Exercício 15 – Space Colector

Temática

- ⊡ Variáveis e expressões
- ⊡ Estruturas de decisão
- ⊡ Estruturas de repetição
- ⊡ Controlo por teclas

Na sua pasta de trabalho crie uma nova pasta com o nome "ex16". Nesta pasta devem ser guardados os ficheiros desenvolvidos nesta aula.

Problema:

Construir um jogo de naves. O jogador poderá controlar os movimentos na horizontal da nave, por teclas, assim como, acelerar e abrandar (movimento na vertical). Durante o jogo irão aparecer alguns obstáculos (outras naves) em sentido contrário, que o jogador deverá evitar ou destruir através do seu poder de fogo.

Neste jogo vamos utilizar um cenário com dimensões fixas (700x700). Por esse motivo, o ficheiro *index.html* invoca uma função para abrir o ficheiro *jogo.html* numa nova janela do *browser* com as características adequadas para este jogo. Verifica como está implementado!

http://labmm.clients.ua.pt/LM3/LM3-p/ex16a

Com base na página *index.html*, *jogo.html* e no ficheiro *jogo.js*, desenvolva o código javascript necessário para implementar as alíneas seguintes.

Parte 1

1. Para este exercício serão necessários alguns dados globais que será conveniente definir inicialmente, nomeadamente:
    a. a resistência do **casco** (100 no início);
    b. **velocidade** (10 no início);
    c. o número de naves abatidas **NumNaves** (0 no início).
    d. *Ao longo do exercício deverá adicionar novos dados globais, de acordo com as soluções que implementar*.

2. A primeira função a implementar deverá ter como responsabilidade preparar o ecrã de início de jogo e dar instrução para iniciar o jogo (use a função **carregaElementos()**):
    a. Sugerimos que comece por **esconder** os elementos desnecesários nesta etapa:
        i. Elemento "fim de jogo";

   ii. Naves inimigas (posicionar fora da área de jogo);
   iii. Elemento de "tiro" (posicionar fora da área de jogo);
b. Deverá também mostrar os valores iniciais dos estados da nave nos respetivos indicadores:
   i. a resistência do **casco** no elemento **cascoSpan**;
   ii. o número de naves abatidas no elemento **navesSpan**.
c. Poderá ainda posicionar o elemento Nave na respetiva posição inicial
   i. Horizontal: centrada;
   ii. Vertical: encostada ao fundo do ecrã de jogo.
d. Por fim deverá invocar uma nova função que será responsável iniciar o jogo (use função **iniciarMotorJogo ()**).
e. Ter em atenção que a função *carregaElementos()* deverá ser invocada apenas quando todos os elementos da página estiverem carregados.

3. Conforme indicado a função *iniciarMotorJogo()* deverá ser responsável por iniciar e gerir os mecanismos que controlam o jogo, nomeadamente:
   a. Criar uma dado global que permita indicar qual o estado do jogo (*1 - a jogar; 2 - jogo acabado*). Neste ponto a variável deverá assumir o valor '1';
   b. Criar um contador que seja responsável por invocar (60 fps) uma função responsável por atualizar os elementos no ecrã de jogo (use função **atualizaJogo()**).
   c. De seguida deverá encontrar uma solução para gestão do tempo de jogo. Esse solução deverá ter em conta os seguintes requisitos
      i. O jogo deverá ter uma duração de 15 segundos.
      ii. O tempo disponível deverá decrescer a cada segundo que passa.
      iii. O tempo disponível deverá ser visível no elemento *tempoSpan*.

4. Implementar a função *atualizaJogo()* tendo em conta as seguintes indicações:
   a. Assim que o tempo disponível termine (igual a 0) o jogo deverá terminar, nomeadamente através da invocação da função **fimJogo()**;
   b. Ajustar o mostrador do tempo disponível, por forma a que quando faltarem 10 segundos, a cor da letra passe a vermelho;
   c. Garantir que o contador de tempo não continua a contar atingir o valor 0 e não continua para os números negativos.

5. Implementar a função *fimJogo()* a qual deve:
   a. Atualizar a variável de **estado do jogo** para 2 (jogo acabado);
   b. Apresentar o elemento "fim de jogo";
   c. Calcular e apresentar a pontuação do utilizador (soma do casco, dobro do número de naves abatidas e metade do tempo que ainda falta decorrer até ao fim do jogo);

Parte 2

6. Criar a função **processaTeclas(event)** para gestão e processamento das teclas pressionadas, a qual deve:
    a. Implementar um mecanismo que permita que sempre que uma tecla é pressionada esta função é invocada.
    b. Obter o código da tecla pressionada;
    c. Processar o valor da tecla pressionada, desde que o **estado de jogo** seja 1 (a jogar);
        i. Ler as teclas direcionais do teclado (códigos 37 a 40) e mediante a tecla pressionada invocar a função **deslocaNave(direcao)**.
        ii. Ter em atenção que a função *deslocaNave(direcao)* apresenta um parâmetro para indicação da direção de deslocamento do elemento Nave. *Sugerimos que* ***direcao*** *corresponda ao código da tecla pressionada*.
    d. Implementar a função **deslocaNave(direcao)**, de acordo com as seguintes indicações:
        i. Mediante a direção pretendida o elemento *Nave* deverá ser deslocado com base na ***velocidade*** (variável global) definida inicialmente.
        ii. A deslocação da nave deverá estar limitada horizontalmente (não deve ultrapassar os limites da área de jogo) e verticalmente (entre o fundo do ecrã e 200px acima.
        iii. Ajustar a **velocidade** de acordo com a posição vertical da *Nave*. Quanto mais afastada do fundo, maior deverá ser a velocidade.

Este exercício contínua na próxima aula, sendo por isso fundamental que todos os desafios presentes estejam solucionados, por forma a não comprometer o desenvolvimento do próximo exercício.