Creating a Role Playing Game with XNA Game Studio 3.0
Part 12
Updating the Tile Engine

To follow along with this tutorial you will have to have read the previous tutorials to understand much of what it going on. You can find a list of tutorials here: XNA 3.0 Role Playing Game Tutorials You will also find the latest version of the project on the web site on that page. If you want to follow along and type in the code from this PDF as you go, you can find the previous project at this link: Eyes of the Dragon - Version 11 You can download the graphics for the these tutorials at this link: Graphics.zip

For this tutorial you will need the new tile set that I have created. You can find it in the graphics file for these tutorials mentioned above. You are free to use these tiles for non-commercial use. If you want to use these tiles for commercial use you have two options. You can mention Spiral Graphics, with their URL http://www.spiralgraphics.biz, in your game or you can purchase a license to use their textures in your game.

In this tutorial I will be updating the tile engine. I will be creating a new class for the tile set. This class will create the rectangles used for drawing the tiles. I am hoping that this will make thing much easier for the tile engine.

To get started the first thing that you will want to do is to add the new tile set to the **TileSets** folder in the **Content** folder. The tile set's file name is tileset1.jpg and like I mentioned is in the Graphics.zip file found in the above link.

With that done, you will want to add a new class to the project called Tileset. As always, I will give you the new code and then explain why I've done what I have done. This is the code for the Tileset class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;

namespace New2DRPG
{
    public class Tileset
    {
        static Texture2D tilesetTexture;
        List<Rectangle> tiles = new List<Rectangle>();
        static int tileWidth;
        static int tileHeight;

        int tilesWide;
        int tilesHigh;

        public Tileset(Texture2D tilesetTexture, int tileWidth, int tileHeight,
                  int tilesWide, int tilesHigh)
        {
            Tileset.tilesetTexture = tilesetTexture;
            Tileset.tileWidth = tileWidth;
            Tileset.tileHeight = tileHeight;
```

```
            this.tilesWide = tilesWide;
            this.tilesHigh = tilesHigh;
            CreateRectangles(tilesWide, tilesHigh);
        }

        public List<Rectangle> Tiles
        {
            get { return tiles; }
        }

        public static Texture2D TilesetTexture
        {
            get { return tilesetTexture; }
        }

        public static int TileWidth
        {
            get { return tileWidth; }
        }

        public static int TileHeight
        {
            get { return tileHeight; }
        }

        public void CreateRectangles(int tilesWide, int tilesHigh)
        {
            Rectangle rectangle = new Rectangle();
            rectangle.Width = Tileset.tileWidth;
            rectangle.Height = Tileset.tileHeight;
            tiles.Clear();
            for (int i = 0; i < tilesWide; i++)
            {
                for (int j = 0; j < tilesHigh; j++)
                {
                    rectangle.X = i * Tileset.tileWidth;
                    rectangle.Y = j * Tileset.tileHeight;
                    tiles.Add(rectangle);
                }
            }
        }
    }
}
```

As you can see, I added in two using statements to the class. One for the XNA framework and one for the XNA Graphics classes. I made this class a public class so it could be used in the **CoreComponents**, like the tile engine with out upsetting the compiler by having it less accessible.

There are six variables in this class. There is a variable for the **Texture2D** of the tile set and variables for the height and width of the tiles. This is an important note. This is not the size of the tiles on the screen. This is the size of the tiles in the tile set. I created the tiles to be 128 pixels by 128 pixels. These variables are static. I decided to only have one tile set for use at a time. This is reasonable because it will help with the rendering process because XNA likes to render textures that are the same.

There is also a **List** of **Rectangles** for the tile set. This list will be the index of the tiles in the tile set. I will explain how these **Rectangles** are created a little later on. To create the **Rectangles** I also need to know how many tiles wide the tiles set is and how many tiles high the tile set is so there are variables for that. That does it for variables, on to the constructor.

The constructor does take five parameters. That is a lot but I think it was necessary. The parameters it takes are the **Texture2D** for the tile set, the width and height of the tiles and how many tiles wide and high the tile set is. The constructor just assigns the variables for the class and calls a method to create the rectangles for the tile set.

There are four properties for this class. They are all get only properties. There is one for the **List** of **Rectangles** for the tile set. The **Texture2D** of the tile set and the height and width of the tiles in the tile set, not the height and width of the tiles on the screen.

That just leaves the method to create the **Rectangles** for the tile set. I will need to explain something here. When I create the **Rectangles** for the tile set, I create them from left to right then top to bottom. This will be important later when I get to creating the editor for creating maps. In the method I create a **Rectangle** object. Since the width and the height of the tiles will not change I set those to the height and the width of the tiles in the tile set. Then I clear the **List** of **Rectangles**. Next there are two for loops. Inside these loops I will set the X and Y values of each **Rectangle** and add the **Rectangle** object to the **List** of **Rectangles**. To find the X and Y values I do the same thing I do in the tile engine to determine where the tiles should be drawn. The **i** value will be the X value and the **j** value will be the Y value. I take the X value and multiply it by the width of the tile and the Y value and multiply it by the height of the tile. Unlike in the tile engine though, the X and Y values are not reversed.

To make use of this new class, I will have to make a few changes to the rest of the project. The first change I will make is in the **ActionScreen** game screen. Since the **ActionScreen** is fairly small, I will give you the new version and then explain the changes I've made.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using New2DRPG.CoreComponents;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Content;
using New2DRPG.SpriteClasses;

namespace New2DRPG
{
    class ActionScreen : GameScreen
    {
        SpriteFont gameFont;
        Texture2D chest;
        Texture2D tilesetTexture;
```

```
        ContentManager Content;
        SpriteBatch spriteBatch;

        string tilesetName;
        TileEngine tileEngine;
        Tileset tileset;

        public ActionScreen(Game game, SpriteFont gameFont, string tilesetName)
            : base(game)
        {
            this.gameFont = gameFont;
            Content =
                (ContentManager)Game.Services.GetService(typeof(ContentManager));

            spriteBatch =
                (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));
            this.tilesetName = tilesetName;
            LoadContent();

            tileset = new Tileset(tilesetTexture, 128, 128, 4, 4);
            tileEngine = new TileEngine(game, this.tileset, 50, 50);
            Components.Add(tileEngine);
            tileEngine.Show();

        }

        protected override void LoadContent()
        {
            base.LoadContent();
            tilesetTexture = Content.Load<Texture2D>(@"Tilesets\" + tilesetName);
            chest = Content.Load<Texture2D>(@"Items\chest");
        }

        public override void Update(GameTime gameTime)
        {
            base.Update(gameTime);

        }

        public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);
        }

        public override void Show()
        {
            base.Show();
            Enabled = true;
            Visible = true;
        }

        public override void Hide()
        {
            base.Hide();
            Enabled = false;
            Visible = false;
        }
    }
}
```

The first change is that I changed the **Texture2D** from **tileset** to **tilesetTexture**. Then I added a **Tileset** object. I changed the constructor to reflect the changes that I made. In the constructor, after I called the **LoadContent** method to load in the texture for the tile set, I create the **Tileset** object. I passed in the width and height of the tiles, 128, and the number of tiles wide and the number of tiles high, 4. Then I changed where I created the **TileEngine** object to take the new **Tileset** object instead of a **Texture2D** as a parameter. Those are the only changes to the **ActionScreen**.

Because the **TileEngine** now takes a **Tileset** object instead of a **Texture2D**, I had to make a few changes to the constructor and the class itself, such as a few variables and one property. This is the code for the variables of the class, the constructor and the changed property.

```
SpriteBatch spriteBatch;
ContentManager Content;
static List<Rectangle> tiles = new List<Rectangle>();
KeyboardState oldState;
KeyboardState newState;

TileMap map;

List<GameComponent> childComponents = new List<GameComponent>();

static int tileWidth = 64;
static int tileHeight = 48;

static int tileMapWidth;
static int tileMapHeight;

static int screenWidth;
static int screenHeight;

static int mapWidthInPixels;
static int mapHeightInPixels;

static float cameraX;
static float cameraY;

Random random = new Random();
Camera camera = new Camera();
static Tileset tileset;

public TileEngine(Game game, Tileset tileset, int tileMapWidth,
          int tileMapHeight)
    : base(game)
{
    spriteBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

    TileEngine.tileset = tileset;
    TileEngine.tileMapWidth = tileMapWidth;
    TileEngine.tileMapHeight = tileMapHeight;

    TileEngine.tiles = tileset.Tiles;

    map = new TileMap(tileMapWidth, tileMapHeight, game);
    childComponents.Add(map);
```

```
            Content =
                (ContentManager)Game.Services.GetService(typeof(ContentManager));

            Texture2D chest = Content.Load<Texture2D>(@"Items\chest");

            ItemSprite tempItemSprite;
            Vector2 position;
            Random random = new Random();

            for (int i = 0; i < 10; i++)
            {
                position = new Vector2(
                    random.Next(0, 10),
                    random.Next(0, 10));

                tempItemSprite = new ItemSprite(game,
                    chest, position);

                childComponents.Add(tempItemSprite);
            }

            screenWidth = game.Window.ClientBounds.Width;
            screenHeight = game.Window.ClientBounds.Height;
        }
```

The first change is that instead of an array of **Rectangles** there is a **List** of **Rectangles** for the tiles. This will not effect the rest of the **TileEngine** or components because the **List** can be accessed just like an array. The next change is that instead of the **tileset** variable being a **Texture2D**, it is a now a **Tileset**. The rest of the variables for the class didn't change.

There were a few changes to the constructor. The first is that instead of taking a **Texture2D** for the tile set, it takes a **Tileset**. The other change is I moved where I created the map. The reason I did this is because I wanted to change the way I created the map to reflect the new tile set.

There is one other change to the **TileEngine** class. I changed the property for getting the **Texture2D** of the tile set. Instead of returning the old **tileset** variable, it returns the **Texture2D** from the tile set object. This is the new **Tileset** property:

```
public static Texture2D TileSet
{
    get { return Tileset.TilesetTexture; }
}
```

There actually was one other change. I had to change the get property for the tiles. Instead of an array of **Rectangles**, it returns a a **List** of **Rectangles**. This is the new **Tiles** property:

```
public static List<Rectangle> Tiles
{
    get { return tiles; }
}
```

I also changed the **TileMap** class to reflect the changes in the tile set. The only change was, instead of using 4 to create a random tile index, I used the **Count** property of the **List** of **Rectangles** to find a random tile. I don't think there is really much else to explain so I will just give you the new constructor of the **TileMap** class.

```
public TileMap(int tileMapWidth, int tileMapHeight, Game game)
    : base(game)
{
    spriteBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

    TileMapLayer layer = new TileMapLayer(tileMapWidth, tileMapHeight);

    for (int x = 0; x < 50; x++)
        for (int y = 0; y < 50; y++)
            layer.SetTile(x, y, random.Next(0, TileEngine.Tiles.Count));
    tileMapLayers.Add(layer);
}
```

Well, that is it for this tutorial. It may seem short, but I did actually make a couple big changes to the project. I will try and have another tutorial available on the web site soon, so you can check out the blog that goes with these tutorials, http://xna-rpg.tutorials.com, for the latest news on these tutorials or the news page on my web site for the latest news on these tutorials.