Relatório de Lcom

Projeto Pro Debugger

Ana Inês Oliveira Barros
up201806593

João Alexandre Lobo Cardoso
up201806531

02/01/2020

ÍNDICE

1 Resumo

O nosso projeto resume-se a um jogo ”top-down”(vista de baixo para cima) com o objetivo de sobreviver o mais tempo possível. Para dar contexto ao projeto, acabamos por optar por nos inspirarmos no estilo do *Windows XP*, sendo que o jogo corre no ambiente de trabalho do *Windows XP*.

Em primeiro lugar, o utilizador tem que fazer *login* para aceder ao ambiente de trabalho e, consequentemente, começar o jogo. Durante o jogo, o objetivo do jogador é proteger o ícon da *Virtual Box*, que se encontra no centro do ecrã, dos insetos (”bugs”) que aparecem de fora do ecrã e que se dirigem para o mesmo. Se um inseto tocar no ´ıcon, o jogador perde o jogo. Para impedir que tal aconteça, o jogador pode matar os ”bugs”, isto é, o jogador possui a capacidade de mandar um míssil de cima para um sítio à escolha e, dessa forma, fazer com que a explosão provocada pelo míssil atinja um inseto.

Também é oferecido ao jogador a possibilidade de utilizar ajudas, mas estas são limitadas, e existe um menu (menu iniciar do *Windows XP*) que o utilizador pode usar para pausar o jogo (voltar à página de ”login”) ou, para sair do programa.

Depois de se perder o jogo, uma caixa de erro do estilo do *Windows XP* aparece e oferece ao jogador a opção de sair ou de reniciar.

2 Instruções de utilização do programa

2.1 Correr o programa

Podemos correr o programa na *Virtual Box* com o comando: lcom run proj. Não são necessários argumentos.

2.2 Ecrã Inicial

Figure 1: Ecrã inicial

Ao iniciar o programa, o utilizador é deparado com este ecrã. O objetivo deste ecrã é representar a página de *login* do *Windows XP* e, ao mesmo tempo, servir de separação entre o início do programa e o ínicio do jogo.

Existem 3 funcinalidades neste ecrã:

1. **Cursor do rato**
   Representa a posição e os cliques do rato.

2. **A caixa de texto (retângulo branco);**
   Quando o utilizador pressiona teclas no teclado (exceto ”ESC” e ”ENTER”) pode utilizar esta caixa para escrever uma palavra-passe. Isto apenas é opcional, ou seja, o utilizador não precisa de introduzir nenhuma palavra-passe para avançar para o jogo. Ao escrever aparece no ecrã várias pintas pretas, tais como as que aparecem normalmente quando se introduz uma palavra-passe. O número de pintas é limitado ao tamanho da caixa mas, a quantidade de carateres continua a acumular (apenas não são desenhadas as pintas). Pressionar o ”BACKSPACE” funciona para apagar carateres/pintas e, o ”ENTER” permite efetuar o *login* / passar ao ecrã seguinte.

Figure 2: Palavra-Passe inserida

3. **Botão verde com Seta branca**
   Se o utilizador colocar o cursor sobre este botão e pressionar o botão esquerdo do rato, o programa avança para o próximo ecrã (simulação de efetuar *login* no *Windows XP*). Pressionar a tecla ”ENTER” permite o mesmo efeito.

Figure 3: Botão de Login

4. **Botão vermelho na zona inferior esquerda do ecrã**
   Se o utilizador colocar o cursor sobre este botão e pressionar o botão esquerdo do rato, o programa termina (simulação de desligar o computador com o *Windows XP*). Pressionar a tecla ”ESC” tem o mesmo efeito.

Figure 4: Botão de Desligar

2.3 Ecrã de jogo

Figure 5: Ecrã do jogo

Este é o ecrã que aparece depois de fazer ”login” e significa o ínicio do jogo.
Em termos de objetos estáticos temos:

- Fundo t´ıpico do *Windows XP*;
- Ícon da *Virtual Box* no centro do ecrã;
- Barra de tarefas na parte inferior do ecrã;

De resto:

- Cursor do rato;
- Botão para abrir o menu inciar;
- Duas barras (retângulo preto) que vão sendo preenchidas de verde durante o jogo;
- Barra de tarefas na parte inferior do ecrã;
- Horas e minutos no canto inferior direito;
- Número de moedas do jogador com um *Sprite* animado de uma moeda a rodar do lado direito;
- Insetos que vão aparecendo de fora do ecrã, de várias direções, e que se dirigem para o ´ıcon da *Virtual Box* ;

2.4 Funcionamento do jogo

2.4.1 Objetivo

Aguentar o mais tempo possível sem perder.

2.4.2 Perder

Perde-se o jogo quando um inseto toca no ´ıcon da *Virtual Box*. Quando se perde, aparece uma janela de erro como a seguinte:

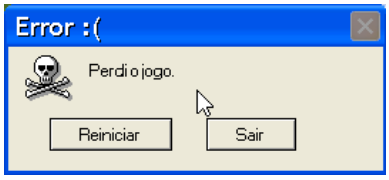

Figure 6: Janela de erro

Esta janela dá a opção ao jogador de reniciar o jogo ou de fechar o programa. Para selecionar a opção basta o jogador colocar o cursor por cima do botão pretendido e pressionar o botão esquerdo do rato. A opção reniciar apenas renicia o jogo e não programa.

2.4.3 Defesa

Para o jogador defender o ´ıcon da *Virtual Box* dos inimigos pode utilizar mísseis. A posição onde vai ser largado o míssil é onde estiver o cursor quando o jogador pressionar o botão esquerdo do rato, desde que não esteja a pressionar o botão do menu iniciar do *Windows XP*.
Quando o jogador escolhe disparar começa a animação do míssil. Primeiro, aparece uma marca em forma de 'X' que vai aumentando (para dar a ideia de que o míssil está a cair de cima), depois, aparece o míssil a cair e a explodir. Se um inseto for atingido pela explosão, este é destruído/desaparece.

Figure 7: Marca do m´ıssil

Figure 8: Explosão do míssil

2.4.4 Ajudas

Decidimos incluir no jogo algumas funcionalidades que permitem ao jogador defender a *Virtual Box* por mais tempo. No entanto, estas ajudas têm as suas limitações.
Mais especificamente, o jogador só consegue usar a ajuda quando possuir moedas suficientes para a comprar. Podemos ver quantas moedas temos na parte esquerda da barra de tarefas junto ao sprite animado da moeda.

Figure 9: Número de Moedas

Adicionalmente, para saber se possuímos moedas suficientes, existe uma barra associada a cada habilidade que se vai enchendo à medida que o jogador

acumula moedas e, quando esta barra ficar cheia, significa que a habilidade está disponível para ser usada.

Figure 10: Barras de habilidade

Para comprar a habilidade, o jogador deve pressionar a tecla respectiva, ou seja, a tecla que se encontra ao lado da barra. Depois de efetuada a compra, o número de moedas diminui e a barra ajusta-se a esse novo número. Existem 2 ajudas:

- *"Nuke"*
  Tecla de ativação: Z
  Número de moedas necessário para compra: 40
  Efeito: Destruir todos os inimigos de momento no ecrã.

Figure 11: Nuke

- *Antiv´ırus*
  Tecla de ativação: X
  Número de moedas necessário para compra: 20
  Efeito: Diminuir a velocidade dos inimigos por 33% e o intervalo de tempo entre o surgir de inimigos por 15%.

Figure 12: Antiv´ırus

2.4.5 Pausar e Sair do jogo

Para sair do jogo existem 2 maneiras distintas. Premir a tecla "ESC" em qualquer altura e em qualquer ecrã termina espontaneamente o programa. A outra maneira requer que o jogador aceda ao `Menu Iniciar` do *Windows XP*.

Menu

Para abrir ou fechar o menu, o jogador deve pressionar o botão verde no canto inferior esquerdo do ecrã. Depois de aberto, o botão fica diferente, ou seja, passa ser desenhado de maneira a parecer pressionado.

Figure 13: Menu Iniciar

Dentro do menu, podemos interagir com dois botões:

- *Botão vermelho*
  A função deste botão é terminar o programa, tal como se desligássemos o computador no *Windows XP*.

Figure 14: Botão de Desligar

- *Botão amarelo*
  A função deste botão é pausar o jogo, tal como se terminássemos a sessão (*logout*) no *Windows XP*. Por outras palavras, quando este botão é pressionado, o ecrã inicial (ecrã de *login*) volta a ser apresentado. Quando fizermos *login* de novo, o jogo é retomado exatamente no ponto antes da pausa.

Figure 15: Botão de Log Off

3 Estado do Projeto

3.1 Funcionalidades da secção anterior não implementadas

Todas as funcionalidades referidas na secção anterior foram implementadas por nós com sucesso.

3.2 Dispositivos de Entrada e Saída

Dispositivo	Função	Interrupções
Timer	Controlo de ”frame rate”	Yes
KBD	Selecionar habilidade, escrever, terminar programa	Yes
Mouse	Disparar, premir botões	Yes
VBE	Mostrar ecrãs, tempos, sprites e menus.	Yes
RTC	Ler tempo e usar alarmes	Yes

Table 1: Dispositivos E/S usados

3.2.1 VBE

1. **Modo de Vídeo**
   O nosso projeto utiliza o ”Real color mode” (0x115 Resolução 800x600 (4:3)) de 24 *bits* por cor, 8 para vermelho, 8 para azul e 8 para verde, assim havendo 256 * 256 * 256 cores disponíveis. Uma destas cores é utilizada como cor de transparência: 0xFF0080. Ou seja, esta cor não é desenhada pelas funções responsáveis pelo desenho de *Sprites*.

2. **”Double Buffering”**
   Utilizamos double buffering atravéz da função memcpy() do C.

3. **”Fonts”**
   Existem duas ”fonts” diferentes utilizadas parar representar números decimais (0-9). A primeira ”font” é utilizada para indicar a quantidade de moedas. Já a segunda é utilizada para mostrar as horas e minutos atuais.

Figure 16: ”Font” do Número de moedas

Figure 17: ”Font” do Relógio

4. Objetos Não Estáticos e Colisões

No nosso projeto, recorremos a duas *structs* criadas por nós para conseguirmos trabalhar com objetos.

Em primeiro lugar, criamos uma *struct* com o nome *Sprite* cujas funções incluem desenhar ”XPM’s”, rodá-los, mudar a sua posição e verifica colisões. Um exemplo de um objeto que utiliza esta *struct* é o ícon da *Virtual Box*.

Com a ajuda da primeira, foi criada uma *struct* para representar *sprites* animados: a *struct Animated*. Esta *struct* tem um parâmetro que é um *array* de ”Sprite*” que nos permite animar objetos tal como o cursor, os inimigos, o botão do menu, etc.

Como já foi mencionado, as colisões são detetadas com uma função da *struct Sprite*.

5. Funções VBE

vg_exit() - unsubscribeAll() - mainLoop.h
vm_map_phys() - setGraphicsMode() - VBE.h
vbe_get_mode_info() - setGraphicsMode() - VBE.h

6. Funções Relevantes

Na função subscribeAll() é invocada a função setGraphicsMode() que por sua vez invoca funções acima mencionadas para podermos utilizar as funcionalidades do VBE no modo 0x115. Na unsibscribeAll() é chamar vg_exit() para o programa sair do modo gráfico.

3.2.2 Keyboard

1. Controlo do programa

Durante todo o decorrer do programa, a tecla ”ESC” permite terminar

imediatamente o programa. No ecrã inicial, a tecla ”ENTER” permite o *login* do utilizador e durante o jogo, as teclas ’Z’ e ’X’ permitem ativar as habilidades especiais, quando estas estiverem disponíveis.

2. **Escrita de texto**
   No ecrã inicial de login, o utilizador pode escrever uma palavra-passe antes de começar o jogo. Os carateres são representados por pintas pretas, como é costume das palavras-passe, e estas são desenhadas até ao limite da caixa de texto apesar de continuarem a acumular. É possível apagar com a tecla ”BACKSPACE” e avançar para o ecrã seguinte com a tecla ”ENTER”.

3. **Funções Relevantes**
   No ínicio da função interrupts(), é chamada subscribeAll() que acaba por subscrever as interrupções do rato com mouse_subscribe_int(). No loop dentro da função interrupts(), quando ocorrer uma interrupção é chamado kbc_ih() e que utiliza read scancode kbd().
   Quando mouse_ih() chega ao final, verifica-se se é *makecode* ou *breakcode*. No final, já depois do loop, é chamada a função unsubscribeAll() que manda de novo comandos e invoca mouse unsubscribe int().

3.2.3 Mouse

1. **Posição**
   O nosso programa utiliza as interrupções do *mouse* PS/2. Essas interrupções incluem as coordenadas ’X’ e ’Y’ do rato relativas às da última interrupção. É com esta informação que conseguimos determinar a posição absoluta do rato no ecrã, na qual é desenhada um cursor.

2. **Botões**
   O botão esquerdo do rato é usado lançar mísseis e para premir botões, se a posição do rato for igual à posição do botão. Quando se lança um míssil, este é desenhado onde o cursor do jogo estiver posicionado.

3. **Funções Relevantes**
   No ínicio da função interrupts(), é chamada subscribeAll() responsável por mandar 2 comandos para o Mouse com send_command_to_mouse() e por subscrever as interrupções do rato com mouse_subscribe_int(). No loop dentro da função interrupts(), quando ocorrer uma interrupção é

chamado mouse ih() e que utiliza readMouseByte().
Quando mouse_ih() chega ao final, com set_packet() para criar o "packet".
No final, já depois do loop, é chamada a função unsubscribeAll() que manda de novo comandos e invoca mouse unsubscribe_int().

3.2.4 RTC

1. **Ler data/tempo**
   Para imitar o relógio no canto inferior direito típico do *Windows*, utilizamos o RTC para ler o tempo.

2. **Gerar alarmes**
   Para saber quando temos que alterar o tempo apresentado no relógio, programamos o RTC para gerar um alarme a cada alteracão dos minutos do RTC.

3. **Interrupções Periódicas**
   Não utilizamos interrupções periódicas no projeto.

4. **Funções Relevantes**
   Na função interrupts)(), primeiro, é chamada a função read_time() para ler o tempo e, mais tarde, é chamada uma função que, por sua vez, chama a função para subscrever as interrupções do RTC, (rtc_subscribe_int)(uint8_t *bit_no).
   Depois chama-se int set_alarm() para ativar o alarme e, no loop, quando ocorrer uma interrupção é chamado rtc_ih() que chama set_alarm() de novo. No final da função interrupts(), chama-se outra função, que invoca a função que chama int(rtc_unsubscribe_int)(), função que cancela as subscrições.

4 Organização de código

4.1 Animated

Possui a struct *Animated*, que inclui um *array* de *Sprites*, para representar *Sprites* animados. Também contém funções necessárias para criar, destruir, animar e mudar a frame específica de um *Animated*.
Percentagem no projeto: 15%

4.2 Keyboard

Contém as funções necessárias para subscrever, cancelar subscrição e tratar das interrupções do teclado. Possui também funções necessárias para a intrepretação e leitura de ”packets” recebidos.
Percentagem no projeto: 10%

4.3 Main Loop

Invoca as funções relacionadas com o jogo, tais como, subscrições das interrupções. Corre a ”loop” de interrupções que controla a maior parte das funções do jogo, como por exemplo, decidir quando se deve desenhar certos *Sprites*.
Percentagem no projeto: 30%

4.4 Mouse

Contém as funções necessárias para subscrever, cancelar subscrição e tratar das interrupções do rato. Possui também funções necessárias para a intrepretação e leitura de ”packets” recebidos e, funções relacionadas com o envio de comandos.
Percentagem no projeto: 10%

4.5 RTC

Contém as funções necessárias para subscrever, cancelar subscrição e tratar das interrupções do RTC. Possui também funções necessárias para corretamente ler o tempo (incluindo esperar pelo momento correto para ler) e inclui uma função para programar o alarme para quando os segundos forem de valor

igual a 0.
Percentagem no projeto: 5%

4.6 Sprites

Contém a struct *Sprite* e as funções necessárias para criar, destruir, desenhar, alterar *Sprites* e para detetar colisões.
Percentagem no projeto: 10%

4.7 Timer

Contém as funções necessárias para subscrever, cancelar subscrição e tratar das interrupções do Timer.
Percentagem no projeto: 7%

4.8 Utils

Contém uma função que invoca sys_inb() mas com um uint8_t* como segundo argumento. Contém, também, algumas funções para comparar e processar números de 32 *bits*.
Percentagem no projeto: 3%

4.9 VBE

Contém a função para mudar o modo do sistema operativo para módo gráfico. Esse modo é o 0x115 (800x600, 24 bits por pixel).
Percentagem no projeto: 10%

4.10 Distribuição de trabalho pelo grupo

Apesar de ás vezes um elemento ter trabalhado mais numa função ou noutra, a distribuição de trabalho pelos 2 membros foi bastante equilibrada, isto é, o projeto foi realizado com bastante cooperação. É nos pedido que indiquemos o trabalho de cada membro nas funções, no entanto, como foi um trabalho com bastante e igual interevenção dos 2 membros, é difícil esclarecer. Por esta razão, é mais fácil afimar que cada membro realizou 50% do trabalho.

4.11 Código da ”Internet”

Para conseguirmos fazer o relógio no canto inferior direito do ecrã, utilizamos a funcionalidade do RTC para obter as horas e minutos atuais. No entanto, este tempo lido está no formato BCD. Para converter o tempo para decimal, acabamos por criar uma função, com código retirado da *Internet*, que faz esta conversão. Com o tempo em decimal, conseguimos fazer os números corresponder a cada sprite, respetivamente e desenhá-los no ecrã.
Função: uint32_t BCDToDecimal(uint32_t BCD)
Localização: utils.h
URL de onde foi copiado o código: http://www.mbeddedc.com/2017/03/decimal-to-binary-coded-decimal-bcd.html

4.12 Gráfico de Invocação de Funções

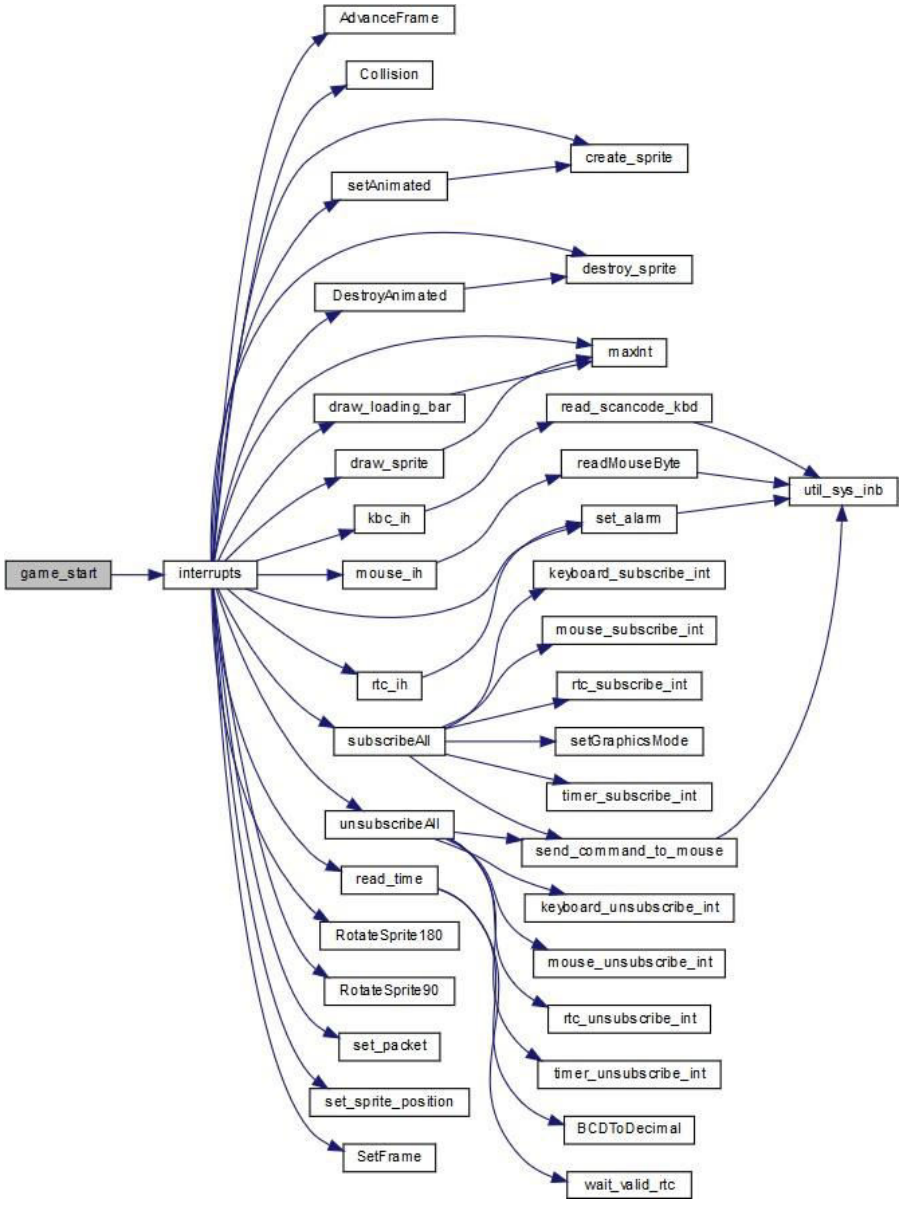

Figure 18: Gráfico

5 Detalhes da Implementação

5.1 Object oriented programming

Utilizamos esta estratégia recorrendo a para *sprites* estáticos e animados, tal como sugerido nas aulas teóricas. Utilizámos ”mallocs” e ”frees” para poder ter *sprites* de tamanhos variáveis, assim tendo duas *structs* genéricas para qualquer sprite que pretendessemos criar.

5.2 State Machines

Para conseguirmos controlar o estado do programa, tivemos que recorrer à implementação de uma máquina de estados. A nossa implemetação baseou-se na utilização de *bools*, ou seja, as acões do utilizador influenciam o valor destas *bools*.
A máquina pode ser visualizada na imagem seguinte:

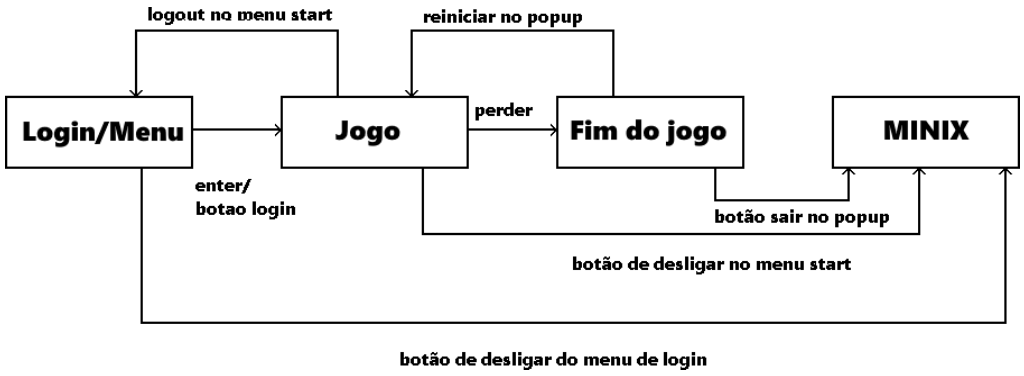

Figure 19: State Machine das fases do jogo

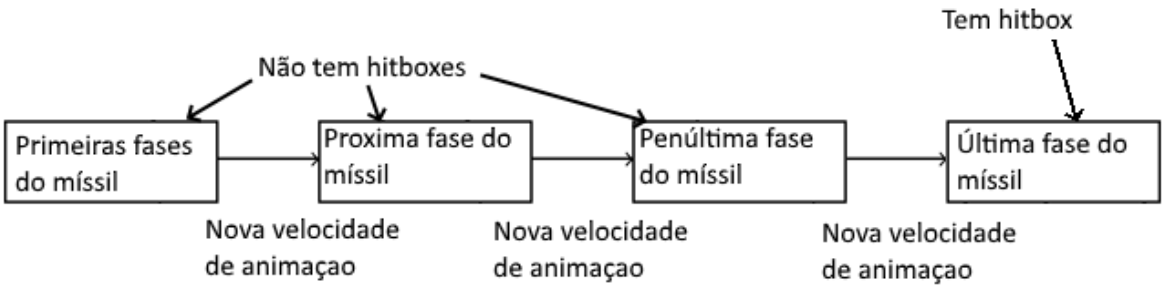

Figure 20: State Machine do míssil

5.3 RTC

O nosso relógio mostra apenas as horas e os minutos. No ínicio, usamos uma função para ler o tempo atual do RTC. De seguida, colocamos uma

interrupção de alarme a ocorrer aos 0 segundos fazendo set do valor do alarme de segundos a 0 e os dos outros alarmes, isto é, minutos e horas, a 0b11000000, ou seja, os 2 bits mais significativos a 1 (don't care value). Assim, quando ocorre um alarme, incrementamos o tempo mostrado, e ativamos o alarme de novo.

5.4 Double Buffering

Com a função memcpy() do C, copiamos o background para o *memory buffer*. Após desenharmos todos os sprites em cima do background no *memory buffer*, voltamos a usar memcpy() mas, desta vez, para copiar desse *buffer* para a memória principal, assim implementando *double buffering*.

5.5 Construção do Código

A tática que utilizámos foi muito semelhante a "Bottom up approach". Construímos primeiro as funções de nível mais baixo (por exemplo, as funções feitas nos labs, como sys_inb()), que são necessárias para correr o jogo. Depois,fizémos as funções secundárias (criar os enimigos, etc). No fim, dedicámo-nos mais a implementar mecânicas opcionais e menos importantes para o bom funcionamento do jogo, (habilidades).

6 Conclusões

6.1 Avaliação da Cadeira

Na nossa opinião, a cadeira de LCOM foi a cadeira mais satisfatória do semestre. Adorámos fazer o projeto ,que foi diferente de tudo o que nos pediram para fazer até agora, mais concretamente, adorámos a liberdade que nos foi oferecida para o projeto. Aprendemos muito sobre a linguagem C, ganhámos muita responsabilidade devido aos trabalhos(labs) feitos ao longo do semestre e aprendemos muito sobre como trabalhar em grupo, o que será muito importante no futuro. No entanto, também achamos que esta cadeira tem muito mais potencial e que podería ser melhor aproveitada.

O ponto negativo que se destaca, são os guiões. A verdade é que passámos mais tempo a enteder o que devemos fazer, em vez de o fazer. Como programador, é importante saber o que se está a programar. Se o programador não entende o que tem que fazer, também não o consegue fazer. Acreditamos que a cadeira seria muito mais enriquecedora se o *website* fosse mais apelativo e organizado, se os *powerpoints* e guiões fossem melhorados e mais explícitos. Isto retira muita da frustração que maior parte dos alunos ,que acabam por não aproveitar a cadeira, sentem.

Concluindo, pela nossa experiência, LCOM é uma cadeira pretinente e das mais interessantes mas, que é completamente mascarada negativamente pela falta de organização dos Labs.