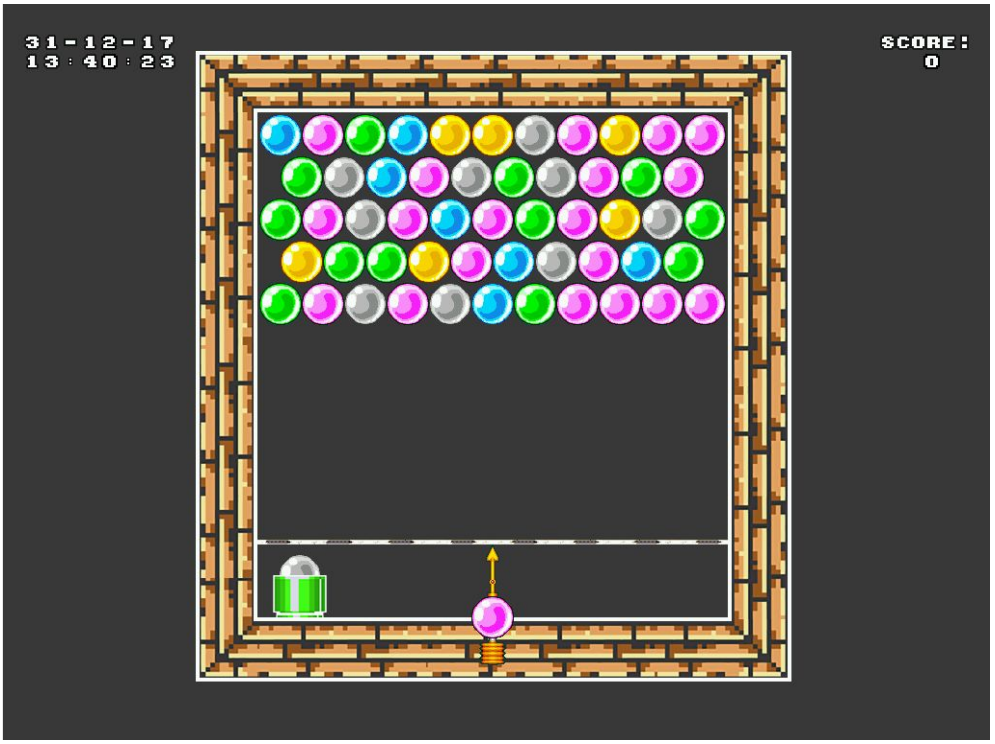
U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Relatório do projeto

“ColourPop”



Laboratório de Computadores 2017/2018

Turma 5 Grupo 6

Beatriz Mendes **up201604253**@fe.up.pt

Joana Silva **up201208979**@fe.up.pt

Índice

Resumo do Jogo

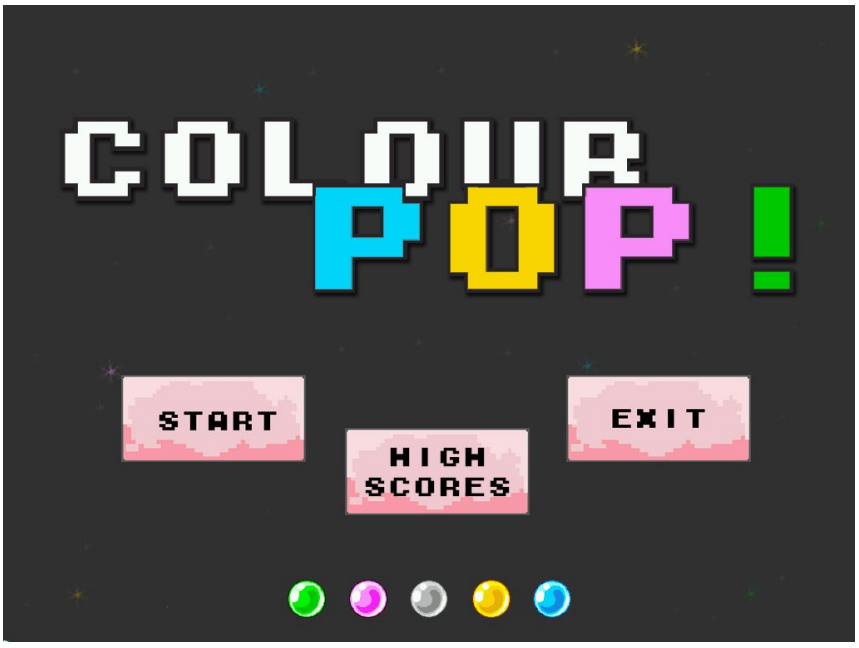
A ideia do jogo é baseada numa fusão dos jogos Puzzle Bobble e Bubble Shooter.

No ecrã é disposta uma matriz de *bubbles* com várias cores. Para as remover, o jogador deverá atirar *bubbles*, uma a uma, de forma em que estas colidem com *bubbles* da mesma cor. Se a *bubble* que é atirada, ao entrar na matriz, formar um grupo de 3 ou mais *bubbles* da mesma cor, este grupo de *bubbles* rebenta, saindo da matriz e aumentando a pontuação do jogador.

Periodicamente é adicionada uma nova linha no início da matriz, tendo do jogador de impedir que as *bubbles* ultrapassem a linha de limite, uma vez que perde o jogo se ultrapassarem.

Instruções de Utilização

Main Menu

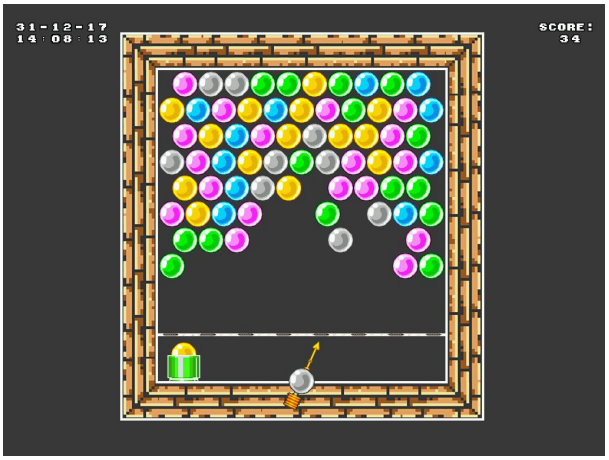


Quando o utilizador inicia o programa aparece o menu principal do jogo, onde são apresentadas 3 opções que podem ser selecionadas com o cursor do rato:

- Start – inicia um novo jogo
- Highscores – mostra as pontuações mais elevadas até ao momento, exibindo também os dias em que foram obtidas
- Exit – sai do programa

Play Mode

Quando o utilizador entra no modo jogo, é pedido para introduzir um nome (com o máximo de 4 caracteres), nome este que depois vai ser mostrado no menu de Highscores, dependendo da pontuação do jogo.



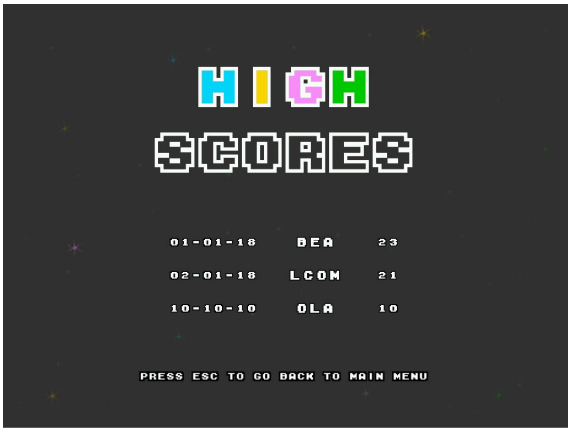
Para jogar, o utilizador pode escolher movimentar a seta e disparar as *bubbles* com o rato ou com o teclado. À medida que vai fazendo com que grupos de três ou mais *bubbles* colidam, a pontuação, disposta no canto superior direito, vai aumentando.

No canto superior esquerdo é mostrada a data e hora atual.

Paused Menu

Ao pressionar a tecla “P”, o jogo é encaminhado para o menu de pausa, para voltar novamente ao modo jogo, basta pressionar a tecla “R”.

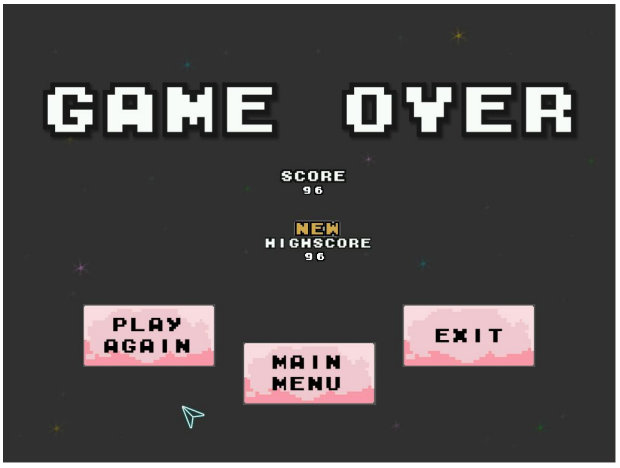
Highscores Menu

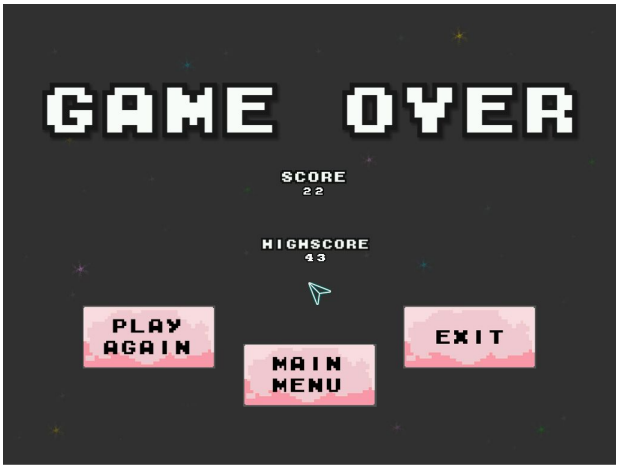


O menu Highscores dispõe a informação sobre as pontuações mais elevadas, tais como, a data em que foram obtidas, o nome do jogador e a pontuação.

Ao pressionar a tecla “ESC”, o jogo volta para o menu principal.

Game Over Menu





Ao deixar que as *bubbles* ultrapassem a linha de limite de jogo, o jogador perde, sendo redirecionado para o menu Game Over. Neste menu, o utilizador consegue ver a sua pontuação final e verificar se conseguiu atingir uma highscore, tendo também a opção de jogar outra vez ao selecionar com o cursor o botão "Play Again", voltar ao menu principal através do botão "Main Menu" ou sair do programa através do botão "Exit".

Periféricos Usados

Na realização deste jogo foram usados todos os periféricos abordados na cadeira, com exceção da porta de série. A tabela em baixo contém a informação das funcionalidades utilizadas de cada dispositivo, assim como também, a forma de como foram usados (com interrupções ou não).

Dispositivo	Funcionalidade	Interrupções
Timer	Atualização do estado do jogo	Sim
Teclado	Navegar entre menus e interface do jogo	Sim
Rato	Navegar entre menus e interface do jogo	Sim
Placa Gráfica	Menus e interface do jogo	Não
RTC	Mostrar data e hora atual, guardar a data em que as highscores foram atingidas	Sim

Timer

Este dispositivo tem a função principal de atualizar o estado do jogo a cada interrupção, mais especificamente de desenhar os gráficos. Através da função *check_timer_events*, dependendo do estado em que se encontra o jogo, é desenhado o menu pretendido, sendo também nesta função onde periodicamente se adiciona uma nova linha ao início da matriz.

Teclado

Este dispositivo, assim como o rato, é usado na lógica do jogo e na navegação de menus.

No início da parte de gameplay do programa, é utilizado para o utilizador introduzir o nome de jogador (limitamos até 4 caracteres). Este é usado para gravar a pontuação do mesmo caso obtenho uma highscore.

Durante o jogo em si, é usado para mover a seta que coordena a direção do lançamento das *bubbles* e também por lançá-las. Através das teclas "A" e "D", a seta movimenta-se para a esquerda e direita, respetivamente, e , ao clicar no espaço, a *bubble* é lançada para a matriz na direção para onde aponta a seta.

No modo jogo também se pode utilizar o teclado para sair do programa, ao clicar na tecla "ESC".

Rato

Dispositivo usado na navegação de menus e na própria interface do jogo em par com o teclado.

Durante o jogo, o utilizador, para além de poder jogar com o teclado, também tem a possibilidade de usar o rato para jogar. Ao mexer o cursor, a seta que indica o local para onde a *bubble* será lançada acompanha o movimento do rato, podendo, de seguida, o jogador lançar a *bubble* ao pressionar o botão do lado esquerdo.

Placa Gráfica

Um dos dispositivos mais importantes do projeto, já que é responsável por desenhar todas as imagens do jogo no modo *RGB 5 6 5* que possui 2^{16} cores. Foi usado o modo gráfico 0x117, tendo uma resolução de 1024*768 pixéis.

De forma a melhorar a jogabilidade e tornar o jogo mais fluído, implementamos a técnica de double buffer, fazendo a cópia para o video mem a cada interrupt do

timer (*flipBuffer* - *video_gr.c*). Usamos também animações para a explosão das *bubbles* e quando o jogador perde o jogo.

Usamos fonts na atualização da data e hora (*RTC*) e na para guardar o nome do jogador e dos jogadores que possuem as 3 maiores high scores.

Para detetar as colisões entre as *bubbles* usamos um algoritmo que verifica se colide num raio igual ao da largura do bitmap usado para as representar.

RTC

O dispositivo RTC (Real Time Clock) foi usado, como o nome indica, para ler a data e a hora atual.

Encontramos duas formas de usar este dispositivo. Durante o modo jogo é usado para mostrar a data (dia, mês e ano) e a hora (horas, minutos e segundos) atuais no ecrã e no menu dos Highscores é mostrada a data do dia em que cada pontuação foi obtida.

Descrição da Arquitetura do Programa
(Módulos)

Main

Módulo por onde começa a ser executado o programa. Inicia-se o modo gráfico pretendido, a máquina de estados e cria-se um apontador ColourPop para um objeto da struct Game, onde se chama a função para iniciar o jogo. De seguida, chama-se a função que irá tratar de todas as interrupções e, por último, quando o jogo acaba, sai-se do modo gráfico, voltando para o modo de texto.

Desenvolvimento: *Beatriz Mendes (50%) | Joana Silva (50%)*
Peso no projeto: *2%*

Bitmap

Módulo responsável por carregar, destruir e desenhar as imagens necessárias para o projeto, as quais foram guardadas em formato bitmap e com cor de fundo *0x006666*. O código usado foi o disponibilizado pelo antigo aluno Henrique Ferrolho (*http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html*), porém foram realizadas algumas alterações de modo a desenhar imagens com transparência. Esta última funcionalidade foi conseguida ao ignorar os pixéis que tivessem a cor que designamos *cor de fundo*.

Desenvolvimento: *Beatriz Mendes (50%) | Joana Silva (50%)*
Peso no projeto: *4%*

Bubbles

Este módulo possui todas a funções relativas a uma *bubble* tais como desenhar, gerar e apagar as *bubbles* usadas no jogo. Neste módulo também estão implementadas as funções de animação das *bubbles* - *explodeBubble*, *turnBlack*. Por último, contém também a função *checkColision* que verifica se uma *bubble* está a colidir com outra.

Desenvolvimento: *Beatriz Mendes*
Peso no projeto: *6%*

Game

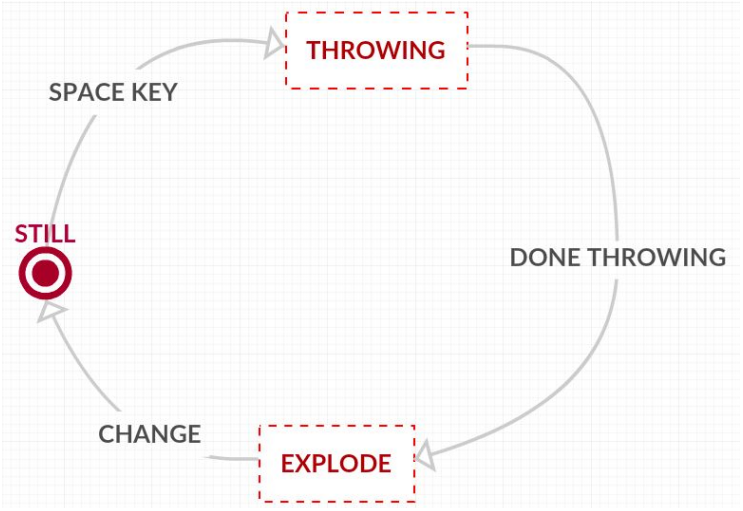
Módulo base de todo o jogo. Apresenta toda a estrutura Game e contém o ciclo principal de todo o jogo, que processa todas as interrupções de todos os dispositivos presentes.

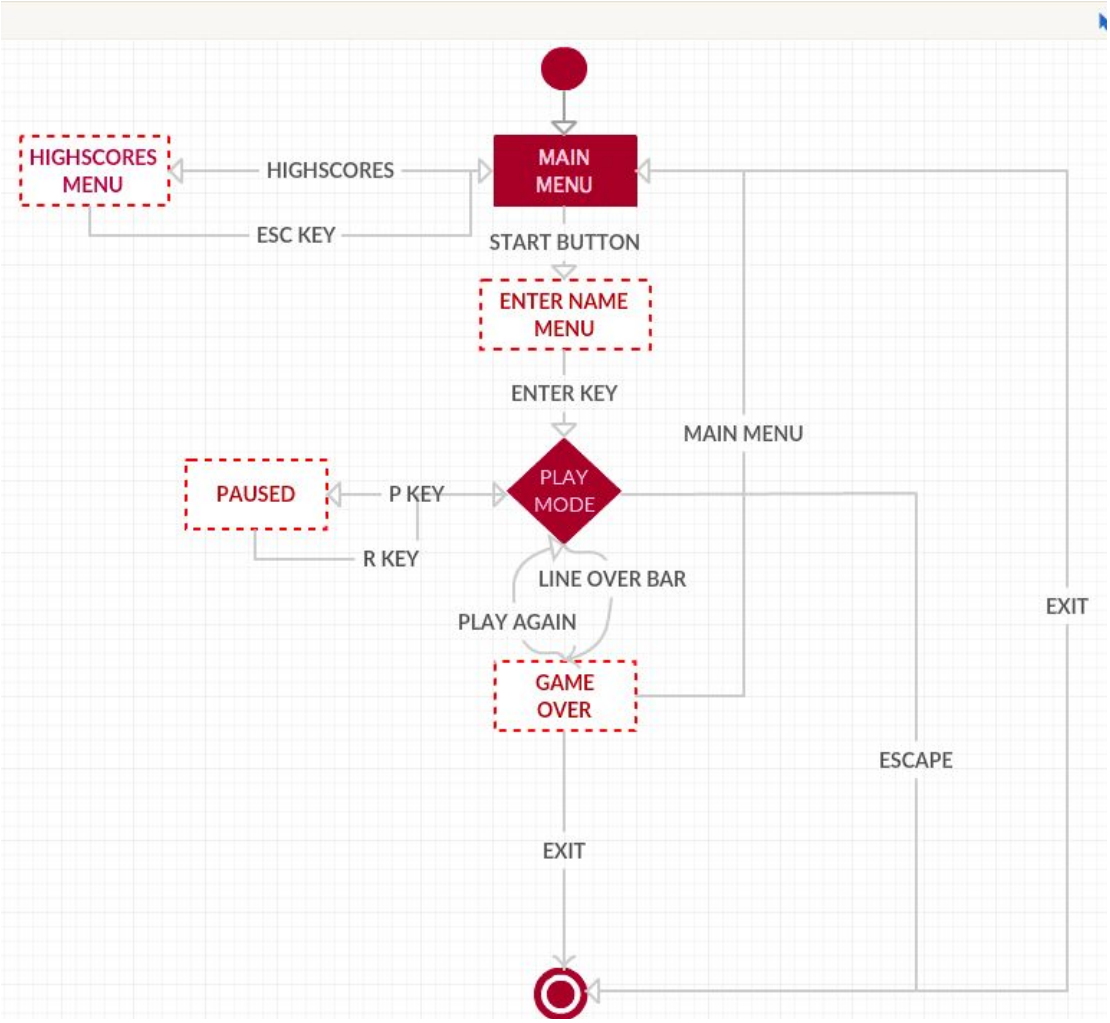
Este módulo possui as seguintes funções:

- *Game * startGame* - inicializa toda a estrutura do jogo. Dá subscribe a todos os dispositivos usados, carrega maior parte dos bitmaps, aloca memória para o jogo e cria um apontador para um objeto da struct Date, onde vai ser coordenada a data lida pelo RTC.
- *int interrupt_h(Game * ColourPop)* - ciclo principal de todo o jogo que processa todas as informações recebidas pelos periféricos e dá unsubscribe a todos os dispositivos utilizados no fim da função.
- *void checkIfLost(Game * ColourPop)* - função que verifica se o número de linhas da matriz excedeu a barra limite de jogo e faz update aos highscores.
- *void playAgain(Game * ColourPop)* - função que aloca na memória um novo jogo.
- *void freeGame(Game * ColourPop)* - liberta a memória usada no jogo.

Desenvolvimento: *Beatriz Mendes (50%) | Joana Silva (50%)*
Peso no projeto: *11%*

Gamestates





Neste módulo estão implementadas duas máquinas de estados: uma referente ao estado do jogo, possuindo estados como: *MENU*, *HIGHSCORES_MENU*, *ENTER_NAME_MENU*, *PLAY_MODE*, *PAUSED*, *GAME_OVER, DONE;* usada na mudança entre menus. A segunda máquina de estados implementada avalia ao estado da *bubble* que é lançada possuindo estados como: CHANGE, THROWING, EXPLODING; usada para considerar todas as ocorrências possíveis.

Desenvolvimento: *Beatriz Mendes*
Peso no projeto: *4%*

Graphics

Este módulo é responsável por desenhar quase todos os bitmaps necessários para o jogo, nomeadamente todos os menus existentes no jogo, a matriz das *bubbles*, a data e hora, a pontuação e o nome do jogador. Neste módulo também encontramos declaradas e definidas as structs Spear, Bar e Score.

Desenvolvimento: *Beatriz Mendes (50%) | Joana Silva (50%)*
Peso no projeto: *4%*

Highscores

Módulo responsável por ler e escrever os nomes dos jogadores, as pontuações mais elevadas e as datas em que foram obtidas no ficheiro *higscores.txt*.

Desenvolvimento: *Beatriz Mendes*
Peso no projeto: *2%*

Matrix

Este módulo também é um dos mais cruciais do projeto, já que contém todas as funções necessárias para o funcionamento da matriz de *bubbles* do jogo. É responsável pelas funções de alocar na memória a matriz (*allocateInitialMatrix* e *allocateRow*), pela função de acrescentar uma linha de *bubbles* ao início da matriz (addRow) e todas as funções que advêm desta ação (checkIfRowAdded e checkIfRowDeleted). Apresenta também a função que coloca as *bubbles* na matriz e as funções que têm a haver com a explosão das mesmas. Por fim, é o módulo que compara as cores das *bubbles* vizinhas através da função *checkNeighbours*.

Desenvolvimento: *Beatriz Mendes (50%) | Joana Silva (50%)*
Peso no projeto: *14%*

Keyboard

Código importado do lab 2, ao qual foi acrescentado a função *check_mouse_events* que faz o update dos elementos controlados pelo rato conforme o estado em que o jogo está.

Desenvolvimento: *Beatriz Mendes (70%) | Joana Silva (30%)*
Peso no projeto: *10%*

Mouse

Código importado do lab 2, ao qual foi acrescentado a função *check_mouse_events* que faz o update dos elementos controlados pelo rato conforme o estado em que o jogo está.

Desenvolvimento: *Beatriz Mendes (10%) | Joana Silva (90%)*
Peso no projeto: *10%*

Timer

Código importado do lab 2, ao qual foi acrescentado a função *check_timer_events* que faz o update da frame conforme o estado em que o jogo está.

Desenvolvimento: *Beatriz Mendes (50%) | Joana Silva (50%)*
Peso no projeto: *10%*

RTC

Este módulo faz a leitura da informação dos registos do RTC (Real Time Clock), nomeadamente da hora e data atual, atualizando-os (*updateHour* e *updateDate*). Uma vez que a informação obtida pode estar em BCD, vamos verificar se isso acontece com a função *CheckIfBCD*, no caso de se confirmar, convertemos a data para binário através de um algoritmo de conversão que foi encontrado na internet (*https://stackoverflow.com/questions/28133020/how-to-convert-bcd-to-decimal*).

Desenvolvimento: *Joana Silva*
Peso no projeto: *3%*

Read_asm

Leitura de informação para o rato e teclado através de assembly. Código importado das aulas do laboratório 3, mas modificado, de modo a ser mais eficiente e a também ler informação para o rato e não só para o teclado.

Desenvolvimento: *Joana Silva*
Peso no projeto: *1%*

Vbe

Código importado das aulas do laboratório 5.
Peso no projeto: *2%*

Video_gr

Código importado das aulas do laboratório 5, ao qual foi acrescentado funções que retornam a resolução usada, os buffers e a função que copia do double buffer para o *video mem* (*getGraphicsBuffer*,*getSecondBuffer*,*flipBuffer*).

Desenvolvimento: *Beatriz Mendes (50%) | Joana Silva (50%)*
Peso no projeto: *5%*

i8042

Código importado das aulas práticas dos laboratórios 3, 4 e 5, ao qual foram adicionadas umas definições relacionadas com o jogo, tais como o números de linhas iniciais da matriz de *bubbles*, número de colunas de *bubbles* por linha, entre outros.

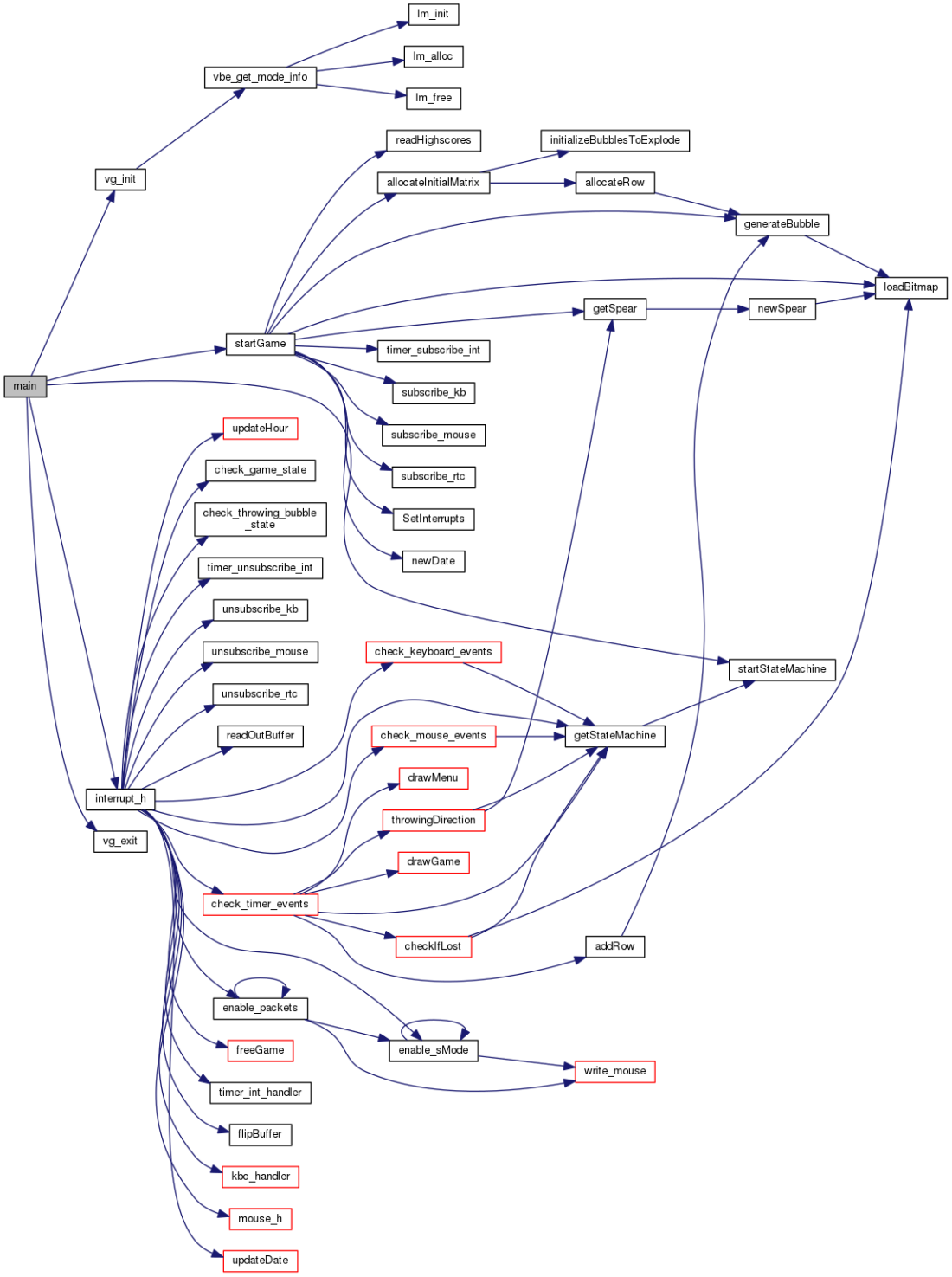
Peso no projeto: *1%*

i8254

Código importado das aulas do laboratório 2.

Peso no projeto: *1%*

Gráfico de chamada de funções



Nota: Está incluída a função *interrupt_h* que chama a *driver_receive.*

Detalhes de implementação

Como referido anteriormente, a forma que usamos para simplificar o código e tornar mais fácil a consideração de todos os eventos que possam ocorrer durante o jogo foi implementar duas máquinas de estados, a primeira para avaliar os estados do jogo e a segunda para avaliar os estados do *bubble* que é lançada.

O desenvolvimento do código foi feito incrementalmente, por camadas, como nos foi aconselhado. Está bastante dividido em funções mais simples de modo a ser mais organizado e conciso.

Achamos relevante referir ainda que a as *bubbles* apresentadas no ecrã durante o jogo são organizadas numa matriz dinâmica conseguida por uma array bidimensional, ao qual são adicionados elementos no início (ao adicionar novas linhas), no fim (ao atirar novas *bubbles*) e são eliminados. As colisões das respetivas bubbles foi feita através de um algoritmo que percorre a área circular de um elemento e vê se há elementos que se sobrepõem.

Considerações finais

Dificuldades no projeto

Na realização deste projeto, a nossa maior dificuldade foi na lógica do jogo. Inicialmente, ao descobrir qual a forma mais eficiente para criar a matriz do jogo e posteriormente, o mais complicado, perceber como guardar as *bubbles* que eram atiradas no lugar certo da matriz, como ver as colisões entre *bubbles*, verificar quando as mesmas deviam de explodir e remover as *bubbles* flutuantes. Contudo, eventualmente essas dificuldades foram sendo ultrapassadas, apesar de permanecer a dificuldade de remover as *bubbles* flutuantes, uma vez que não conseguimos arranjar uma forma ideal de apagar as *bubbles* que deixaram de estar a colidir com a *bubble* anteriormente atirada e que deixaram de estar na matriz.

Como é normal, no início do projeto e no seu desenvolvimento foram aparecendo algumas dúvidas e problemas, no entanto, com o esforço e empenho de ambos os membros e das várias horas dedicadas ao mesmo, esses problemas foram sendo superados.

Quanto ao código em assembly, decidimos que era mais benéfico melhorar o código da leitura do teclado anteriormente feito no laboratório 3 e aplicá-lo também à leitura do rato. Também pensamos que poderíamos fazer o RTC em assembly, porém, não vimos vantagens na implementação do nosso código, uma vez que o código em C nos deu mais alternativas para a realização do nosso projeto.

O *RTC*, embora não trabalhado nas aulas práticas, foi implementado com alguma facilidade.

Por último, inicialmente, tinhamos planeado implementar UART, no entanto, decidimos, por falta de tempo, dedicar a nossa atenção completa às restantes componentes do projeto.

Avaliação da unidade curricular

Esta unidade curricular é bastante enriquecedora para nosso crescimento enquanto programadores, já que obtemos um vasto conhecimento e experiência de programar em linguagem C. Contudo, achamos que está implementada de uma forma um pouco desvantajosa, dado que, no início é complicado perceber o que fazer nos laboratórios, não havendo uma explicação muito detalhada de como proceder. Achamos, portanto, que seria mais benéfico para os alunos haver um maior acompanhamento ou uma documentação mais pormenorizada no início da cadeira.

Instruções de instalação

Para a compilação e execução do programa usamos scripts para automatizar este processo para qualquer *MINIX*.

 # cd proj
 # su
 # sh scripts/install.sh
 # sh scripts/compile.sh
 # sh scripts/run.sh

O terceiro comando serve para instalar o ficheiro de configuração do sistema necessário para a execução do programa e facilitar o acesso às imagens da pasta *res.* Caso não seja a primeira execução do programa, este comando não precisa de ser executado.