

XNA Basics

Sprites

Table of Contents

3. Project Files
4. Getting the files into the project
5. Creating memory for the file
6. Loading the file into memory
7. Drawing the sprite on the screen

This tutorial will cover drawing sprites to the screen

The first thing to do is to get the project files from:

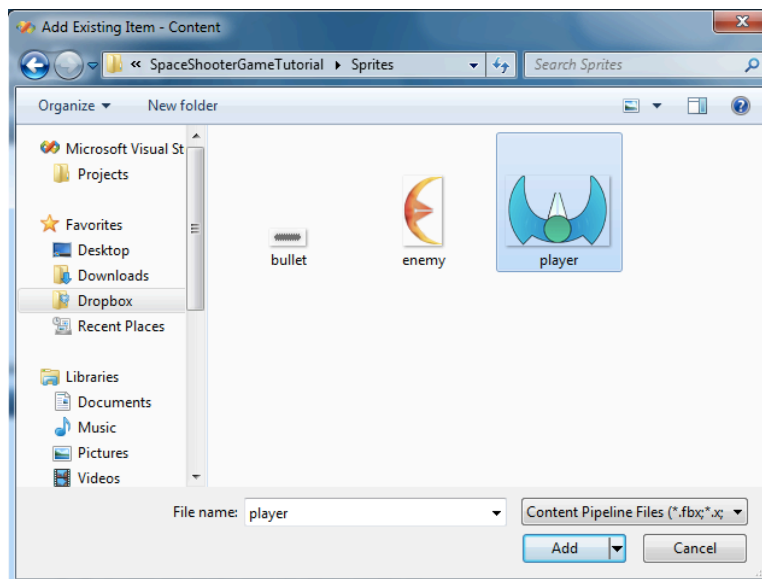
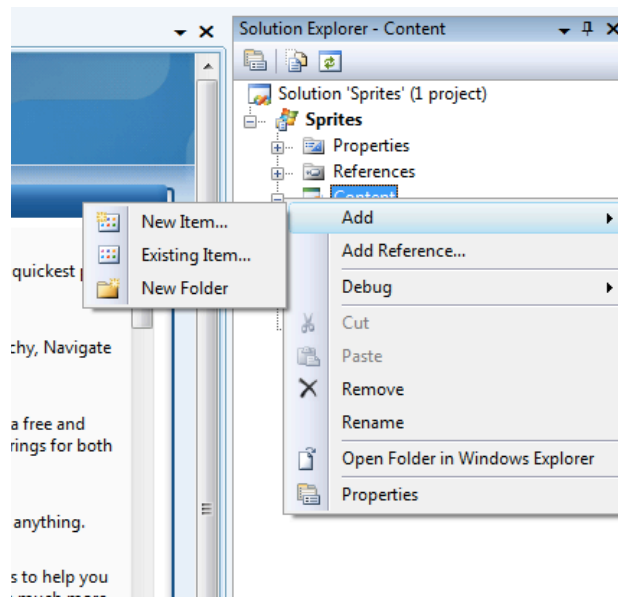
<http://www.phstudios.com/tutorials/xna/SpaceShooter/Sprites.zip>

The only sprite that we will be concerned with in this tutorial is the player.png file.

Now that you have the project files that we are using in this tutorial it is time to create a new project. Refer to the Introduction tutorial if you can't remember how to do this.

The first thing to do is to add a file from the project files that you downloaded.

Right Click Content in the solution explorer ⇒ Add ⇒ Existing Item... ⇒ Navigate to the project files ⇒ player.png



Now to start adding code.

The line in bold is the line that needs to be added to your project.

```
namespace Sprites
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class SpritesGame :
Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Texture2D playerSprite;

        public SpritesGame()
        {
            graphics = new
GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }
    }
}
```

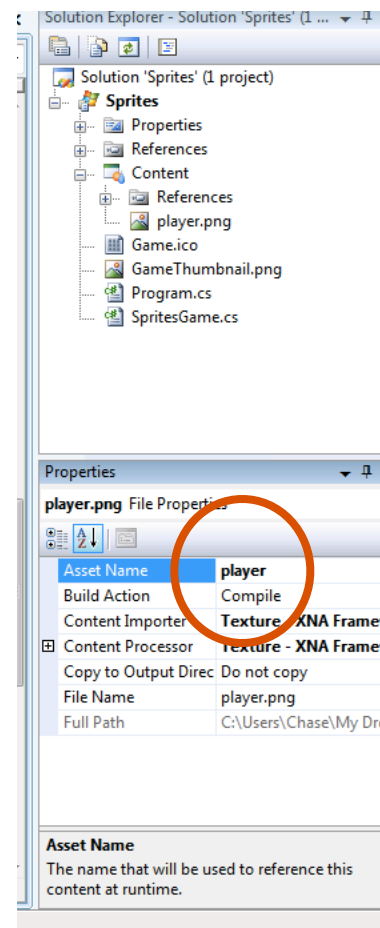
The Texture 2D line that you just added gives the player.png file memory by creating an object that identifies the player.png file as a picture.

The next thing you will want to do is load the content. You do not want to initialize it since the sprite you are loading comes from the content folder. Remember if it is stored in the content folder it should be loaded and unloaded later when you are done with it.

```
/// <summary>
/// LoadContent will be called once per game and is the place to load
/// all of your content.
/// </summary>
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
    playerSprite = Content.Load<Texture2D>("player");
}
```

“player” come from the name of the file in Content as it is shown in the properties window in the solution explorer.



The final step is to actually draw the sprite on the screen. To do that you need to specify a beginning and ending to what the computer will draw and how.

```
/// <summary>
/// This is called when the game should draw itself.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);

    // TODO: Add your drawing code here
    spriteBatch.Begin();
    spriteBatch.Draw(playerSprite,
        new Rectangle(0, 0, playerSprite.Width, playerSprite.Height), Color.White);
    spriteBatch.End();

    base.Draw(gameTime);
}
```

If you noticed the spriteBatch.Draw has a dialog box that pops up as soon as you type the open parenthesis. If you arrow down and up you will notice the different sets of parameters that are accepted by this method. This is an example of method overloading.

To learn more about method overloading look here:

<http://msdn.microsoft.com/en-us/library/xxfyae0c%28VS.71%29.aspx>

Google will also reveal more sources to learn about method overloading.