Projeto de Introdução à Arquitetura de Computadores

LEIC, IST

Beyond Mars: passagem pela cintura de asteroides

2022/2023

1 – Objetivos

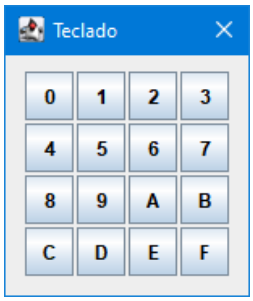
Este projeto pretende exercitar os fundamentos da área de Arquitetura de Computadores, nomeadamente a programação em linguagem *assembly*, os periféricos e as interrupções.

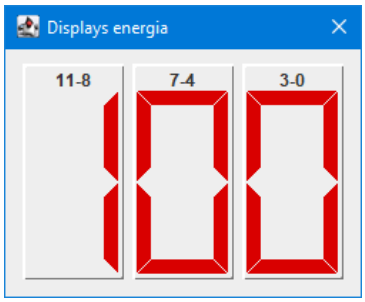
Nas próximas décadas, a humanidade prosseguirá a sua exploração do sistema solar, levando astronautas aonde ainda ninguém terá ido antes. O objetivo deste projeto consiste em concretizar um jogo de simulação da viagem interplanetária de uma nave espacial tripulada por intrépidos astronautas, que se aventuram para além de Marte e têm de atravessar a cintura de asteroides, situada entre as órbitas de Marte e de Júpiter.

Zona perigosa, dado o elevado número de asteroides, sendo crucial evitar uma colisão que será fatal. É possível disparar uma sonda com um míssil para destruir um asteroide, mas tal gasta energia preciosa. No entanto, alguns asteroides contêm metais e compostos valiosos que podem ser minerados por um robô enviado pela nave para aumentar a sua energia, tão necessária para os propulsores iónicos.

A interface do jogo consiste num ecrã, um teclado para controlo/navegação e um conjunto de displays, para mostrar a energia da nave. A imagem ilustra um possível ecrã de início de jogo.

O módulo MediaCenter do simulador possui variadas capacidades multimédia, permitindo definir imagens de fundo, reproduzir sons e vídeos, vários planos de imagens construídas pixel a pixel, um cenário frontal para letreiros, etc. O guião de laboratório 4 fornece mais detalhes sobre este módulo.





O teclado permite, fazendo clique em algumas das teclas, fazer o comando do jogo (*start*, *pause* e *stop*, para além de controlar o lançamento de sondas que ou explodem asteroides ou os mineram para enviar energia para a nave).

Os displays permitem exibir a energia atual da nave, que vai variando com o tempo. Diminui gradualmente pelo simples facto de a nave se mover (a propulsão iónica gasta muita energia) e por cada disparo de uma sonda. Aumenta quando uma sonda encontra um asteroide minerável e o respetivo robô produz energia a partir do asteroide e a envia para a nave.

Cada estado do jogo (inicial, a jogar, em pausa, terminado, etc.) deve ter um cenário ou vídeo de fundo diferente, ou um letreiro (cenário frontal, uma imagem com letras e fundo transparente), de forma a dar uma indicação visual do estado do jogo. A figura da página anterior ilustra um possível cenário de entrada, em que o utilizador tem de premir a tecla C para iniciar o jogo. Também pode ser um vídeo, com as letras sobrepostas por meio de um cenário frontal (com fundo transparente).

Junto com este enunciado dispõe de um pequeno vídeo (**demo.mp4**) que ilustra as principais situações do jogo.

2 – Descrição genérica do jogo

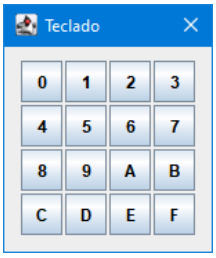
O jogo consiste genericamente nas seguintes ideias:

- O ecrã durante o jogo representa a janela frontal da nave (vista pelos astronautas), com um painel de instrumentos ao meio. Só há 3 comandos possíveis (por clique em 3 teclas diferentes do teclado): lançar uma sonda para a esquerda (a 45°), em frente ou para a direita (a 45°);
- As sondas seguem em linha reta até colidirem com um asteroide ou saírem fora do alcance da nave (máximo 12 movimentos, um pixel de cada vez), altura em que desaparecem. A imagem anterior ilustra duas sondas (pixels a roxo) lançadas em diagonal, enquanto que a sonda lançada em frente atingiu o asteroide que se vê com o efeito de explosão;
- É possível lançar sondas nas 3 direções concorrentemente, mas numa dada direção só é possível lançar de novo depois de a anterior desaparecer (por colisão ou por sair do alcance);
- Do topo de ecrã vão aparecendo os asteroides, que podem ser de dois tipos: mineráveis (verdes, no exemplo) ou não mineráveis (vermelhos, no exemplo). O tipo deve ser aleatório, numa proporção de 1 minerável para 3 não mineráveis. Em caso de colisão de uma sonda com um asteroide, há uma explosão (efeito visual e sonoro) se este for não minerável e um efeito diferente se for minerável (no exemplo, o asteroide vai reduzindo de tamanho até desaparecer, à medida que é minerado/consumido, e há um som de "nice work!");
- Se um asteroide (de qualquer tipo) não for intercetado por uma sonda e colidir com a nave (com o painel de instrumentos), a nave é destruída e o jogo acaba;
- Em cada instante, deve haver 4 asteroides no ecrã. Quando um asteroide desaparece (por explosão, mineração ou por se perder pela linha do fundo), outro deve aparecer no topo do ecrã, com as seguintes características escolhidas aleatoriamente:
  - Tipo: minerável (25% das vezes) ou não minerável (75%);
  - Coluna: no canto superior esquerdo do ecrã, no meio ou no canto superior direito;
  - Direção: os que saem dos cantos deslocam-se na diagonal (a 45°), em direção ao interior do ecrã. Os que saem do meio podem descer na vertical ou na diagonal, para a esquerda ou para a direita (estes últimos não podem colidir com a nave, mas mesmo assim há interesse em lançar uma sonda para apanhar os mineráveis, para obter energia);
  - As 5 combinações possíveis de coluna/direção devem ser equiprováveis;
- A nave gasta energia simplesmente por funcionar (a um ritmo constante) e de cada vez que lança uma sonda. Ganha energia por cada sonda que alcança um asteroide minerável. Há uma energia inicial (100%), que pode subir ou descer. Se a energia chegar a 0, a nave deixa de funcionar e o jogo acaba;
- O objetivo do jogo é prolongá-lo o maior tempo possível, evitando a destruição da nave (colisão com um asteroide) e o esgotamento total da energia;
- Existem também comandos (teclas no teclado) para controlo do jogo: *start*, *pause* e *stop*. Todos devem produzir efeitos sonoros e mostrar uma imagem diferente no ecrã.

3 – Detalhes do projeto

3.1 – Teclado e controlo do jogo

O jogo é controlado pelo utilizador por meio de teclas num teclado (atuado por clique do rato), tal como o da figura seguinte:



A atribuição de teclas a comandos é livre e ao seu gosto. Uma possível atribuição é a seguinte:

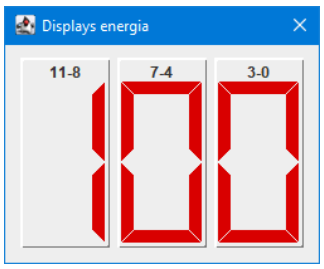
- Teclas 0, 1 e 2: lançar uma sonda para a esquerda, em frente e para a direita, respetivamente;
- Tecla C: começar o jogo (deve reiniciar a energia da nave a 100%);
- Tecla D: suspender/continuar o jogo;
- Tecla E: terminar o jogo (deve manter visível a energia final da nave).

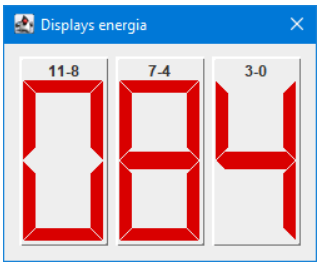
IMPORTANTE - Cada tecla dever executar apenas um comando. Para novo comando, mesmo que igual, tem de se largar a tecla e carregar de novo.

O guião de laboratório 3 ensina a trabalhar com o teclado.

3.2 – Displays e energia da nave

Existem três displays de 7 segmentos, que devem mostrar a energia da nave em percentagem do valor inicial, em cada instante (em decimal, o que implica conversão a partir dos valores hexadecimais que o PEPE usa). As figuras seguintes ilustram a energia inicial e um possível valor após o jogo evoluir:





A energia começa com 100 (%) e deve ser decrementada de 3% periodicamente (de 3 em 3 segundos).

Por cada sonda lançada, a energia diminui de 5%. Por cada asteroide minerável alcançado por uma sonda, a energia aumenta em 25%, como resultado da absorção da energia desse asteroide. Dependendo da sorte e da perícia do piloto da nave, a energia pode subir bastante acima de 100%. Se a energia chegar a 0%, o jogo termina. A imagem de cenário de jogo terminado por falta de energia deve ser diferente da resultante da colisão de um asteroide com a nave. Idem para os efeitos sonoros.

Se o jogo terminar por esta colisão, a energia que tinha na altura deve manter-se. A energia só deve ser reposta a 100% quando um novo jogo for iniciado.

O guião de laboratório 3 ensina a trabalhar com os displays.

3.3 – Ecrã, cenários, sons e bonecos

O ecrã de interação com o jogador tem 32 x 64 pixels (linhas x colunas). Cada pixel pode ter uma cor diferente, em 4 componentes (Alpha, Red, Green e Blue, ou ARGB), todas com 4 bits (valores sem sinal, de 0 a F). A primeira componente define a opacidade (0 é totalmente transparente, F totalmente opaco). O pixel pode estar ligado (com a cor definida pelas 4 componentes) ou desligado (com tudo a zero, caso em que não se vê pois fica transparente).

Painel de instrumentos da nave, asteroides e sondas são bonecos desenhados pixel a pixel.

Por trás deste ecrã de pixels está a imagem ou vídeo de fundo (a que se vê nas imagens anteriores), que tipicamente tem uma resolução bem superior (embora deva ter um fator de forma semelhante, retangular 2 x 1, para que não apareça de forma distorcida).

É possível alterar a imagem/vídeo de fundo através do programa do PEPE. Por isso, espera-se que use um cenário diferente para cada situação. Nem sempre uma imagem tem de variar. Pode editá-la, colocando texto que indique qual a situação (pausa, por exemplo), gerando assim variantes da mesma imagem. Ou pode usar um cenário frontal, com uma imagem com as letras e fundo transparente, que aparece à frente da imagem ou vídeo de fundo. Há também comandos que o programa pode dar para ligar e desligar este cenário frontal.

Não é fornecida nenhuma imagem nem nenhum vídeo para cenários, mas existem inúmeras imagens adequadas que pode obter de forma gratuita na Web para este uso pessoal. O design multimédia fica ao seu gosto. A figura seguinte ilustra um possível cenário após a colisão de um asteroide com a nave.

Também devem existir efeitos sonoros, que mais uma vez podem ser obtidos na Web. Ficheiros de som pequenos é o ideal. O módulo MediaCenter permite, no entanto, restringir os tempos de início e fim de um som (ou de um vídeo), caso haja necessidade.

Estes efeitos sonoros devem ser reproduzidos quando é lançada uma sonda, um asteroide é destruído, a energia de um asteroide é minerada por uma sonda, o jogo termina por falta de energia ou por colisão, etc. Isto é um jogo!

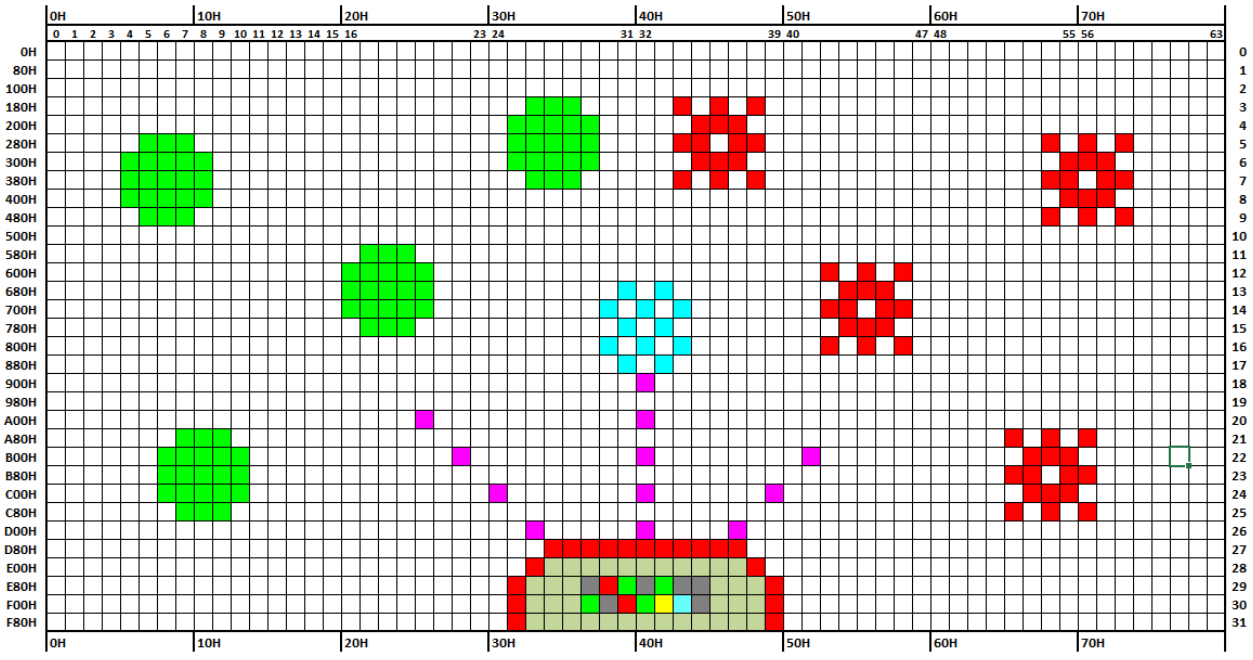
FUNDAMENTAL – Depois de definir os ficheiros de imagem, vídeo e som que quiser acrescentar ao módulo MediaCenter, deve salvar o circuito num diretório que contenha esses ficheiros (ou diretamente ou num subdiretório desse). Esta é a única forma de os ficheiros serem incluídos no ficheiro de descrição do circuito de forma portável. Caso contrário, é guardado o *path* absoluto do ficheiro, e depois não funciona noutro computador (nomeadamente, no do docente que vai avaliar o projeto!).

Desenhar um boneco no ecrã (à frente do cenário de fundo) é desenhar os seus pixels, com cor ou desligado (transparente), em posições (linha e coluna) adjacentes do ecrã, de acordo com a forma definida para esse boneco.

Toma-se um dos pixels do boneco (canto superior esquerdo, por exemplo) como indicador da posição no ecrã desse boneco e todos os restantes pixels desse boneco são desenhados em posições relativas a esse pixel de referência.

Mover um boneco é apagá-lo na sua posição atual, mudar a sua posição e desenhá-lo na nova posição. O efeito visual é o boneco parecer mover-se.

O ficheiro Excel **ecrã-32x64.xlsx**, que é fornecido, simula as quadrículas do ecrã e mostra que o ecrã é na realidade uma memória (note os endereços de cada linha no lado esquerdo), em que cada pixel é uma palavra (2 bytes) dessa memória, com a cor do pixel. Há 64 pixels, ou 128 bytes, por cada linha do ecrã.



Pode usar este ficheiro para planear o tamanho e forma dos seus bonecos, pois não têm de ter as formas e tamanhos mostrados na figura anterior, nem as mesmas cores (e cada boneco pode ter pixels de cores diferentes). A figura é apenas uma sugestão ilustrativa. Aqui pode inventar.

Cada pixel pode ser desenhado escrevendo diretamente na memória do ecrã ou por meio de um comando. Escolha o seu método. O guião de laboratório 4 tem os detalhes de como usar o módulo MediaCenter.

Pretendem-se 4 asteroides simultaneamente, mas pode começar apenas por um numa versão inicial do programa. Podem aparecer já completamente visíveis no topo de ecrã, mas os que chegarem ao fundo do ecrã devem desaparecer graciosamente (linha a linha, até a última desaparecer).

As sondas (bonecos de 1 x 1) saem do painel de instrumentos da nave no seu topo (a meio e nos cantos direito e esquerdo) e devem ter um alcance limitado (sugerem-se 12 movimentos no máximo), ao fim do qual se extinguem se nada atingirem. Podem-se lançar 3 sondas quase ao mesmo tempo, mas em direções diferentes. Só se pode lançar uma nova sonda numa dada direção depois de a anterior nessa direção desaparecer (por limite de alcance ou colisão).

O painel de instrumentos da nave tem um conjunto de luzes em variação rápida. Trata-se de um apontamento meramente cosmético (sempre dá um ar mais SciFi...) e pode ser implementado com um conjunto de bonecos com a mesma forma, mas cores diferentes, em que se vai (re)desenhando cada um deles na mesma posição, em rotação periódica. O exemplo do vídeo **demo.mp4** usa 8 bonecos.

O módulo MediaCenter possibilita que cada boneco seja desenhado num ecrã de pixels diferente (com todos os ecrãs de pixels sobrepostos). Isto tem a vantagem que os asteroides que seguem o mesmo caminho se sobreponham de forma graciosa (como se fossem janelas diferentes num sistema operativo), em vez de um destruir o desenho do anterior.

Dado que a nave e as sondas nunca se sobrepõem, podem usar o mesmo ecrã de pixels. Terá de definir o número de ecrãs de pixels que usar no MediaCenter, em modo Design no seu painel de configuração.

O guião de laboratório 4 ensina a trabalhar com o módulo MediaCenter, em termos de pixels (e desenhar bonecos), ecrãs de pixels, cenários e sons.

3.4 – Temporizações

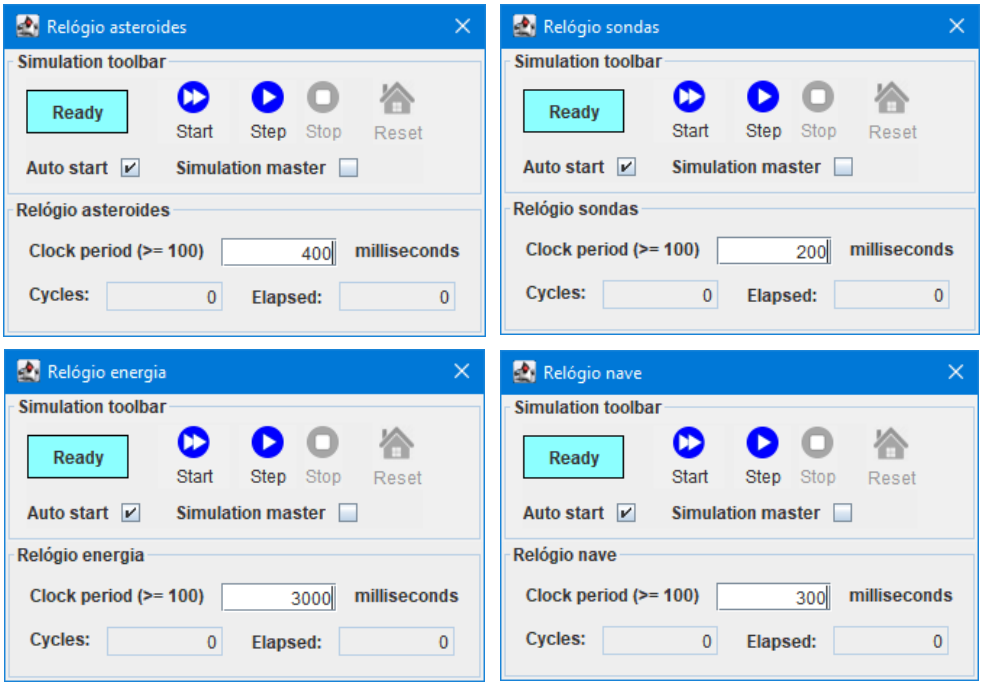
A evolução do jogo requer 4 temporizações diferentes (que pode variar a seu gosto):

- Movimento dos asteroides (período de 400 milissegundos);
- Movimento das sondas (período de 200 milissegundos);
- Decréscimo periódico da energia da nave (período de 3000 milissegundos, ou 3 segundos);
- Variação da configuração das luzes no painel de instrumentos da nave (300 milissegundos).

Os períodos indicados, que marcam o ritmo a que cada um dos eventos ocorre, são gerados por 4 relógios de tempo real, que geram um sinal de um bit que varia periodicamente entre 0 e 1, com um dado período. Sem o tempo real marcado por estes relógios, o jogo evoluiria de forma muito mais

rápida e de forma não controlável, dependendo apenas da velocidade de processamento do computador que executa o simulador.

Estes relógios estão incluídos no circuito usado neste jogo e estão pré-programados com estes tempos, mas pode alterá-los para melhorar a jogabilidade do jogo ou fazer testes.



O arranque dos relógios é automático, pelo que nem precisa de abrir as janelas de simulação deles.

O guião de laboratório 6 ensina a utilizar relógios para marcação de temporizações.

3.5 – Escolhas pseudo-aleatórias

Quando um asteroide “nasce” (aparece no topo de ecrã), deve decidir se vai ser minerável ou não, tomando a cor e forma respetivas, qual a coluna em que deve aparecer, e qual a direção a tomar.

Pretende-se que cerca de 25% de asteroides sejam mineráveis e 75% sejam não mineráveis.

Para a coluna, no canto superior esquerdo do ecrã, no meio ou no canto superior direito. Note que, dada a dimensão dos asteroides, a coluna do pixel de referência de cada asteroide pode não ser mesmo ao meio ou no canto do ecrã.

Para a direção, os que saem dos cantos deslocam-se na diagonal (a 45°), em direção ao interior do ecrã. Os que saem do meio podem descer na vertical ou na diagonal, para a esquerda ou para a direita. As 5 combinações possíveis de coluna/direção devem ser equiprováveis (20% de hipóteses cada).

Estas escolhas devem ser feitas de forma razoavelmente aleatória. Como o PEPE não tem mecanismos para gerar valores aleatórios, usa-se um truque simples. A leitura de um periférico de entrada (PIN, no circuito do projeto) gera valores aleatórios nos bits que estejam “no ar”, ou seja, não ligados a algo que

force um valor. Tal é o caso dos bits 7 a 4 do PIN, cujos bits 3 a 0 ligam ao teclado mas nada liga aos bits 7 a 4. Faz-se uma leitura do periférico (com MOVB, são lidos 8 bits), seguida de um deslocamento para a direita (instrução SHR) de 4 bits, o que coloca os bits 7 a 4 (aleatórios) do periférico nos bits 3 a 0 do registo, ficando-se assim com um valor aleatório entre 0 e 15. A partir daqui, pode obter:

- o tipo do asteroide de forma aleatória (isolando os 2 bits de menor peso, o que dá 4 hipóteses, e uma delas é usada para gerar asteroides mineráveis);
- o par coluna-direção (fazendo o resto da divisão por 5 com a instrução MOD, por exemplo). Cada uma das 5 combinações diferentes pode depois indexar uma tabela em que cada elemento tem dois valores (coluna inicial e direção: -1, 0 ou 1, isto é, o valor a somar à coluna do asteroide de cada vez que este se movimenta).

4 – Faseamento do projeto

O projeto decorrerá em duas fases, versão intermédia e final.

IMPORTANTE – Não se esqueça de identificar os ficheiros de código (grupoXX.asm, em que XX é o número do grupo) em comentários, logo no início do programa, com:

- o número do grupo;
- o número e nome dos alunos que participaram no desenvolvimento do programa.

VERSÃO INTERMÉDIA:

- Vale 25% da nota do projeto (ou 10% da nota final de IAC);
- Deve ser submetida no Fenix (Projeto da versão intermédia) até às **23h59** do dia **26 de maio de 2023**, através de um ficheiro (**grupoXX.zip**, em que XX é o número do grupo) com os seguintes ficheiros:
  - Um ficheiro **grupoXX.asm** com o código, pronto para ser carregado no simulador e executado (deve ter o número do grupo, números de aluno e nomes). Deve criar uma cópia da versão mais recente do código, limpando eventual "lixo" e coisas temporárias, de modo a compilar e executar a funcionalidade pedida. Organização do código e comentários serão avaliados, tal como na versão final;
  - Todos os ficheiros de imagem e som usados no módulo "MediaCenter";
  - Um ficheiro **projetoXX.cir** com o circuito do projeto, mas guardado depois de definir no módulo "MediaCenter" todos os ficheiros de imagem e som usados. Não se esqueça que estes ficheiros devem estar guardados no mesmo diretório do circuito, ou num subdiretório deste;
- **IMPORTANTE** - Use o circuito do projeto, **projeto.cir** (e não o de qualquer guião de laboratório). Note que o periférico dos displays é de 16 bits (deve usar MOV) e não de 8 bits (MOVB);

- **Deve cumprir os seguintes objetivos** (na versão intermédia):
    - O teclado deve estar completamente funcional, detetando todas as teclas;
    - Deve desenhar o painel de instrumentos da nave;
    - Deve ter um cenário de fundo;
    - Deve desenhar um asteroide (minerável ou não), no topo esquerdo do ecrã. Esse asteroide deve descer na diagonal uma linha no ecrã sempre que se carrega numa tecla (escolha qual), mas apenas uma linha por cada clique na tecla. Não tem de detetar a eventual colisão com a nave;
    - Deve ter um efeito sonoro, de cada vez que se carrega na tecla para o asteroide descer;
    - Deve desenhar uma sonda, inicialmente na coluna do meio da linha por cima do painel de instrumentos da nave. Essa sonda deve subir verticalmente uma linha no ecrã sempre que se carrega numa tecla (escolha qual), mas apenas uma linha por cada clique na tecla;
    - Use outras duas teclas à escolha para aumentar e diminuir de uma unidade o valor nos displays, por cada clique na tecla. Para já pode ser em hexadecimal, mas na versão final terá de fazer uma rotina para converter um número hexadecimal qualquer para dígitos em decimal.

VERSÃO FINAL:

- Vale 75% da nota do projeto (ou 30% da nota final);
- **Deve cumprir todas as especificações do enunciado**;
- Deve ser submetida no Fenix (Projeto da versão final) até às **23h59** do dia **9 de junho de 2023**, através de um ficheiro (**grupoXX.zip**, em que XX é o número do grupo) com os seguintes ficheiros:
    - Um ficheiro **grupoXX.pdf**, relatório muito simples de formato livre, mas com a seguinte informação (juntamente com este enunciado, é fornecido um possível modelo de relatório):
        - Identificação do número do grupo, números de aluno e nomes;
        - Definições relevantes, se tiverem feito algo diferente do que o enunciado pede ou indica (teclas diferentes, funcionalidade a mais, etc.);
        - Indicação concreta das funcionalidades pedidas que o código enviado NÃO satisfaz;
        - Eventuais outros comentários ou conclusões.
    - Um ficheiro **grupoXX.asm** com o código, pronto para ser carregado no simulador e executado (também deve ter o número do grupo, números de aluno e nomes);
    - Todos os ficheiros de imagem, vídeo e som usados no módulo “MediaCenter”;

- Um ficheiro **projetoXX.cir** com o circuito do projeto, mas guardado depois de definir no módulo "MediaCenter" todos os ficheiros de imagem, vídeo e som usados. Não se esqueça que estes ficheiros devem estar guardados no mesmo diretório do circuito, ou num subdiretório deste.

5 – Estratégia de implementação

Os guiões de laboratório estão alinhados com objetivos parciais a atingir, em termos de desenvolvimento do projeto. Tente cumpri-los, de forma a garantir que o projeto estará concluído nas datas de entrega, quer na versão intermédia, quer na versão final.

Devem ser usados rotinas cooperativas ou processos cooperativos (guião de laboratório 7) para suportar as diversas ações do jogo, aparentemente simultâneas. Recomendam-se os seguintes processos:

- Controlo (para tratar das teclas de começar, suspender/continuar e terminar o jogo).
- Teclado (varrimento e leitura das teclas, tal como descrito no guião do laboratório 3);
- Nave (para desenhar o painel de instrumentos e produzir o efeito das luzes);
- Energia da nave (para implementar o gasto periódico de energia);
- Sonda (para controlar o lançamento, implementar o movimento, o limite do alcance e a deteção de colisão de cada sonda);
- Asteroide (para controlar as ações e evolução de cada um dos asteroides, incluindo verificação de colisão com a nave).

Como ordem geral de implementação do projeto, recomenda-se a seguinte estratégia (pode naturalmente adotar outra):

1. Teclado e displays;
2. Rotinas de ecrã, para desenhar/apagar:
   - um pixel numa dada linha e coluna (de 0 a 31 e 0 a 63, respetivamente);
   - um boneco genérico, descrito por uma tabela que inclua a sua largura, altura e a cor ARGB de cada um dos seus pixels. Use um desses pixels, por exemplo o canto superior esquerdo do boneco, como referência da posição do boneco (linha e coluna) e desenhe os restantes pixels relativamente às coordenadas desse pixel de referência;
3. Desenho do painel de instrumentos da nave;
4. Um só asteroide, movimentado primeiro por uma tecla e mais tarde por interrupção;
5. Uma só sonda, movimentada primeiro por uma tecla e mais tarde por interrupção;
6. Gasto periódico de energia da nave, por meio de uma interrupção;
7. Processos, de forma a poder controlar os vários aspetos do jogo de forma independente;
8. Deteção de colisão (entre a sonda e um asteroide e entre um asteroide e a nave);

9. Resto das especificações, incluindo extensão para os 4 asteroides e as 3 sondas. Esta extensão tem algumas complicações, pelo que é melhor ter um só asteroide e uma só sonda a funcionar do que tentar logo tudo e correr o risco de não conseguir nada a funcionar.

IMPORTANTE:

- As rotinas de interrupção param o programa principal enquanto estiverem a executar. Por isso, devem apenas assinalar aos processos quando ocorrem, por meio de variáveis (LOCKs, no caso de usar processos cooperativos). O processamento do jogo deve ser feito pelos processos e não pelas rotinas de interrupção;
- Se usar valores negativos (por exemplo, -1 para somar à coluna de um boneco para ele se deslocar para a esquerda), as variáveis correspondentes devem ser de 16 bits (declaradas com WORD, e não BYTE).

Tenha ainda em consideração as seguintes recomendações:

- Faça PUSH e POP de todos os registos que use numa rotina e não constituam valores de saída (mas não sistematicamente de todos os registos, de R0 a R11!). É muito fácil não reparar que um dado registo é alterado durante um CALL, causando erros que podem ser difíceis de depurar. Atenção ao emparelhamento dos PUSHs e POPs, bem como à ordem relativa;
- 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, quer de constantes, quer de variáveis, e rotinas auxiliares de implementação de cada processo junto a ele;
- Produza comentários abundantes, não se esquecendo de cabeçalhos para as rotinas com descrição, registos de entrada e de saída (veja exemplos nos guiões de laboratório);
- Não coloque constantes numéricas (com algumas exceções, como 0 ou 1) pelo meio do código. Defina constantes simbólicas no início e use-as depois no programa;
- Como boa prática, as variáveis em memória devem ser de 16 bits (WORD), para suportarem valores negativos sem problemas. O PEPE só sabe fazer aritmética em complemento para 2 com 16 bits;
- Os periféricos de 8 bits e as tabelas com BYTE devem ser acedidos com a instrução MOVB. As variáveis definidas com WORD (que são de 16 bits) e periféricos de 16 bits devem ser acedidos com MOV;
- **ATENÇÃO!!!** Ao contrário dos guiões de laboratório, o periférico POUT-1 é de 16 bits (por causa dos 3 displays) e não de 8 (deve ser acedido com MOV, e não MOVB);
- Não duplique código (com *copy-paste*). Use uma rotina com parâmetros para contemplar os diversos casos em que o comportamento correspondente é usado.

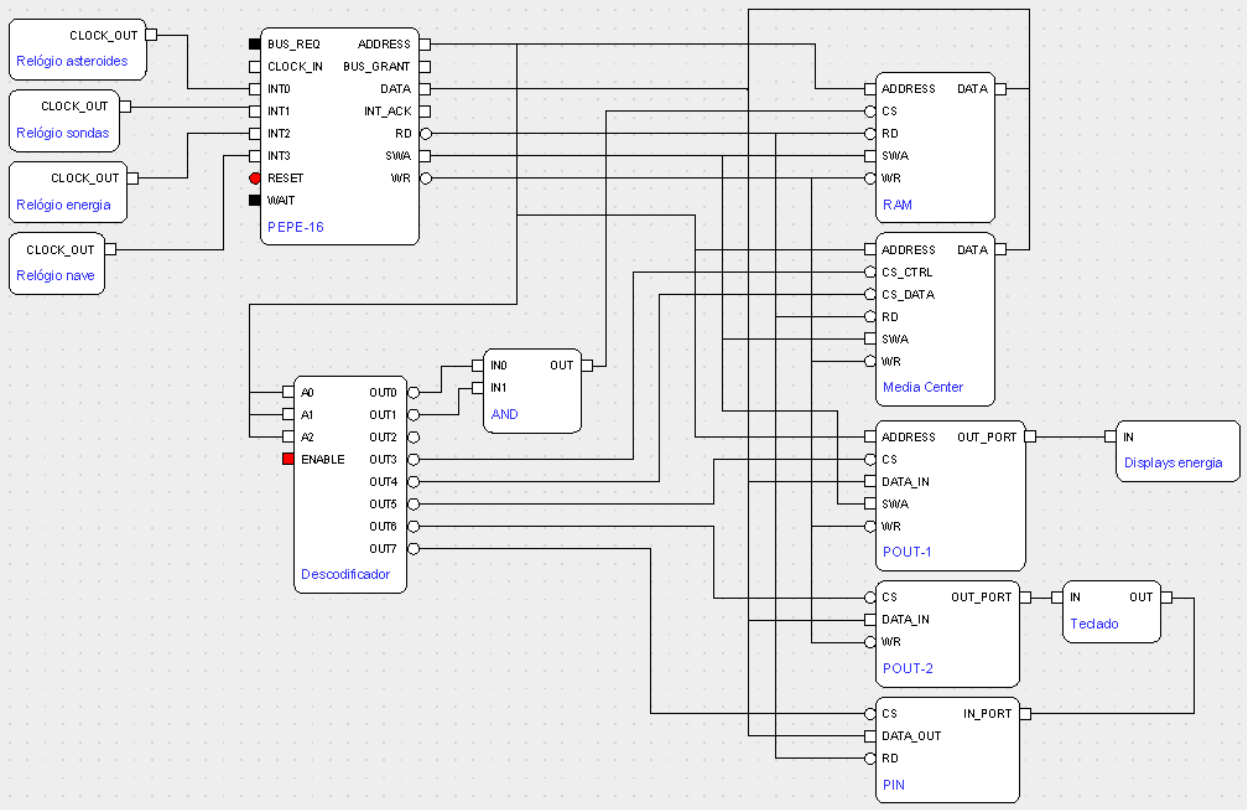
6 – Critérios de avaliação

Os critérios de avaliação e o seu peso relativo na nota final do projeto (expressos numa cotação em valores) estão indicados na tabela seguinte:

Critério	Versão intermédia	Versão final
Funcionalidade de base	1	3
Estrutura dos dados e do código	3	6
Comentários	1	2
Vários asteroides e sondas: dados e código		4
Total	**5**	**15**

7 – Circuito do projeto

A figura seguinte mostra o circuito a usar (fornecido, ficheiro **projeto.cir**). Use este circuito para o projeto (os dos guiões não servem).



Os módulos seguintes têm painel de controlo em execução (modo "Simulation"):

- Relógio asteroides – Relógio de tempo real, para ser usado como base para a temporização do movimento dos asteroides. Está ligado ao pino de interrupção 0 do PEPE;
- Relógio sondas – Relógio de tempo real, para ser usado como base para a temporização do movimento das sondas. Está ligado ao pino de interrupção 1 do PEPE;
- Relógio energia – Relógio de tempo real, para ser usado como base para a temporização da diminuição periódica de energia da nave. Está ligado ao pino de interrupção 2 do PEPE;
- Relógio nave – Relógio de tempo real, para ser usado como base para a temporização do conjunto de luzes no painel de instrumentos da nave. Está ligado ao pino de interrupção 3 do PEPE;
- MediaCenter – módulo multimédia que inclui um ecrã de 32 x 64 pixels. Este ecrã é acedido por comandos ou como se fosse uma memória de 2048 pixels (ou 4096 bytes: 128 bytes em cada linha, 32 linhas). Este periférico tem 2 *chip select*s, um para acesso pela memória e outro para acesso pelos comandos. Pode ver no ficheiro de excel **ecrã-32x64.xlsx** os endereços de cada byte (relativos ao endereço de base do ecrã). O guião de laboratório 4 fornece mais detalhes;
- Três displays de 7 segmentos, ligados aos bits 11-8, 7-4 e 3-0 do periférico POUT-1, para mostrar a energia da nave. **ATENÇÃO!!!**: ao contrário dos guiões de laboratório, este periférico é de 16 bits e deve ser acedido com MOV (e não MOVB);
- Teclado, de 4 x 4 teclas, com 4 bits ligados ao periférico POUT-2 e 4 bits ligados ao periférico PIN (bits 3-0). A deteção de qual tecla foi carregada é feita por varrimento. Atenção, que estes periféricos são de 8 bits e devem ser acedidos com MOVB (e não MOV). Note também que só os 4 bits de menor peso (3 a 0) são significativos. Os restantes (7 a 4) estão no ar e leem valores aleatórios. Por isso, deve usar uma máscara para os eliminar ao tentar detetar teclas premidas;
- Memória (RAM), que tem 16 bits de dados e 14 bits de endereço, com capacidade de endereçamento de byte, tal como o PEPE e o MediaCenter;
- PEPE-16 (processador de 16 bits).

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

Dispositivo	Endereços
RAM	0000H a 3FFFH
MediaCenter (acesso aos comandos)	6000H a 6069H (ver guião de laboratório 4)
MediaCenter (acesso à sua memória)	8000H a 8FFFH
POUT-1 (periférico de saída de 16 bits)	0A000H a 0A001H
POUT-2 (periférico de saída de 8 bits)	0C000H
PIN (periférico de entrada de 8 bits)	0E000H