Exercício 16b – Space Colector (continuação)

Temática

- Variáveis, expressões, Propriedades
- Estruturas de decisão e estruturas de repetição
- Controlo por teclas e Temporizadores
- Deslocação e Colisão

Na sua pasta de trabalho crie uma nova pasta com o nome “ex15b”, para onde deverá copiar os ficheiros implementados na aula anterior. Nesta pasta devem ser guardados os ficheiros desenvolvidos nesta aula. Substitua também os ficheiros “index.html” e “estilo.css” pelas novas versões fornecidas, faça o mesmo com a pasta “imagens”.

Problema:

Pretende-se concluir a construção do jogo das naves. O jogador poderá controlar os movimentos da nave, por teclas, assim como, acelerar e abrandar (distância para cima). Durante o jogo alguns obstáculos (outras naves) irão aparecer em sentido contrário. O jogador deverá evitar as naves inimigas ou abatê-las com o seu poder de fogo.

Com base no código javascript desenvolvido na aula anterior, desenvolva o necessário para implementar as alíneas seguintes.

A versão resolvida pode ser consultada no seguinte URL:
http://labmm.clients.ua.pt/LM3/LM3-p/ex16b/

Parte 1

1. Criar a função **deslocaNavesInimigas()** a qual deve, para cada uma das naves inimigas:
    a. Deslocar verticalmente, de acordo com a velocidade pré-definida globalmente para as naves inimigas;
    b. Assim que a nave ultrapassar o limite inferior da área de jogo, deverá ser reposicionada:
        i. Verticalmente: imediatamente antes do topo da área de jogo;
        ii. Horizontalmente: aleatoriamente dentro dos limites horizontais da área de jogo.
        iii. Deverá garantir que a posição horizontal coloca a nave totalmente na área de jogo, evitando situações em que apenas parte da nave é visível.

c. Adicionar, na função *atualizaJogo(),* uma invocação à função *deslocaNavesInimigas(),* para que este seja invocada repetidamente resultando no efeito de deslocação das naves inimigas;

d. A velocidade das naves inimigas deverá ser ajustada de acordo com o posicionamento vertical da Nave do jogador (acelerar quando a Nave estiver mais subida no ecrã, desacelerar até à velocidade normal quando a Nave do Jogador estiver no fundo do ecrã).

e. O mecanismo de colisão será adicionado mais à frente no exercício.

2. Criar a função **ativaTiro()** a qual deverá ser responsável por verificar, sempre que a tecla espaço for pressionada, se:

   a. Existe ou não algum tiro ativo (tiro ativo corresponde a um tiro em deslocação na área de jogo):

   b. Caso **não** exista tiro ativo, deverá ser colocado um tiro na mesma localização da nave do jogador;

   c. A deslocação do Tiro deverá ser tratada na função seguinte.

3. Criar função **moverTiro()** que deverá ser responsável pela deslocação do tiro na área de jogo:

   a. A velocidade de deslocação do tiro deverá ser igual a da nave do jogador;

   b. Quando o tiro atingir o limite superior da área de jogo, deverá ser possível disparar novo tiro,

   c. A função atual deverá ser invocada repetidamente, por forma a criar o movimento do tiro, **sempre que exista um tiro ativo**. Em que ponto do código atual deverá ser adicionada?

Parte 2

Nesta parte do exercício iremos abordar o mecanismo de colisão entre elementos. Propõem-se a criação de uma função única que permita verificar a colisão de dois elementos distintos, sendo que ao longo do jogo esta função será utilizada para verificar a colisão entre a Nave do Jogador e as naves inimigas, bem como entre o tiro e as naves inimigas.

4. Criar a função que verifica a colisão entre dois elementos **verificaColisao (elemento1, elemento1Altura, elemento1Largura, elemento2, elemento2Altura, elemento2Largura)**:

   a. Verificar colisão corresponde a verificar se existe intersecção na vertical e horizontal entre ambos os elementos (e.g. Nave e nave Inimiga, Tito e Nave Inimiga)

   b. A função atual deverá responder TRUE caso exista colisão entre os dois elementos

5. Agora que existe uma função para verificar colisões, a mesma deverá ser invocada a partir de um ponto estratégico do jogo que permita aferir colisões entre a Nave ou o Tiro com as Naves Inimigas.
    a. Sugerimos que o ponto de invocação seja no fim da função **deslocaNavesInimigas()**. Está de acordo ou na sua ótica deveria ser num momento diferente?
    b. Deverá então ser invocada a função **verificaColisao(),** sendo que os elementos a passar seriam: Nave do Jogador e cada uma das Naves Inimigas. Em caso de colisão:
        i. O valor do "casco" deve ser decrementado 10 valores;
        ii. Se o valor do "casco" foi igual a zero o jogo deve terminar.
        iii. A nave inimiga reposicionada no topo do ecrã;
    c. No caso de existir um Tiro ativo, deverá ser invocada a função **verificaColisao()**, sendo que os elementos a passar seriam: Tiro e cada uma das Naves Inimigas. Em caso de colisão:
        i. O número de naves abatidas deve ser incrementado;
        ii. O tiro deve desaparecer;
        iii. A nave inimiga reposicionada imediatamente antes de entrar no topo do ecrã.

Avançada

1. Desenvolva um sistema de *parallax* recorrendo para isso às *divs* 'star-field1', 'star-field2' e 'star-field3'. Todas se devem deslocar na vertical, com velocidades distintas, simulando o efeito de profundidade;
2. Adapte o efeito de *parallax* de modo a que, quando a nave se desloca, também as estrelas efetuam uma ligeira deslocação horizontal.
3. Deverá proceder ao ajuste do mecanismo de processa teclas, de modo a proporcionar uma deslocação mais fluída da nave do jogador. Alguma ideia? E que tal utilizando em conjunto com o *onkeydown* o evento de *onkeyup*?