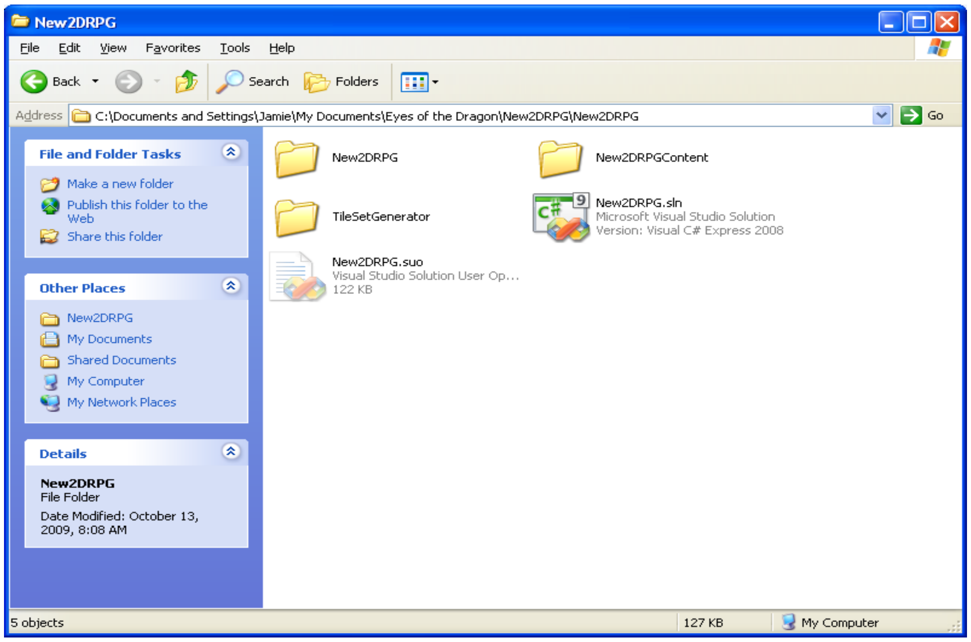
# Creating a Role Playing Game with XNA Game Studio 3.0
# Part 26
# Adding the Tile Set Generator
# A Little Refactoring

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 25 You can download the graphics from this link: Graphics.zip

This is a rather short tutorial. I have been talking about the Tile Set Generator program that I wrote to create **Tileset** objects that the game can read in. In this tutorial I will be adding the Tile Set Generator to the game. To do this tutorial you will need to download the Tile Set Generator from the following link: http://xna.jtmbooks.com/Downloads/TileSetGenerator.zip. I will also do a little refactoring to help shorten a little code.

Once you have downloaded the Tile Set Generator project extract the files. Find the folder named **TileSetGenerator**. You will want to copy that folder and add it the **New2DRPG** folder that has the file **New2DRPG.sln** file. That folder should now look like this.


New2DRPG
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Documents and Settings\Jamie\My Documents\Eyes of the Dragon\New2DRPG\New2DRPG
Go
File and Folder Tasks
Make a new folder
Publish this folder to the Web
Share this folder
Other Places
New2DRPG
My Documents
Shared Documents
My Computer
My Network Places
Details
New2DRPG
File Folder
Date Modified: October 13, 2009, 8:08 AM
New2DRPG
New2DRPGContent
TileSetGenerator
New2DRPG.sln
Microsoft Visual Studio Solution
Version: Visual C# Express 2008
New2DRPG.suo
Visual Studio Solution User Op...
122 KB
5 objects
127 KB
My Computer

Now load the last version of the game. You will want to right click the solution in the **Solution Explorer** window. Now you will choose the **Add** then **Existing project** item. Now you will navigate to the **TileSetGenerator** folder. You need to find the **TileSetGenerator.csproj** file and then select it. This will add the project to your solution. You should now have three projects in your solution.

Now when you want to create a **Tileset** object there is an easy way to do it. What you can do is right click the **TileSetGenerator** project and select the **Set as Startup Project** opition. When you run the program the **TileSetGenerator** project will be run. When you want to run the game you then right click that project and choose the **Set as Startup Project** option.

Now I will do a little refactoring to make some things a little easier. When you create enums you can declare them at the namespace level so you do not have to qualify them with the class name. I've been wanting to do this for a while and thought this would be the perfect opportunity to do so. If you are unaware of what refactoring is, it is the process of making code better organized and easier to read.

There are three classes that have enums in them: **PlayerCharacter**, **CharacterAbilities** and **Difficulty** that I will refactor. Doing this will cause a few errors in the program that you will have to fix. I will start with the **PlayerCharacter** class. Remove the enum **CharClass** from the class and move it so it is in the namespace. The **PlayerCharacter** class should resemble something like this:

```
namespace New2DRPG
{
    public enum CharClass { Fighter = 0, Wizard = 1, Priest = 2, Thief = 3 };

    class PlayerCharacter : Microsoft.Xna.Framework.DrawableGameComponent
    {
        static string[] classNames = Enum.GetNames(typeof(CharClass));
```

This will create a lot of errors in the rest of the game. The errors related to the **PlayerCharacter** class will be in the **CreatePlayerCharacter** method in the **Game1** class. Everywhere that you had **PlayerCharacter.CharClass** you will want to replace that with just **CharClass** as follows.

```
private void CreatePlayerCharacter()
{
    if (createPCScreen.CharacterClass == CharClass.Fighter)
    {
        playerCharacter = new FighterCharacter(
            createPCScreen.CharacterName,
            createPCScreen.CharacterGender,
            createPCScreen.DifficultyLevel, this);
    }
    if (createPCScreen.CharacterClass == CharClass.Priest)
    {
        playerCharacter = new PriestCharacter(
            createPCScreen.CharacterName,
            createPCScreen.CharacterGender,
            createPCScreen.DifficultyLevel, this);
    }
    if (createPCScreen.CharacterClass == CharClass.Thief)
    {
        playerCharacter = new ThiefCharacter(
            createPCScreen.CharacterName,
            createPCScreen.CharacterGender,
            createPCScreen.DifficultyLevel, this);
    }
    if (createPCScreen.CharacterClass == CharClass.Wizard)
    {
        playerCharacter = new WizardCharacter(
            createPCScreen.CharacterName,
            createPCScreen.CharacterGender,
            createPCScreen.DifficultyLevel, this);
    }
    actionScreen.SetPlayerCharacter(playerCharacter);
}
```

There will also be a few errors in the **CreatePCScreen** class. There will be errors with the **characterClass** field and the **CharacterClass** property. This is the code for the **characterClass** field and **CharacterClass** property.

```
CharClass characterClass = CharClass.Fighter;

public  CharClass CharacterClass
{
    get { return characterClass; }
}
```

There will be a few other errors in the **CreatePCScreen** class. They are in the **ChangeClass** method. This is the code for the **ChangeClass** method.

```
public void ChangeClass(int className)
{
    this.className = className;
    switch (className)
    {
        case 0:
            characterClass = CharClass.Fighter;
            break;
        case 1:
            characterClass = CharClass.Wizard;
            break;
        case 2:
            characterClass = CharClass.Priest;
            break;
        case 3:
            characterClass = CharClass.Thief;
            break;
    }
}
```

The last error is in the **ClassPopUpScreen** class. The error is in the constructor where you got the names of the classes for the player character. This is the update constructor.

```
public ClassPopUpScreen(Game game)
    : base(game)
{
    LoadContent();

    Components.Add(new BackgroundComponent(game, image, false));

    imagePosition = new Vector2();
    imagePosition.X = (game.Window.ClientBounds.Width - image.Width) / 2;
    imagePosition.Y = (game.Window.ClientBounds.Height - image.Height) / 2;

    string[] items = Enum.GetNames(typeof(CharClass));

    menu = new ButtonMenu(game, spriteFont, buttonImage);
    menu.SetMenuItems(items);
    Components.Add(menu);
}
```

Now I will refactor the **CharacterAbilities** class. You will want to remove the **Abilities** enum from the class and make it part of the namespace. The **CharacterAbilities** class should resmeble the following:

```
namespace New2DRPG
{
    public enum Abilities
    {
        Strength = 0,
        Stamina = 1,
        Agility = 2,
        Speed = 3,
        Intellect = 4,
        Luck = 5
    };

    class CharacterAbilities
    {
        const float minAbilityScore = 1f;
```

Doing this causes just one error in the program and that is in the **ViewCharacterScreen** class. It will be with the **abilitiyWords** field. You will want to change it to the following.

```
string[] abilityWords = Enum.GetNames(typeof(Abilities));
```

Now I will change the **Difficulty** class. You will want to take the **Level** enum out of the class and make it part of the namespace. The **Difficulty** class should resemble this code.

```
namespace New2DRPG
{
    public enum Level { Easy = 0, Normal = 1, Hard = 2, Ultimate = 3 };

    public class Difficulty
    {
```

This will create many errors and one of them will not be as easy to fix as the others and I will deal with that one last. Two errors will be with the **difficulty** field and the associated property **Difficulty** in the **PlayerCharacter** class. Modify the **difficulty** field and **Difficulty** property to the following.

```
        protected Level difficulty;

        public Level DifficultyLevel
        {
            get { return difficulty; }
        }
```

There will also be an error in the constructor of the **PlayerCharacter** class. The new constructor for the **PlayerCharacter** class follows.

```
public PlayerCharacter(Game game)
    : base(game)
{
    this.game = game;
    spriteBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));
    Content =
        (ContentManager)Game.Services.GetService(typeof(ContentManager));

    difficulty = Level.Normal;
    level = 1;
    experience = 0;
    gold = 0;

    hpspDisplay = Content.Load<Texture2D>(@"GUI\hpspborder");
    blueBar = Content.Load<Texture2D>(@"GUI\bluebar");
    redBar = Content.Load<Texture2D>(@"GUI\redbar");
    spriteFont = Content.Load<SpriteFont>(@"smallFont");
    hpspPosition = new Vector2(10, 10);
}
```

This is unfortunately going to create another error. The name of the enum **Level** is going to conflict with the propery in the **PlayerCharacter** class **Level**. To fix this error I am going to rename the property **Level** to **PlayerLevel**. This is the code for the new property. This will create an error I will fix shortly.

```
public int PlayerLevel
{
    get { return level; }
}
```

You will also have the same error in the **FighterCharacter**, **PriestCharacter**, **ThiefCharacter** and **WizardCharacter** classes and that is in definition the constructors. Change the constructors for each of the classes to the following.

```
public FighterCharacter(string name,
        bool gender,
        Level difficulty,
        Game game)
    : base(game)

public PriestCharacter(string name,
        bool gender,
        Level difficulty,
        Game game)
    : base(game)

public ThiefCharacter(string name,
        bool gender,
        Level difficulty,
        Game game)
    : base(game)

public WizardCharacter(string name,
        bool gender,
        Level difficulty,
        Game game)
    : base(game)
```

This also create a few errors in the **CreatePCScreen** class. Two errors are the **difficultyLevel** field and the associated property **DifficultyLevel.** Change the **difficultyLevel** field and property to the following.

```
        Level difficultyLevel;

        public Level DifficultyLevel
        {
            get { return difficultyLevel; }
        }
```

There is going to be an error in the constructor for the **DifficultyPopUpScreen**. To fix this error change the constructor to the following.

```
        public DifficultyPopUpScreen(Game game)
            : base(game)
        {
            LoadContent();

            Components.Add(new BackgroundComponent(game, image, false));

            imagePosition = new Vector2();
            imagePosition.X = (game.Window.ClientBounds.Width - image.Width) / 2;
            imagePosition.Y = (game.Window.ClientBounds.Height - image.Height) / 2;

            string[] items = Enum.GetNames(typeof(Level));

            menu = new ButtonMenu(game, spriteFont, buttonImage);
            menu.SetMenuItems(items);
            Components.Add(menu);
        }
```

There will be an error in the **CreatePCScreen** class related to the change. That is in the field **difficultyLevels**. This is the new definition of the **difficultyLevels** field.

```
        string[] difficultyLevels = Enum.GetNames(typeof(Level));
```

One last error is in the **HandleStartScreenInput** method in the **Game1** class. It is in the switch statement when I created a default character. This is the updated **HandleStartScreenInput** method.

```
        private void HandleStartScreenInput()
        {
            if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
            {
                switch (startScreen.SelectedIndex)
                {
                    case 0:
                        activeScreen.Hide();
                        activeScreen = createPCScreen;
                        activeScreen.Show();
                        break;
                    case 1:
                        activeScreen.Hide();
                        playerCharacter = new FighterCharacter(
                            "Evander",
                            false,
                            Level.Normal,
                            this);
                        actionScreen.SetPlayerCharacter(playerCharacter);
                        activeScreen = actionScreen;
                        actionScreen.Show();
                        break;
                    case 2:
                        activeScreen.Hide();
                        activeScreen = helpScreen;
                        activeScreen.Show();
                        break;
                    case 3:
                        activeScreen.Hide();
                        activeScreen = creditScreen;
                        activeScreen.Show();
                        break;
                    case 4:
                        activeScreen.Enabled = false;
                        activeScreen = quitPopUpScreen;
                        activeScreen.Show();
                        break;
                }
            }
        }
```

There will be one last error to fix because of the changes and it is in the **Draw** method of the **ViewCharacterScreen**. It will be where I create the **statValues** variable. Change the **statValues** variable to the following.

```
            string[] statValues = {
                    playerCharacter.ClassName,
                    playerCharacter.PlayerLevel.ToString(),
                    playerCharacter.HitPointsCurrent.ToString() + "/" +
                        playerCharacter.HitPointsMax.ToString(),
                    playerCharacter.SpellPointsCurrent.ToString() + "/" +
                        playerCharacter.SpellPointsMax.ToString(),
                    playerCharacter.Experience.ToString(),
                    playerCharacter.Gold.ToString() };
```

Well that is it for this tutorial. I am already working on coding the next part of Eyes of the Dragon so I encourage you to keep either visiting my site http://xna.jtmbooks.com or my blog, http://xna-rpg.blogspot.com for the latest news on these tutorials.