Creating a Role Playing Game with XNA Game Studio 3.0
Part 19
Cleaning Things Up

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 18 You can download the graphics for the these tutorials at this link: Graphics.zip

I will be doing a lot in this tutorial. I will be cleaning things up and make a lot of changes to the game. I will also be changing a few screens. The first thing to do is to add a new folder to the project called **TileEngine**. I will be moving everything that has to do with the tile engine into this folder. Open the **CoreComponents** folder and drag the **TileEngine**, **TileMap**, **TileMapLayer** and **Camera** file to the **TileEngine** folder. Also drag the **TileSet** file into this folder. Now open each of the files and change the namespace to **New2DRPG**. I was going to change it to **New2DRPG.TileEngine** but the namespace would conflict with the **TileEngine** class.

I have made a change to the **GameScreen** class. What I have done is added in a **protected SpriteBatch** object and a **ContentManager** object. This way I they will be available to any screen that might need them. I also made the list of child components **protected** so they will be available to any chlild classes. I also added in a **protected Game** object that I set in the constructor so it will be available to any child classes. This is the code for the new **GameScreen** 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 class GameScreen : Microsoft.Xna.Framework.DrawableGameComponent
    {
        protected List<GameComponent> childComponents;
        protected SpriteBatch spriteBatch;
        protected ContentManager Content;
        protected Game game;
```

```
public GameScreen(Game game)
    : base(game)
{
    this.game = game;
    spriteBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));
    Content =
        (ContentManager)Game.Services.GetService(typeof(ContentManager));
    childComponents = new List<GameComponent>();
    Visible = false;
    Enabled = false;
}

public List<GameComponent> Components
{
    get { return childComponents; }
}

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

    base.Initialize();
}

public override void Update(GameTime gameTime)
{
    foreach (GameComponent child in childComponents)
    {
        if (child.Enabled)
        {
            child.Update(gameTime);
        }
    }

    base.Update(gameTime);
}

public override void  Draw(GameTime gameTime)
{
    foreach (GameComponent child in childComponents)
    {
        if ((child is DrawableGameComponent) &&
            ((DrawableGameComponent) child).Visible)
        {
            ((DrawableGameComponent) child).Draw(gameTime);
        }
    }
      base.Draw(gameTime);
}
public virtual void Show()
{
    Visible = true;
    Enabled = true;
}

public virtual void Hide()
{
    Visible = false;
```

```
            Enabled = false;
        }
    }
}
```

I am going to make a change to the **BackgroundComponent** next. I am going to add a **bool** parameter to the constructor called **fill**. If **fill** is true the image will fill the screen. If it is false the image will be centered on the screen. This will simplfy a few of the other screens. As always I will show you the new code and then explain the changes. This is the new **BackgroundComponent**.

```
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 class BackgroundComponent :
            Microsoft.Xna.Framework.DrawableGameComponent
    {
        Texture2D background;
        SpriteBatch spriteBatch = null;
        Rectangle bgRect;

        public BackgroundComponent(Game game, Texture2D texture, bool fill)
            : base(game)
        {
            this.background = texture;
            spriteBatch =
                (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));
            if (fill)
            {
                bgRect = new Rectangle(0,
                            0,
                            Game.Window.ClientBounds.Width,
                            Game.Window.ClientBounds.Height);
            }
            else
            {
                bgRect = new Rectangle(
                    (Game.Window.ClientBounds.Width - texture.Width) / 2,
                    (Game.Window.ClientBounds.Height - texture.Height) / 2,
                    texture.Width,
                    texture.Height);
            }
        }
```

```
        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)
        {
            spriteBatch.Draw(background, bgRect, Color.White);
            base.Draw(gameTime);
        }

    }
}
```

The constructor was changed to take a **bool** parameter. In the construtor there if-esle statement. If the **bool** parameter is **true** then I create a rectangle that will fill the screen using the height and the width of the screen. If it is set to **false** then I create a rectangle that has the height and the width of the texture. Then I center the texture on the screen. To center something on the screen you take the width of the screen and subtract the width of the texture and divide that by 2. That would give you the X coordinate of the screen. For the Y coordinate you take the height of the screen and subtract the height of the texture and divide that by 2.

Now I had to change all of the other screens to use the new changes. I also made a few other changes to the screens. The **CreatePCScreen** will have quite a few changes to it so I will deal with it after I have added in everything else. I will start with the **IntroScreen**. This is the code for the new **IntroScreen** class.

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

namespace New2DRPG
{
    class IntroScreen : GameScreen
    {
        Texture2D introduction;
        TimeSpan myTimeSpan;
        Vector2 position;
        bool fadeFinished = false;
        bool topReached = false;
        byte alphaValue = 254;
        Color tintColor = Color.White;
        Texture2D background;

        public IntroScreen(Game game)
            : base(game)
        {
            LoadContent();
            Components.Add(new BackgroundComponent(game, background, true));
            position = new Vector2(0, Game.Window.ClientBounds.Height);
        }

        protected override void LoadContent()
        {
            background = Content.Load<Texture2D>(@"Backgrounds\introbackground");
            introduction = Content.Load<Texture2D>(@"Backgrounds\introduction");
            base.LoadContent();
        }
        public bool IntroFinished
        {
            get { return fadeFinished; }
        }

        public override void Update(GameTime gameTime)
        {
            myTimeSpan += gameTime.ElapsedGameTime;
            if (myTimeSpan > TimeSpan.FromMilliseconds(25) && !topReached)
            {
                position.Y -= 2.0f;
                myTimeSpan -= TimeSpan.FromMilliseconds(25);
                if (position.Y < 0.0f)
                {
                    position.Y = 0.0f;
                    topReached = true;
                }
            }
            if (myTimeSpan > TimeSpan.FromMilliseconds(15) && topReached)
            {
                if (alphaValue > 0)
                    alphaValue--;
                else
```

```
                    fadeFinished = true;
                tintColor.A = alphaValue;
                myTimeSpan -= TimeSpan.FromMilliseconds(15);
            }
            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);
            spriteBatch.Draw(introduction, position, tintColor);
        }
    }
}
```

The first thing that I did was add in a **Texture2D** field for the background image of the screen. I also add in a **LoadContent** method. After loading in the content for the screen I created a **BackgroundComponent** and passed **true** as the parameter for the **fill** parameter. Those were all of the changes to this class.

I will deal with the **HelpScreen** next. This is the new **HelpScreen** class.

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

namespace New2DRPG
{
    class HelpScreen : GameScreen
    {
        ButtonMenu buttonMenu;
        Texture2D background;
        Texture2D buttonImage;
        SpriteFont spriteFont;

        public HelpScreen(Game game)
            : base(game)
        {
            LoadContent();
            Components.Add(new BackgroundComponent(game, background, true));

            string[] items = { "RETURN TO MENU" };

            buttonMenu = new ButtonMenu(game, spriteFont, buttonImage);
            buttonMenu.SetMenuItems(items);
            Components.Add(buttonMenu);

            base.Hide();
        }

        protected override void LoadContent()
        {
            background = Content.Load<Texture2D>(@"Backgrounds\helpscreen");
            buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackground");
```

```
            spriteFont = Content.Load<SpriteFont>(@"normal");
            base.LoadContent();
        }

        public int SelectedIndex
        {
            get { return buttonMenu.SelectedIndex; }
        }

        public override void Show()
        {
            buttonMenu.Position = new Vector2((Game.Window.ClientBounds.Width -
                                    buttonMenu.Width) / 2, 700);
            base.Show();
        }

        public override void Hide()
        {
            base.Hide();
        }
    }
}
```

The first thing that I did was add in fields for the background image, the image for the button menu and the **SpriteFont**. I changed the constructor to have only one parameter, the **Game** object. I added in a **LoadContent** method to load in the content for the screen. In the constructor when I add the **BackgroundComponent** I again pass **true** for the **fill** parameter. I of course had to change the **LoadHelpScreen** method in the **Game1** class. I will deal with all of the changed methods in the **Game1** class later on. I also moved the **helpscreen.jpg** file to the **Backgrounds** folder.

I also changed the **CreditScreen** class. I will give you the code for the new **CreditScreen** class and then explain the changes. They are pretty similar to the previous two screens.

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

namespace New2DRPG
{
    class CreditScreen : GameScreen
    {
        Texture2D background;
        Texture2D credits;
        Texture2D buttonImage;
        SpriteFont spriteFont;
        TimeSpan myTimeSpan;
        Vector2 position;
        bool fadeFinished = false;
        byte alphaValue = 0;
        Color tintColor = Color.White;
        ButtonMenu buttonMenu;
```

```
        public CreditScreen(Game game)
            : base(game)
        {
            LoadContent();
            Components.Add(new BackgroundComponent(game, background, true));
            string[] items = { "RETURN TO MENU" };

            buttonMenu = new ButtonMenu(game, spriteFont, buttonImage);
            buttonMenu.SetMenuItems(items);
            Components.Add(buttonMenu);

            position = new Vector2(0, 0);
        }

        protected override void LoadContent()
        {
            background = Content.Load<Texture2D>(@"Backgrounds\creditsbackground");
            credits = Content.Load<Texture2D>(@"Backgrounds\credits");
            buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackground");
            spriteFont = Content.Load<SpriteFont>("normal");
            base.LoadContent();
        }

        public override void Update(GameTime gameTime)
        {
            myTimeSpan += gameTime.ElapsedGameTime;
            if (myTimeSpan > TimeSpan.FromMilliseconds(15))
            {
                if (alphaValue < 254)
                    alphaValue++;
                tintColor.A = alphaValue;
                myTimeSpan -= TimeSpan.FromMilliseconds(15);
            }
            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);
            spriteBatch.Draw(credits, position, tintColor);
        }

        public override void Show()
        {
            buttonMenu.Position = new Vector2((Game.Window.ClientBounds.Width -
                                        buttonMenu.Width) / 2, 700);
            alphaValue = 0;
            base.Show();
        }
    }
}
```

Again I added in a **LoadContent** method to load in the textures and the font. I also added in the fields to load these in to. I changed the constructor to only take the **Game** object as a parameter. In the constructor I no longer needed to get the **SpriteBatch** object or the **ContentManager** object. I called the **LoadContent** method to load in the textures and the font. I added the **BackgroundComponent** with the **fill** parameter set to **true** to fill the screen. Again I had to change the **LoadCreditScreen** method.

Next I changed the **StartScreen** class. I did the same thing as I did with the other screens. This is the code for the new **StartScreen**.

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

namespace New2DRPG
{
    class StartScreen : GameScreen
    {
        ButtonMenu buttonMenu;
        SpriteFont spriteFont;
        Texture2D background;
        Texture2D buttonImage;

        public StartScreen(Game game)
            : base(game)
        {
            LoadContent();
            Components.Add(new BackgroundComponent(game, background, true));

            string[] items = { "THE STORY BEGINS",
                               "THE STORY CONTINUES",
                               "HELP",
                               "CREDITS",
                               "QUIT" };

            buttonMenu = new ButtonMenu(
                game,
                spriteFont,
                buttonImage);

            buttonMenu.SetMenuItems(items);
            Components.Add(buttonMenu);
        }

        public int SelectedIndex
        {
            get { return buttonMenu.SelectedIndex; }
        }

        protected override void LoadContent()
        {
            background = Content.Load<Texture2D>(@"Backgrounds\titlescreen");
            buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackground");
            spriteFont = Content.Load<SpriteFont>(@"normal");
            base.LoadContent();
        }

        public override void Show()
        {
            buttonMenu.Position = new Vector2((Game.Window.ClientBounds.Width -
```

```
                                        buttonMenu.Width) / 2, 450);
            base.Show();
        }

        public override void Hide()
        {
            base.Hide();
        }
    }
}
```

I did pretty much the same thing that I did with the other screens. I added in fields for the textures that are needed and a field for the font. I add in a **LoadContent** method to load in the textures and the font. I changed the constructor to only take the **Game** object as a parameter. I called the **LoadContent** method in the constructor. When I added the **BackgroundComponent** I passed **true** as the **fill** parameter so the background would fill the screen. I also had to change the **Game1** class to reflect the changes I made to the constructor. Again, I will deal with all of the changes to the **Game1** class later. I also moved the **titlescreen.jpg** file from the root folder in the **Content** folder to the **Backgrounds** folder.

The next screen that I worked on was the **CreatePCScreen**. Again, I did pretty much the same thing. I will give you the new code and then go over it. This is the **CreatePCScreen** code.

```
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;

namespace New2DRPG
{
    class CreatePCScreen : GameScreen
    {
        ButtonMenu buttonMenu;

        string name = "Evander";
        bool gender = false;
        int difficultyLevel = 1;
        int className = 0;

        Texture2D buttonImage;
        Texture2D background;

        int screenWidth;
        int screenHeight;
        PlayerCharacter.CharClass characterClass =
            PlayerCharacter.CharClass.Fighter;

        string[] classNames = PlayerCharacter.ClassNames;

        string[] menuItems = {
                  "FORSEE HERO'S NAME",
                  "FORSEE HERO'S GENDER",
                  "FORSEE HERO'S CLASS",
                  "FORSEE DIFFICULTY LEVEL",
                  "BACK TO MENU",
                  "BEGIN THE ADVENTURE" };

        string[] difficultyLevels = {
                    "Easy",
                    "Normal",
                    "Hard",
                    "Ultimate" };

        SpriteFont spriteFont;

        public CreatePCScreen(Game game)
            : base(game)
        {
            LoadContent();

            Components.Add(new BackgroundComponent(game, background, true));

            buttonMenu = new ButtonMenu(game, spriteFont, buttonImage);
            buttonMenu.SetMenuItems(menuItems);
            Components.Add(buttonMenu);

            screenWidth = Game.Window.ClientBounds.Width;
            screenHeight = Game.Window.ClientBounds.Height;
```

```
}

public PlayerCharacter.CharClass CharacterClass
{
    get { return characterClass; }
}

public bool CharacterGender
{
    get { return gender; }
}

public string CharacterName
{
    get { return name; }
}

public int SelectedIndex
{
    get { return buttonMenu.SelectedIndex; }
}

protected override void LoadContent()
{
    background = Content.Load<Texture2D>(@"Backgrounds\creatpcscreen");
    buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackground");
    spriteFont = Content.Load<SpriteFont>("normal");
    base.LoadContent();
}

public void ChangeName(string name)
{
    this.name = name;
}

public void ChangeGender(bool gender)
{
    this.gender = gender;
}

public void ChangeDifficulty(int difficultyLevel)
{
    this.difficultyLevel = difficultyLevel;
}

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

```
                case 3:
                    characterClass = PlayerCharacter.CharClass.Thief;
                    break;
            }
        }

        public override void Show()
        {
            buttonMenu.Position = new Vector2((screenWidth -
                              buttonMenu.Width) / 2,
                              screenHeight - buttonMenu.Height - 10);
            base.Show();
        }

        public override void Draw(GameTime gameTime)
        {
            Vector2 position = new Vector2();
            string characterString;

            characterString = name + " the ";

            base.Draw(gameTime);

            characterString += classNames[className];
            Vector2 stringSize = spriteFont.MeasureString(characterString);
            position.X = (screenWidth - stringSize.X) / 2;
            position.Y = 280;

            spriteBatch.DrawString(spriteFont,
                characterString,
                position + Vector2.One * 3,
                Color.Black);

            spriteBatch.DrawString(spriteFont,
                characterString,
                position,
                Color.White);

            characterString = "Playing in " + difficultyLevels[difficultyLevel];
            characterString += " mode";
            stringSize = spriteFont.MeasureString(characterString);
            position.X = (screenWidth - stringSize.X) / 2;
            position.Y += spriteFont.LineSpacing;

            spriteBatch.DrawString(spriteFont,
                characterString,
                position + Vector2.One * 3,
                Color.Black);

            spriteBatch.DrawString(spriteFont,
                characterString,
                position,
                Color.White);
        }
    }
}
```

The first thing I did was move the **createpcscreen.jpg** file to the **Backgrounds** folder in the **Content** folder. I removed the **Content** and **spriteBatch** fields because they are both now in the parent

class **GameScreen**. I added in a **Texture2D** field to hold the background for the screen. I removed the **SpriteFont** parameter from the constructor so that the constructor only takes a **Game** object as a parameter. In the constructor I no longer had to get the **ContentManager** or **SpriteBatchObjects**. When I create the **BackgroundComponent** for this screen I again passed **true** for the **fill** parameter so the background would fill the screen.

I have also decided to replace all of the **PopUpScreens** with individual screens that derive from **GameScreen**. I will be starting with the **quitPopUpScreen**. All of these screens are pretty much the same. All that will be different in them is the menu items for the menu and the texture that is loaded in for background image. The one for chosing the class of the player character will also be some what different. After I give you the code for the new screens I will give a quick explanation of them.

The first screen that I will be adding is the **QuitGameScreen**. This screen is the screen that will be displayed when the player chooses the **Quit Game** option from the **StartScreen**. Right click the **Screens** folder and add a new class called **QuitGameScreen**. This is the code for the **QuitGameScreen**.

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

namespace New2DRPG
{
    class QuitGameScreen : GameScreen
    {
        ButtonMenu menu;
        Texture2D image;
        Texture2D buttonImage;
        SpriteFont spriteFont;
        Vector2 imagePosition;
        public QuitGameScreen(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 = { "YES", "NO" };
            menu = new ButtonMenu(game, spriteFont, buttonImage);
            menu.SetMenuItems(items);
            Components.Add(menu);
        }

        public int SelectedIndex
        {
            get { return menu.SelectedIndex; }
        }
```

```
        protected override void LoadContent()
        {
            image = Content.Load<Texture2D>(@"GUI\quitpopupbackground");
            buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackgroundshort");
            spriteFont = Content.Load<SpriteFont>(@"normal");
            base.LoadContent();
        }

        public override void Show()
        {
            base.Show();
            menu.Position = new Vector2((image.Width -
                                menu.Width) / 2 + imagePosition.X,
                                image.Height - menu.Height - 10 + imagePosition.Y);
        }
    }
}
```

As you can see the screen is pretty much the same as the old **PopUpScreen** with elements from the new **StartScreen**. The first thing I did was change the namespace from **New2DRPG.Screens** to **New2DRPG**. There is a field in the class for a **ButtonMenu**, a **Texture2D** for the image and the button, the **SpriteFont** and a **Vector2**. The **Vector2** holds where to draw the **ButtonMenu** of the screen.

The constructor for the screen only takes one parameter, the **Game** object. The constructor then calls the **LoadContent** method to load in the content for the screen. After calling **LoadContent** it then adds a **BackgroundComponent** passing in **false** as the **fill** parameter. This will center the image on the screen. It then finds where the image will be positioned on the screen using the same method that I used in the **BackgroundComponent** constructor. It then creates a **ButtonMenu** and adds it to the list of components.

There is a get only property to get the current menu item that is selected **SelectedIndex**. The **LoadContent** method loads in the proper images and the font. I made an override of the **Show** method to position the menu using the **Vector2** the holds where the image is being drawn on the screen. It is a pretty simple screen. I didn't have to override any other methods for these screens.

The next screen I added was a screen to select the gender of the player character. It is identical to the other screen except in two respects. The first is that instead of **Yes** and **No** for menu items the items are **Female** and **Male**. The other difference is that I load in the texture to ask what gender the player character will be. Right click the **Screens** folder and add a class called **GenderPopUpScreen**. Since I have already explained the differences I will just give you the code for the new screen.

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

namespace New2DRPG
{
    class GenderPopUpScreen : GameScreen
```

```
    {
        ButtonMenu menu;
        Texture2D image;
        Texture2D buttonImage;
        SpriteFont spriteFont;
        Vector2 imagePosition;

        public GenderPopUpScreen(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 = { "Female", "Male" };
            menu = new ButtonMenu(game, spriteFont, buttonImage);
            menu.SetMenuItems(items);
            Components.Add(menu);
        }

        public int SelectedIndex
        {
            get { return menu.SelectedIndex; }
        }

        protected override void LoadContent()
        {
            image = Content.Load<Texture2D>(@"GUI\maleorfemale");
            buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackgroundshort");
            spriteFont = Content.Load<SpriteFont>(@"normal");
            base.LoadContent();
        }

        public override void Show()
        {
            base.Show();
            menu.Position = new Vector2((image.Width -
                                menu.Width) / 2 + imagePosition.X,
                                image.Height - menu.Height - 10 + imagePosition.Y);
        }
    }
}
```

The next screen I added was the **ClassPopUpScreen** that will be used to select the class of the player character. Right click the **Screens** folder and add a new class called **ClassPopUpScreen.** While almost the same as the others there is a difference in how I got the menu items. I used the **enum** in the **PlayerCharacter** class to get class names. There is a class **System.Enum**. It has a method that you can use this class to get the names of an **enum**. For example, the enum has the values **Fighter**, **Wizard**, **Priest** and **Thief**. You can use the method **Enun.GetNames** to get the actual words used: **Fighter**, **Wizard**, **Priest** and **Thief** in the **enum**. This is the code for the **ClassPopUpScreen** class.

```
using System;
using System.Collections.Generic;
```

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

namespace New2DRPG
{
    class ClassPopUpScreen : GameScreen
    {
        ButtonMenu menu;
        Texture2D image;
        Texture2D buttonImage;
        SpriteFont spriteFont;
        Vector2 imagePosition;

        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(PlayerCharacter.CharClass));

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

        public int SelectedIndex
        {
            get { return menu.SelectedIndex; }
        }

        protected override void LoadContent()
        {
            image = Content.Load<Texture2D>(@"GUI\chooseclass");
            buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackgroundshort");
            spriteFont = Content.Load<SpriteFont>(@"normal");
            base.LoadContent();
        }

        public override void Show()
        {
            base.Show();
            menu.Position = new Vector2((image.Width -
                                menu.Width) / 2 + imagePosition.X,
                                image.Height - menu.Height - 10 + imagePosition.Y);
        }
    }
}
```

There is one **PopUpScreen** left that just has a **ButtonMenu**, the **PopUpScreen** that is used to

select the difficulty level. Right click the **Screens** folder and add a new class **DifficultyPopUpScreen**. For this class I added a new **enum** to **PlayerCharacter** for the difficulty level called **Difficulty**. I made another change to the **PlayerCharacter** class as well. It was suggested to me to make the player's hit points and spell points partly transparent so I did that. I will give you the new **PlayerCharacter** class a little later. The only other change was that I loaded in the image that asks the player to choose a difficulty level. The is the code for the **DifficultyPopUpScreen**.

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

namespace New2DRPG
{
    class DifficultyPopUpScreen : GameScreen
    {
        ButtonMenu menu;
        Texture2D image;
        Texture2D buttonImage;
        SpriteFont spriteFont;
        Vector2 imagePosition;

        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(PlayerCharacter.Difficulty));

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

        public int SelectedIndex
        {
            get { return menu.SelectedIndex; }
        }

        protected override void LoadContent()
        {
            image = Content.Load<Texture2D>(@"GUI\choosedifficulty");
            buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackgroundshort");
            spriteFont = Content.Load<SpriteFont>(@"normal");
            base.LoadContent();
        }

        public override void Show()
```

```
        {
            base.Show();
            menu.Position = new Vector2((image.Width -
                            menu.Width) / 2 + imagePosition.X,
                            image.Height - menu.Height - 10 + imagePosition.Y);
        }
    }
}
```

In the **PlayerCharacter** class I added in an **enum** that will hold the different difficulty levels. I also changed the **Draw** method to draw the hit points and spell points semi-transparent. To make them semi-transparent I created a variable called **tintColor** which is a **Color**. To have no tint for an item that is drawn you use **Color.White**. I set the **tintColor** variable to **Color.White** and then I set the alpha-channel of the color to 128, 50% transparency. Then when I was drawing instead of using **Color.White** I used **tintColor**. This is the code for the **PlayerCharacter** class.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
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
{
    class PlayerCharacter : Microsoft.Xna.Framework.DrawableGameComponent
    {
        public enum CharClass { Fighter = 0, Wizard = 1, Priest = 2, Thief = 3};
        public enum Difficulty { Easy, Normal, Hard, Ultimate };

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

        protected string name;
        protected bool gender;

        protected int[] hitPoints = new int[2];
        protected int[] spellPoints = new int[2];

        protected CharacterAbilities abilities = new CharacterAbilities();

        protected string className;
        protected Game game;

        protected SpriteBatch spriteBatch;
        protected ContentManager Content;

        protected Texture2D hpspDisplay;
        protected Texture2D blueBar;
        protected Texture2D redBar;

        protected SpriteFont spriteFont;
```

```
protected Vector2 hpspPosition;

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

public string Name
{
    get { return name; }
}

public string ClassName
{
    get { return className; }
}

public static string[] ClassNames
{
    get { return classNames; }
}

public int HitPointsMax
{
    get { return hitPoints[1]; }
}

public int HitPointsCurrent
{
    get { return hitPoints[0]; }
}

public int SpellPointsMax
{
    get { return spellPoints[1]; }
}

public int SpellPointsCurrent
{
    get { return spellPoints[0]; }
}

public virtual CharacterAbilities Abilities
{
    get { return abilities; }
}

public void Show()
```

```
        {
            Visible = true;
        }

        public void Hide()
        {
            Visible = false;
        }

        public override void Draw(GameTime gameTime)
        {
            DrawHitPointsSpellPoints();

            base.Draw(gameTime);
        }

        private void DrawHitPointsSpellPoints()
        {
            Vector2 barPosition = new Vector2();
            Vector2 textPosition = new Vector2();
            Vector2 stringSize;
            string text;
            Color tintColor = Color.White;
            tintColor.A = 128;

            spriteBatch.Draw(hpspDisplay, hpspPosition, tintColor);

            barPosition = hpspPosition + Vector2.One * 5;
            spriteBatch.Draw(redBar, barPosition, tintColor);

            textPosition = barPosition;
            textPosition.Y -= 5;
            text = hitPoints[0].ToString() + "/" + hitPoints[1].ToString();
            stringSize = spriteFont.MeasureString(text);
            textPosition.X += (200 - stringSize.X) / 2;
            textPosition.Y += (20 - spriteFont.LineSpacing) / 2;
            spriteBatch.DrawString(spriteFont, text, textPosition, tintColor);

            barPosition.Y += 19;
            spriteBatch.Draw(blueBar, barPosition, tintColor);

            textPosition = barPosition;
            textPosition.Y -= 5;
            text = spellPoints[0].ToString() + "/" + spellPoints[1].ToString();
            stringSize = spriteFont.MeasureString(text);
            textPosition.X += (200 - stringSize.X) / 2;
            textPosition.Y += (20 - spriteFont.LineSpacing) / 2;
            spriteBatch.DrawString(spriteFont, text, textPosition, tintColor);
        }
    }
}
```

Now I just have to give you the updates to the **Game1** class. There were quite a lot of them actually. I will give you all of the new code and then go over all of the changes. This is the new **Game1** 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;
using New2DRPG.CoreComponents;

namespace New2DRPG
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        StartScreen startScreen;
        CreatePCScreen createPCScreen;
        HelpScreen helpScreen;
        ActionScreen actionScreen;
        CreditScreen creditScreen;
        IntroScreen introScreen;

        QuitGameScreen quitPopUpScreen;
        GenderPopUpScreen genderPopUpScreen;
        ClassPopUpScreen classPopUpScreen;
        DifficultyPopUpScreen difficultyPopUpScreen;
        InputScreen nameInputScreen;

        GameScreen activeScreen;

        SpriteFont normalFont;

        Texture2D background;
        KeyboardState newState;
        KeyboardState oldState;

        PlayerCharacter playerCharacter;

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            graphics.PreferredBackBufferWidth = 1024;
            graphics.PreferredBackBufferHeight = 768;
            graphics.PreferredBackBufferFormat = SurfaceFormat.Bgr32;
            graphics.IsFullScreen = false;
            this.Window.Title = "Eyes of the Dragon";
            Content.RootDirectory = "Content";
        }

        protected override void Initialize()
        {
            base.Initialize();
```

```
}

protected override void LoadContent()
{
    spriteBatch = new SpriteBatch(GraphicsDevice);
    Services.AddService(typeof(SpriteBatch), spriteBatch);
    Services.AddService(typeof(ContentManager), Content);

    normalFont = Content.Load<SpriteFont>("normal");

    LoadCreatePCScreen();
    LoadStartScreen();
    LoadHelpScreen();
    LoadActionScreen();
    LoadQuitPopUpScreen();
    LoadGenderPopUpScreen();
    LoadClassPopUpScreen();
    LoadDifficultyPopUpScreen();
    LoadNameInputScreen();
    LoadCreditScreen();
    LoadIntroScreen();

    creditScreen.Hide();
    startScreen.Hide();
    helpScreen.Hide();
    createPCScreen.Hide();

    activeScreen = introScreen;
    activeScreen.Show();
}

private void LoadIntroScreen()
{
    introScreen = new IntroScreen(this);
    Components.Add(introScreen);
}

private void LoadCreditScreen()
{
    creditScreen = new CreditScreen(this);
    Components.Add(creditScreen);
}

private void LoadNameInputScreen()
{
    background = Content.Load<Texture2D>(@"GUI\choosename");
    nameInputScreen = new InputScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackgroundshort"));
    Components.Add(nameInputScreen);
}

private void LoadDifficultyPopUpScreen()
{
    difficultyPopUpScreen = new DifficultyPopUpScreen(this);
    Components.Add(difficultyPopUpScreen);
}
```

```
private void LoadClassPopUpScreen()
{
    classPopUpScreen = new ClassPopUpScreen(this);
    Components.Add(classPopUpScreen);
}

private void LoadGenderPopUpScreen()
{
    genderPopUpScreen = new GenderPopUpScreen(this);
    Components.Add(genderPopUpScreen);
    genderPopUpScreen.Hide();
}

private void LoadQuitPopUpScreen()
{
    quitPopUpScreen = new QuitGameScreen(this);
    Components.Add(quitPopUpScreen);
    quitPopUpScreen.Hide();
}

private void LoadActionScreen()
{
    actionScreen = new ActionScreen(this, normalFont, "tileset1");
    Components.Add(actionScreen);
    actionScreen.Hide();
}

private void LoadHelpScreen()
{
    helpScreen = new HelpScreen(this);
    Components.Add(helpScreen);
}

private void LoadStartScreen()
{
    startScreen = new StartScreen(this);
    Components.Add(startScreen);
}

private void LoadCreatePCScreen()
{
    createPCScreen = new CreatePCScreen(this);
    Components.Add(createPCScreen);
}

protected override void UnloadContent()
{
}

protected override void Update(GameTime gameTime)
{
    newState = Keyboard.GetState();

    if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
          ButtonState.Pressed)
        this.Exit();

    if (activeScreen == startScreen)
    {
```

```
            HandleStartScreenInput();
        }
        else if (activeScreen == helpScreen)
        {
            HandleHelpScreenInput();
        }
        else if (activeScreen == createPCScreen)
        {
            HandleCreatePCScreenInput();
        }
        else if (activeScreen == quitPopUpScreen)
        {
            HandleQuitPopUpScreenInput();
        }
        else if (activeScreen == genderPopUpScreen)
        {
            HandleGenderPopUpScreenInput();
        }
        else if (activeScreen == classPopUpScreen)
        {
            HandleClassPopUpScreenInput();
        }
        else if (activeScreen == difficultyPopUpScreen)
        {
            HandleDifficultyPopUpScreenInput();
        }
        else if (activeScreen == nameInputScreen)
        {
            HandleNameInputScreenInput();
        }
        else if (activeScreen == introScreen)
        {
            HandleIntroScreenInput();
        }
        else if (activeScreen == creditScreen)
        {
            HandleCreditScreenInput();
        }
        else if (activeScreen == actionScreen)
        {
            HandleActionScreeenInput();
        }
        oldState = newState;

        base.Update(gameTime);
    }

    private void HandleActionScreeenInput()
    {
        if (CheckKey(Keys.Escape))
        {
            this.Exit();
        }
    }

    private void HandleCreditScreenInput()
    {
        if (CheckKey(Keys.Enter))
        {
```

```
            activeScreen.Hide();
            activeScreen = startScreen;
            activeScreen.Show();
        }
    }

    private void HandleIntroScreenInput()
    {
        if (CheckKey(Keys.Escape) || CheckKey(Keys.Space) ||
              introScreen.IntroFinished)
        {
            activeScreen.Hide();
            activeScreen = startScreen;
            activeScreen.Show();
        }
    }

    private void HandleNameInputScreenInput()
    {
        if (CheckKey(Keys.Enter))
        {
            createPCScreen.ChangeName(nameInputScreen.Text);
            activeScreen.Hide();
            activeScreen = createPCScreen;
            activeScreen.Show();
        }
    }

    private void HandleDifficultyPopUpScreenInput()
    {
        if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
        {
            createPCScreen.ChangeDifficulty(
                    difficultyPopUpScreen.SelectedIndex);
            activeScreen.Hide();
            activeScreen = createPCScreen;
            activeScreen.Show();
        }
    }

    private void HandleClassPopUpScreenInput()
    {
        if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
        {
            createPCScreen.ChangeClass(classPopUpScreen.SelectedIndex);
            activeScreen.Hide();
            activeScreen = createPCScreen;
            activeScreen.Show();
        }
    }

    private void HandleGenderPopUpScreenInput()
    {
        if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
        {
            switch (genderPopUpScreen.SelectedIndex)
            {
                case 0:
                    createPCScreen.ChangeGender(true);
```

```
                activeScreen.Hide();
                activeScreen = createPCScreen;
                activeScreen.Show();
                break;
            case 1:
                createPCScreen.ChangeGender(false);
                activeScreen.Hide();
                activeScreen = createPCScreen;
                activeScreen.Show();
                break;
        }
    }
}

private void HandleQuitPopUpScreenInput()
{
    if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
    {
        switch (quitPopUpScreen.SelectedIndex)
        {
            case 0:
                this.Exit();
                break;
            case 1:
                activeScreen.Hide();
                activeScreen = startScreen;
                activeScreen.Show();
                break;
        }
    }
}

private void HandleHelpScreenInput()
{
    if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
    {
        switch (helpScreen.SelectedIndex)
        {
            case 0:
                activeScreen.Hide();
                activeScreen = startScreen;
                activeScreen.Show();
                break;
        }
    }
}

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,
                      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;
        }
    }
}

private void HandleCreatePCScreenInput()
{
    if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
    {
        switch (createPCScreen.SelectedIndex)
        {
            case 0:
                activeScreen.Enabled = false;
                activeScreen = nameInputScreen;
                activeScreen.Show();
                break;
            case 1:
                activeScreen.Enabled = false;
                activeScreen = genderPopUpScreen;
                activeScreen.Show();
                break;
            case 2:
                activeScreen.Enabled = false;
                activeScreen = classPopUpScreen;
                activeScreen.Show();
                break;
            case 3:
                activeScreen.Enabled = false;
                activeScreen = difficultyPopUpScreen;
                activeScreen.Show();
                break;
            case 4:
                activeScreen.Hide();
                activeScreen = startScreen;
                activeScreen.Show();
                break;
```

```
                case 5:
                    activeScreen.Hide();
                    CreatePlayerCharacter();
                    activeScreen = actionScreen;
                    activeScreen.Show();
                    break;
            }
        }
    }

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

    private bool CheckKey(Keys theKey)
    {
        return oldState.IsKeyDown(theKey) && newState.IsKeyUp(theKey);
    }

    /// <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);

        spriteBatch.Begin(SpriteBlendMode.AlphaBlend);
        base.Draw(gameTime);
        spriteBatch.End();
    }
}
}
```

As you can see I had to change the fields of the class to use the new screens. I also changed the constructor a little. I changed the **PreferedBackBufferFormat** to **Surface.Bgr32** to make sure that the screen will be running in 32-bit mode with each channel having 8 bits. 8 bits for each of the red, blue and green channels and 8 bits for the alpha channel.

Almost all of the **Load** methods have changed as well. Because I am now loading all of the content for the screens in the screens themselves I don't need to pass anything other than the **Game** object to the constructors.

I added in a new method called **HandleActionScreenInput** that will be called from the **Update** method if the active screen is set to the action screen. I removed the part in the **Update** method where the game will exit if the player presses the escape key. In the **HandleActionScreenInput** method the game will exit if the player presses the escape key.

Like I said this was a long tutorial but I feel that it needed to be done. 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.