Creating a Role Playing Game with XNA Game Studio 3.0
Part 7
Adding Sprites

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. 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 6

In today's tutorial I'm going to add a sprite game component to the project and test it using a derived class for items. To get started you will need to add a new **GameComponent** to the **CoreComponents** folder called **Sprite**. This component is meant to be a base component. You should inherit form this component. I will, as always, give you the code and then explain what it does and why I did what I did. This is the code for the **Sprite** class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;


namespace New2DRPG.CoreComponents
{
    /// <summary>
    /// This is a game component that implements IUpdateable.
    /// </summary>
    public abstract class Sprite : Microsoft.Xna.Framework.DrawableGameComponent
    {
        protected Texture2D texture;

        protected Rectangle sourceRectangle;

        protected Vector2 position;
        protected Vector2 velocity;
        protected Vector2 center;

        protected float scale;
        protected float rotation;

        protected SpriteBatch spriteBatch;

        public Sprite(Game game, Texture2D texture)
            : base(game)
        {
```

```
        spriteBatch =
            (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

        this.texture = texture;

        position = Vector2.Zero;
        velocity = Vector2.Zero;
        center = new Vector2(texture.Width / 2,
            texture.Height / 2);

        scale = 1.0f;
        rotation = 0.0f;

        sourceRectangle = new Rectangle(0,
                                        0,
                                        texture.Width,
                                        texture.Height);
    }

    public Sprite(Game game, Texture2D texture, Rectangle sourceRectangle)
        : base(game)
    {
        spriteBatch =
            (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

        this.texture = texture;
        this.sourceRectangle = sourceRectangle;

        position = Vector2.Zero;
        velocity = Vector2.Zero;
        center = new Vector2(sourceRectangle.Width / 2,
            sourceRectangle.Height / 2);

        scale = 1.0f;
        rotation = 0.0f;
    }

    public virtual Texture2D Texture
    {
        get { return texture; }
    }

    public virtual Vector2 Position
    {
        get { return position; }
    }

    public virtual Vector2 Velocity
    {
        get { return velocity; }
    }

    public virtual Vector2 Center
    {
        get { return center; }
    }

    public virtual float Scale
    {
```

```
            get { return scale; }
        }

        public virtual float Rotation
        {
            get { return rotation; }
        }

        public override void Initialize()
        {
            // TODO: Add your initialization code here

            base.Initialize();
        }

        public override void Update(GameTime gameTime)
        {
            // TODO: Add your update code here

            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {

            base.Draw(gameTime);
        }

        public virtual void Show()
        {
            Enabled = true;
            Visible = true;
        }

        public virtual void Hide()
        {
            Enabled = true;
            Visible = true;
        }
    }
}
```

The first change is to make the component inherit from **DrawableGameComponent** instead of **GameComponent** because it is a visual component that needs to be drawn. All of the fields for this class are **protected**. The reason I made them **protected** is because I didn't want them accessed outside of the component but I wanted the class that inherits from it to have these variables.

The varaibles for this class include the **Texture2D** of the sprite. The next one is a **Rectangle** for the source rectangle of the sprite. The reason for this will become clearer when I get to the constructors for this class. (Notice I said constructors not just constructor.)

There are **Vector2**s for the position, velocity and the center of the sprite as these are all useful attributes of a sprite. There are also two **float**s for the scale and rotation of the sprite. Again, both useful attributes of a sprite. Finally there is a **SpriteBatch** object.

There are two constructors for this class. One that takes a **Game** object and a **Texture2D**. The other

takes a **Game** object, a **Texture2D** and a **Rectangle**. There are two so you can have just a single sprite or a sprite sheet and specify where the sprite is in the sprite sheet.

The first consturctor gets the **SpriteBatch** object and sets all the velocity and position **Vector2**s to the zero vector. The center of the sprite is calculated using the height and width of the sprite divided by two. The constructor sets the scale to 1.0f and the rotation to 0 radians. Then the constructor creates the source rectangle of the sprite using the height and width of the texture. The second contructor does pretty much the same. The only difference is it sets the source rectangle to the rectangle it was passed and the center of the sprite is calculated using the height and width of the source rectangle.

I made all of the properties of the class public and virtual. They are all get only methods but you can change that behavior in the derived class. All the **Update** and **Draw** methods do is call the **Update** and **Draw** methods of their base class. There are also virtual **Hide** and **Show** methods to hide and show the sprite.

Now I'm going to make a class to test this component. I added a new folder to the project called **SpriteClasses**. In this folder I added a new class called **ItemSprite**. This is actually a very small class as all I wanted to do is test to see if the component I made worked. This is the class that I made.

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

namespace New2DRPG.SpriteClasses
{
    class ItemSprite : Sprite
    {
        public ItemSprite(Game game, Texture2D texture, Vector2 position)
            : base(game, texture)
        {
            this.position = position;
        }

        public override void Draw(GameTime gameTime)
        {
            spriteBatch.Draw(texture,
                new Rectangle((int)position.X * TileEngine.TileWidth
                        - TileEngine.CameraXPosition,
                    (int)position.Y * TileEngine.TileHeight
                        - TileEngine.CameraYPositon,
                    TileEngine.TileWidth,
                    TileEngine.TileHeight),
                    Color.White);

            base.Draw(gameTime);
        }

    }
}
```

As you can see there is only a construtor and the **Draw** method. I needed to add the **using** statements

for the **CoreComponents** and a few of the XNA Framework classes. As you can see I inherited the class from the new **Sprite** component.

The constructor for this class takes three parameters. It takes a **Game** object, to pass to the base class, the **Texture2D** for the sprite and a **Vector2** for the position of the sprite. This position is measured in **tiles**. It is the tile where the item will be located on the map. I created the class so that as the map is scrolled the sprites will scroll with the map. If you don't the sprite will just float on top of the map, not exactly what I was looking for.

If you look at the **Draw** method I used the same method to draw the sprite as I used to draw a tile. I created a destination rectangle where I wanted to draw the sprite. I calculate the position of the rectangle using it's position, multipling it by the height and width of the tile and subtracting the position of the camera. I finish the rectangle using the height and the width of a tile.

The last thing to do is to use this class to draw some sprites. I did this in the **TileEngine**. I am just going to give you the new constructor for the **TileEngine** and the **using** statement that had to be added to the **TileEngine** class to be able to use the new class. I also added a new variable for a **ContentManager** class. Here is the new code, and again I will explain why I did what I did.

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;
using New2DRPG.SpriteClasses;

        ContentManager Content;

        public TileEngine(Game game, Texture2D tileset, int tileMapWidth,
            int tileMapHeight)
            : base(game)
        {
            map = new TileMap(tileMapWidth, tileMapHeight, game);
            childComponents.Add(map);

            spriteBatch =
                (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

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

            tiles[0] = new Rectangle(0, 0, 128, 128);
            tiles[1] = new Rectangle(128, 0, 128, 128);
            tiles[2] = new Rectangle(0, 128, 128, 128);
            tiles[3] = new Rectangle(128, 128, 128, 128);

            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 new thing I did was get the **ConentManager** object using the **GetService** method I use to get the SpriteBatch object. I added a new folder to the **Content** folder called **Items**. In this folder added a sprite that I'm working on that will represent a chest. Please forgive me I'm not a good artist so it isn't all that great looking. I'm adding the texture to this zip file: Graphics. I used the **ContentManager** object to load in the texture of the chest. There are three new variables in the class. An **ItemSprite** variable to hold a temorary sprite. There is a **Vector2** for the position of the sprite and a **Random** object to place some chests on the map randomly. Inside a for loop I add 10 sprites to the list of child components of the tile engine. I used the **Next** method of the **Random** variable to get integers between 0 and 10, excluding 10 so you wouldn't have to scroll through then entire map looking for the sprites and they would all appear on the first part of the screen.

Well, that is all for this tutorial. There is a screen shot of the new output on my web site. I will try and have a new tutorial on my web site soon so I hope you will keep coming back and looking for more.