TI24X7

CURSOS ONLINE EM TECNOLOGIA

Curso XNA Desenvolvimento de jogos

Módulo 6

Autor: Fernando Amaral

WWW.TI24X7.COM.BR

6. Incluindo Som em seu Jogo

Já temos quase todos os elementos necessários para criarmos um jogo 2D completo, falta o SOM! Existem duas formas de tocar som em um jogo XNA: usando a API de som e tocando arquivos diretamente, ou utilizando o XACT, uma ferramenta para organização e configuração de áudio para um jogo XNA. Em nosso primeiro exemplo, vamos usar a API para adicionar som quando houver colisão no jogo desenvolvido no capitulo passado. Em seguida, utilizando o XACT, onde, alem do som de colisão, vamos adicionar som de fundo ao nosso jogo.

Adicionando som usando a API de som do XNA

Faça uma cópia do projeto do capitulo passado. É importante você lembrar-se de fazer uma copia para que, na seção seguinte, possamos utilizar novamente o jogo sem as alterações feitas nesta seção.

Com a solução Colisao aberta, crie uma nova pasta no projeto de conteúdo chamada Audio. Clique com o botão direito sobre a pasta, selecione Add, Existem Item. Localize e adicione o arquivo explosao.wav no diretório de exemplos deste capitulo.

Em seguida, abra o arquivo Game1.cs, junto as variáveis declaradas a nível de classe explosao, do tipo SoundEffect, e instanciaexplosao, do tipo SoundEffectInstance:

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

    Texture2D caveira;
    Vector2 vetorcaveira;
    Vector2 velocidadecaveira = new Vector2(6, 6);

    Texture2D nave;
    Vector2 vetornave;


    SpriteFont fonte;
    int pontos;
    int numaleatorio;

    SoundEffect explosao;
    SoundEffectInstance instanciaexplosao;
```

No método LoadContent, vamos adicionar duas linhas de código ao código já existente. Primeiro carregamos o conteúdo, o arquivo de áudio, para a variável explosao. Em seguida criamos uma instancia da explosao, inicializando a variável instanciaexplosao, deixando-o pronta para tocar o áudio:

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

    caveira = Content.Load<Texture2D>(@"Imagens\Caveira");
    nave = Content.Load<Texture2D>(@"Imagens\NaveEspacial");
    fonte = Content.Load<SpriteFont>(@"Fontes\SpriteFont1");
    explosao = Content.Load<SoundEffect>(@"Audio\Explosao");
    instanciaexplosao = explosao.CreateInstance();


    vetorcaveira = new Vector2(10, 10);
    vetornave = new Vector2(Window.ClientBounds.Width / 2 - nave.Width /
           2, Window.ClientBounds.Height / 2 - nave.Height / 2);


    // TODO: use this.Content to load your game content here
}
```

Agora basta tocar o áudio no método Update. Observe a chamada da função de detecção de colisão, caso a colisão seja detectada, além de incrementar os pontos, chamamos o método Play da instancia da explosão: instanciaexplosao.Play(); Faça a alteração em seu código e rode a aplicação. Note que sempre que houver a intersecção entre sprites, um som de explosão é executado.

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

    KeyboardState teclado = Keyboard.GetState();
    //Verifica se alguma tecla de movimento esta pressionada
    if (teclado.IsKeyDown(Keys.Left))
           vetornave.X -= 3;
    if (teclado.IsKeyDown(Keys.Right))
            vetornave.X += 3;
    if (teclado.IsKeyDown(Keys.Up))
            vetornave.Y -= 3;
    if (teclado.IsKeyDown(Keys.Down))
             vetornave.Y += 3;

    //detecta limites horizontais da tela e inverte a velocidade
    if (vetorcaveira.X > Window.ClientBounds.Width - caveira.Width ||
                    vetorcaveira.X < 0)
        velocidadecaveira.X *= -1;
    //detecta limites verticais da tela e inverte a velciodade
    if (vetorcaveira.Y > Window.ClientBounds.Height - caveira.Height ||
                    vetorcaveira.Y < 0)
        velocidadecaveira.Y *= -1;

    //Muda a direção da caveira, aleatóriamente
    numaleatorio = GeraAleatorio(1, 100);
    switch (numaleatorio)
    {   case 2:
            velocidadecaveira.Y *= -1;
            break;
        case 3:
            velocidadecaveira.X *= -1;
            break;
        case 4:
            velocidadecaveira.Y *= -1;
            velocidadecaveira.X *= -1;
            break;
    }


    //atualiza a posição do sprite
    vetorcaveira.X += velocidadecaveira.X;
    vetorcaveira.Y += velocidadecaveira.Y;

    //detecta colisão, reduz pontos em caso de verdadeiro,
    //incrementa em caso negativo
    if (DetectaColisao())
    {
        pontos -= 15;
        instanciaexplosao.Play();
    }
    else
    pontos +=1;
```

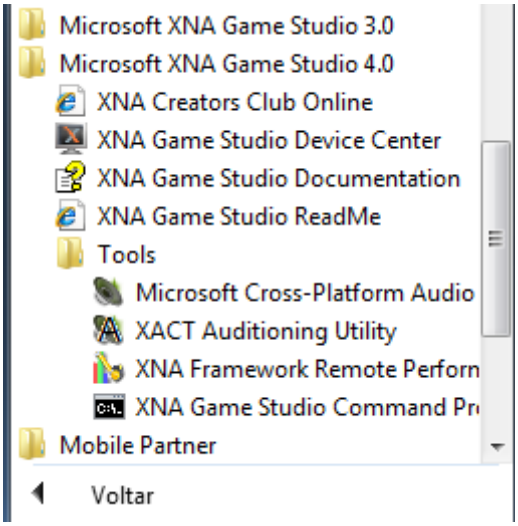
```
        base.Update(gameTime);
    }
```

Além do método play, a API dispõe diversos outros métodos, que permitem parar, pausar e até alterar o volume do áudio.

Utilizando o XACT

Neste exemplo crie uma nova cópia do jogo colisão do capitulo passado. Além de adicionar o som de explosão na colisão, vamos criar um som ambiente – de fundo – que será executado durante a duração do jogo. Abra a copia do projeto e na solução de recursos adicione uma pasta chamada Audio.

O XACT é uma ferramenta autônoma de áudio que é instalada junto com o XNA Studio 4.0. Você poderá encontrá-la no sub menu Tools de sua instalação do XNA, sob o nome Microsoft Cross-Platform Audio Creation Tool 3 (XACT3). No mesmo sub menu você encontra a ferramenta XACT Auditioning Utility, que utilizaremos em seguida para testar os sons do XACT.

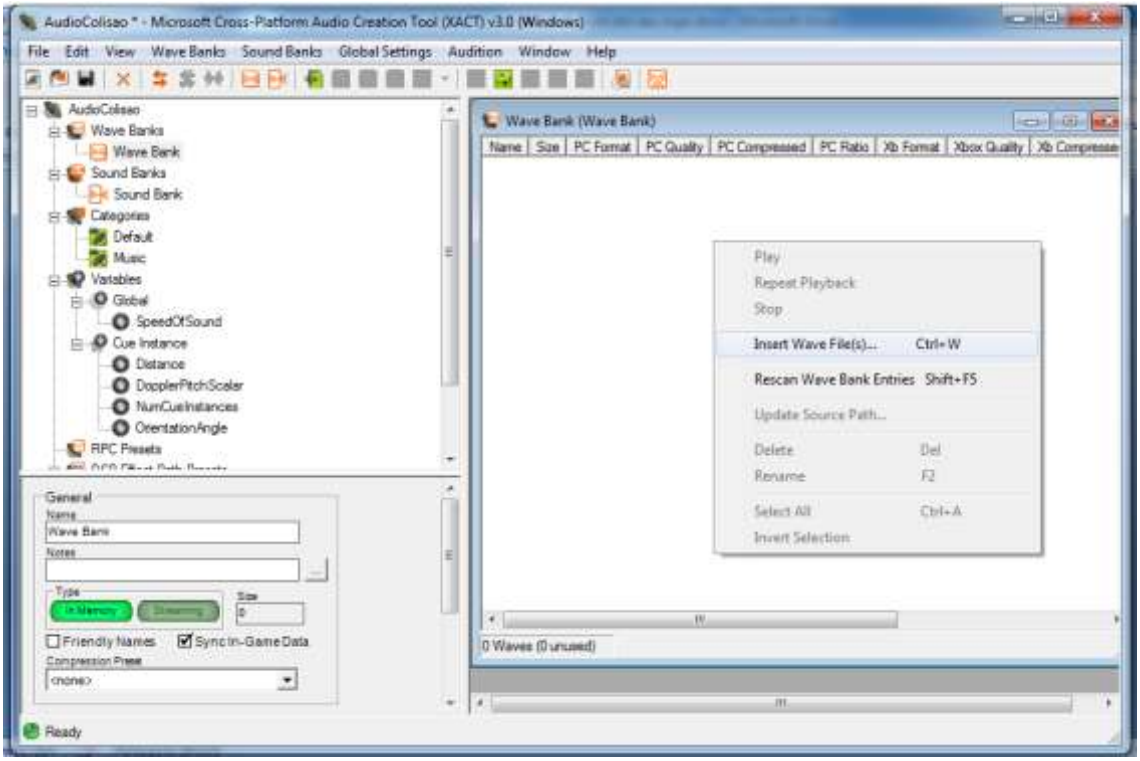


Para começar, abra o XACT clicando sobre o menu Microsoft Cross-Platform Audio Creation Tool 3:

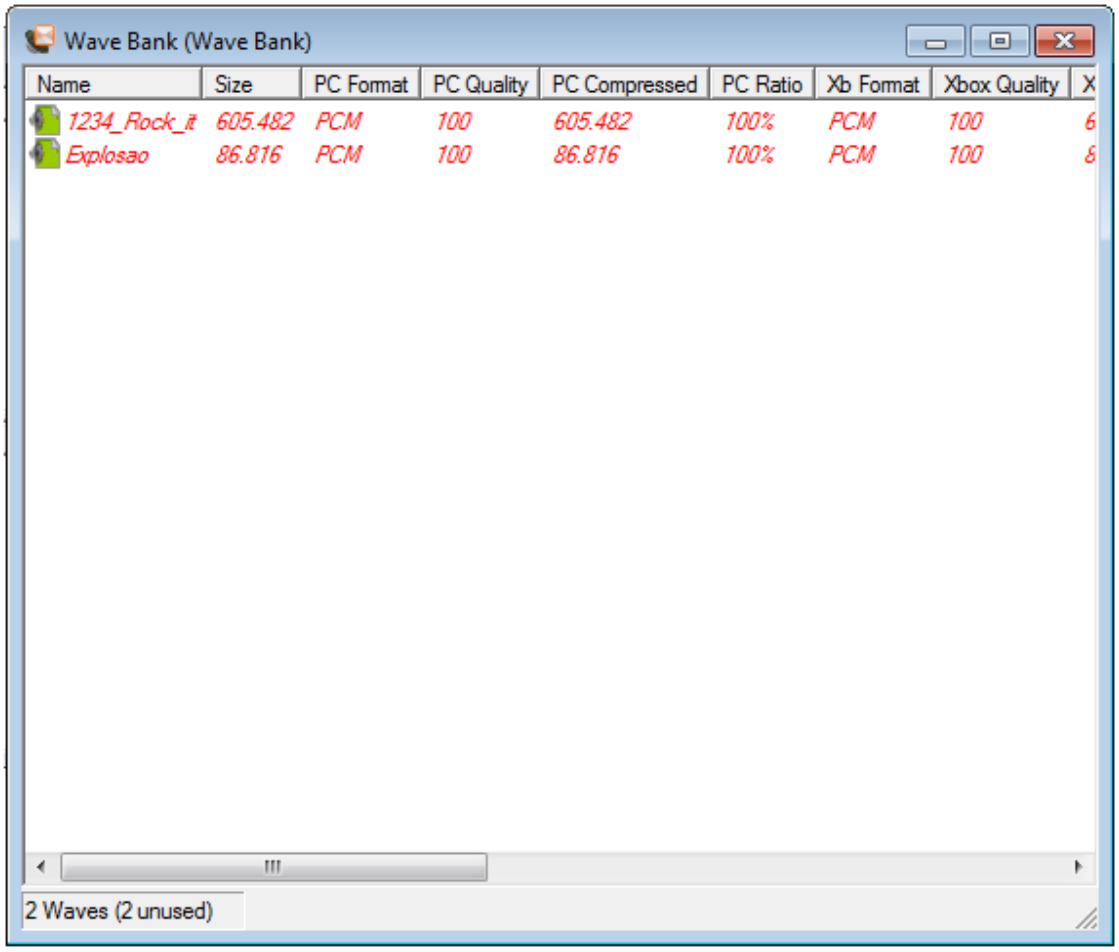
Inicialmente vamos criar um novo projeto, para isto clique em File, New Project. Para evitar confusão, é importante você salvar o projeto dentro da pasta Audio criada a pouco no seu projeto de Recursos. De o nome do Projeto do XACT de AudioColisao.

Clique com o botão direto sobre Wave Bank e selecione New Wave Bank. Mantenha o nome padrão “Wave Bank”. Faça o mesmo procedimento para Sound Bank, também mantendo nome padrão “Sound Bank”.

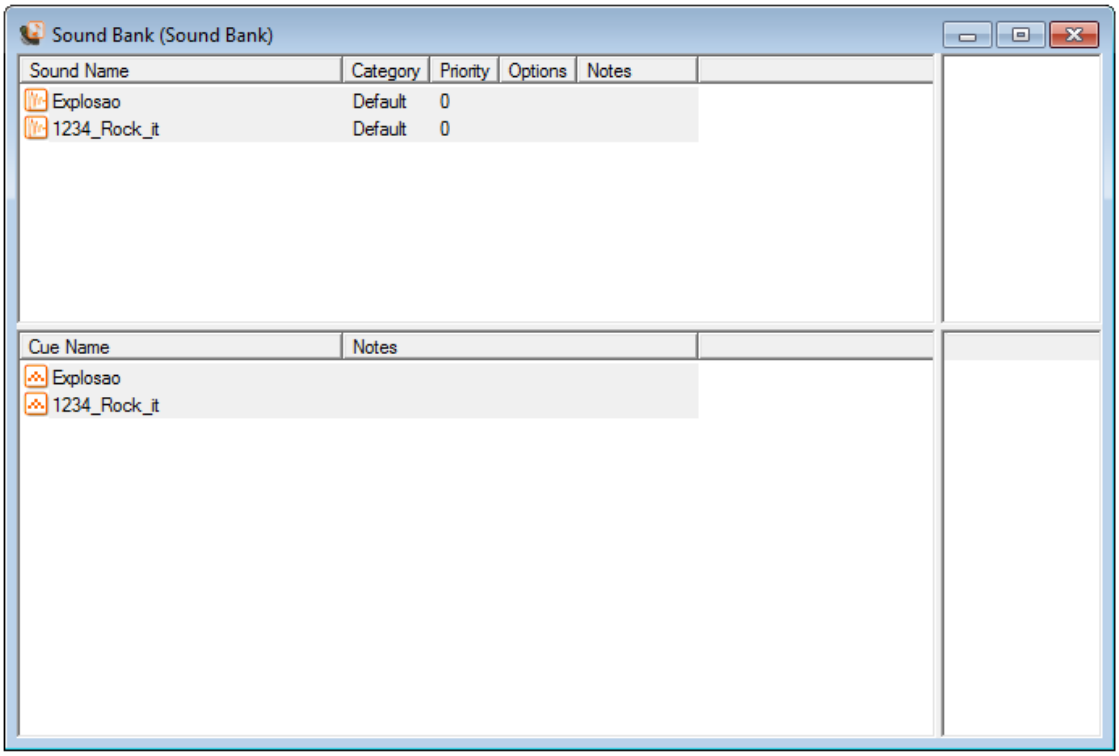
Agora vamos adicionar nossos arquivos de áudio ao Wave Bank, para isto clique com o botão direito sobre a área do banco e selecione a opção Insert Wave File(s).



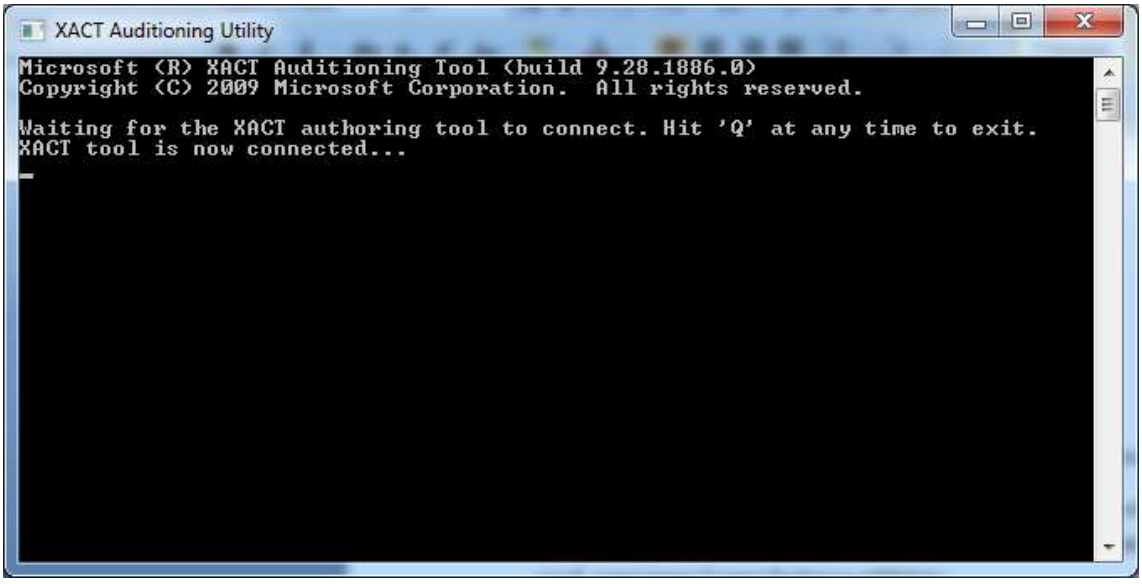
Selecione os arquivos 123_Rock_it.wav e Explosao.wav do diretório de exemplos que acompanham o curso. Seu Wave Bank deve estar semelhante a imagem abaixo:



Agora você precisa adicionar os arquivos ao Sound Bank, para isto selecione os dois arquivos de áudio e os arraste para a área Cue Name, do Sound Bank. Cue Name se refere ao nome em que faremos referencia ao som dentro de nosso Jogo XNA – o mesmo principio do Asset Name de um arquivo de conteúdo. Note que os nomes também foram adicionados a área superior do Sound Bank:

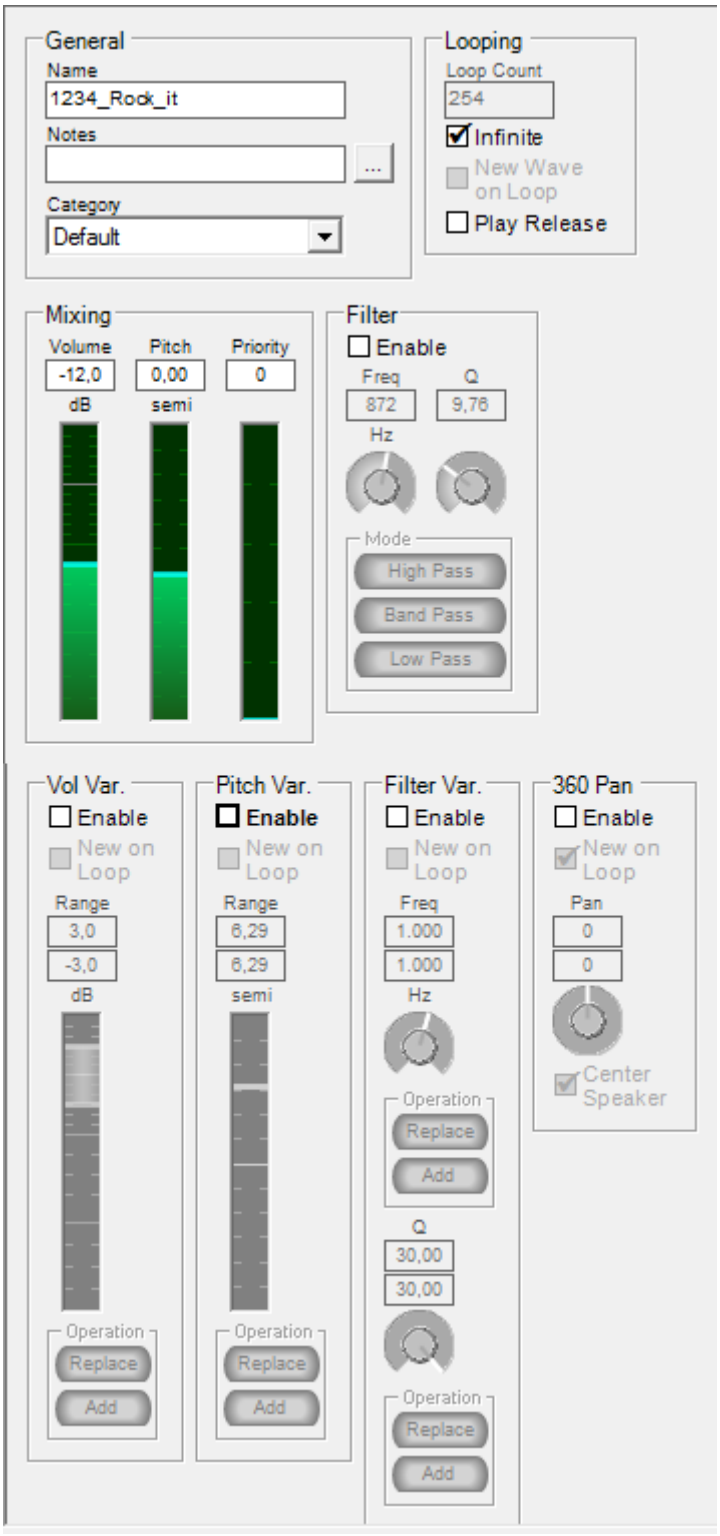


Agora você pode testar os arquivos de áudio, porém antes você deve executar o XACT Auditioning Utility, que abrirá uma tela de comando esperando um som ser executado ou que você pressione Q para fechar o utilitário:



Para executar um som, basta selecioná-lo e clicar no botão verde em forma de seta na barra de ferramentas.

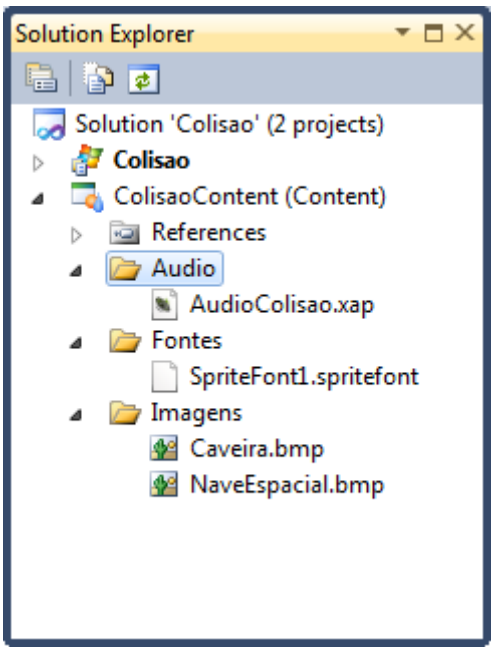
Selecione um som em Sound Bank, e note que na área a esquerda, diversos recursos de mixagem e efeitos estão disponíveis, alguns devem ser habilitados na caixa de verificação logo acima do efeito. Sugiro que você perca alguns minutos testando as possibilidades diversas.



Para nosso jogo, queremos que um som de fundo seja executado durante todo jogo. Para obter este efeito, selecione o arquivo 1234_Rock_It e marque a opção Infinite, como na imagem acima.

Preparando o Jogo

Agora precisamos adicionar o arquivo AudioColisao.xap criado ao nosso projeto de recurso de nosso jogo. Já salvamos o arquivo na mesma pasta, basta adicioná-lo ao projeto clicando com o botão direito sobre a pasta Audio Add, Existem Item, localize e selecione o arquivo AudioColisao.xap. Seu projeto deve parecer como a imagem abaixo:



Em nível de classe vamos declarar 5 novas variáveis em nosso jogo: motoraudio, do tipo AudioEngine, bancowave, do tipo WaveBank, bancosom, do tipo SoundBank, e somFundo e somColisao, do tipo Cue:

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

    Texture2D caveira;
    Vector2 vetorcaveira;
    Vector2 velocidadecaveira = new Vector2(6, 6);

    Texture2D nave;
    Vector2 vetornave;
```

```
SpriteFont fonte;
int pontos;
int numaleatorio;

AudioEngine motoraudio;
WaveBank bancowave;
SoundBank bancosom;
Cue somfundo;
Cue somcolisao;
```

No método LoadContent vamos instanciar nossas variáveis motoraudio, bancowave e bancosom. Observe que a forma como indicamos a arquivo de recurso é diferenciada dos demais recursos, primeiro, temos que usar como diretório base a pasta Content. Wave Bank e Audio Bank são os nomes que demos a nossos bancos de waves e áudios, respectivamente, em nosso projeto do XACT. Instanciado estes objetos, agora basta inicializar a variável Cue somfundo, como nomeados em nosso projeto do XACT. Por fim, chamamos o método Play do Cue somfundo, já que este som deverá ser executado durante toda a duração do jogo. O outro som, para a explosão, só será carregado no método Update, na próxima selção vamos explicar o motivo.

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

     caveira = Content.Load<Texture2D>(@"Imagens\Caveira");
     nave = Content.Load<Texture2D>(@"Imagens\NaveEspacial");
     fonte = Content.Load<SpriteFont>(@"Fontes\SpriteFont1");

     vetorcaveira = new Vector2(10, 10);
     vetornave = new Vector2(Window.ClientBounds.Width / 2 - nave.Width /
             2, Window.ClientBounds.Height / 2 - nave.Height / 2);


     motoraudio = new AudioEngine(@"Content\Audio\AudioColisao.xgs");
     bancowave = new WaveBank(motoraudio, @"Content\Audio\Wave Bank.xwb");
     bancosom = new SoundBank(motoraudio, @"Content\Audio\Sound
             Bank.xsb");

     somfundo = bancosom.GetCue("1234_Rock_it");

     somfundo.Play();

     // TODO: use this.Content to load your game content here
 }
```

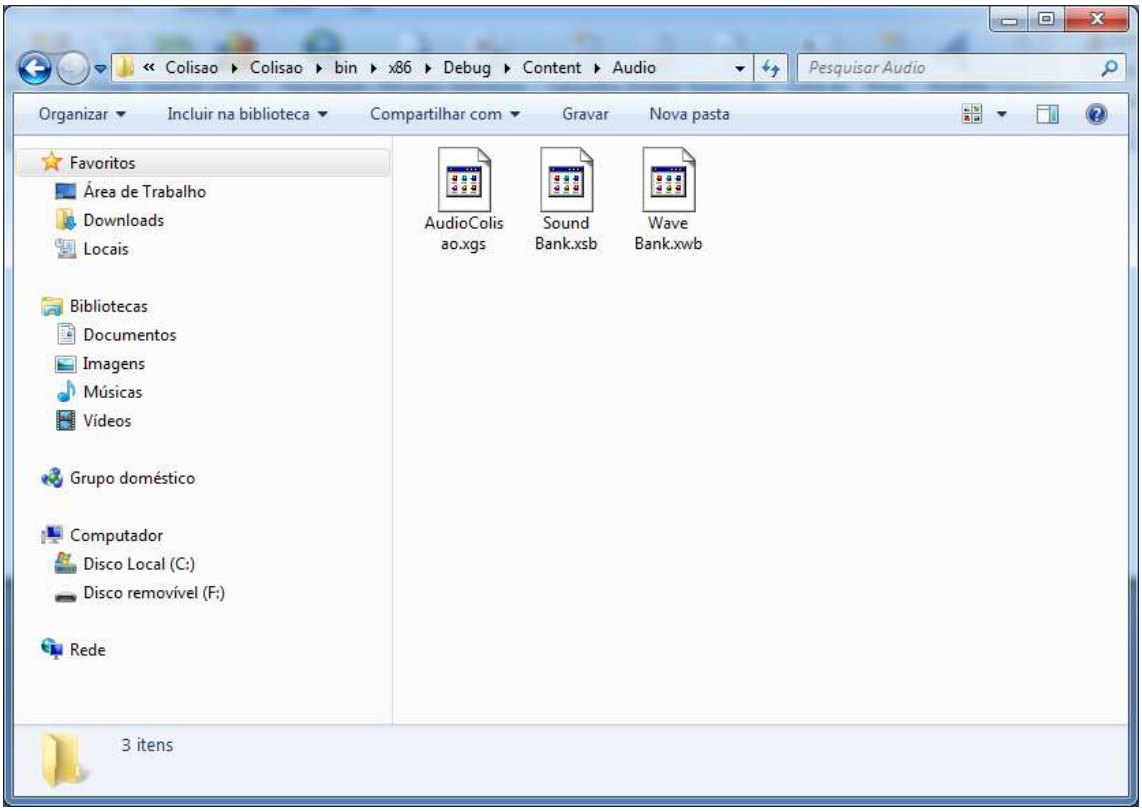
Por fim, alteramos nosso método Update, para caso alguma colisão seja detectada, a Cue comcolisao seja executada. Uma Cue só pode ser executada uma vez depois de carregada, se precisamos executá-la novamente, é preciso novamente chamar o método GetCue, a carga do áudio é feito somente no método Update. Repare que um bloco “if” verifica se o método é igual a nulo ou se o som esta parado. Caso contrário, o som será reiniciado várias vezes, enquanto estiver havendo colisão.

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

     KeyboardState teclado = Keyboard.GetState();
     //Verifica se alguma tecla de movimento esta pressionada
     if (teclado.IsKeyDown(Keys.Left))
            vetornave.X -= 3;
     if (teclado.IsKeyDown(Keys.Right))
             vetornave.X += 3;
     if (teclado.IsKeyDown(Keys.Up))
             vetornave.Y -= 3;
     if (teclado.IsKeyDown(Keys.Down))
              vetornave.Y += 3;

     //detecta limites horizontais da tela e inverte a velocidade
     if (vetorcaveira.X > Window.ClientBounds.Width - caveira.Width ||
             vetorcaveira.X < 0)
         velocidadecaveira.X *= -1;
     //detecta limites verticais da tela e inverte a velciodade
     if (vetorcaveira.Y > Window.ClientBounds.Height - caveira.Height ||
             vetorcaveira.Y < 0)
         velocidadecaveira.Y *= -1;

     //Muda a direção da caveira, aleatóriamente
     numaleatorio = GeraAleatorio(1, 100);
     switch (numaleatorio)
     {   case 2:
             velocidadecaveira.Y *= -1;
             break;
         case 3:
             velocidadecaveira.X *= -1;
             break;
         case 4:
             velocidadecaveira.Y *= -1;
             velocidadecaveira.X *= -1;
             break;
     }

      //atualiza a posição do sprite
     vetorcaveira.X += velocidadecaveira.X;
     vetorcaveira.Y += velocidadecaveira.Y;

     //detecta colisão, reduz pontos em caso de verdadeiro,
     //incrementa em caso negativo
      if (DetectaColisao())
      {
          pontos -= 15;
       if (somcolisao == null || somcolisao.IsStopped)
             {
              somcolisao = bancosom.GetCue("Explosao");
             somcolisao.Play();
             }
      }
      else
      pontos +=1;
```

```
        base.Update(gameTime);
    }
```

Rode a aplicação e veja que agora temos um som de fundo executado durante toda a execução do Jogo. Na colisão, nosso som de explosão é executado.

Agora vamos voltar um pouco no código que escrevemos. Observe que ao carregar o conteúdo, usamos como nome dos recursos arquivos com nomes como AudioColisao.xgs, sendo que o arquivo XACT que criamos se chama AudioColisao.xap, além de nomes como Sound Bank.xsb e Wave Bank.xwb. Na verdade estes são os arquivos criados pelo XNA em tempo de execução, contendo os recursos usados em nossos áudios. Se você abrir a pasta de depuração do Visual Studio após executar o jogo, poderá encontrá-los:



Conclusão

Agora temos todos os elementos necessários para criar jogos completos em 2D. Esta é a tarefa do próximo capítulo, construir um jogo completo!