U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Christmas Run

Laboratório de Computadores 2016/2017

Turma 4 – Grupo 7

2 de janeiro de 2017

Bárbara Silva **up201505628**@fe.up.pt

Julieta Frade **up201506530**@fe.up.pt

Índice

Instruções de Utilização

Main Menu

Ao iniciar o programa, é mostrado ao utilizador o menu inicial com as seguintes opções:

- Play, inicia um novo jogo.
- Options, redireciona para um menu de escolha de personagem.
- Exit, sai do programa.

O menu permite usar o rato para selecionar uma das 3 opções, no caso da opção Exit, poderá também pressionar a tecla “ESC”.

Play

O objetivo do jogo é evitar todos os obstáculos que irão aparecer no seu caminho durante o maior tempo possível, usando quer o teclado como o rato para controlo da personagem. Relativamente ao teclado, ao pressionar a tecla W ou espaço, a personagem salta, e ao pressionar a tecla S, a personagem baixa, caso continue com esta tecla pressionada, a personagem irá correr sempre agachada. Caso o utilizador deseje usar o rato, o botão esquerdo serve para saltar e o direito para baixar.

Durante o jogo, no canto superior direito, vai sendo incrementada a pontuação dessa jogada (*score*).

Quando a personagem colidir com um obstáculo, perde o jogo, sendo direcionado para um menu de *Game Over*.

Decidimos implementar também a desistência do jogo, se o jogador quiser sair, basta pressionar a tecla "ESC" que o encaminha para o menu de *Game Over*.

Options Menu

Neste menu, o jogador pode escolher a sua personagem com o rato, entre 3 opções, sendo indicada a sua atual com uma placa em cima do retângulo com a imagem escolhida. Ao iniciar o programa pela primeira vez, a rena será a personagem pré-definida para o jogo, podendo ser alterada pelo o utilizador.

Após estar satisfeito com a sua escolha, tem a opção de voltar ao menu inicial (*Home*).

Game Over Menu

Sempre que o jogador perder ou desistir, é direcionado para este menu, onde pode: visualizar a sua pontuação naquela jogada e escolher com o rato, jogar novamente (*Play Again*) ou voltar ao menu inicial (*Home*).

Estado do Projeto

Dispositivos Usados

Dispositivo	Utilização	Interrupção?
Timer	Atualização do estado do jogo	Sim
Teclado	Interface com o jogo e atalho no menu	Sim
Rato	Navegação nos menus e interface com o jogo	Sim
Placa Gráfica	Desenho de imagens	Não
Real Time Clock	Contagem decrescente para uma dada data	Não

Timer

A principal função do timer é a atualização do estado do jogo, nomeadamente para desenhar os gráficos do jogo, isto é, atualizar as coordenadas da personagem, dos obstáculos, do rato, e também dos fundos – *graphics.c*. Trata-se do dispositivo mais importante do jogo, pois todo o tipo de informação é manuseado pelas interrupções do timer – *game.c*.

Teclado

Este dispositivo é usado na lógica do jogo, para além do rato. No menu inicial é usado para sair do programa (opção *Exit*) quando carregada a tecla "ESC" e durante o jogo esta tecla é usada para desistir.

Durante o jogo em si, serve para movimentar a personagem, nomeadamente faze-la saltar (tecla W ou Espaço) ou baixar (tecla S), tendo em conta que se deixar a tecla S pressionada, ela ira correr sempre abaixada – *xmasrun.c ; void kbdManager | graphics.c ; int jump, void movedown.*

Rato

Este, é também um dos dispositivos mais importantes do jogo, pois é a forma de navegação em todos os menus, a partir da posição do cursor, assim como uma opção de controlo sob a personagem, se pressionar o botão esquerda ela salta, se for o direito ela baixa.

Durante o jogo, é uma alternativa que dispomos ao utilizador, tanto pode usar o rato como teclado para movimentar a personagem.

Placa Gráfica

Este dispositivo é a base de todo o jogo, visto que é utilizado para desenhar todas as imagens [bitmap, modo RGB (5:6:5)] usadas no programa. O modo vídeo que decidimos usar é 0x117, de resolução 1024 por 768 pixéis – *vídeo_gr.c*.

De modo a tornar o jogo fluído, decidimos implementar a técnica de *double buffering*, assim como a animação de *sprites*, para simular o movimento da personagem, que foi implementada da seguinte forma: alternação entre duas *frames* a cada 10 interrupções – *graphics.c ; void updateFrame*. Quanto à deteção de colisões, achamos mais indicado desenvolver um algoritmo de deteção de colisões entre retângulos – *graphics.c ; int collision*.

As fontes são usadas para apresentar a *score* no ecrã e a contagem decrescente no menu através de um algoritmo que a partir de um inteiro devolve o bitmap do numero a dar display – *graphics.c | Bitmap* numberToBitmap*

Real Time Clock (RTC)

O RTC é usado para ler a data e hora atual.

O melhor partido que decidimos tirar do mesmo foi: efetuar uma contagem decrescente com os dias, horas, minutos e segundos para o próximo Natal. Assim, sempre que o utilizador estiver no menu inicial, poderá visualizar a mesma.

Achamos indicado utilizar esta funcionalidade tendo em conta ser o tema do trabalho – *rtc.c | graphics.c ; int daysLeft, int rdn, int secondsLeft, int minutesLeft, int hoursLeft, void countDown.*

Organização de Estrutura do Código

Bitmap

Este módulo é responsável por carregar, desenhar e apagar imagens bitmap, cujo código é da autoria de Henrique Ferrolho, publicado no seu blog (source), com pequenas alterações em termos de desenho da imagem, onde aplicamos o método de desenhar imagens com transparência. As imagens são feitas no programa Adobe Photoshop, com fundo cor de rosa puro (0xffffffff8). Quando a imagem for desenhada, concebemos um algoritmo que ignora esta mesma cor, de modo a desenhar só a parte da imagem que queremos, e assim, concebendo transparência.

Módulo desenvolvido por: Bárbara Silva e Julieta Frade (90%/10%).

Game

Este é o módulo principal, onde se inicializa a estrutura *game,* onde ficarão guardadas várias informações fundamentais ao programa, como o número de interrupções, o *scan* do teclado, todas as imagens, e informação proveniente do RTC.

Tem 4 funções:

- *Game* startGame* – que cria a estrutura, inicializa todas as variáveis da mesma (acima referidas), e chama as funções de *subscribe* e *unsubscribe*. Resumidamente, inicializa o programa de maneira a poder receber toda a informação dos periféricos.
- *void interruptHandler* – é a função principal, pois contém o ciclo principal do jogo, onde processa todas as notificações recebidas dos periféricos até ao ponto de paragem se verificar, quando o estado do jogo for *COMP*.
- *void deleteBitmaps* – liberta a memória usada para guardar a imagem, destruindo-a.
- *void leaveGame* – como o nome indica, esta é a função que apaga a estrutura criada, e faz *unsubscribe* às interrupções subscritas.

Módulo desenvolvido por: Bárbara Silva.

Graphics

Este módulo também é dos principais, pois é o responsável pelo display de todas as imagens, como também trata todas as alterações feitas nas coordenadas das estruturas.

Tem funções fundamentais ao jogo, como a inicialização de todas as coordenadas de todos os objetos, algoritmos de movimentação da personagem e obstáculos, verificação de colisões, desenho do cursor do rato, algoritmos de display e atualização da pontuação (*score*) e contagem decrescente.

O aparecimento de obstáculos é feito na função *void obj* de modo a que o programa escolha de forma aleatória uma de três sequências de obstáculos constituídas pelos obstáculos gift, candycane e pine, e que no final do aparecimento de uma dessas sequências apareça ou não (escolhido de forma aleatória também) um obstáculo voador (snowball).

Este módulo tem também as funções de seleção das opções de todos os menus (*int clicksMainMenu, int clicksOptionsMenu, int clicksGameOver*). Estas funções são chamada no módulo *xmasrun* quando o utilizador clica no botão esquerdo do rato e verificam as coordenadas onde tal aconteceu, se coincidem com as coordenadas de alguma opção desse menu retornam o inteiro correspondente.

Resumindo, todo o tratamento de imagem é efetuado neste módulo, e o display do ecrã completo em cada interrupção, após todas ou nenhumas modificações efetuadas pelo utilizador é feito pela função *void updateFrame*.

Achamos útil criar as seguintes estruturas com respetivos objetos:

- ***Character***: player.
  Contém toda a informação sobre a personagem a ser utilizada durante o jogo no seu estado normal, ou seja, tem as duas *frames* para fazer a animação, as coordenadas e as dimensões da imagem.

- ***DownCharacter***: dplayer.
  Contém toda a informação sobre a personagem a ser utilizada durante o jogo no seu estado agachado, ou seja, quando o jogador faz com que a sua personagem se abaixe. Assim, temos apenas uma *frame*, as coordenadas e as dimensões da imagem.

- ***Gift***: present | ***Cane***: candyCane | ***Tree***: pine | ***Snow***: ball.
  Contém as coordenadas e dimensões da imagem de cada um dos quatro obstáculos.

- ***Clouds***: clouds e clouds2.
  Contém as coordenadas e dimensões da imagem das nuvens, de modo a criar uma animação de movimento no fundo.

- ***Plaque***: plaque.
  Contém as dimensões da imagem de uma placa, que no menu *Options*, a sua posição é modificada de modo a indicar a personagem atual escolhida para o jogo.

- ***Score***: gameScore.
  Contém o valor da pontuação em inteiro, e as imagens dos números aos quais vai dar *display* no ecrã.

- ***Countdown***: countdown.
  Contém os dias, horas, minutos e segundos até ao próximo natal em inteiros, e as imagens dos números aos quais vai dar *display* no ecrã.

- ***Mouse***: mouse.
  Contém as coordenadas e dimensões da imagem do cursor do rato.

A função *int rdn* é a implementação do algoritmo Rata Die e foi baseada numa função que encontramos online (source). Este algoritmo é uma maneira de contar os dias desde 1 de Janeiro do ano 1 até uma data passada como argumento.

Módulo desenvolvido por: Bárbara Silva e Julieta Frade (50%/50%).

I8042

Importado do código das aulas práticas dos laboratórios 3/4/5, com algumas modificações.

I8254

Importado do código das aulas praticas do laboratório 2.

Keyboard

Importado do código das aulas práticas do laboratório 3, com pequenas modificações.

Mouse

Importado do código das aulas práticas do laboratório 4.

Proj

Este módulo é a base de todos os outros.

É o que inicializa todos os serviços, com chamas às funções *vg_init*, *startGame*, *newGameStates*, *defaultPlayer*, *positionMouseInit*. Após estar tudo inicializado, chama a função que executa o jogo *interruptHandler*.

Assim que o ciclo principal acabar e retornar da função anterior, trata de chamar *leaveGame* e *vg_exit*, encerrando o programa a seguir.

Módulo desenvolvido por: Bárbara Silva e Julieta Frade (50%/50%).

RTC

Este módulo lê a informação proveniente dos registos do *Real Time Clock* nomeadamente hora atual e data atual (*void getHour* e *void getDate*). Como esta informação pode estar em *BCD*, analisamos esta condição (*int isBCD*) e se se verificar convertemos para binário (*unsigned long BCDtoBinary*). O algoritmo de conversão foi encontrado online (sourse).

Módulo desenvolvido por: Julieta Frade.

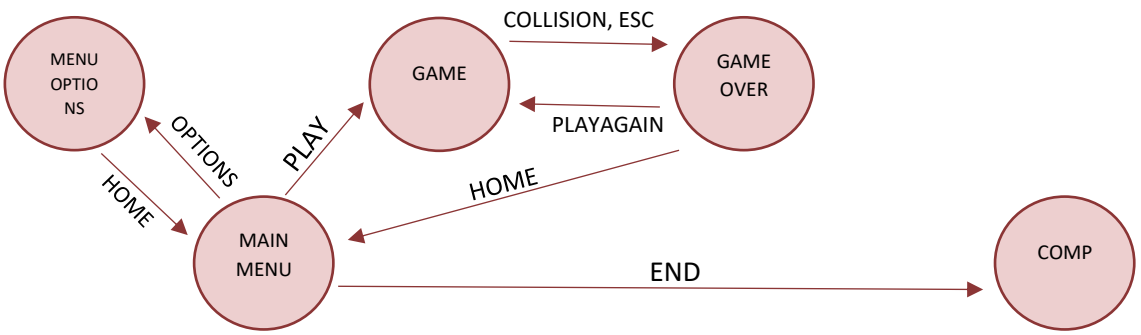
State Machine

Este é um módulo fundamental para o desenvolvimento do jogo, pois inclui duas maquinas de estado:

- ***gameState***

  Estado relativo ao jogo, isto é, em que menu de jogo se encontra.

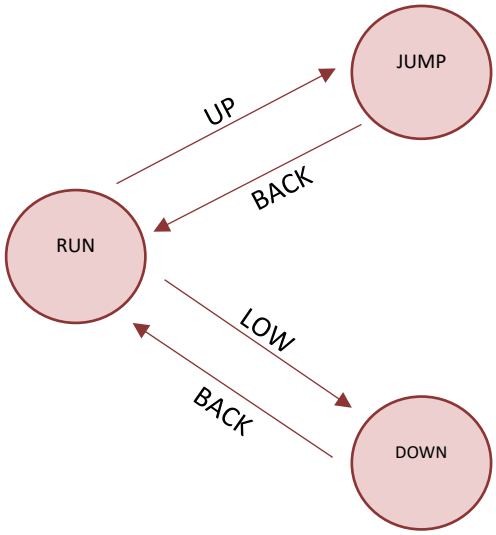
  Contém os estados: MAIN_MENU (menu inicial), GAME (jogo), GAMEOVER (menu game over, quando o jogador perde), MENU_OPTIONS (menu de mudança de personagem), COMP (fim do programa, exit).



- ***playerState***

  Estado da ação da personagem durante o jogo.

  Contém os estados: RUN (estado normal, personagem corre), JUMP (personagem salta), DOWN (personagem baixa).



Decidimos que seria mais vantajoso criar uma estrutura *States* que guarda ambos os estados, de modo a que o acesso a ambos seja mais fácil. Foi criado um objeto desta estrutura, *gameStates*, na função *newGameStates,* que é chamada em *Proj*, e inicializa o membro *gameState* a MAIN_MENU e *playerState* a RUN.

Módulo desenvolvido por: Bárbara Silva.

Timer

Importado do código das aulas práticas do laboratório 2.

VBE

Importado do código das aulas práticas do laboratório 5.

Video_GR

Importado do código das aulas práticas do laboratório 5.

XmasRun

Este módulo é o que trata a informação recebida pelos dispositivos usados, em cada interrupção.

Inclui 3 funções distintas:

- ***kbdManager***
  Analisa o *scan* do teclado.

- ***timeManager***
  Com base nos estados, tem como função chamar as funções do *graphics.c* que fazem o tratamento e display da imagem.

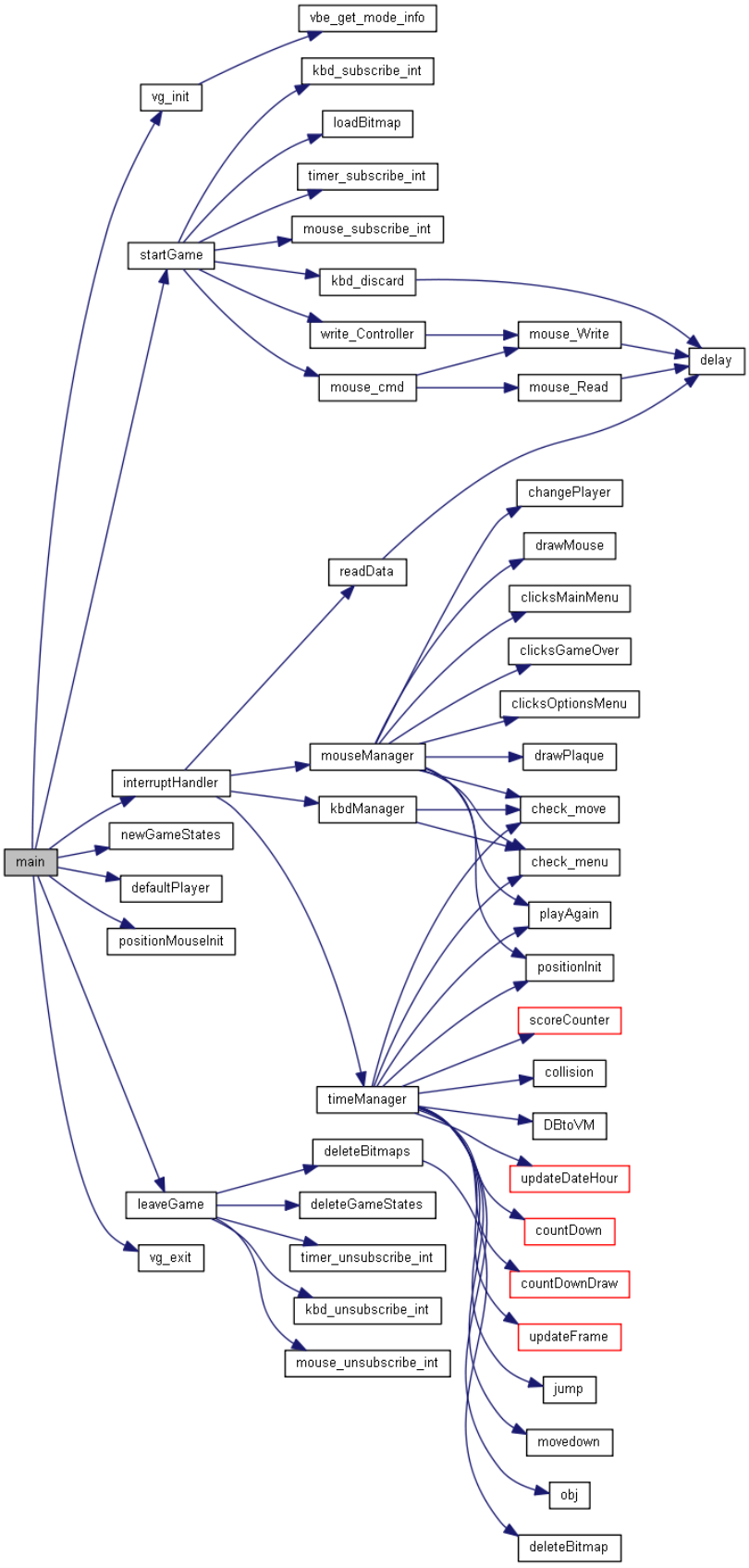
- ***mouseManager***
  Analisa os *packets* emitidos pelo rato.

A forma de como cada função atua, está dependente do *gameState* e *playerState*.

Módulo desenvolvido por: Bárbara Silva e Julieta Frade (50%/50%).

Módulo	***Peso Relativo no Projeto (%)***
Bitmap	7
Game	10
Graphics	10
I8042	4
I8254	4
Keyboard	8
Mouse	8
Proj	6
RTC	5
State Machine	8
Timer	9
VBE	5
Video_GR	6
XmasRun	10

Call Graph


vbe_get_mode_info
kbd_subscribe_int
vg_init
loadBitmap
timer_subscribe_int
mouse_subscribe_int
startGame
kbd_discard
write_Controller
mouse_Write
delay
mouse_cmd
mouse_Read
changePlayer
drawMouse
readData
clicksMainMenu
clicksGameOver
clicksOptionsMenu
mouseManager
drawPlaque
interruptHandler
kbdManager
check_move
newGameStates
check_menu
main
defaultPlayer
playAgain
positionMouseInit
positionInit
scoreCounter
collision
timeManager
DBtoVM
deleteBitmaps
updateDateHour
leaveGame
deleteGameStates
countDown
vg_exit
timer_unsubscribe_int
countDownDraw
kbd_unsubscribe_int
updateFrame
mouse_unsubscribe_int
jump
movedown
obj
deleteBitmap

Detalhes da Implementação

Achamos relevante mencionar a forma como implementamos duas máquinas de estado (ver página 11), que nos permitiu simplificar o código e contruir um jogo bastante fluído e consistente, assim como a criação de estruturas que vieram facilitar a partilha de informação e acesso entre módulos.

Como recomendado, decidimos utilizar código por camadas de modo a facilitar a utilização e escrita de funções mais complexas, assim como por questões de organização e compreensão.

Achamos também relevante detalhar o algoritmo que usamos para a implementação da contagem decrescente no menu principal. Em primeiro lugar há que mencionar que é uma contagem decrescente até ao próximo dia de Natal, não tendo um dia fixo, ou seja, quando o próximo Natal chegar, a contagem vai atualizar para começar a contar para o próximo dia de Natal depois desse. Para contar dias foi utilizado o algoritmo Rata Die que conta os dias desde o dia 1 de Janeiro do ano 1 até à data passada como argumento. Para contar os dias desde a data atual até ao próximo Natal foi feita uma subtração usando o algoritmo mencionado (*int daysLeft*). Foi depois também calculado os segundos até ao próximo minuto (*int secondsLeft*), os minutos até à próxima hora (*int minutesLeft*) e as horas até ao próximo dia (*int hoursLeft*). Na função *void countdown* é onde obtemos os números finais para a contagem. Nesta função analisamos se o número de segundos é maior que zero, deve ser retirada uma unidade aos minutos, se os segundos ou os minutos são maiores que zero, é retirada uma unidade às horas e se os segundos ou os minutos ou as horas são maiores que zero, é retirada uma unidade aos dias, isto porque é uma contagem decrescente, por exemplo, se queremos uma contagem decrescente até às 24h00 e são 23h55, faltam 5 minutos e não 1 hora e 5 minutos. Esta implementação foi toda feita no módulo *graphics*.

Conclusão

Ambas sentimos que o aproveitamento tanto das aulas teóricas como práticas foi muito positivo, embora as práticas tenham sido mais benéficas, pelo que sem os laboratórios não teríamos tido tanto à vontade para a elaboração deste projeto. No entanto, achamos que ambas as aulas deviam estar mais a par uma da outra, visto que, por vezes, enquanto nas aulas práticas nos estávamos a focar num periférico, nas teóricas já íamos bem mais avançados, o que notamos ser um pouco prejudicial para o acompanhamento das aulas.

Outra melhoria que gostaríamos de propor, era a atualização dos slides e dos guiões, pois ao longo desde semestre, deparamo-nos com alguns erros nos mesmos.

Sentimos uma maior dificuldade na organização do código, devido à junção de todos os periféricos, e na gestão de memória em relação às imagens. As colisões também foram desafiantes de implementar.

Gostaríamos também de mencionar que, achamos muito útil consultar o blog do Henrique Ferrolho (source), que ajudou na elaboração deste projeto.

Em suma, gostaríamos de frisar que ambos os elementos do grupo se empenharam de igual forma e dedicaram muitas horas para complementar todos os objetivos, tendo sido todo o trabalho igualmente dividido.

Apêndice

Instruções de Instalação

Devido à utilização de imagens Bitmap e de forma a evitar a utilização de caminhos absolutos, implementamos uma forma diferente de compilar o programa.

Caso seja a primeira compilação deverá ser feito o seguinte:

- dentro da pasta *proj*, entrar na **pasta *scripts***
- correr "**sh install.sh**"
- correr "**sh compile.sh**"
- correr "**sh run.sh**"

O primeiro script serve para instalar o ficheiro de configuração do sistema necessário para a execução do programa e para copiar todas as imagens da pasta *res* para uma outra no diretório: ***/home/proj/res***, caso esta seja a mesma, não haverá problema. Assim, é possível compilar o programa em qualquer MINIX, tendo sempre acesso às imagens.

Caso o primeiro script já tenha sido feito pelo menos uma vez, sempre que quiser correr o programa basta correr os dois últimos.