Faculdade de Engenharia da Universidade do Porto

LCOM

T3-G01- Target Practice Shooting

André Malheiro - 201706280
Juan Bellon - 201908142

Índice

Introdução

O nosso projeto, “Target Practice Shooting”, trata-se de uma simulação de tiro ao alvo no seu estilo mais tradicional.

O jogo consiste em tentar acertar no maior número de alvos, tendo o jogador 15 balas ao seu dispor. Quando o número de balas é igual a 0, o jogo termina automaticamente. Os alvos são gerados num ponto aleatório do ecrã e movimentam-se segundo uma trajetória definida inicialmente. Quando o alvo no ecrã chega ao limite do mesmo, é apagado e gera-se um novo alvo.

O jogador pode apontar a mira usando o rato e disparar com o botão esquerdo do mesmo. Quando o utilizador acerta no alvo, este ganha 1 ponto e o alvo é pintado. Tendo a arma só 3 balas, o utilizador é obrigado a muni-la (premindo a tecla R do teclado) para continuar a disparar. A munição da pistola pode ser feita a qualquer altura do jogo, desde que haja balas disponíveis para tal.

Após as 15 balas serem gastas, o ecrã de final de jogo é automaticamente apresentado, mostrando ao utilizador a sua pontuação obtida. Neste modo, o utilizador pode inserir o seu nome e voltar ao menu inicial.

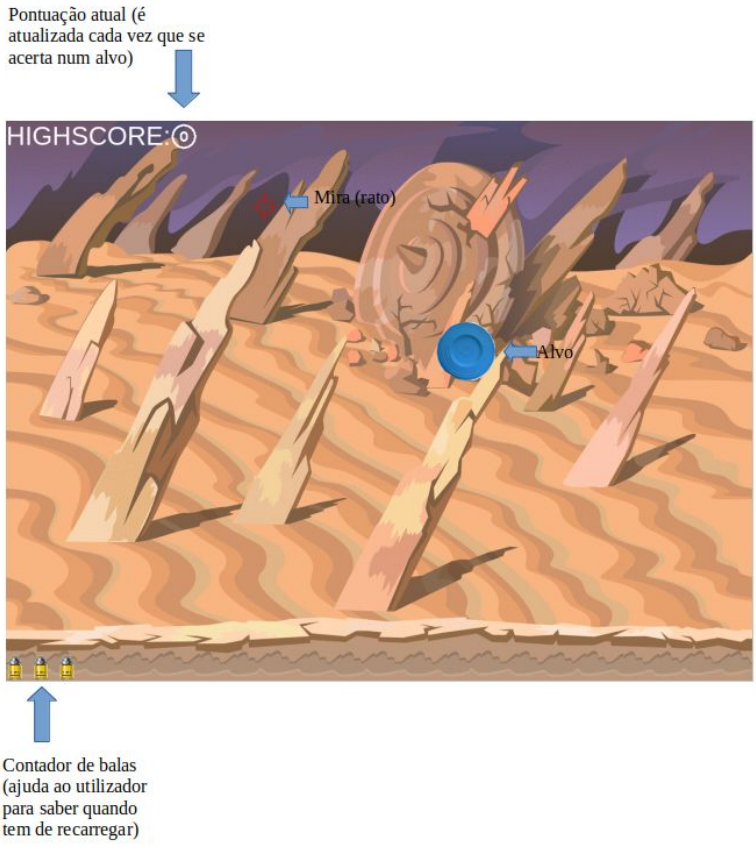
O objetivo do projeto foi aplicar o conhecimento adquirido nas aulas:

1. Programar o hardware
2. Desenvolver programas de baixo nível
3. Implementar diferentes softwares de programação a grande escala

Utilização do Jogo (Test-Run)

Menu Principal.Escolhe com o botão esquerdo a opção que quer.

Se escolher Play:



Quando o jogo acaba automaticamente:

Se escolher Exit:

Funcionalidades Implementadas

Periférico	Utilização	Interrupções
Timer	Atualização do jogo e dos gráficos apresentados	Sim
Teclado	Interação com a Interface(High Scores, Reload, e introdução de nome)	Sim
Rato	Interação com a Interface (Disparar e Seleção no menu)	Sim
Placa Gráfica	Visualização do ecrã de jogo	Não

Timer:

As interrupções do timer 0 (configuradas para 60 por segundo) são utilizadas para:

- atualizar a imagem;
- atualizar o estado do jogo;
- contar o tempo entre disparos consecutivos (utilizador só pode disparar de 40 ticks a 40 ticks), e o tempo de reload (60 ticks);

Teclado:

As interrupções do teclado são utilizadas para descobrir as teclas pressionadas, e assim:

- recarregar a arma, premindo a tecla R durante o jogo;
- permitir que o utilizador introduza o seu nome no final do jogo (apenas lidas teclas de A a Z);
- permitir ao jogador por o jogo em pausa, premindo a tecla ESC, e permitindo ao jogador continuar o jogo, voltando a premir

a mesma tecla;

Rato:

As interrupções do rato são utilizadas para ler os pacotes de dados do rato, e com esses atualizar a classe Mouse. Desta forma, as interrupções estão diretamente associadas:

- à atualização das coordenadas do rato, levando ao movimento do rato no ecrã;
- ao “disparo” das balas durante o jogo, usando o botão esquerdo do rato;
- à seleção das opções em qualquer menu do jogo, usando o botão esquerdo do rato;

Placa Gráfica:

Neste projeto decidimos usar o modo gráfico 0x14C, cuja resolução é 1152x864 pixéis.

Inicialmente implementámos um sistema de double-buffering. Seguindo o conselho de um antigo colega do MIEIC, acrescentámos um buffer auxiliar para o desenho do rato, que nos facilitou a exibição da imagem do rato no ecrã.

Ao decorrer da evolução do projeto encontrámos problemas conhecidos (cintilação das imagens) com as animações existentes. A solução encontrada foi a implementação de um novo buffer para o desenho do background. Este buffer só é utilizado durante o jogo, e será atualizado sempre que se disparar uma bala, acertar num alvo ou se recarrega a arma.

Deste modo, o programa desenha o background inicialmente no buffer *“background_buffer”*. De seguida, copia o conteúdo desse buffer para o *“buffer”*, onde vai desenhar a imagem do alvo nos pixéis corretos. Por último, o programa chama uma função *“update_mouse”*

que copia o conteúdo do *“buffer”* para o *“mouse_buffer”*, onde desenha a imagem do rato. Após estas chamadas, o programa copia o conteúdo do *“mouse_buffer”* para a memória VRAM.

Estrutura do Código

Inicialmente, fomos organizando o código por módulos, de maneira a dividir a implementação das funcionalidades, e a melhorar a legibilidade para quem o visse no futuro. Contudo, com o passar do tempo e devido à falta do mesmo até à data final da entrega, o programa foi ficando mais centrado num módulo principal.

- **proj.c:**

  Este módulo foi inicialmente fornecido pelos professores, mas muito modificado.

  Consiste maioritariamente num ciclo de interrupções, que chama as funções corretas para lidar com os periféricos.

  Contribuição: André Malheiro / Juan Bellon (80% / 20%).

  Peso: 3%

- **game.c:**

  Este módulo é o principal. É responsável por:

  - criar uma instância da classe *“Game”*;
  - fazer a ligação entre o timer,rato, teclado e o programa;
  - iniciar o jogo;
  - fazer as atualizações do jogo e do seu estado;
  - gerir os clicks do botão esquerdo do rato;
  - eliminar a instância da classe *“Game”*;

  Contribuição: André Malheiro / Juan Bellon (75% / 25%).

Peso: 28%

- **disk.c:**

  Este módulo é responsável pela criação de uma instância da classe *“Disk”*, pela atribuição de coordenadas iniciais ao alvo e pela sua trajetória, e pela atualização das suas coordenadas.

  Contribuição: André Malheiro / Juan Bellon (100% / 0).

  Peso: 5%

- **highscore.c:**

  Este módulo seria responsável pela escrita/leitura num ficheiro de texto *“highscores.txt”* dos nomes dos jogadores e das suas pontuações.

  Devido à falta de tempo, esta funcionalidade não pode ser implementada.

  Contribuição: André Malheiro / Juan Bellon (0% / 100%).

  Peso: 0%

- **kbc.c:**

  Este módulo é responsável por toda a comunicação com o Keyboard Controller. Logo, todas as operações de leitura ou escrita no teclado ou rato.

  Desenvolvida durante as aulas laboratoriais, mas ligeiramente otimizada durante o projeto.

  Contribuição: André Malheiro / Juan Bellon (50% / 50%).

  Peso: 15%

- **keyboard.c:**

Este módulo é responsável pelas subscrições das interrupções do teclado e pelo cancelamento da mesma.

Processa também as interrupções do mesmo, criação dos pacotes de dados, e a verificação da tecla que foi premida.

Contribuição: André Malheiro / Juan Bellon (50% / 50%).

Peso: 10%

- **menu.c:**

Este módulo é responsável pela criação de duas instâncias da classe *“Menu”* e pela sua destruição.

Contribuição: André Malheiro / Juan Bellon (80% / 20%).

Peso: 2%

- **mouse.c:**

Este módulo é responsável pela subscrição das interrupções do rato e pelo cancelamento da mesma.

É também responsável por criar os pacotes de dados do rato, definir o modo do rato ao criar o jogo, definir o modo do rato para o default do Minix ao sair do jogo, e atualizar as coordenadas da instância da classe *“Mouse”*.

O código usado foi maioritariamente reutilizado das aulas laboratoriais, mas com ligeiras alterações para ir ao encontro das nossas necessidades.

Contribuição: André Malheiro / Juan Bellon (70% / 30%).

Peso: 15%

● timer.c:

Este módulo é responsável pela subscrição das interrupções do timer e pelo cancelamento da mesma.

O código usado foi desenvolvido durante as aulas laboratoriais.

Contribuição: André Malheiro / Juan Bellon (50% / 50%).

Peso: 5%

● utils.c:

Este módulo é responsável por executar a função *“sys_inb”* o byte menos significativo.

Código desenvolvido durante as aulas laboratoriais.

Contribuição: André Malheiro / Juan Bellon (50% / 50%).

Peso: 2%

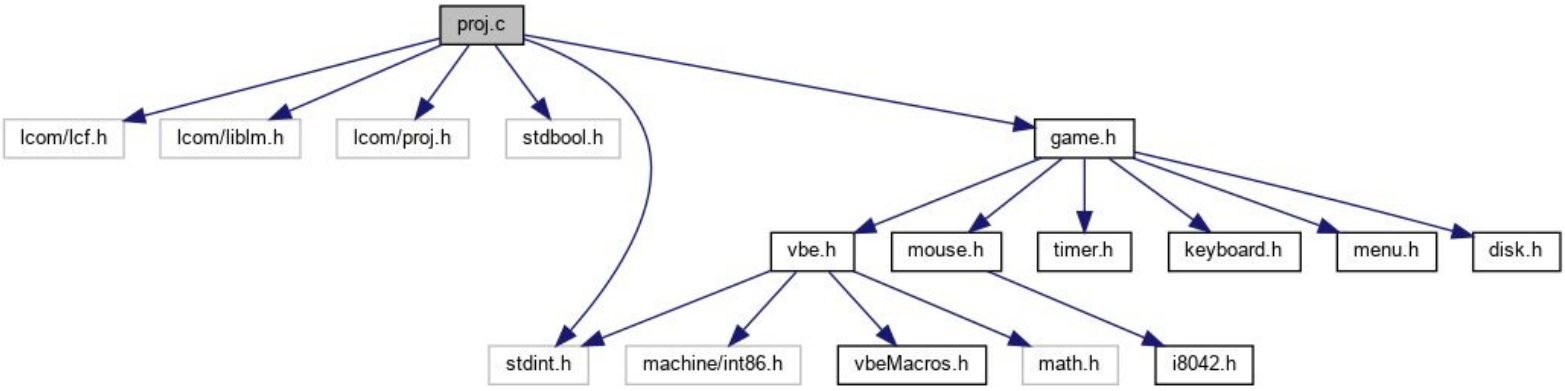
● vbe.c:

Este módulo é responsável por configurar o modo gráfico para o desejado no início do jogo, e sair do modo gráfico no final do mesmo. É também responsável pelo desenho das imagens xpm nos buffers, e pela atualização da VRAM.

Código maioritariamente desenvolvido durante as aulas laboratoriais, mas modificado de modo a satisfazer as nossas necessidades.

Contribuição: André Malheiro / Juan Bellon (75% / 25%).

Peso: 15%



Detalhes de Implementação

Em relação a detalhes de implementação, o projeto está dividido por 2 camadas. A camada mais baixa, composta por funções relativas aos periféricos. E a camada superior, onde se controla a maioria dos aspetos do jogo e faz-se a ligação à camada inferior para recolha de dados.

O projeto baseia-se num ciclo de interrupções onde o programa reage de acordo com o dispositivo que gera a interrupção. Deste modo, o jogo é atualizado com as interrupções do timer de forma a atingir uma taxa constante de atualização (60Hz), e conseguir animar os objetos.

Relativamente a estruturação de código, implementámos várias estruturas para obter um programa orientado a objetos.

Conclusão

A falta de documentação acessível aos estudantes em conjunto com a dificuldade da matéria em questão, torna a disciplina bastante frustrante para vários alunos.

Contudo, e apesar de ser demasiado trabalhosa para os créditos que tem, ambos achamos a Unidade Curricular bastante importante. A programação a baixo nível é algo que iremos voltar a ver ao longo do nosso percurso, portanto foi uma aprendizagem definitivamente útil. Para além do mais, aprender a configurar os periféricos é interessante.

Em suma, poder aplicar os conhecimentos acumulados ao longo do semestre e finalmente ver o resultado prático do esforço e dedicação, é motivador e gratificante.