U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

"PANG"

Relatório do Projeto

Laboratório de Computadores

Autoria:

MIEIC 2º Ano - Turma 5 - Grupo 01

(LCOM1617-T5G01)

João Carvalho - up201504875- up201504875@fe.up.pt

Renato Campos - up201504942 - up201504942@fe.up.pt

Índice

1. Introdução

No âmbito da unidade curricular de Laboratório de Computadores foi-nos proposto modelar um programa que usasse os conhecimentos detidos nas aulas e tentar aprender mais com isso.

Assim sendo, inicialmente tínhamos pensado em desenvolver um programa que desse para tocar alguns instrumentos sincronizados em diferentes computadores, mas como a conclusão do projeto iria depender muito da utilização de som e poderia haver alguns problemas foi-nos aconselhado partir para outra abordagem. Foi assim que pensamos no *PANG.*

1.1. *PANG*

PANG, também conhecido por *Pomping World*, é um jogo de cooperação entre dois jogadores, lançado para o mercado no ano de 1989 por *Mitchell Corporation*, a ultima versão lançado do jogo foi este presente ano de 2016, com o nome de *PANG ADVENTURES*, desenvolvido pela *DotEmu.*

O objetivo principal do jogo é salvar várias cidades do planeta Terra dos ataques mortíferos de bolas saltitantes. Para isso os jogadores têm de utilizar as suas armas para destruir as bolas, mas estas vão dividindo em bolas mais pequenas até serem realmente destruídas.

2. Instruções de Utilização

2.1. *Menu Inicial*

Quando iniciado o jogo o utilizador é deparado com o menu inicial, com o qual pode interagir através do rato para a seleção de opções. Podendo escolher entre 3 opções “Local Game”, “2PC’s Game” ou “Exit”.

2.2. Menu Local Game

Quando selecionado o menu do Local Game existe 3 opções “Single Palyer”, Multiplayer” e “Back”. “Single Player” para iniciar o jogo em modo de um jogador, “Multiplayer” para iniciar o jogo em modo dois jogadores e “Back” para voltar ao menu inicial.

Single Player

Quando iniciado o jogo em modo Single player o utilizador deve movimentar a sua personagem com as setas do teclado, para a esquerda e para a direita, e pode disparar com a barra de espaços a sua corda para a destruição das bolas saltitonas.

Multiplayer

Quando iniciado o jogo em modo Multiplayer os utilizadores devem movimentar a sua personagem com as setas do teclado, para a esquerda e para a direita, e pode disparar com a barra de espaços a sua corda, no caso do jogador 1, e com as teclas A e D para movimentar e o SHIFT para disparar a sua corda, no caso do jogador 2, para em conjunto destruírem as bolas saltitonas.

2.3. *2PC's. Game*

No modo 2PC's Game, o jogo é para ser jogado em dois computadores os quais comunicam através da porta de série. Para o jogo começar a é necessário ser pressionada a barra de espaços, a qual informa que o jogador 1 está pronto para jogar e a tecla de SHIFT, a qual informa que o jogador 2 também está pronto.

2.4. *Defeat*

Quando os jogadores são atingidos pelas bolas eles perdem as suas vidas. Inicialmente eles começam com 3 vidas e morrem quando têm zero vidas.

2.5. *Victory*

Quando os jogadores destruírem as bolas ambos ganham o jogo e podem voltar ao menu inicial carregando na tecla ESC.

3. Estado do Projeto

Como foi descrito na especificação do projeto foram implementadas todas as funcionalidades pretendidas e ainda realizada uma função em assembly, que não estava previsto.

Periférico	Resumo do uso do periférico	Interrupção
Timer	Temporização do jogo e atualização de variáveis	SIM
Teclado	Ações das personagens	SIM
Rato	Deslocamento no menu e interação com as opções	SIM
Placa Gráfica	Exibe os gráficos necessários	NÃO
RTC	Altera o fundo a ser desenhado dependendo da hora	NÃO
Porta de Série	Envio e receção de ações feitas entre computadores	SIM e NÃO

3.1. *Timer*

O timer está a ser atualizado para gerir o *refresh rate* do ecrã, e a cada interrupção do timer é atualizado o que aparece no ecrã.

Também usado para fazer uma contagem decrescente do tempo da ronda a ser jogada e quando chega a zero perdem o jogo.

Para além disso é também usado para fazer a atualização das posições das personagens e testar as colisões dos objetos.

Ficheiros onde é implementado: timer.c, timer.h e i8042

3.2. *Teclado*

O teclado é apenas usado para ler os *scancodes* das teclas pressionadas, durante o jogo.

Quando pressionadas as teclas, dependendo das teclas que forem, faz o movimento das personagens e o disparo da corda.

Ficheiros onde é implementado: keyboard.c, keyboard.h e i8254

3.3. *Rato*

O rato é usado para a interação com os menus existentes e para a seleção da opção pretendida.

Ficheiros onde é implementado: mouse.c, mouse.h e i8254

3.4. *Placa Gráfica*

A placa gráfica é usada para o desenho dos gráficos no ecrã, quer relativos ao menu quer ao jogo.

Foi usada no modo 0x117, tendo uma resolução de 1024*768, usado a palete 16bits do RGB com a possibilidade de fazer 64K de cores diferentes.

Implementados *double buffer*, de modo a que o movimento dos objetos seja mais fluido.

Não criamos uma fonte completamente nova apenas criamos imagens que as quais já tinham a nossa fonte.

Criamos também sprites animadas para objetos como as personagens que altera os bitmaps, quando corre para os lados e quando dispara a corda.

Ficheiros onde é implementado: graphics.c, graphics.h, vbe.c, vbe.h, BMPLoaded.c e BMPLoaded.h

3.5. *RTC*

O RTC é usado para alterar o fundo do ecrã quando esta a ser jogado dependendo da hora do dia. Esta foi implementada em assembly.

Ficheiros onde é implementado: rtc.c, rtc.h e rtc_asm.s

3.6. *Porta de Série*

A porta de série foi implementada de forma a receber dados sempre que existe uma interrupção desta e são enviadas informações pela porta de série sempre que é pressionada uma tecla.

Ficheiros onde é implementado: uart.c, uart.h

4. Organização de código e Estrutura

4.1. *bitmap.c*

Este módulo contém funções para carregar bitmaps e para os destruir. Este modulo não é da nossa autoria, tendo sendo inteiramente copiado do fórum do Ferolho.

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

Membro do grupo responsável: João Carvalho

Peso do modulo no projeto: 2%

4.2. *BMPLoaded.c*

Neste modulo são implementadas todas as funções onde carregamos os bitmaps necessários para o projeto, contendo também as *sprites* das personagens e das bolas.

As sprites foram criadas de forma a conter um *array* de bitmaps para fazer as animações necessárias.

Também foram criadas as funções para desenhar os bitmaps e as *sprites* no buffer.

Membro do grupo responsável: João Carvalho

Peso do modulo no projeto: 9%

4.3. *game.c*

Este modulo é o modulo principal do projeto.

Foi implementado o principal ciclo de receção de interrupções do teclado, do timer e da porta de serie, que depois chama as funções necessárias para lidar com cada interrupção, desde o handler do keyboard [keyboard_Handler()] ao handler das personagens [handleRH(), handleBH()], …

Também foi implementado, neste modulo uma função de teste de colisões a qual primeiramente testa se as caixas das imagens se estão a intercetar. Caso não estejam, retorna dizendo que não há colisão, no outro caso testa com mais precisão a qual cria um array do mesmo tamanho do buffer e primeiro escreve uma imagem nesse array e depois escreve a segunda imagem no caso de quando está a escrever a segunda imagem se naquela posição do array já tiver sido escrito alguma coisa significa que houve colisão das duas imagens.

Membro do grupo responsável: João Carvalho

Peso do modulo no projeto: 30%

4.4. *graphics.c*

Neste modulo foi implementada as funções que controlam a placa gráfica, desde o inicio do modo gráfico até á alteração da cor dos pixéis.

É neste que também foi implementado a estrutura das sprites e as funções correspondentes.

Membro do grupo responsável: João Carvalho

Peso do modulo no projeto: 9%

4.5. *i8042*

É um módulo apenas de macros.

Membro do grupo responsável: João Carvalho

Peso do modulo no projeto: 3%

4.6. *i8254*

É um módulo apenas de macros.

Membro do grupo responsável: Renato Campos

Peso do modulo no projeto: 3%

4.7. *keyboard.c*

Neste modulo foram implementadas as funções de *subscribe* e *unsubscribe* relativas ao teclado, bem como funções para a leitura do *buffer e sua limpeza.*

Membro do grupo responsável: Renato Campos

Peso do modulo no projeto: 6%

4.8. *main.c*

Este é o modulo de inicio como o nome indica, apenas executa a inicialização do modo gráfico e chama a função do menu inicial.

Membro do grupo responsável: Renato Campos

Peso do modulo no projeto: 2%

4.9. *menu.c*

Este modulo foi implementado à base de uma maquina de estados, a qual varia com o estado atual e o evento criado que faz com que esta troque de estado ou apenas desenhe um título diferente no ecrã.

É usado um ciclo principal que recebe as interrupções do rato que depois chama uma função que recebe o *packet* do rato de modo a atualizar as informações da maquina de estados e da posição do rato.

Membro do grupo responsável: João Carvalho

Peso do modulo no projeto: 10%

4.10. *mouse.c*

Neste módulo são implementadas as funções correspondentes ao *subscribe* do rato e ao *unsubscribe* das interrupções, assim como a leitura do buffer.

Para além das funções já descritas foram criadas também funções para ativar e desativar o rato, ou seja, receber e parar de receber dados do rato.

Membro do grupo responsável: Renato Campos

Peso do modulo no projeto: 7%

4.11. *rtc.c e rtc_asm.s*

Este modulo foi implementado de forma a chamar uma função que lê os dados do rtc em assembly, lendo a hora e os minutos.

Membro do grupo responsável: Renato Campos

Peso do modulo no projeto: 4%

4.12. *timer.c*

Neste modulo foram implementadas as funções correspondentes à manipulação do timer como o *subscribe* e *unsubscribe* e uma função que atrasa a execução de instruções quando necessário.

Membro do grupo responsável: Renato Campos

Peso do modulo no projeto: 7%

4.13. *uart.c*

Este modulo corresponde às funções implementadas para manipular a porta de serie. Desde o *subscribe* e o *unsubscribe*, à leitura e escrita no buffer da porta de serie.

Membro do grupo responsável: João Carvalho

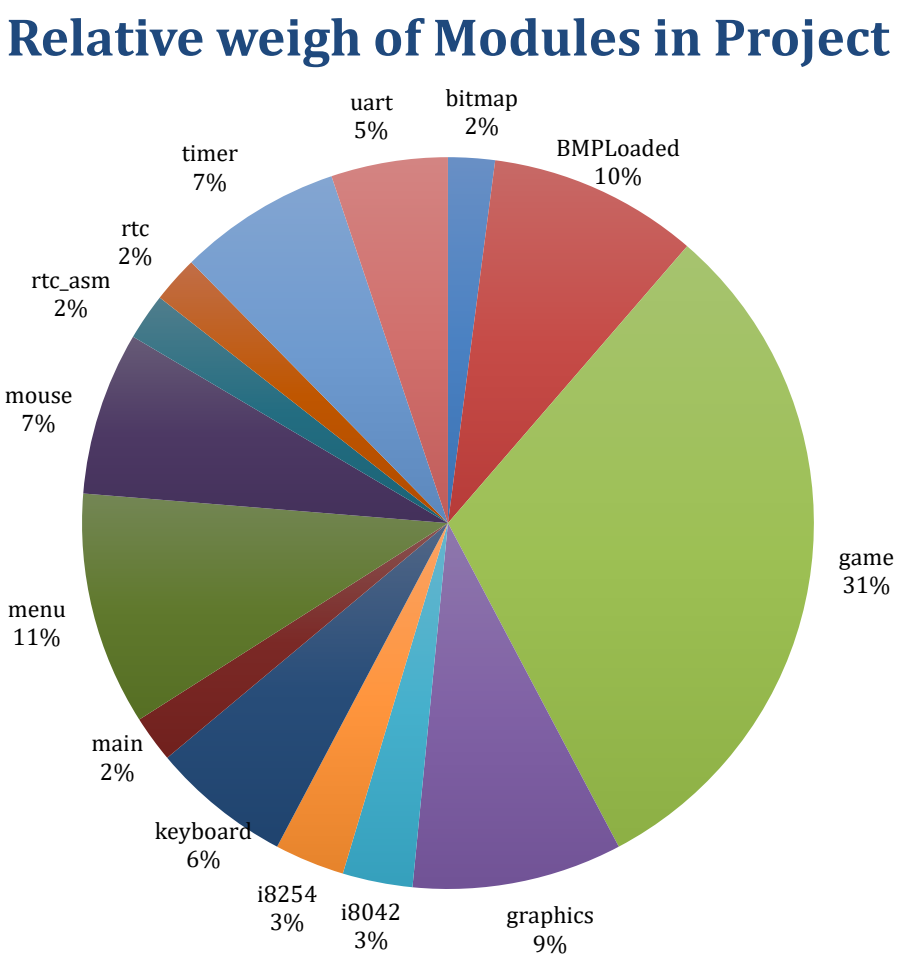
Peso do modulo no projeto: 5%

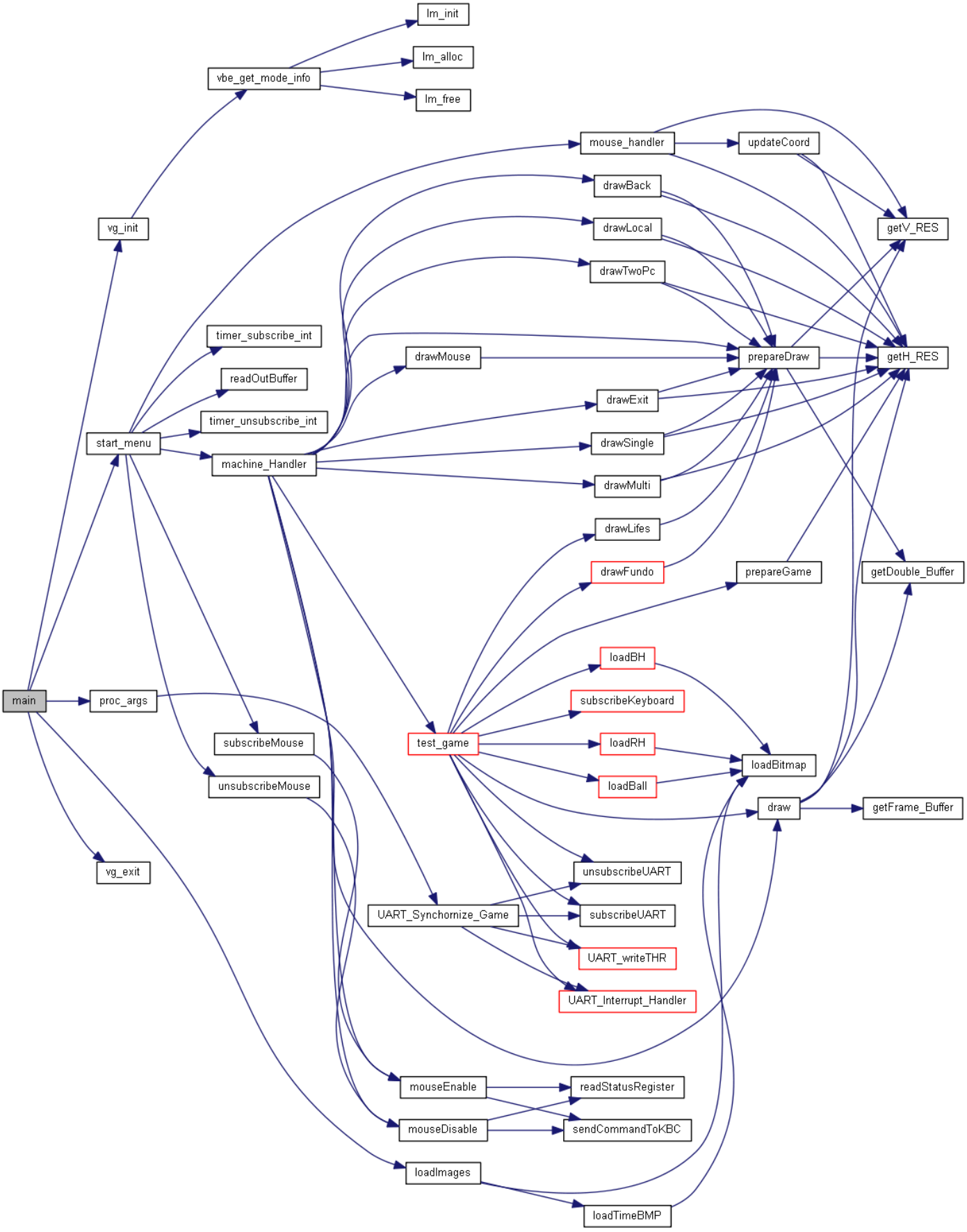
4.14. *vbe.c*

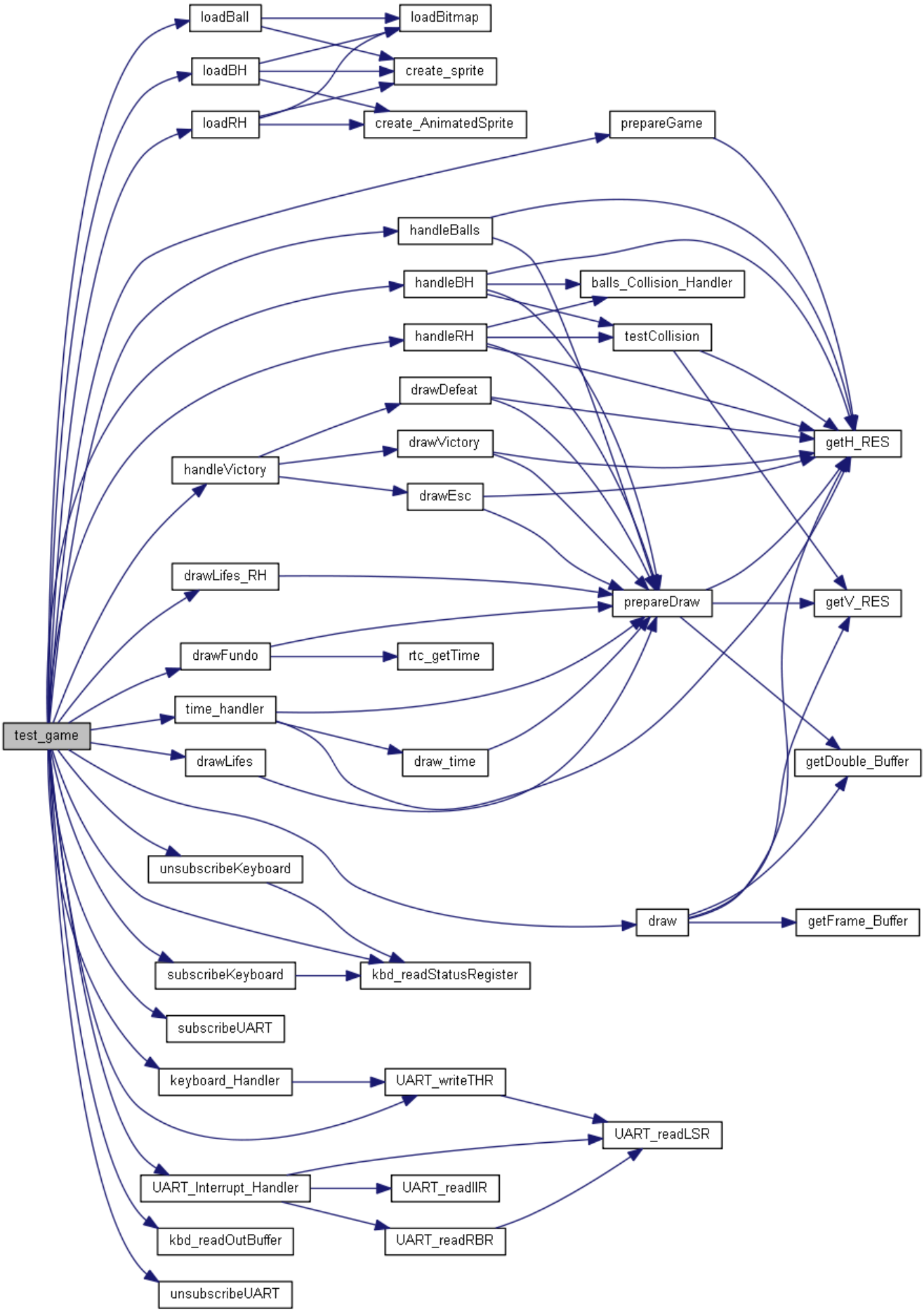
Este modulo como feito no lab da placa gráfica é utilizado apenas para criar o buffer na memoria virtual correspondente ao buffer da memoria física.

Membro do grupo responsável: Renato Campos

Peso do modulo no projeto: 3%


Relative weigh of Modules in Project
uart 5%
bitmap 2%
BMPLoaded 10%
timer 7%
rtc 2%
rtc_asm 2%
mouse 7%
game 31%
menu 11%
main 2%
keyboard 6%
i8254 3%
i8042 3%
graphics 9%


lm_init
lm_alloc
lm_free
vbe_get_mode_info
vg_init
mouse_handler
updateCoord
drawBack
drawLocal
drawTwoPc
getV_RES
getH_RES
timer_subscribe_int
readOutBuffer
timer_unsubscribe_int
drawMouse
prepareDraw
drawExit
drawSingle
drawMulti
drawLifes
drawFundo
start_menu
machine_Handler
prepareGame
getDouble_Buffer
loadBH
subscribeKeyboard
loadRH
loadBall
loadBitmap
draw
getFrame_Buffer
main
proc_args
subscribeMouse
unsubscribeMouse
test_game
vg_exit
unsubscribeUART
subscribeUART
UART_writeTHR
UART_Interrupt_Handler
UART_Synchornize_Game
mouseEnable
readStatusRegister
mouseDisable
sendCommandToKBC
loadImages
loadTimeBMP


loadBall
loadBitmap
loadBH
create_sprite
loadRH
create_AnimatedSprite
prepareGame
handleBalls
handleBH
balls_Collision_Handler
handleRH
testCollision
drawDefeat
getH_RES
drawVictory
handleVictory
drawEsc
drawLifes_RH
prepareDraw
getV_RES
drawFundo
rtc_getTime
time_handler
test_game
drawLifes
draw_time
getDouble_Buffer
unsubscribeKeyboard
draw
getFrame_Buffer
subscribeKeyboard
kbd_readStatusRegister
subscribeUART
keyboard_Handler
UART_writeTHR
UART_readLSR
UART_Interrupt_Handler
UART_readIIR
kbd_readOutBuffer
UART_readRBR
unsubscribeUART

5. Detalhes da Implementação

5.1. *Colisões*

Uma das implementações que achamos relevante falar um pouco é sobre a função de teste de colisões.

```
int testCollisio (Bitmap *obj1, int xObj1, int yObj1, Bitmap *obj2, int xObj2, int yObj2);
```

Esta função pode ser separada um duas partes.

A parte inicial na qual apenas testa se as caixas das imagens estão a se intercetar e caso isso não se verifique retorna 0 de modo a informar que não houve colisão.

A segunda parte é um método mais complexo, tanto algoritmicamente como computacionalmente, mas só entra nesta fase caso a primeira parte se verificar. Foi importante ser implementada porque como usamos imagens com formas um pouco complexas, assim evitamos que a colisão seja detetada de uma forma grosseira com *hitboxs* que apenas se aproximariam à imagem real.

Para isso foi criado um array do tamanho do buffer da memoria gráfica. Posteriormente é escrita a primeira imagem da mesma forma que seria como se fosse no buffer gráfico e depois a segunda imagem. Para testar a colisão é no ato de escrita da segunda imagem no buffer que é verificado se aquela posição no array já contêm alguma informação correspondente à primeira imagem, isto é, se nessa posição for diferente a 0 significa que já foi escrita alguma cor primeiramente detetando assim colisão e sai da função retornando 1.

5.1. *UART*

Para além do teste das colisões achamos relevante falar da porta de serie dado que não é um laboratório desenvolvido nas aulas e teve de ser feito por inteira interpretação dos slides.

No desenvolvimento da UART, achamos que não era necessário utilizar em modo FIFO uma vez que nós só tínhamos dados a enviar quando era pressionada alguma tecla.

Para simplificar usamos o modo que transmite 8 bits de informação, assim podemos enviar um caracter que corresponde à tecla carregada.

6. Considerações Finais

6.1. *Instalação*

Para iniciar o programa é necessário colocar o repositório do projeto no path:

/home/lcom/svn/

Posteriormente basta entrar na pasta do projeto e executar o script run.sh, usando o comando:

sh run.sh

6.2. *Avaliação global da Unidade Curricular*

Achamos que a unidade curricular de LCOM é muito vantajosa para o nosso desenvolvimento enquanto programadores. Não só por ficarmos com um vasto conhecimento de linguagem C, mas também do funcionamento dos periféricos e principalmente de programação de mais baixo nível ao que estamos habituados.

Contudo achamos que a primeira abordagem que temos é muito "agressiva". No início mesmo indo a aulas teóricas é muito complicado de perceber a logica necessária a ter para fazer o primeiro laboratório, apesar de agora nos parecer uma coisa simples, mas se no inicio houvesse mais ajuda ou então as coisas fossem explicadas com mais pormenor seria claramente mais fácil.