Curso: Desenvolvendo Jogos 2d Com C# E Microsoft XNA

Conteudista: André Luiz Brazil

Aula 7: ACRESCENTANDO TIROS E UM CENÁRIO ESPACIAL AO SEU JOGO

META

Atirar com a espaçonave e movimentar o cenário de fundo do jogo.

OBJETIVOS

Ao final da aula, você deve ser capaz de:

1. Atirar com a espaçonave do seu projeto de jogo;
2. Movimentar o cenário de fundo do seu projeto de jogo de acordo com o movimento da espaçonave.

PRÉ-REQUISITOS

1. Estar familiarizado com a ferramenta XNA Game Studio, conceito abordado na aula 2;
2. Possuir um computador com as ferramentas Visual C# e XNA Game Studio instaladas, conforme explicado na aula 3**;**
3. Reconhecer os conceitos de classe e objeto, abordados na aula 4;
4. Ter o seu projeto de jogo com a ferramenta Visual C# atualizado para exibir e movimentar a espaçonave do jogador, conforme visto na aula 6.

Introdução

Na aula anterior, aprendemos como exibir e movimentar as espaçonaves dentro do jogo. Agora podemos acrescentar alguns detalhes para tornar o jogo mais interessante. Que tal fazermos a espaçonave do jogador a atirar? Assim ela poderá enfrentar os seus inimigos, os soldados do espaço.



Figuras 7.1, 7.2 e 7.3 – Exemplos de tiros

Fontes: (Jefferson, favor redesenhar)

- http://7scgng.bay.livefilestore.com/y1p_W-Q3sq-YXBc0mC46U3IozorrStF0WkUsHPp4QMt4BaL9ysP-UrvpXJ6et4gcfxUhyFiMdF3OJg
- http://www.af.mil/shared/media/ggallery/hires/AFG-071127-003.jpg
- http://img511.imageshack.us/img511/5111/66564696aj6.jpg

Figura 7.4 – Robô de XNA

Fonte: Eduweb

Veja três características que o tiro possui:

- Potência
- Direção
- Velocidade

Além disso, precisamos armazenar informações sobre a localização e o visual do tiro, para fazer o tiro aparecer dentro da tela do nosso jogo.

Criação da classe Tiro

Lembra da classe Espaçonave, que criamos na Aula 4? Vamos agora criar uma uma classe para os tiros disparados no jogo. Basta seguir os passos abaixo:

Passo1: Criar os atributos e o método de criação da classe **Tiro**.

A **classe Tiro** possuirá os seguintes **atributos: Potência, Direção, Velocidade, Posição e Imagem**.

Veja como ficou a nossa classe Tiro:

```
public class Tiro
{
        private int _potencia;
        private int _direcao;
        private int _velocidade;
        private int _posicaox;
        private int _posicaoy;
        private Texture2D _textura;

        public Tiro(int p_potencia, int p_direcao, int p_velocidade, int
         p_posicaox, int p_posicaoy)
        {
            _potencia = p_potencia;
            _direcao = p_direcao;
            _velocidade = p_velocidade;
            _posicaox = p_posicaox;
            _posicaoy = p_posicaoy;
        }
}
```

Repare no método **Tiro**, que é o método de criação da classe **Tiro**. Observe que ao contrário da espaçonave, que não precisava receber nenhuma informação especial durante a sua criação, cada tiro requer as seguintes informações no momento da sua criação: potência do tiro (**p_potencia**), direção do tiro (**p_direcao**), velocidade do tiro (**p_velocidade**) e posição inicial do tiro (**p_posicaox** e **p_posicaoy**).

Passo 2: Criar o método **Mover** para a classe **Tiro**.

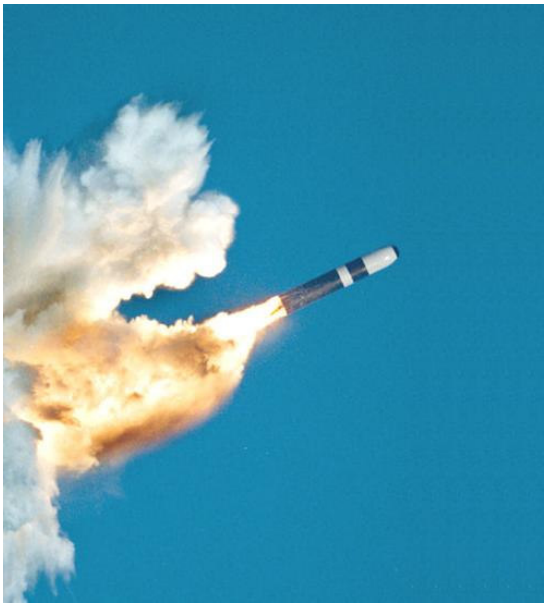

Figura 7.5 – O tiro em movimento

Fonte: http://www.defensetech.org/images/Trident_missile_image.jpg (Jefferson, favor redesenhar)

Assim como a espaçonave, o tiro também não fica parado. Tente identificar quais ações um tiro vai desempenhar no jogo:

- Mover;
- Explodir.

Repare que nem todos os tiros irão explodir. Desta forma, vamos nos concentrar primeiro em criar um método para movimentar os tiros: o método Mover da classe Tiro. Ele é uma versão simplificada do método **Mover** da classe **Espaçonave**.

Observe que o tiro possui sua própria direção. Desta forma, o método mover da classe Tiro não receberá a direção como parâmetro. Também não precisaremos verificar se o tiro "saiu da tela", como fazíamos no método **Mover** da classe **Espaçonave**. Veja como ficou:

```
public void Mover()
{
    // Variáveis temporárias
    int deslocamento_x, deslocamento_y;
    deslocamento_x = 0;
    deslocamento_y = 0;

    // Deslocamento para cima
    if (_direcao == 1)
    {
        deslocamento_y = - _velocidade;
    }

    // Deslocamento para a direita
    if (_direcao == 2)
```

```
    {
        deslocamento_x = + _velocidade;
    }

    // Deslocamento para baixo
    if (_direcao == 3)
    {
        deslocamento_y = + _velocidade;
    }

    // Deslocamento para a esquerda
    if (_direcao == 4)
    {
        deslocamento_x = - _velocidade;
    }

    // Executando o movimento
    _posicaox = _posicaox + deslocamento_x;
    _posicaoy = _posicaoy + deslocamento_y;
}
```

Passo 3: Criar os demais métodos da classe Tiro

Perceba que o tiro, além de se mover, também precisa ser exibido em uma determinada posição da tela do jogo. Para que isso aconteça, vamos construir os seguintes métodos:

- **PosicaoX** e **Posicao Y**: Apenas para retornar a posição atual (coordenadas x e y) do tiro;
- **CarregarTextura**: Assim como na classe espaçonave, este método serve para atribuir uma imagem ao tiro;
- **Textura**: Apenas para retornar a imagem atual que foi associada ao tiro;
- **Desenhar**: Para fazer o tiro aparecer na tela do jogo.

Veja a seguir como foram construídos estes métodos:

```
public int PosicaoX()
{
    return _posicaox;
}

public int PosicaoY()
{
    return _posicaoy;
}

public void CarregarTextura(GraphicsDevice p_dispositivo, string
 p_local)
{
    _textura = Texture2D.FromFile(p_dispositivo, p_local);
}

public Texture2D Textura()
{
    return _textura;
}
```

```
        public void Desenhar(SpriteBatch p_sprite)
        {

            p_sprite.Begin();
            p_sprite.Draw(_textura, new Rectangle(_posicaox, _posicaoy,
              _textura.Width, _textura.Height), Color.White);
            p_sprite.End();
        }
```

Pronto! A nossa classe **Tiro** já está completa! Observe que os métodos **CarregarTextura** e **Desenhar** são idênticos aos da classe **Espaçonave**.

Atividade Prática 1 – Atende ao Objetivo 1

Seguindo os três passos descritos anteriormente, abra o seu projeto de jogo espacial na ferramenta Visual C# e crie o código da classe **Tiro** dentro do seu projeto de jogo.

Fim da Atividade Prática 1

Acrescentando os tiros ao jogo

Agora que a classe **Tiro** já foi construída, vamos ver como incorporar os tiros ao nosso projeto de jogo.

Pense sobre este assunto e analise as seguintes questões:

- Quem vai controlar os tiros?
- Quem irá disparar os tiros?
- Os tiros existirão para sempre?

Encontrou as respostas? Vamos a elas então:

Questão 1: Quem vai controlar os tiros?

Resposta: O jogo. Vamos criar dentro do jogo uma lista para guardar todos os tiros que já foram disparados. Assim poderemos controlar a movimentação dos tiros no jogo.

```
namespace JogoEspacial
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        // Definindo a espaçonave do jogador no jogo
        Espaconave NaveJogador;
```

```
// Definindo um leitor de teclas do teclado
KeyboardState teclado;

// Criando uma lista de tiros disparados no jogo
List<Tiro> TirosDisparados;
```

Para criar uma lista, usamos o comando **List<tipo>**, onde **<tipo>** define quais são os elementos que farão parte desta lista. No nosso caso criamos uma lista de elementos do tipo **Tiro** e chamamos a lista de **TirosDisparados**.

Precisamos também inicializar a lista de tiros disparados no método de inicialização (**Initialize**) do jogo:

```
protected override void Initialize()
{
    // Criando efetivamente a espaçonave do jogador
    NaveJogador = new Espaconave();

    // Colocando um nome para a nave do jogador
    NaveJogador.Nome("Falcão Justiceiro");

    // Inicializando a lista de tiros do jogo
    TirosDisparados = new List<Tiro>();

    base.Initialize();
}
```

Repare que ainda falta controlarmos a movimentação de todos os tiros que existem no jogo. Podemos fazer isso no método de atualização do jogo (**Update**). Para isso, precisaremos movimentar cada um dos tiros do jogo percorrendo a lista de tiros disparados. Veja como:

```
protected override void Update(GameTime gameTime)
{
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
                ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here
            teclado = Keyboard.GetState();
            if (teclado.IsKeyDown(Keys.Up))
            {
                NaveJogador.Mover(1, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Right))
            {
                NaveJogador.Mover(2, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Down))
            {
                NaveJogador.Mover(3, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Left))
            {
```

```
            NaveJogador.Mover(4, this.Window.ClientBounds.Width,
            this.Window.ClientBounds.Height);
        }

        // Movendo os tiros do jogo
        foreach (Tiro oTiro in
            TirosDisparados.GetRange(0,TirosDisparados.Co
            unt))
        {
            oTiro.Mover();
        }

        base.Update(gameTime);
}
```

Perceba que utilizamos um novo comando para percorrer a lista de tiros, o commando **foreach**. Para cada tiro na lista de tiros disparados, chamamos o método mover desse tiro (**oTiro.Mover**) para que ele atualize a sua posição e se movimente pela tela do jogo.

O método de exibição do jogo (**Draw**) também precisa ser atualizado. Acrescente uma instrução para desenhar cada um dos tiros do jogo, de forma semelhante ao que fizemos anteriormente para movimentar os tiros:

```
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.Black);

    // TODO: Add your drawing code here
    NaveJogador.Desenhar(spriteBatch);

    // Desenhando os tiros do jogo
    foreach (Tiro oTiro in TirosDisparados)
    {
        oTiro.Desenhar(spriteBatch);
    }

    base.Draw(gameTime);
}
```

Questão 2: Quem irá disparar o tiro?

Resposta: A espaçonave, quando o jogador apertar algum botão.

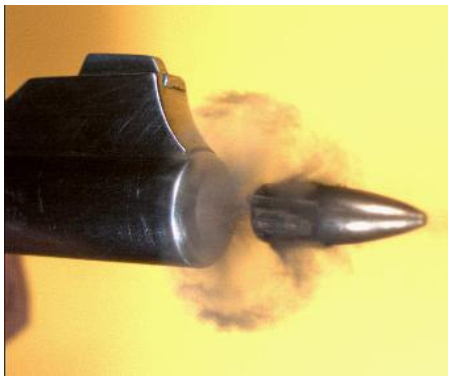

Figura 7.6 – O ato de atirar

Fonte: http://blog.wired.com/photos/uncategorized/2007/06/27/bullet_from_revolver_1.jpeg (Jefferson, favor redesenhar)

Certo! Vamos então programar o método **Atirar** da classe **Espaçonave** para que ela produza os tiros.

```
public void Atirar(GraphicsDevice p_dispositivo_grafico, string
   p_local_textura_tiro, List<Tiro> p_lista_tiros)
{
      // Cria um tiro
      Tiro oTiro;
      oTiro = new Tiro(_potencia_tiro, 1, _velocidade_tiro, _posicaox,
   _posicaoy);

      // Carrega a textura do tiro
      oTiro.CarregarTextura(p_dispositivo_grafico, p_local_textura_tiro);

      // Acrescenta o tiro na lista de tiros do jogo
      p_lista_tiros.Add(oTiro);
}
```

Observe que o método **Atirar** recebe alguns parâmetros importantes:

- **p_dispositivo_grafico** - O dispositivo gráfico do jogo, onde será exibido o tiro;
- **p_local_textura_tiro** – O local onde se encontra a imagem que será usada para o tiro;
- **p_lista_tiros** – A lista de tiros disparados no jogo.

Em seguida, ele cria um tiro chamado **oTiro** e na sua criação define a potência (**_potencia_tiro**), a direção (**1**), a velocidade (**_velocidade_tiro**) e a posição inicial do tiro (**_posicaox** e **_posicaoy**). Repare que a posição inicial do tiro é a mesma da espaçonave que atirou.

Por fim, ele carrega a imagem do tiro como textura utilizando o método **oTiro.CarregarTextura** e adiciona o tiro criado à lista de tiros do jogo usando o comando **Add**.

Adicione também à sua classe Espaçonave um valor inicial para a potência do tiro. Isto pode ser feito ajustando o método de criação da classe Espaçonave:

```
public Espaconave()
{
    _energia = 50;
    _velocidade = 5;
    _velocidade_tiro = 5;
    _posicaox = 10;
    _posicaoy = 10;
    _potencia_tiro = 10;
}
```

Ainda falta um detalhe: O tiro da espaçonave do jogador só é disparado quando o jogador aperta algum botão.

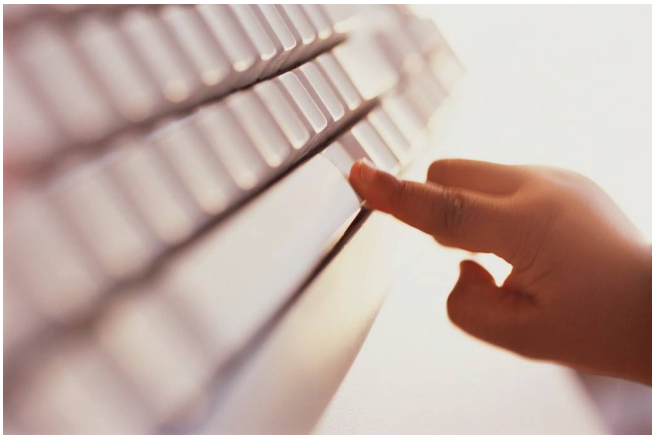

Figura 7.7 – Pressionando a barra de espaços

Fonte: http://intuicare.com/images/photos/keypress.jpg (Jefferson : favor redesenhar)

Vamos configurar a barra de espaços para ser o nosso botão de disparar os tiros. Isto pode ser feito dentro do método de atualização do jogo (**Update**). Observe:

```
protected override void Update(GameTime gameTime)
{
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
                ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here
            teclado = Keyboard.GetState();
            if (teclado.IsKeyDown(Keys.Up))
            {
                NaveJogador.Mover(1, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Right))
            {
                NaveJogador.Mover(2, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Down))
            {
                NaveJogador.Mover(3, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Left))
            {
                NaveJogador.Mover(4, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            //Apertou o tiro
            if (teclado.IsKeyDown(Keys.Space))
            {
                NaveJogador.Atirar(graphics.GraphicsDevice,
                "../../../Content/Imagens/tiro.png",
                TirosDisparados);
```

```
            }

            // Movendo os tiros do jogo
            foreach (Tiro oTiro in TirosDisparados)
            {
                oTiro.Mover();
            }

            base.Update(gameTime);
}
```

Utilizamos o comando **IsKeyDown(Keys.Space)** para verificar se a barra de espaços foi pressionada. Caso seja, chamamos o método **Atirar** da espaçonave do jogador, passando para ele o dispositivo gráfico do jogo, o local da imagem do tiro e a lista de tiros do jogo.

Repare também que utilizamos a imagem **tiro.png** para os tiros da espaçonave do jogador. Este arquivo de imagem precisa ser acrescentado ao projeto de jogo, da mesma forma como fizemos na Aula 6 para a imagem da espaçonave do jogador.

Questão 3: Os tiros existirão para sempre?

Resposta: Não. É importante retirarmos da lista todos os tiros que já "saíram da tela do jogo".

Para limpar a lista de tiros, mantendo nela apenas os tiros que ainda estão dentro da tela do jogo, podemos acrescentar um teste ao método de atualização do jogo (**Update**), para verificarmos se cada um dos tiros já ultrapassou os limites da tela e removê-los da lista de tiros do jogo. Veja como ficou:

```
protected override void Update(GameTime gameTime)
{
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
                ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here
            teclado = Keyboard.GetState();
            if (teclado.IsKeyDown(Keys.Up))
            {
                NaveJogador.Mover(1, this.Window.ClientBounds.Width,
                 this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Right))
            {
                NaveJogador.Mover(2, this.Window.ClientBounds.Width,
                 this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Down))
            {
                NaveJogador.Mover(3, this.Window.ClientBounds.Width,
                 this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Left))
            {
```

```
                NaveJogador.Mover(4, this.Window.ClientBounds.Width,
                 this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Space)) //Apertou o tiro
            {
                NaveJogador.Atirar(graphics.GraphicsDevice,
                 "../../../Content/Imagens/tiro.png", TirosDisparados);
            }

            // Movendo os tiros do jogo
            foreach (Tiro oTiro in TirosDisparados.GetRange(0,TirosDisparados.
                     Count))
            {
                oTiro.Mover();

               // Remove os tiros que saíram da tela do jogo
               if (oTiro.PosicaoX() < 1 ||
                   oTiro.PosicaoY() < 1 ||
                   oTiro.PosicaoX() >
                    this.Window.ClientBounds.Width - 1 ||
                   oTiro.PosicaoY() >
                    this.Window.ClientBounds.Height - 1)
                  {
                        TirosDisparados.Remove(oTiro);
                  }
            }
}
```

Repare que colocamos a verificação logo após a movimentação do tiro (**oTiro. Mover**). Testamos se a posição do tiro (coordenadas x e y) são menores que 1 ou maiores que a largura (**this.Window.ClientBounds.Width**) ou a altura (t**his.Window. ClientBounds.Height**) da tela. Neste caso, retiramos o tiro da lista de tiros do jogo através do comando **Remove(oTiro)**.

Atividade Prática 2 – Atende ao Objetivo 1

Seguindo as respostas dadas para as três questões levantadas, abra o seu projeto de jogo espacial na ferramenta Visual C# e incorpore a criação de tiros ao seu jogo, de forma que eles sejam criados sempre que o jogador pressionar a barra de espaços.

Fim da Atividade Prática 2

Atividade Prática 3 – Atende ao Objetivo 1

Existe uma pequena falha no jogo que faz com que vários tiros sejam criados de uma só vez quando o jogador aperta apenas uma vez a barra de espaços. Ajuste o código do jogo de forma que cada vez que o jogador aperte a barra de espaços, apenas um único tiro seja gerado.

Dicas:

- Utilize o comando **IsKeyUp(Keys.Space)** para verificar quando a barra de espaços deixou de ser apertada;
- Crie no jogo um atributo do tipo verdadeiro/falso (**boolean**), para controlar exatamente quando a barra de espaços já foi pressionada ou solta;

Fim da Atividade Prática 3

Atividade Prática 4 – Atende ao Objetivo 1

Existe um outro pequeno problema no jogo. Quando o tiro é criado, ele sai do canto esquerdo da espaçonave. Modifique o código do jogo para fazer com que o tiro saia do meio da espaçonave.

Dicas:

- Crie na classe **Tiro** os métodos para atualizar a posição do tiro (coordenadas x e y);
- Utilize o comando **_textura.Width** para obter a largura da textura da espaçonave.

Fim da Atividade Prática 4

Acrescentando um cenário de fundo ao jogo

Vejamos então agora, como acrescentar um cenário de fundo móvel ao jogo, de forma a dar uma sensação de movimento constante da nave. Para tal, siga os passos descritos a seguir:

Passo 1: Acrescentar dois novos atributos ao nosso jogo:

- Uma textura para usar como cenário de fundo, que chamaremos de **cenário_fundo**;
- A posição vertical atual deste cenário de fundo do jogo, que vamos chamar de **posicaoy_cenario_fundo**;

Observe agora os atributos do jogo:

```
public class Game1 : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;

    // Definindo a espaçonave do jogador no jogo
    Espaconave NaveJogador;

    // Definindo um leitor de teclas do teclado
    KeyboardState teclado;

    // Criando uma lista de tiros disparados no jogo
    List<Tiro> TirosDisparados;
```

```
// Atributos do cenário de fundo do jogo
Texture2D cenario_fundo;
int posicaoy_cenario_fundo;
```

Passo 2: Alterar o método de inicialização do jogo (**Initialize**) para configurar uma posição vertical inicial para o cenário de fundo do jogo. Inicialmente, colocaremos o valor zero, para que o fundo seja projetado inicialmente cobrindo toda a tela, do início ao fim.

```
protected override void Initialize()
{
    // Criando efetivamente a espaçonave do jogador
    NaveJogador = new Espaconave();

    // Colocando um nome para a nave do jogador
    NaveJogador.Nome("Falcão Justiceiro");

    // Inicializando a lista de tiros do jogo
    TirosDisparados = new List<Tiro>();

    // Definindo a posição inicial do cenário de fundo
    posicaoy_cenario_fundo = 0;

    apertou_tiro = false;

    base.Initialize();
}
```

Passo 3: Ajustar no método de carregamento do conteúdo (**LoadContent**), para carregar a textura do cenário de fundo do jogo:

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // TODO: use this.Content to load your game content here
    NaveJogador.CarregarTextura(graphics.GraphicsDevice,
    "../../../Content/Imagens/nave.PNG");

    // Carregando o cenário de fundo do jogo
    cenario_fundo = Texture2D.FromFile(
     graphics.GraphicsDevice,
     "../../../Content/Imagens/espaco.jpg");
}
```

Repare também que utilizamos a imagem **espaco.jpg** como o cenário de fundo do jogo. Este arquivo de imagem precisa ser acrescentado ao projeto de jogo da mesma forma como fizemos na Aula 6, para a imagem da espaçonave do jogador.

Passo 4: Mover o cenário de fundo do jogo conforme a movimentação da espaçonave. Para tal, vamos atualizar constantemente a posição y do cenário de fundo no método de atualização do jogo (**Update**). Desta forma, cada vez o cenário de fundo

será desenhado numa posição vertical diferente da anterior, trazendo a sensação de movimento. Veja:

```
protected override void Update(GameTime gameTime)
{
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
                ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here
            teclado = Keyboard.GetState();
            if (teclado.IsKeyDown(Keys.Up))
            {
                NaveJogador.Mover(1, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Right))
            {
                NaveJogador.Mover(2, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Down))
            {
                NaveJogador.Mover(3, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Left))
            {
                NaveJogador.Mover(4, this.Window.ClientBounds.Width,
                this.Window.ClientBounds.Height);
            }
            if (teclado.IsKeyDown(Keys.Space) && !apertou_tiro) //Apertou o tiro
            {
                NaveJogador.Atirar(graphics.GraphicsDevice,
                "../../../Content/Imagens/tiro.png", TirosDisparados);
                apertou_tiro = true;
            }
            if (teclado.IsKeyUp(Keys.Space) && apertou_tiro)
            {
                apertou_tiro = false;
            }

            // Movendo os tiros do jogo
            foreach (Tiro oTiro in
                TirosDisparados.GetRange(0,TirosDisparados.Count))
            {
                oTiro.Mover();

                // Remove os tiros que saíram da tela do jogo
                if (oTiro.PosicaoX() < 1 ||
                    oTiro.PosicaoY() < 1 ||
                    oTiro.PosicaoX() > this.Window.ClientBounds.Width - 1 ||
                    oTiro.PosicaoY() > this.Window.ClientBounds.Height - 1)
                {
                    TirosDisparados.Remove(oTiro);
                }
            }
```

```
// Movimentando o cenário de fundo do jogo
posicaoy_cenario_fundo += NaveJogador.Velocidade() / 2;
if (posicaoy_cenario_fundo >= this.Window.ClientBounds.
    Height) posicaoy_cenario_fundo = 0;
```

```
}
```

Repare que existe um pequeno teste que faz a posição y do cenário de fundo retornar a zero quando o valor desta excede a altura da tela de jogo.

Passo 5: Ajustar o método de desenhar do jogo (Draw), para desenharmos a textura do cenário de fundo do jogo antes de todos os outros objetos, de forma que esta fique por baixo das outras texturas. Utilizaremos a posição y definida para a textura de cenário de fundo como posição inicial. Veja como foi feito:

```
protected override void Draw(GameTime gameTime)
{
      GraphicsDevice.Clear(Color.Black);

      // Desenhando o cenário de fundo do jogo
      spriteBatch.Begin();
      spriteBatch.Draw(cenario_fundo, new Rectangle(0,
       posicaoy_cenario_fundo,
       this.Window.ClientBounds.Width,
       this.Window.ClientBounds.Height), Color.White);
      spriteBatch.Draw(cenario_fundo, new Rectangle(0,
       posicaoy_cenario_fundo -
       this.Window.ClientBounds.Height,
       this.Window.ClientBounds.Width,
       this.Window.ClientBounds.Height), Color.White);
      spriteBatch.End();

      // TODO: Add your drawing code here
      NaveJogador.Desenhar(spriteBatch);

      // Desenhando os tiros do jogo
      foreach (Tiro oTiro in TirosDisparados)
      {
          oTiro.Desenhar(spriteBatch);
      }

      base.Draw(gameTime);
}
```

Observe que o cenário de fundo é desenhado duas vezes! O segredo consiste em desenhar uma segunda vez o mesmo cenário de fundo do jogo, posicionando-o de forma a completar o buraco deixado pelo primeiro desenho, devido ao aumento da posição y.

A primeira vez que o cenário de fundo é desenhado, é utilizada a posição y para definir o ponto inicial do desenho. Já na segunda vez, utilizamos como ponto inicial a posição y menos a altura da tela (**this.Window.ClientBounds.Height**). Isto fará com que o mesmo cenário de fundo seja desenhado duas vezes, preenchendo todo o espaço da tela e dando uma sensação de movimento a cada momento do jogo, já que os cenários de fundo estarão sempre sendo desenhados com base numa posição y diferente.

Figura 7.8 - Tela do jogo com tiros e cenário de fundo

Fonte: Visual C#

Atividade Prática 5 – Atende ao Objetivo 2

Seguindo os cinco passos descritos anteriormente, abra o seu projeto de jogo espacial na ferramenta Visual C# e altere-o para carregar um cenário de fundo ao jogo.

Fim da Atividade Prática 5

Atividade Prática 6 – Atende ao Objetivo 2

Agora faça com que o cenário de fundo do jogo se mova na posição horizontal ao invés da vertical. **Dica:** Utilize a instrução **this.Window.ClientBounds.Height** para obter a altura da tela do jogo.

Fim da Atividade Prática 6

CAIXA DE FÓRUM **Informação sobre Fórum**

Figura 7.9
Fonte: http://www.stockxpert.com/browse_image/view/28331341/?ref=sxc_hu (Jefferson- favor redesenhar)

Você teve alguma dificuldade para adicionar tiros ou o cenário de fundo ao jogo? Entre no fórum da semana e compartilhe suas dúvidas e experiências com os seus amigos.

FIM DE CAIXA DE FÓRUM

CAIXA DE ATIVIDADE **Informação sobre Atividade on-line**

Figura 7.10
Fonte: http://www.sxc.hu/photo/1000794 (Jefferson : favor redesenhar)

Agora que você já está com o seu código de projeto do jogo ajustado para adicionar tiros e o cenário de fundo ao jogo, vá à sala de aula virtual e resolva as atividades propostas pelo tutor.

FIM CAIXA DE ATIVIDADE

Resumo

- Da mesma forma que você criou uma classe para as suas espaçonaves, precisamos criar uma classe para todos os tiros disparados no jogo, a classe **Tiro.** Ela será composta pelos atributos potência, direção, velocidade, posição e imagem e os métodos **Mover, CarregarTextura, Desenhar, Textura, PosicaoX** e **PosicaoY**;
- Observe que o tiro possui sua direção própria. Desta forma, o método mover da classe Tiro não receberá a direção como parâmetro. Também não precisaremos verificar se o tiro "saiu da tela", como fazíamos no método **Mover** da classe **Espaçonave**;

- Precisaremos criar uma lista para controlar os tiros que foram disparados dentro do jogo, para que possamos exibi-los e movimentá-los corretamente com o decorrer do tempo de jogo;
- Será necessário programar o método **Atirar** da classe **Espaçonave**, para que ela produza os tiros e os acrescente na lista de tiros do jogo;
- É importante acrescentarmos também uma rotina para retirar da lista de tiros do jogo os tiros que já "saíram da tela do jogo". Isto pode ser feito verificando se cada tiro já ultrapassou os limites da tela do jogo;
- Para movimentar o cenário de fundo do jogo, precisaremos acrescentar dois atributos ao projeto de jogo: a textura **cenario_fundo** e a posição y dessa textura.
- Para dar a sensação de movimento ao cenário de fundo do jogo, basta exibirmos este cenário de fundo utilizando uma posição y inicial diferente a cada exibição. É necessário também fazer uma segunda exibição do cenário de fundo de forma a complementar o espaço vazio que foi deixado pela primeira exibição.

Fim do resumo

Informações sobre a próxima aula

Na próxima aula, veremos como acrescentar sons ao projeto de jogo. Até Lá!