Programando Jogos em Delphi - Parte 3

Animação, Lógica e Controle
versão orientada à objetos

Antônio Sérgio de S. Vieira

[1]Inimigos - 19/05/2011

sergiosvieira@hotmail.com

1. Inimigos

Dando continuidade ao desenvolvimento de um jogo de nave completo em Delphi, nesta apostila veremos como adicionar alguns inimigos. Será criada mais uma classe para representar a nave inimiga (*TNaveInimiga*) e daremos a ela um movimento todo especial utilizando a função cosseno.

De forma semelhante as classes criadas nas apostilas anteriores, nesta, também serão incluídas variáveis de posicionamento é métodos para mover e desenhar a nave inimiga.

Como a nave utilizará cosseno para movimentação, a primeira diferença é que em vez de utilizar variáveis do tipo *Integer* utilizaremos variáveis *Double*, assim, poderemos utilizar valores "quebrados" para posicionamento. Porém, quando formos desenhar a nave na tela com *offscreen_.Canvas.Pixels*, teremos que arredondar as coordenadas com o comando *Round*.

O cosseno é bastante interessante para movimentar as naves inimigas, isto ocorre por conta de seu comportamento periódico. Utilizando esta função para movimentar a nave, em vez de apenas incrementar com a velocidade da nave a coordenada y, faremos com que a coordenada x seja incrementada com um valor de cosseno de *alpha*, onde *alpha* é uma variável *Double* que cresce a cada nova iteração do laço principal do jogo.

Para exemplificar melhor a influência do cosseno na coordenada x, veja alguns valores calculados abaixo:

cos(alpha)	resultado
cos(0.00)	= 1.000000
cos(0.68)	= 0.777573
cos(1.36)	= 0.209239
cos(2.04)	= -0.452176
cos(2.72)	= -0.912438
cos(3.4)	= -0.966798
cos(4.08)	= -0.591073
cos(4.76)	= 0.047593
cos(5.44)	= 0.665088
cos(6.12)	= 0.986715

Observando a tabela anterior, percebe-se que a medida que *alpha* é incrementado o resultado torna-se positivo, negativo e novamente positivo. Isto faz com que

a nave mova-se para direita (positivo), para a esquerda (negativo) e novamente para a direita (positivo). Esta periodicidade continua indefinidamente e ela faz com que a nave inimiga movimente-se zigue-zagueando enquanto desce pela tela (Figura 1).

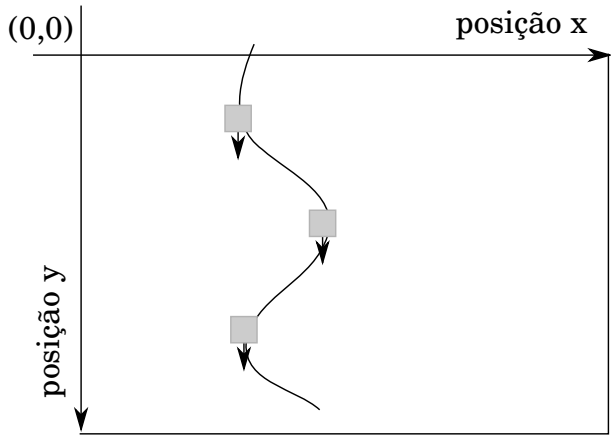

Figura 1. Movimento em zigue-zague da nava inimiga

2. Passo-a-Passo

1. abra o aquivo Unit1.pas modificado na apostila passada, logo depois da classe *TEstrela* adicione a classe *TNaveInimiga*:

```
TNaveInimiga = class
private
  alpha: Double;
  velocidade: Double;
  color: TColor;
public
  x, y: double;
  constructor create(color_: TColor; velocidade_: Double);
  procedure mover();
  procedure desenhar(offscreen_: TBitmap);
end;
```

   As novidades dessa classe são: coordenadas *Double* e a cor da nave inimiga que é escolhida através do construtor da classe.

2. Adicione o construtor da classe *TNaveInimiga* como mostrado abaixo:

```
constructor TNaveInimiga.create(color_: TColor; velocidade_: Double);
var i: Integer;
begin
  color := color_;
  velocidade := velocidade_;
  x := RandomRange(1, 640);
  y := RandomRange(1, 480) * -1;
  alpha := RandG(0.5, 0.8);
end;
```

   De forma semelhante a classe *TEstrela*, no construtor da classe *TNaveInimiga*, as coordenadas *x* e *y* são geradas aleatoriamente, lembrando que *y* é multiplicado por -1 para que a nave surja na parte superior não-visível da tela. A

novidade fica por conta da função *RandG* que retorna um *Extend* (valor "quebrado") dentro do intervalo $(0.5, 0.8)$, isto é feito para que as naves criadas não iniciem seu movimento na mesma direção.

3. Em seguida adicione o método *mover* como exibido abaixo:

```
procedure TNaveInimiga.mover;
var i: Integer;
begin
  alpha := alpha + 0.01;
  y := y + velocidade;
  x := x + cos(alpha);
  if (y + 16 >= 480) then
  begin
    x := RandomRange(1, 640);
    y := RandomRange(1, 480) * -1;
  end;
end;
```

4. Insira o código do método *desenhar* da seguinte forma:

```
procedure TNaveInimiga.desenhar(offscreen_: TBitmap);
var i, j: Integer;
begin
  for i:= 0 to 15 do
    for j := 0 to 15 do
      offscreen_.Canvas.Pixels[Round(x) + i, Round(y) + j] := color;
end;
```

   Será desenhado um quadrado de 16x16 *pixels* e deve-se atentar ao comando *Round* agora necessário, pois, x e y agora são do tipo *Double* enquanto que *Pixels* só aceita valores inteiros.

5. Para adicionar as naves inimigas ao jogo crie um *Array* de naves inimigas e as instancie como exibido abaixo:

```
procedure TForm1.FormActivate(Sender: TObject);
var
  estrelas: Array[1..100] of TEstrela;
  naves_inimigas: TANave;
  i: Integer;
begin
  for i := 1 to 100 do estrelas[i] := TEstrela.create(clWhite, 1);
  for i := 1 to 20 do naves_inimigas[i] := TNaveInimiga.create(clWhite, 0.3);
(...)
```

   O tipo `TANave` deve ser colocado logo abaixo da classe *TNaveInimiga* da seguinte forma *TANave = Array[1..20] of TNaveInimiga;*. O uso desse tipo ficará mais claro na próxima apostila. Por enquanto apenas o use.

6. A última coisa a fazer é desenhar e mover as naves inimigas. Veja como fica o novo laço principal:

```
(...)
while not application.Terminated do
  begin
    OffScreen.Canvas.Brush.Color := clBlack;
    OffScreen.Canvas.FillRect(Rect(0, 0, 640, 480));
    for i := 1 to 100 do
    begin
```

```
        estrelas[i].desenhar(OffScreen);
        estrelas[i].mover();
      end;
      for i := 1 to 20 do
      begin
        naves_inimigas[i].desenhar(OffScreen);
        naves_inimigas[i].mover();
      end;
  (...)
```

7. Aperte F9 pra mais uma versão do jogo. :)

3. Próxima Apostila

Na próxima apostila será explicado como realizar detecção de colisão entre a nave inimiga e o tiro da nave. Qualquer dúvida ou sugestão, ou mesmo se você encontrar algum erro nesta apostila, envie-me um email. Boa sorte!!!