Projeto de Arquitetura de Computadores

LEIC, LERC e LEE

Jogo de batalha espacial

IST - Taguspark

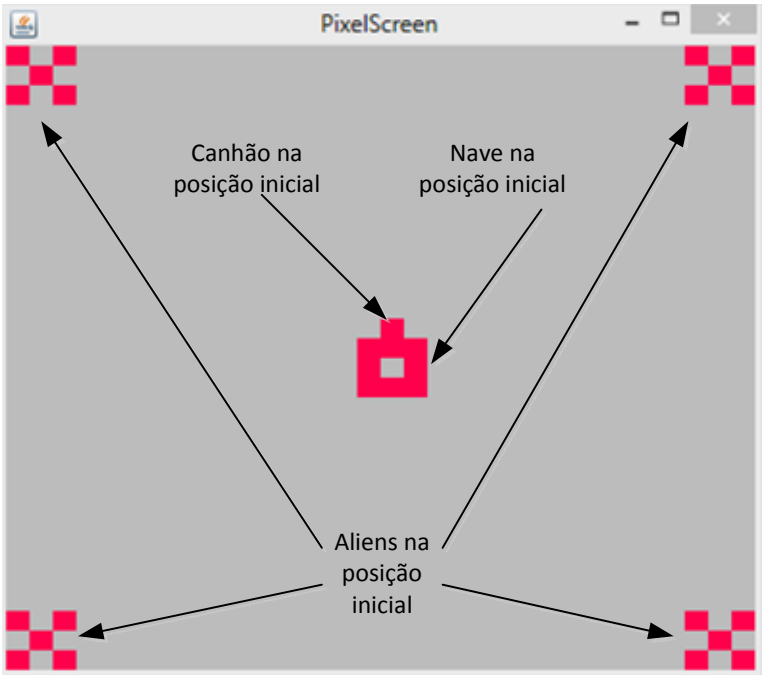
2012/2013

1 – Objetivos

Este projeto pretende exercitar as componentes fundamentais de uma disciplina de Arquitetura de Computadores, nomeadamente a programação em linguagem *assembly*, os periféricos e as interrupções.

O objetivo deste projeto consiste em concretizar um jogo de batalha espacial, em que vários aliens se tentam aproximar da nave do jogador para colidirem com ela e a destruírem. A nave possui um canhão, que pode ser orientado em todas as direções, e consegue disparar um raio que destrói os aliens, mas cujo alcance é infelizmente limitado. A nave pode movimentar-se em qualquer direção, mas gasta energia com isso, tal como disparar o raio. Destruir um alien dá um bónus de energia. O jogo termina quando um alien conseguir destruir a nave ou a energia desta se esgotar.

A figura seguinte exemplifica a janela de ecrã do jogo no instante inicial, com algumas anotações.



2 – Especificação do projeto

O ecrã de interação com o jogador tem 32 x 32 pixels. Tanto a nave como cada alien ocupa 9 pixels. O aspeto de cada um é à escolha do grupo (a figura mostra apenas um exemplo).

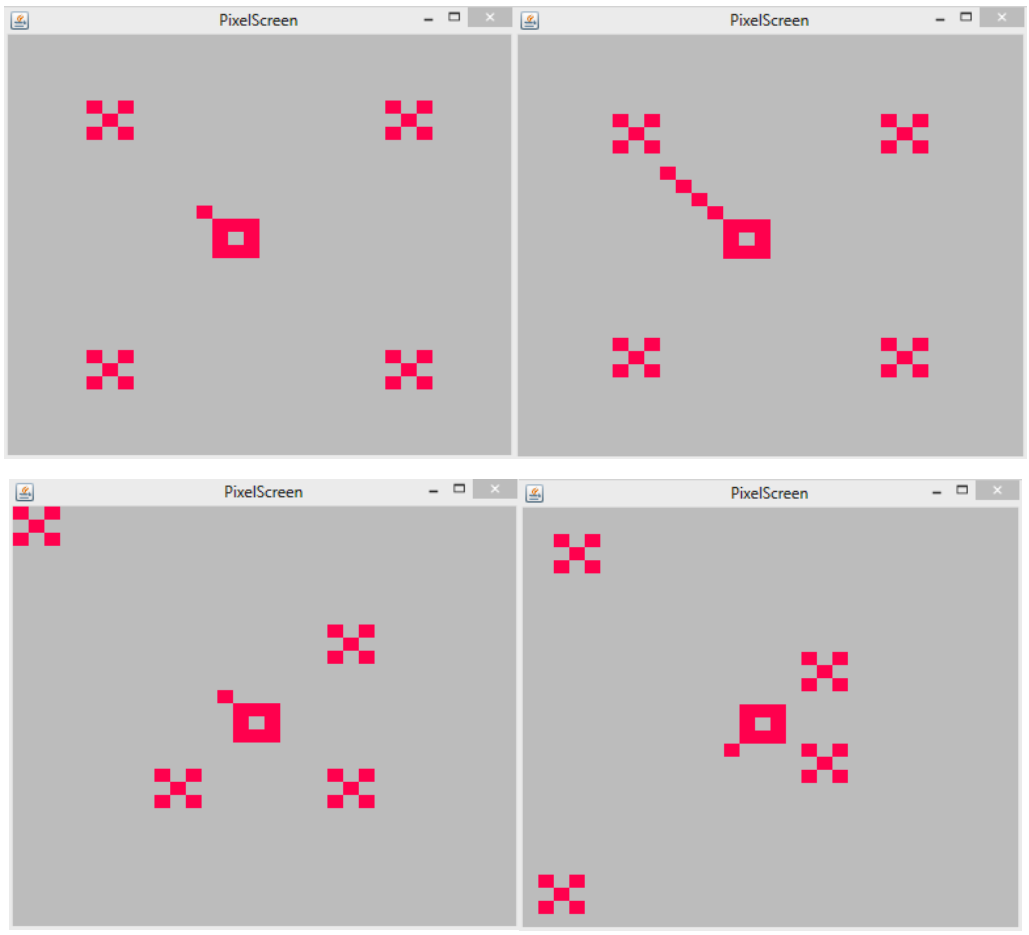
A nave tem um canhão de um pixel que pode circular pelos 8 pixels adjacentes à nave segundo as orientações verticais, horizontais e intermédias. O canhão pode disparar um raio, com um alcance de 3 pixels, na mesma direção para que o canhão está orientado.

A nave pode ser deslocada, segundo as mesmas 8 direções.

Se a ponta do raio entrar no espaço ocupado por um alien, destrói-o. Um alien destruído recomeça na sua posição inicial.

Os aliens evoluem de forma autónoma e automática, reduzindo de uma unidade, em cada evolução, a diferença em linha e coluna da sua posição face à posição da nave. Se o seu centro coincidir com o centro da nave, há uma colisão, esta é destruída e o jogo acaba. O ritmo de evolução dos aliens deve ser programável (sugere-se 3 a 5 segundos de período).

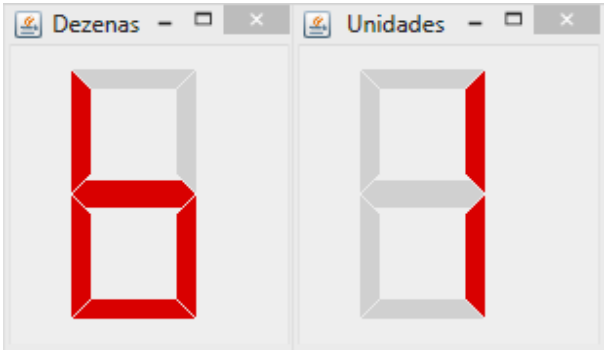
Os dois ecrãs seguintes mostram o jogo após algumas evoluções e o canhão já noutra orientação, o disparo do raio (que neste exemplo não atingiu o alien) e a destruição e recomeço dos aliens.



O número de aliens deve poder ser alterado (por uma constante no programa) entre 0 e 4.

As posições iniciais dos aliens, tal como da nave e a orientação do canhão, são à escolha do grupo (os ecrãs anteriores são apenas exemplos).

A nave tem uma reserva de energia, que é mostrada em valor decimal em dois displays de 7 segmentos, tal como a figura seguinte exemplifica:



Mover a nave e disparar o canhão gastam energia. O valor inicial da energia, bem como a energia gasta no movimento e no disparo, são à escolha do grupo (sugere-se 99 para valor inicial, 1 para mover a nave de uma posição e 2 para disparar o canhão).

Destruir um alien dá um bónus de energia, também configurável por uma constante no programa (à escolha do grupo, sugerindo-se 10).

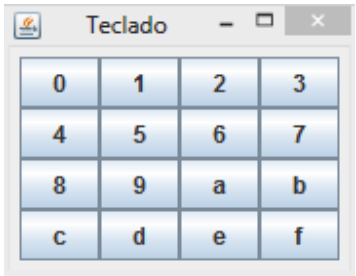
Para evitar a tentação de ligar o raio e ficar à espera que o alien chegue, a manutenção do raio ligado gasta também energia por cada unidade de tempo (sugere-se 2 unidades de energia por cada 0,5 segundos, para além das 2 unidades consumidas pelo ligar do raio). Esta cadência de consumo de energia pelo facto de o raio estar ligado deve ser programável.

O valor da energia disponível deve ser atualizado nos displays sempre que mude e o programa deve terminar se alguma vez a energia ficar negativa como resultado de alguma operação que consuma energia.

A situação de disparar o raio para as “paredes”, com a nave muito perto das fronteiras do ecrã, de modo que o raio saia fora do ecrã, deve estar prevista e protegida.

De igual forma, a nave não se deve poder aproximar demasiado das fronteiras do ecrã, devendo sempre guardar espaço para poder desenhar a nave completamente mais o seu canhão, em qualquer direção.

O comando do jogo é feito por um teclado, tal como o da figura seguinte:



São necessários os seguintes comandos:

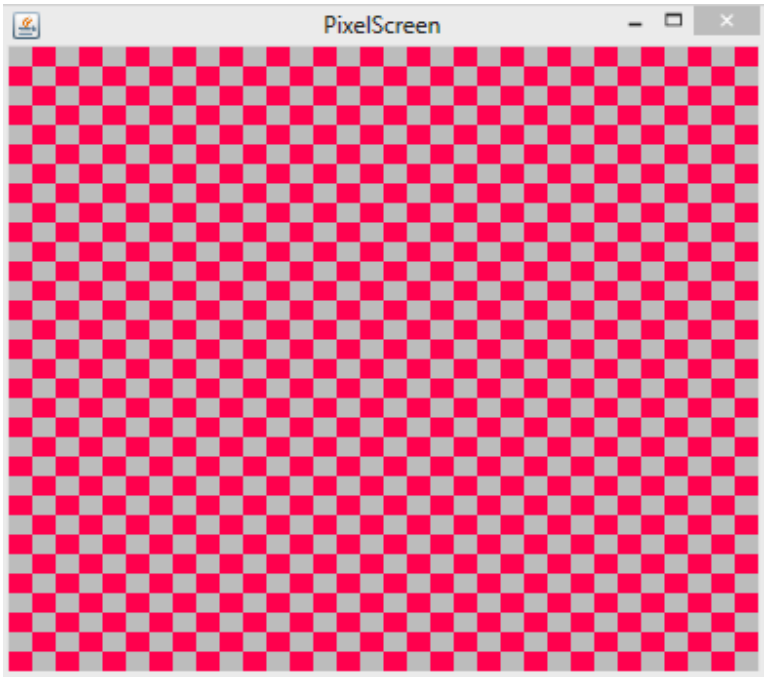
- Movimentar a nave nas 8 direções;
- Rodar o canhão no sentido dos ponteiros do relógio;
- Rodar o canhão no sentido contrário ao dos ponteiros do relógio;
- Disparar canhão;
- Reiniciar o jogo;
- Suspender o jogo (para recomeçar, carrega-se novamente na mesma tecla);
- Terminar o jogo (para recomeçar, carrega-se na tecla de reiniciar).

A escolha de que tecla faz o quê é à escolha do grupo.

Todas as teclas funcionam ao ser carregadas, mas só depois de ser largada é que pode funcionar novamente, com exceção da tecla de disparo do canhão. Por exemplo, para movimentar a nave de vários pixels ou rodar o canhão de várias direções, tem de se carregar e largar a respetiva tecla várias vezes.

A tecla de disparo do canhão faz ligar o raio ao ser carregada e desligar o raio ao ser largada. Enquanto a tecla estiver carregada, o raio deve estar ligado. No entanto, a energia deve ir sendo consumida, tal como descrito atrás, enquanto esta tecla estiver carregada.

Um jogo terminado deve mostrar um ecrã à escolha do grupo, que tanto pode ser sintetizado pelo programa (com um ciclo) ou copiado de uma tabela (sempre é mais bonito…). A figura seguinte mostra um exemplo muito simples:



Juntamente com este documento, encontrará um ficheiro excel (**screen.xlsx**), que reproduz os pixels do ecrã. Foi usado para realizar algumas experiências na conceção deste trabalho e poderá também usá-lo para desenhar algumas letras grandes (exemplo: GAME OVER) e traduzi-las para uma tabela, de forma a produzir um ecrã de fim de jogo mais interessante. Este aspeto é opcional.

NOTA - Cada grupo é livre de mudar as especificações do jogo, desde que não seja para ficar mais simples e melhore o jogo em si. A criatividade e a demonstração do domínio da tecnologia são sempre valorizadas!

3 – Estratégia de implementação

Alguns dos guiões de laboratório contêm objetivos parciais a atingir, em termos de desenvolvimento do projeto. Tente cumpri-los, de forma a garantir que o projeto estará concluído na data de entrega.

Devem ser usados processos cooperativos para suportar as diversas ações do jogo, aparentemente simultâneas. Recomendam-se os seguintes processos:

- Teclado (varrimento e leitura das teclas, tal como descrito no guião do laboratório 4);
- Nave (para controlar a nave o canhão);
- Aliens (para controlar os aliens que houver);
- Controlo (para tratar das teclas de reiniciar, pausa e terminar).

Como ordem geral de implementação, recomenda-se a seguinte:

- Rotinas de ecrã (desenhar/apagar um pixel numa dada linha e coluna, desenhar/apagar nave/alien – represente os objetos pelas coordenadas do seu pixel central e desenhe-os à volta dele);
- Teclado (um varrimento de cada vez, inserido num ciclo principal onde as rotinas que implementam processos vão ser chamadas);
- Nave (primeiro circular o canhão à volta da nave fixa, depois mover a nave e o canhão, disparar o raio e controlo de energia);
- Controlo;
- Interrupções (testando o controlo de energia do raio ligado);
- Aliens (comece por ter apenas um, já com interrupções, depois generalize para vários);

Note que muita coisa gira à volta da direção do canhão (que é também a do raio). Uma rotina que calcula a posição seguinte a outra, segundo uma dada direção, será muito útil.

Use tabelas extensivamente. Por exemplo:

- A nave e os aliens podem ser descritos por uma tabela de 3 bytes, de que só se aproveitam os 3 bits de menor peso. Para os desenhar é só copiar esses bits para 3 linhas seguidas. Naturalmente, poderá usar outra implementação
- A posição do canhão face à nave pode ser guardada como um valor de 0 a 7 (8 posições), valor esse que depois pode ser usado para indexar uma tabela, em que para cada entrada se têm dois

valores (que podem ser -1, 0 ou 1) que indicam o deslocamento em linha e coluna, face a uma dada posição, que definem uma dada direção. Por exemplo, na página 2, a direção do canhão, quando dispara o raio, corresponde a (-1, -1), isto é, para obter o pixel seguinte nessa direção tem de se subtrair 1 quer à linha, quer à coluna;

- Não vai certamente testar as 8 teclas de controlo de movimento da nave uma a uma! Sugere-se que o varrimento da tecla gere um valor entre 0 e 15 (correspondente à tecla), valor esse que depois é usado como índice numa tabela de 16 entradas, em que para aquelas 8 teclas se faz corresponder um valor da direção (0 a 7), que por sua vez pode ser usado como índice na tabela de direções do canhão, e assim obter os valores (-1, 0 ou 1) a somar à linha e coluna onde a nave está antes de se mover;

Para cada processo, recomenda-se:

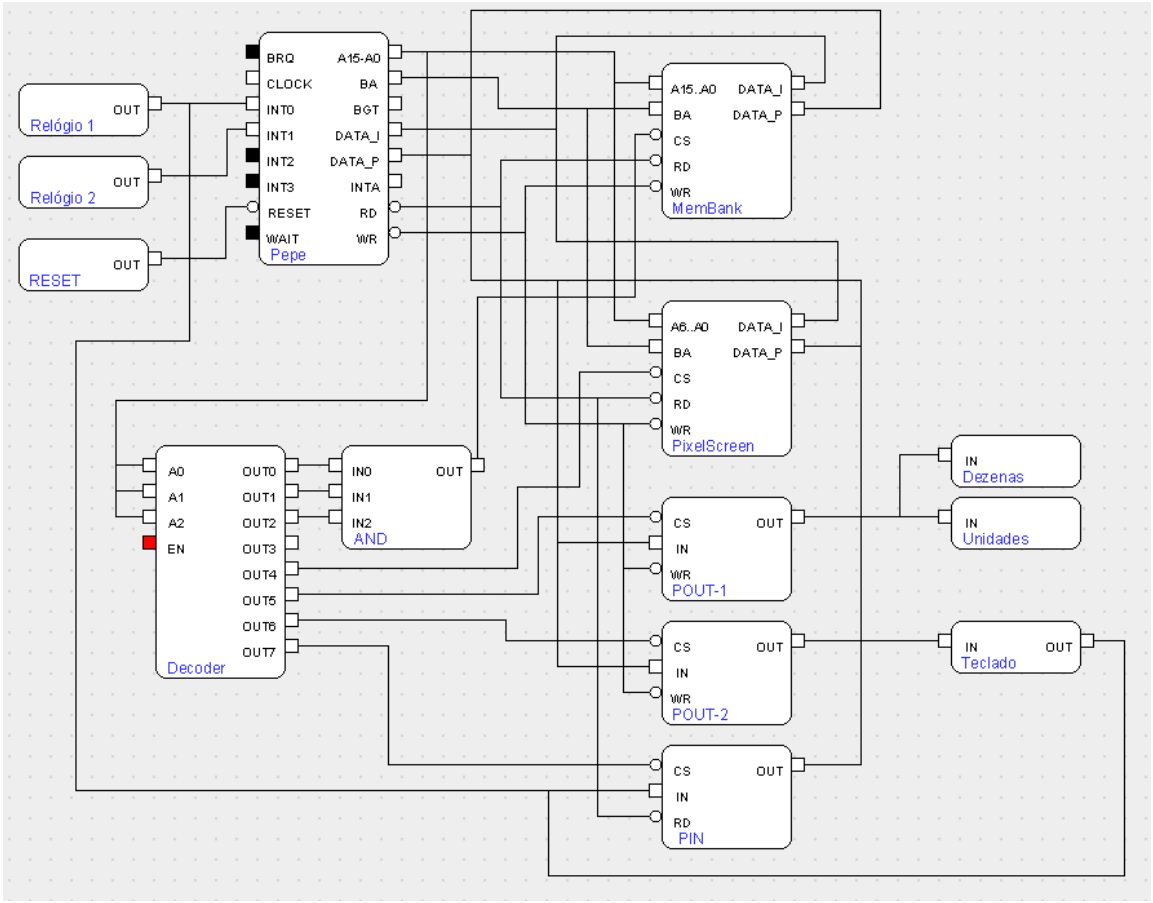
- Um estado 0, de inicialização. Assim, fazer reset ao jogo é pôr todos os processos no estado 0, em que cada um inicializa as suas próprias variáveis. Fica mais modular;
- Planeie os estados (situações estáveis) em que cada processo poderá estar. O processamento (mover a nave, evoluir um alien) é feito ao transitar entre estados;
- Veja que variáveis são necessárias para manter a informação relativa a cada processo, entre invocações sucessivas (posição da nave, direção do canhão, energia da nave, posição de cada alien, etc.)
- O processo alien trata de todos os aliens que houver. Logo, terá de ter uma tabela, com uma entrada (contendo a informação de estado) para cada alien, e fazer um ciclo para os tratar a todos;

Finalmente:

- Como norma, faça PUSH e POP de todos os registos que use numa rotina e não constituam valores de saída. É muito fácil não reparar que um dado registo é alterado durante um CALL, causando erros que podem ser difíceis de depurar;
- Vá testando todas as rotinas que fizer e quando as alterar. É muito mais difícil descobrir um bug num programa já complexo e ainda não testado;
- Estruture bem o programa, com zona de dados no início e rotinas auxiliares de implementação de cada processo junto a eles;
- Não coloque constantes numéricas (com algumas exceções, como 0 ou 1) pelo meio do código. Defina constantes simbólicas e use-as depois no programa;
- Produza comentários abundantes, não se esquecendo de cabeçalhos para as rotinas com descrição, registos de entrada e de saída.

4 – Implementação

A figura seguinte mostra o circuito a usar (fornecido, ficheiro **jogo.cmod**).



Podem observar-se os seguintes módulos, cujo painel de controlo deverá ser aberto em execução (modo Simulação):

- Relógio 1 – Relógio de tempo real, para ser usado como base para as temporizações de evolução dos aliens. Em versões intermédias pode ser útil lê-lo num ciclo de instruções. Por isso, também liga ao bit 4 do porto de entrada PIN;
- Relógio 2 – Relógio de tempo real, para ser usado como base para a temporização de gasto de energia pelo facto de o raio estar ligado;
- Matriz de pixels (PixelScreen) – ecrã de 32 x 32 pixels. É acedido como se fosse uma memória de 128 bytes (4 bytes em cada linha, 32 linhas). Atenção, que o pixel mais à esquerda em cada byte corresponde ao bit mais significativo desse byte;
- Dois displays de 7 segmentos, ligados aos bits 7-4 e 3-0 do porto POUT-1, para mostrar a energia disponível na nave;
- Teclado, de 4 x 4 botões, com 4 bits ligados ao porto POUT-2 e 4 bits ligados ao porto PIN (bits 3-0). A deteção de qual botão está carregado é feita por varrimento.

O mapa de endereços (em que os dispositivos podem ser acedidos pelo PEPE) é o seguinte:

Dispositivo	Endereços
RAM (MemBank)	0000H a 5FFFH
PixelScreen	8000H a 807FH
POUT-1 (porto de saída de 8 bits)	0A000H
POUT-2 (porto de saída de 8 bits)	0C000H
PIN (porto de entrada de 8 bits)	0E000H

Notas:

- Os periféricos de 8 bits devem ser acedidos com a instrução MOVB;
- A quantidade de informação mínima a escrever no PixelScreen é de um byte. Por isso, para alterar o valor de um pixel, tem primeiro de se ler o byte em que ele está localizado, alterar o bit e escrever de novo no mesmo byte;
- O Relógio 1 (bit 4) e o teclado (bits 3-0) partilham o mesmo porto de entrada (PIN). Por isso, terá de usar uma máscara ou outra forma para isolar os bits que pretender, após ler este porto.