Relatório do Projeto de LCOM
2016/2017

Bow And Arrow

Porto, 2 de Janeiro de 2017

Turma 6, Grupo 16
Hugo Diogo Queirós Cunha - up201404587
António Miguel Silva Pereira - up201307910

Índice

Introdução **2**

1. Instruções do jogo **3**
- Menu Inicial 3
- Dentro do jogo 4

2.Estado do Projeto **11**
- Timer 11
- Teclado 12
- Rato 12
- Placa de Vídeo 12
- RTC 12

3.Organização do código/estrutura **13**
- Arrow 13
- Bitmap 13
- Bow and Arrow 13
- Balloon 14
- Hero 14
- Menu 14
- Main 15
- Teclado 15
- Rato 15
- RTC 16
- Timer 16
- Utils 16
- VBE 16
- Video_GR 17

Call Graph **18**

4.Detalhes da implementação **19**
- Gráficos 19
- Estados 19
- Colisões 19

5.Conclusões **20**

6.Autoavaliação **20**

7.Instruções para correr o jogo **20**

Introdução

Neste projeto, temos como objetivo implementar um jogo no sistema operativo Minix com a utilização de periféricos, previamente implementados nas aulas práticas.

No nosso caso, resolvemos fazer uma versão adaptada do jogo *Bow and Arrow,* que é um *shooter* em 2D que tem como objetivo destruir, com setas, o maior número de balões.

De seguida, iremos explicar mais aprofundadamente o funcionamento do mesmo, assim como alguns aspetos relevantes da implementação do jogo.

1. Instruções do jogo

Menu Inicial

Imagem 1 : Menu Inicial

Quando iniciámos o nosso programa, esta é a primeira imagem que nos é apresentada, representando o menu inicial. A seta ao lado do botão *Play* indica que a funcionalidade, neste caso *Play*, está indicada.

Neste momento, podemos mover a seleção para cima e para baixo de acordo com o que desejamos escolher. Caso escolhamos *Play*, de imediato, iniciamos o jogo, em *Exit* saímos do programa.

Dentro do jogo

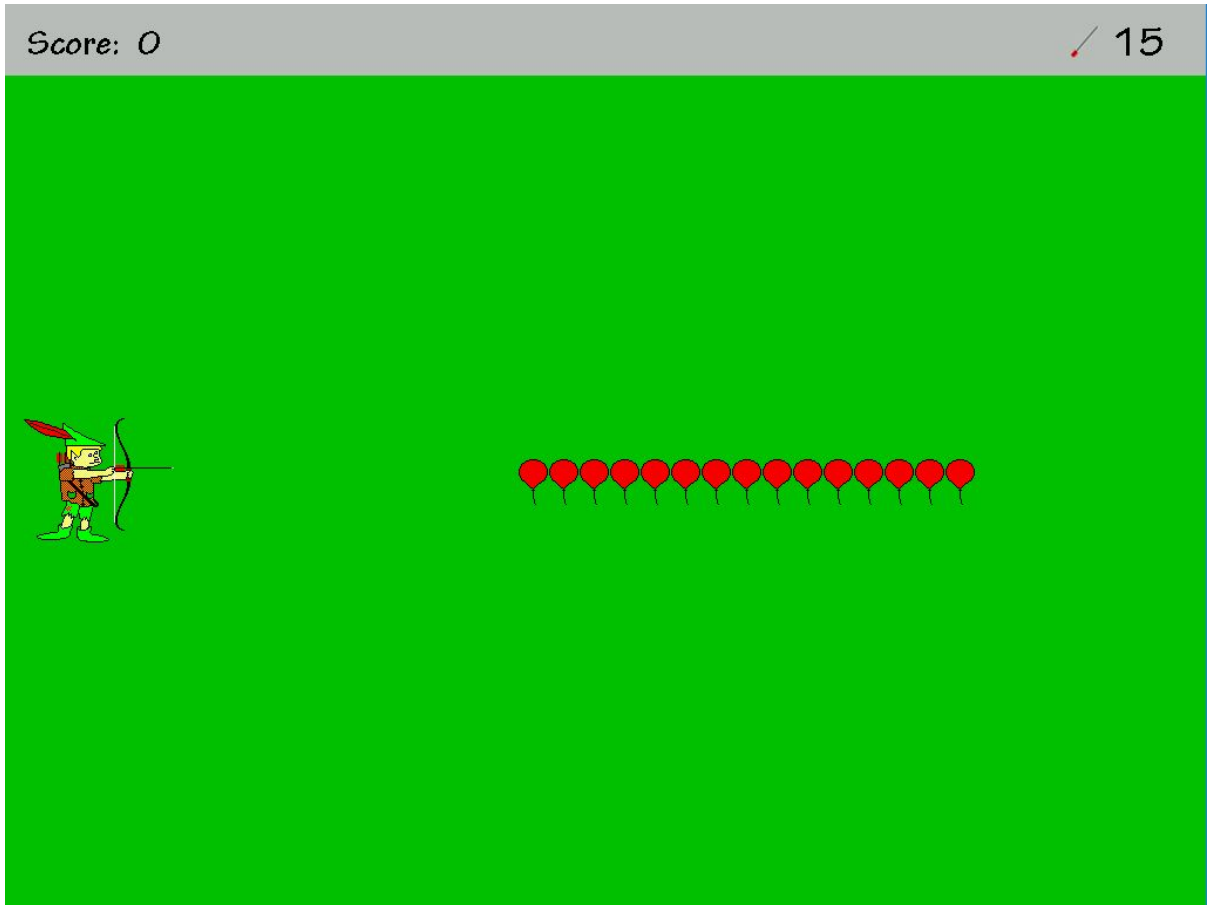


Imagem 2 : Começo do jogo após entrada.

Após a entrada no jogo, é-nos apresentada esta imagem, onde consta o nosso herói, do lado esquerdo, no centro e, em movimento ascendente constante, uma série de balões. Temos também na parte superior do jogo uma barra de informação com o *score* do jogador e também o número de flechas restantes. O objetivo, como referido atrás, será destruir o maior número de balões possíveis.

Nesta altura, o movimento do rato é transmitido ao herói, fazendo com que ele se mova para cima e para baixo, mantendo sempre a mesma posição em x.

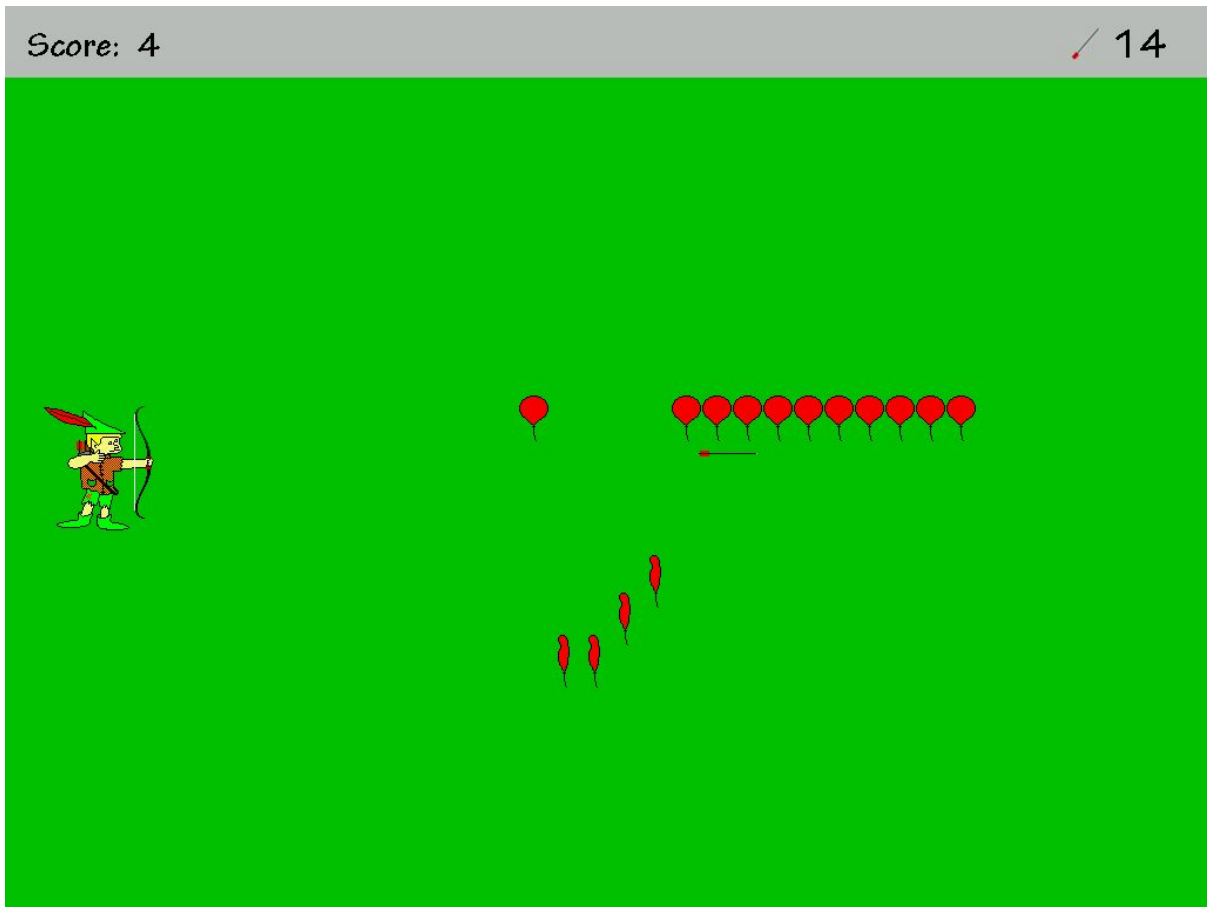


Imagem 3 : Gameplay do primeiro nível do jogo.

Para lançar uma seta devemos manter premida a barra de espaços, que apenas permite lançar a seta caso deixemos de a premir. Enquanto a barra de espaços estiver premida é possível movimentar o herói.

Quando uma seta é largada é necessário adquirir uma nova. Isso é possível carregando no botão do lado direito do rato.

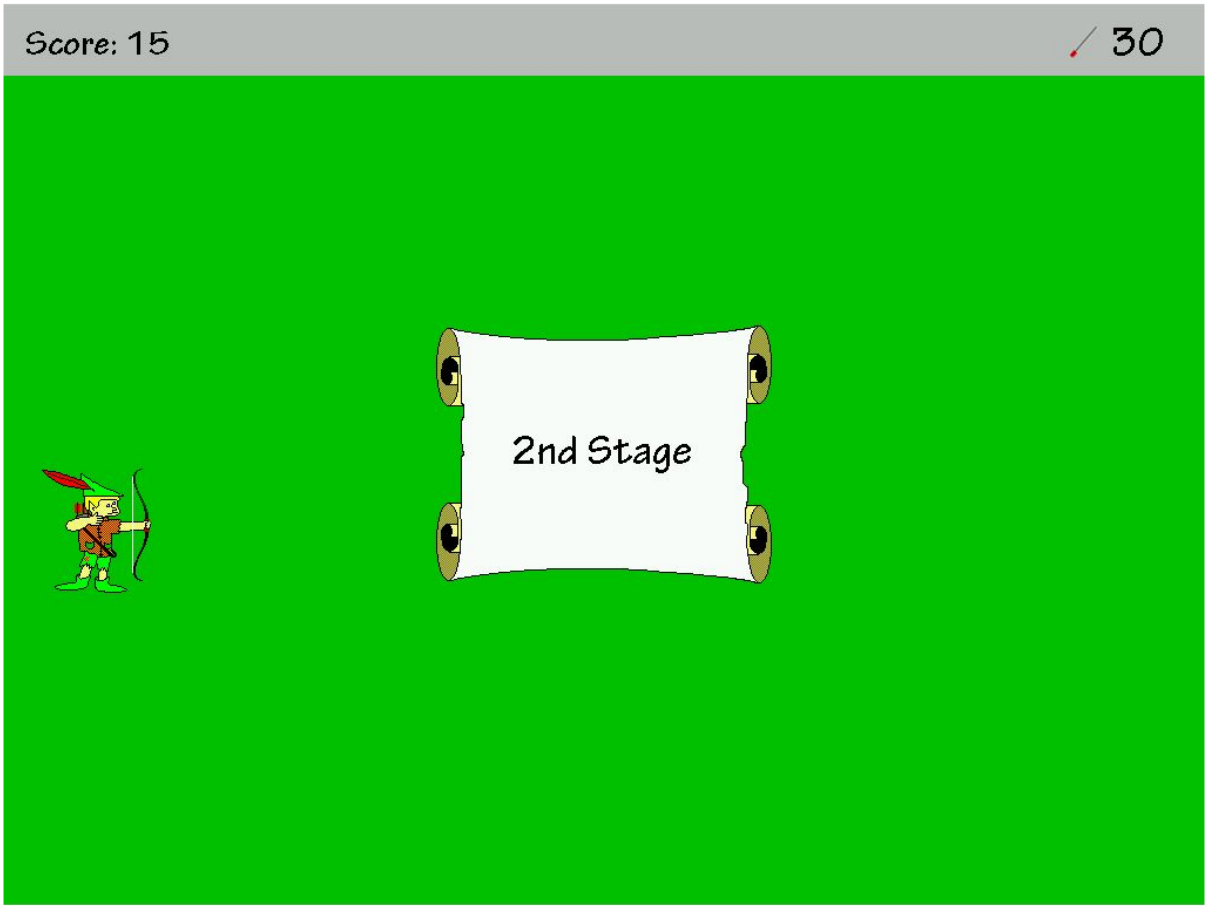


Imagem 4 : Imagem de transição do nível 1 para o nível 2.

Neste podemos visualizar a imagem de transição do primeiro nível para o segundo nível, com um pergaminho a dizer “2nd Stage”. Passados alguns segundos passaremos para o segundo nível.

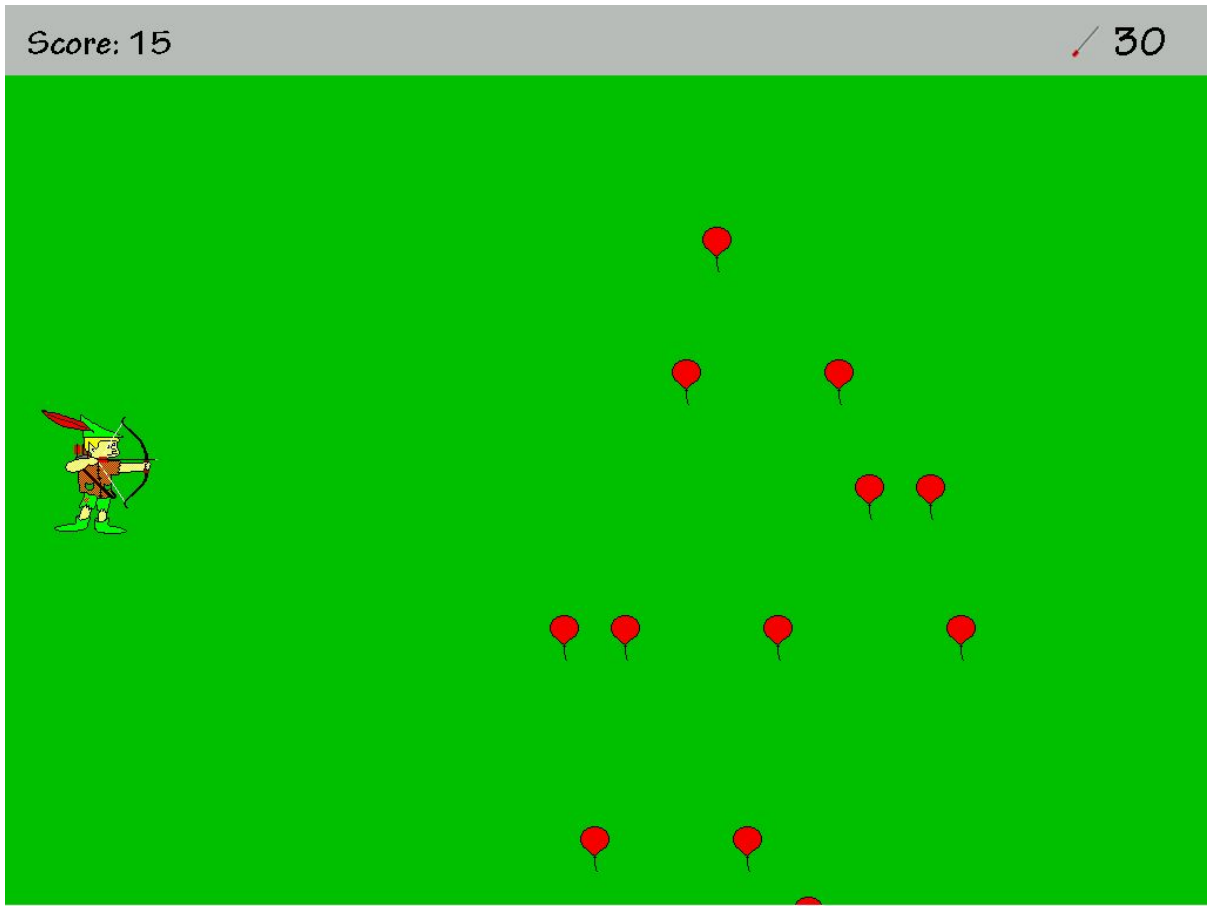


Imagem 5 : Início do nível 2.

Aqui deparamos-nos com o segundo nível do jogo, onde é possível observar que os começam a aparecer no ecrã dependendo do número dos segundos lidos.

Imagem 6 : Ecrã do jogador quando este perde.

Quando perdemos (o número de flechas que o herói tem é igual a 0) e ainda existem balões a voar. Deparamos-nos com este ecrã que passados alguns segundos volta para o main menu.

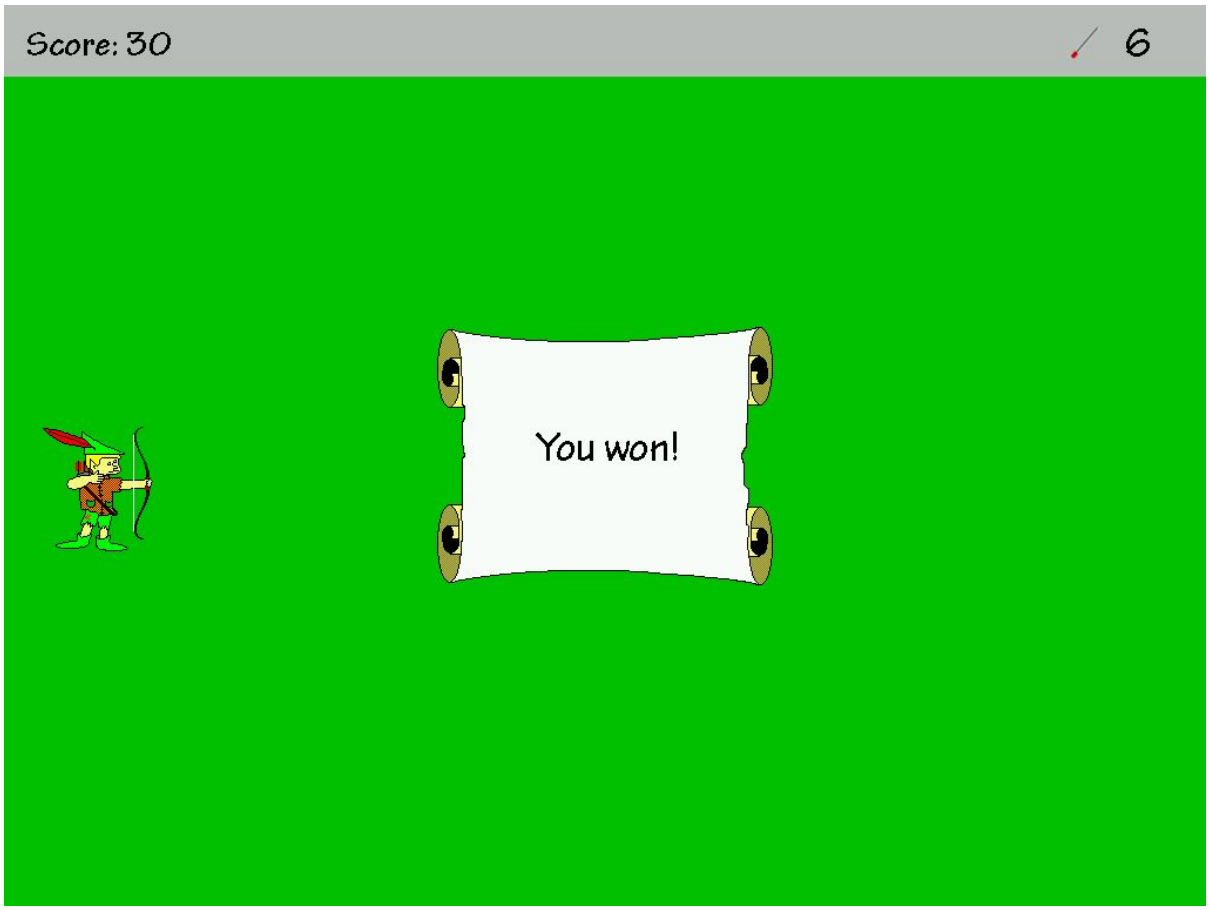


Imagem 7 : Ecrã do jogador quando este ganha.

Chegamos a este ecrã (o de vitória) quando passamos o primeiro e o segundo nível do jogo. Passados alguns segundos passamos para o main menu.

Imagem 8 : Ecrã de pausa.

A qualquer momento do jogo podemos carregar na tecla ESCAPE e este menu é apresentado, que representa o menu de pausa. Neste, é possível voltar ao jogo, que se encontra no momento em que o deixamos, ou voltar ao menu principal, desistindo daquela partida.

2.Estado do Projeto

Excluindo a parte de mostrar *highscores,* a qual não implementamos, todas as funcionalidades foram concluídas com sucesso.

Dispositivo	Utilização	Interrupções
Timer	Atualizar o ecrã, animação do herói e controlar uma série de eventos	Sim
Teclado	Mover seleção no menu e lançamento de setas no jogo	Sim
Rato	Controlo do movimento do herói e carregar setas	Sim
Placa vídeo	Desenhar todos os *bitmaps* do jogo	Não
RTC	Gerar números aleatórios para controlar balões	Não

Timer

As funções do timer são vastas, no entanto, no nosso projeto pode-se destacar o seu uso na medida em que o timer é responsável pela atualização do ecrã a cada 1/60 (60 fps) segundos e também é responsável na transição da posição de descanso do herói para a posição de *powered-up*. Existe, ainda, mais uma série de variáveis de controlo de importância reduzida que são atualizadas com o timer. Os ficheiros relativos ao *timer* são o *timer.h* e *timer.c.* Estas funções são utilizadas no ficheiro: *bow_and_arrow*.c.

Teclado

O teclado é, como o *timer,* fundamental para o nosso projeto pois permite controlar a seleção de opções no menu, usando as setas. Dentro do jogo é essencial para o utilizador poder largar uma seta, utilizando a barra de espaços. As funções do teclado encontram-se declaradas no ficheiro kbd.h e implementadas no kbd.c, sendo utilizadas no ficheiro *bow_and_arrow*.c.

Rato

O rato permite mover o nosso herói dentro do ecrã de jogo com a sua movimentação, mas também recarregar uma seta ao carregar no botão do lado direito.

As funções estão declaradas no ficheiro *mouse*.h e implementadas no *mouse*.c, sendo utilizadas no *bow_and_arrow*.c e no *hero*.c.

Placa de Vídeo

Provavelmente um dos mais importante do nosso projeto porque vai permitir a apresentação de tudo que esteja envolvido a gráficos (bitmaps), sendo todos estes mapeadas nesta.

Dado que a simples utilização de um único *buffer* gráfico fazia, muitas vezes, piscar o ecrã, implementamos, também, técnica de *double-buffering* que consiste na utilização de dois *buffers* e apenas a cópia de memória (com a função *memcpy)* do segundo *buffer* para a *video_mem.*

A placa de vídeo, em modo de vídeo é também utilizada para a deteção de colisões por comparação de posições de *bitmaps.*

As funções da placa de vídeo estão declaradas nos ficheiros vbe.h, video_gr.h e Bitmap.h e estão implementadas nos ficheiros vbe.c,video_gr.c e Bitmap.c.

RTC

No nosso projeto. o RTC tem como objetivo gerar números aleatórios (leitura dos segundos) mediante a altura do dia que vai influenciar a altura em que os balões começaram a ser lançados.

As funções referentes ao mesmo estão declaradas no ficheiro rtc.h e implementadas em rtc.c.

3.Organização do código/estrutura

Arrow

Este módulo trata, em modo exclusivo, as funções relacionadas com as setas e implementa-as. O grupo resolveu representar uma seta como uma struct na qual inclui atributos importantes: a posição (x e y), uma variável de controlo *inside_screen* que nos indica se uma seta está dentro dos limites do ecrã, ou não e, ainda, o *bitmap* associado a essa seta.

- Contribuição: 50/50
- Importância para o projeto: 5%

Bitmap

No módulo *Bitmap* estão as funções necessárias ao *load* dos *bitmaps* para o programa, que mais tarde serão desenhados. É de realçar a utilização de código alheio, neste caso com autoria de Henrique Ferrolho e previamente autorizado pelo professor.

- Importância para o projeto: 20%

Bow and Arrow

Provavelmente o módulo mais importante pois é neste onde são tratadas as iterações do jogo e a sua lógica.

De facto, através de interrupções são atualizadas variáveis dos controladores respetivos (*keyboard/timer/mouse*) através do ciclo verificador de interrupções.

Neste são também verificadas colisões (a cada iteração), atualização de posições de balões, do herói e das setas, preparando o ecrã para o seguinte *frame*.

- Contribuição 50/50
- Importância para o projeto: 30%

Balloon

No *balloon.c* são implementadas funções relacionadas com movimento, estado e *bitmap* associado a cada balão. É de realçar que cada balão funciona como um *mini* máquina de estados, através da variável *b_state*, a qual guarda o estado em que se encontra o balão: *BLOWN, DESTROYED* ou *INTACT.* Esta informação é importante para as colisões, evitando iterações desnecessárias, caso um balão esteja, por exemplo, destruído.

- Contribuição 90/10 (Hugo Cunha)
- Importância para o projeto: 5%

Hero

Neste módulo estão guardadas todas as funções referentes ao herói, incluindo funções que permitem movê-lo, controlar número de setas, o seu *score*, o seu estado (WITHOUT_ANY_ARROW, REST_NO_ARROW,REST_WITH_ARROW, POWERING_UP,FULL_POWER), a sua posição e o *bitmap* associado.

- Contribuição 90/10 (António Pereira)
- Importância para o projeto: 5%

Menu

Este módulo é responsável pelo menu inicial do programa. Neste módulo, utilizamos o teclado como auxiliar que permite ao utilizador mover dentro das opções dadas. As opções neste são apenas dois diferentes *bitmaps* alternando entre si.

- Contribuição 50/50
- Importância para o projeto: 5%

Main

Este é possivelmente o módulo mais simples do projeto, no qual é apenas feita a chamada *vg_init* que permite a entrada no modo de vídeo e um ciclo de controlo entre jogo e menus.

- Contribuição 50/50
- Importância para o projeto: 1%

Teclado

O teclado é utilizado quase ao longo de todo o programa. No menu responsável pela alteração da seleção e dentro do jogo permite lançar setas. Principalmente, estão implementadas funcionalidades que permitem a leitura do *scan code* de teclas por meio de interrupções.

- Contribuição 50/50
- Importância para o projeto: 5%

Rato

O rato, desenvolvido em aulas laboratoriais, é utilizado dentro do jogo para controlar o movimento do herói e carregamento de novas setas.

O rato, tal como o teclado, funciona por intermédio de interrupções.

Durante o desenvolvimento do projeto, adaptamos a implementação do rato, estando agora, associada uma *struct* na qual temos acesso aos *packets* lidos do controlador, tentando aproximar de uma programação orientada a objetos (encapsulamento).

- Contribuição 50/50
- Importância para o projeto: 10 %

RTC

O *RTC* é importante no nosso projeto pois permite a geração de posições aleatórias para o segundo nível do jogo de acordo com a leitura dos segundos atuais. As funções associadas ao mesmo permitem obter informações sobre a hora, dia, mês reais através do periférico *RTC*. No entanto, apenas utilizamos, no projeto, leitura dos segundos.

- Contribuição 50/50
- Importância para o projeto: 2%

Timer

O timer, também desenvolvido em exclusivo em aulas laboratoriais, possui uma implementação simples mas é sem dúvida fulcral para o nosso projeto pois é através das interrupções do mesmo que conseguimos atualizar o ecrã de jogo.

- Contribuição 50/50
- Importância para o projeto: 5%

Utils

Este módulo possui algumas variáveis importantes transversais a um número considerável de módulos, tentando assim evitar a repetição de código.

- Contribuição 60/40 (António Pereira)
- Importância para o projeto: 3%

VBE

Contém funções importantes para a entrada/saída de modo de vídeo. Desenvolvido em aulas laboratoriais.

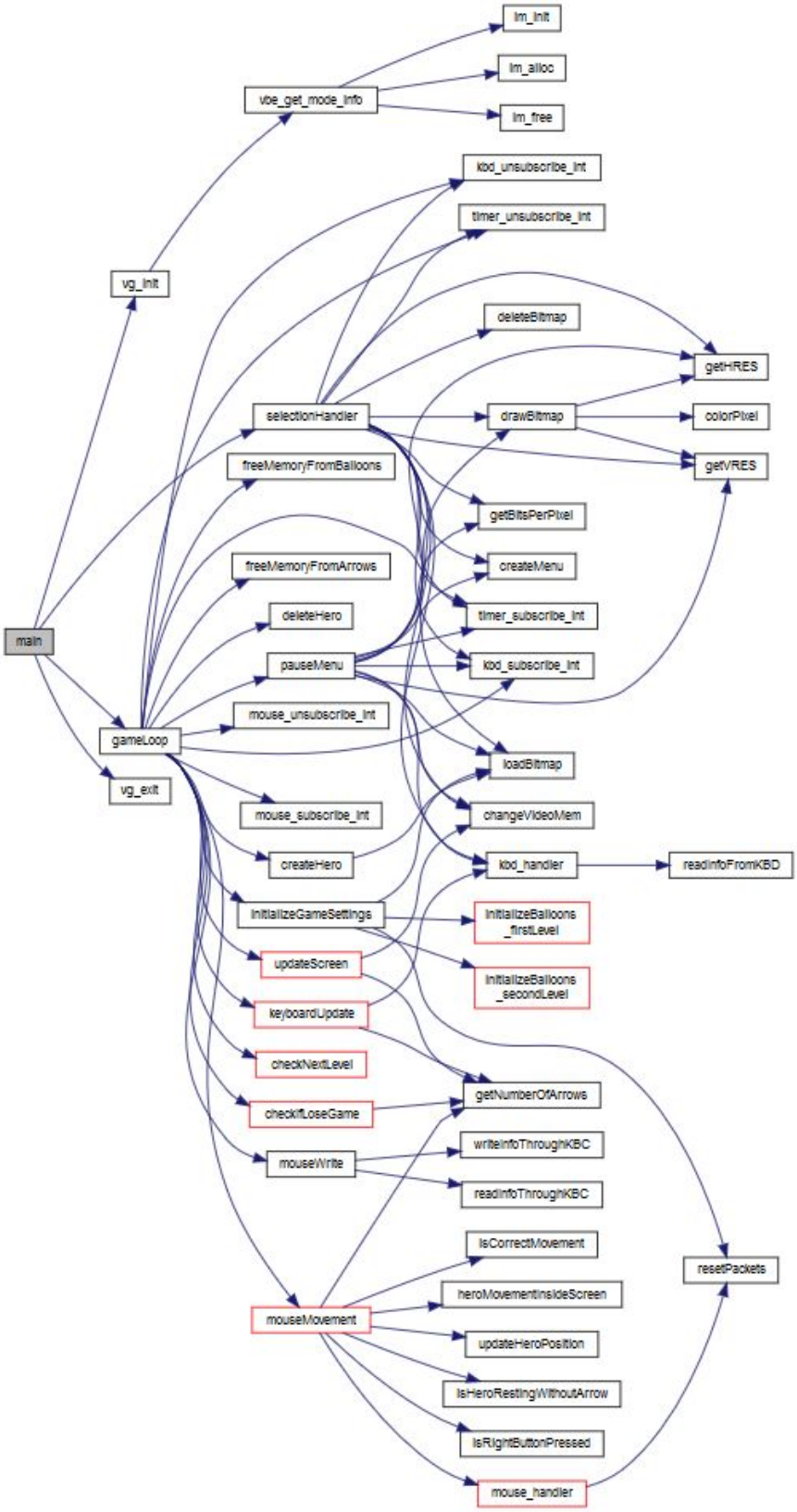
- Contribuição 50/50
- Importância para o projeto: 1%

Video_GR

Assim como o módulo VBE, este módulo foi desenvolvido em aulas laboratoriais e tem como objetivo a entrada em modo gráfico/vídeo. Além disso, resolvemos juntar a este módulo outra funções relevantes e que concerne a parte gráfica do jogo, por exemplo, pintar pixel ou desenhar bitmaps.

- Contribuição 50/50
- Importância para o projeto:3%

Call Graph


lm_init
lm_alloc
lm_free
vbe_get_mode_info
kbd_unsubscribe_int
timer_unsubscribe_int
vg_init
deleteBitmap
getHRES
selectionHandler
drawBitmap
colorPixel
freeMemoryFromBalloons
getVRES
getBitsPerPixel
freeMemoryFromArrows
createMenu
deleteHero
timer_subscribe_int
main
pauseMenu
kbd_subscribe_int
mouse_unsubscribe_int
gameLoop
loadBitmap
vg_exit
changeVideoMem
mouse_subscribe_int
kbd_handler
readInfoFromKBD
createHero
initializeGameSettings
initializeBalloons
_firstLevel
updateScreen
initializeBalloons
_secondLevel
keyboardUpdate
checkNextLevel
getNumberOfArrows
checkIfLoseGame
writeInfoThroughKBC
mouseWrite
readInfoThroughKBC
isCorrectMovement
resetPackets
heroMovementInsideScreen
mouseMovement
updateHeroPosition
isHeroRestingWithoutArrow
isRightButtonPressed
mouse_handler

4.Detalhes da implementação

Gráficos

Gostaríamos de destacar a utilização da técnica de *double-buffering* que faz com que as diferentes *frames* sejam apresentadas de forma suave, sem cansar o utilizador. Isto permite remover o "tilintar" das imagens causado pelo mapeamento da memória de vídeo.

Estados

Implementamos algumas máquinas de estados para ser mais fácil perceber o estado do jogo. Os módulos os quais receberam máquinas de estados foram: balloon, hero, arrow, bow_and_arrow.

Nos balões foi necessário para perceber se eles estão intactos, a cair (por terem sido abatidos) ou destruídos (já não são pintados no ecrã).

No herói para controlar os *bitmaps* associados a estar parado sem flecha, parado com flecha, a "carregar força" ou com seta carregada, e ainda também o bitmap associado de o herói com nenhuma flecha em sua posse.

No caso da classe bow_and_arrow serve para distinguirmos as diversas fases de jogo, que implica a organização do código para diferentes fases de jogo.

Colisões

No caso de colisões implementamos, também, um sistema simples de colisões entre setas e balões que consiste em verificar, a cada iteração, se a ponta direita da seta está "dentro" de um balão (parte de cima,vermelha). Acrescentamos uma certa tolerância de pixels para tornar um pouco mais realista a colisão.

5.Conclusões

Primeiramente, gostaríamos de destacar alguns pontos positivos da unidade curricular. Achamos bastante positivo o tema ser livre, pois permite ter outra *approach* a algo que, em princípio, terá mais interesse pois foi escolhido pelo grupo. Em segundo, gostaríamos de destacar, positivamente, o interesse do conteúdo lecionado ao longo do ano, pois a interação com os diversos periféricos foi algo que, pelo menos para o grupo, trouxe muito conhecimento. Por último, dado que um dos membros já está a frequentar esta unidade pela segunda vez, achamos que aulas teóricas de apenas uma hora são mais vantajosas (em relação ao ano passado que eram de duas) pois permite estar mais atento e guardar mais facilmente a informação dada em cada aula.

Agora mais negativo, pensamos que a informação dos “labs” estava muito dispersa (slides de aulas teóricas, guiões, algumas páginas externas) e pouco objetiva o que, por vezes, levava a alguma perda de tempo e, também, a alguma frustração. Em segundo, como para a unidade curricular é necessária utilização da linguagem C, achamos um pouco complicada a integração dado que C difere bastante de C++ e isso pode tornar-se difícil. Por último, o grupo pensa que a extensa procura às contribuições de cada elemento para uma determinada funcionalidade ou módulo é demasiada elevada.

6.Autoavaliação

Hugo Cunha:

- Contribuição 50%

António Pereira:

- Contribuição: 50%

7.Instruções para correr o jogo

Necessário colocar o ficheiro **proj**, disponível na pasta do projeto, dentro da pasta **conf**, na pasta **etc/system.conf.d.** Quando estiver dentro do diretório ***src*** basta correr o comando **make** e depois é apenas necessário correr com o comando: **service run `pwd`/proj.**