Creating a Role Playing Game with XNA Game Studio 3.0
Part 10
Changing the Character Generator

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 9

I did quite a bit of programming on Eyes of the Dragon this afternoon. I did a major rewrite of the character generator. I would like to think it is a lot nicers than the old one as well as easier to use. The first thing I did was make a change to the **PopUpScreen** class. **Yes** and **No** questions are nice, but wouldn't it be better if you could have more than two items? Well, I did exactly that. All I had to do was add a public method to the **PopUpScreen** class to set the menu items outside of the class. The method takes an array of strings as a paramater. This is the code for the updated **PopUpScreen** class.

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

namespace New2DRPG
{
    class PopUpScreen : GameScreen
    {
        ButtonMenu menu;
        Texture2D image;
        SpriteBatch spriteBatch;
        Rectangle imageRectangle;

        public PopUpScreen(Game game, SpriteFont spriteFont, Texture2D image,
                Texture2D buttonImage)
            : base(game)
        {
            this.image = image;
            spriteBatch =
                (SpriteBatch)game.Services.GetService(typeof(SpriteBatch));
            imageRectangle = new Rectangle();
            imageRectangle.X = (game.Window.ClientBounds.Width - image.Width) / 2;
            imageRectangle.Y =
                (game.Window.ClientBounds.Height - image.Height) / 2;
            imageRectangle.Width = image.Width;
            imageRectangle.Height = image.Height;

            string[] items = { "YES", "NO" };
            menu = new ButtonMenu(game, spriteFont, buttonImage);
            menu.SetMenuItems(items);
            Components.Add(menu);
        }
        public int SelectedIndex
        {
            get { return menu.SelectedIndex; }
        }

        public void SetMenuItems(string[] items)
        {
            menu.SetMenuItems(items);
        }

        public override void Draw(GameTime gameTime)
        {
            spriteBatch.Draw(image, imageRectangle, Color.White);
            base.Draw(gameTime);
        }

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

As you can see, it is pretty much the same but with extending the number of items available it makes a much more powerful menu.

Next I made a lot of changes to the **CreatePCScreen**. It was pretty much a total rewrite. As I usually do, I will give you the new code and then try and explain why I did what I did. This is the code for the new **CreatePCScreen**.

```
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;

        int name = 0;
        bool gender = false;
        int difficultyLevel = 1;
        int className = 0;
        Texture2D buttonImage;
        int screenWidth;
        int screenHeight;
```

```
string[] classNames = {
          "Fighter",
          "Wizard",
          "Thief",
          "Priest" };

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" };

public static readonly string[] MaleNames = {
            "Aris",
            "Barton",
            "Evander",
            "Kalven",
            "Llelwyn" };

public static readonly string[] FemaleNames = {
           "Anwyn",
           "Bronwyn",
           "Cantrian",
           "Julia",
           "Zoey" };

SpriteFont spriteFont;
SpriteBatch spriteBatch;
ContentManager Content;

public CreatePCScreen(Game game, SpriteFont spriteFont)
    : base(game)
{
    spriteBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));
    Content =
        (ContentManager)Game.Services.GetService(typeof(ContentManager));
    this.spriteFont = spriteFont;
    Components.Add(new BackgroundComponent(game,
        Content.Load<Texture2D>("createpcscreen")));
    buttonImage = Content.Load<Texture2D>(@"GUI\buttonbackground");
    buttonMenu = new ButtonMenu(game, spriteFont, buttonImage);
    buttonMenu.SetMenuItems(menuItems);
    Components.Add(buttonMenu);

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

public bool Gender
{
    get { return gender; }
}

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

public void ChangeName(int 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;
}

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;

    if (gender)
    {
        characterString = FemaleNames[name] + " the ";
    }
    else
    {
        characterString = MaleNames[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 you will notice is I added in a using statement for the **Content** name space. Instead of passing in the background image and the image of the buttons, I decided to use the **LoadContent** method to get the textures. The next change was that I switched from a regular text menu to a graphic menu, to try and give the generator a nicer look like the rest of the game. There is a variable for the image of the button and a variable for the **ContentManager** that I will be using. I also changed the text of the menu, to try and give the character generator a little bit of atmosphere. I also made the names public, static and readonly. The reason I did that was because I wanted to be able to get the names outside of this class. I probably didn't need to make them static but I thought it would be easier that way.

In the contructor for the class, I retrieved the **ContentManager** object the same way I did in the **TileEngine** component. I create a background for the character generator called **createpcscreen.jpg**. I've placed it, and all other graphics I created for this tutorial, in the Graphics.zip file. I added it to the **Content** folder by right clicking it and selecting: **Add existing item**. I used the **ContentManager** to load in the background and create a **BackgroundComponent** and add it to the list of components for the screen. After that I create a button menu and it to the list of components for the screen. I also got the height and width of the screen so I could position the menu nicely. There are two fields **screenWidth** and **screenHeight** to hold the width and height of the screen. The are set in the constructor using the **ClientBounds** property of the **Window** property of the **Game** class.

I added a new property to the class as well for the gender of the character. I will use this property later when I get to the game. The old **ChangeGender**, **ChangeName**, **ChangeClass** and **ChangeDifficulty** methods were scrapped and replace with new ones. They now take parameters. The **ChangeGender** takes a bool that will set the gender. The **ChangeName**, **ChangeClass** and **ChangeDifficulty** methods both take integers. This integer will be used to select the name and the difficulty from the arrays for their respective arrays.

I changed the **Show** method to set the position of the **ButtonMenu**. I set the position using the height and width of the screen I got from the constructor.

The **Draw** method was also scrapped and replaced with a new one. You will see that I have two variables in the **Draw** method. The first one, a **Vector2**, holds the position where I will be drawing the text. The second one holds the string that I will be drawing. You will notice that I do the rendering after the call to **base.Draw**. If you don't, when **base.Draw** is called it will draw over everything you render. The position of the strings on the Y axis were arbitrary choices. They just looked good to me. The first string I will draw was created using the name from the list of names according to the gender the player selected. After that I added in the class that the player selected. I centered the string on the screen horizontally. I first draw a shadow of the text in black, slightly offsetted, and then the text in white. I added in the line spacing of the font to the Y coordinate of the position where I would draw the text. Then I centered the text horizontally on the screen as well. I again draw the shadow and the text in black and white.

There were quite a few changes to the **Game1** class. I will give you the code for the new **Game1** class and then go over 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;

        PopUpScreen quitPopUpScreen;
        PopUpScreen genderPopUpScreen;
        PopUpScreen classPopUpScreen;
        PopUpScreen difficultyPopUpScreen;
        PopUpScreen namePopUpScreen;

        GameScreen activeScreen;
```

```
SpriteFont normalFont;

Texture2D background;

KeyboardState newState;
KeyboardState oldState;

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

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

/// <summary>
/// LoadContent will be called once per game and is the place to load
/// all of your content.
/// </summary>
protected override void LoadContent()
{
    spriteBatch = new SpriteBatch(GraphicsDevice);
    Services.AddService(typeof(SpriteBatch), spriteBatch);
    Services.AddService(typeof(ContentManager), Content);

    normalFont = Content.Load<SpriteFont>("normal");
    createPCScreen = new CreatePCScreen(this, normalFont);
    Components.Add(createPCScreen);

    background = Content.Load<Texture2D>("titlescreen");
    startScreen = new StartScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackground"));
    Components.Add(startScreen);

    background = Content.Load<Texture2D>("helpscreen");
    helpScreen = new HelpScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackground"));
    Components.Add(helpScreen);

    actionScreen = new ActionScreen(this, normalFont, "tileset1");
    Components.Add(actionScreen);
    actionScreen.Hide();

    background = Content.Load<Texture2D>(@"GUI\quitpopupbackground");
    quitPopUpScreen = new PopUpScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackgroundshort"));
    Components.Add(quitPopUpScreen);
    quitPopUpScreen.Hide();

    background = Content.Load<Texture2D>(@"GUI\maleorfemale");

    genderPopUpScreen = new PopUpScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackgroundshort"));
    string[] genderItems = new string[] { "Female", "Male" };
    genderPopUpScreen.SetMenuItems(genderItems);
    Components.Add(genderPopUpScreen);

    background = Content.Load<Texture2D>(@"GUI\chooseclass");

    classPopUpScreen = new PopUpScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackgroundshort"));
    string[] classItems = new string[] { "Fighter",
                "Wizard",
                "Thief",
                "Priest" };

    classPopUpScreen.SetMenuItems(classItems);
    Components.Add(classPopUpScreen);

    background = Content.Load<Texture2D>(@"GUI\choosedifficulty");

    difficultyPopUpScreen = new PopUpScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackgroundshort"));

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

    difficultyPopUpScreen.SetMenuItems(difficultyItems);
    Components.Add(difficultyPopUpScreen);

    background = Content.Load<Texture2D>(@"GUI\choosename");

    namePopUpScreen = new PopUpScreen(this,
        normalFont,
        background,
        Content.Load<Texture2D>(@"GUI\buttonbackgroundshort"));
    namePopUpScreen.SetMenuItems(CreatePCScreen.MaleNames);
    Components.Add(namePopUpScreen);

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

    activeScreen = startScreen;
}
```

```
        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// all content.
        /// </summary>
        protected override void UnloadContent()
        {
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            newState = Keyboard.GetState();

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

            if (newState.IsKeyDown(Keys.Escape))
                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 == namePopUpScreen)
            {
                HandleNamePopUpScreenInput();
            }
            oldState = newState;

            base.Update(gameTime);
        }

        private void HandleNamePopUpScreenInput()
        {
            if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
            {
                createPCScreen.ChangeName(namePopUpScreen.SelectedIndex);
                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();
                activeScreen = actionScreen;
                actionScreen.Show();
                break;
            case 2:
                activeScreen.Hide();
                activeScreen = helpScreen;
                activeScreen.Show();
                break;
            case 3:
                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;
                        if (createPCScreen.Gender)
                            namePopUpScreen.SetMenuItems(CreatePCScreen.FemaleNames
);
                        else
                            namePopUpScreen.SetMenuItems(CreatePCScreen.MaleNames);
                        activeScreen = namePopUpScreen;
                        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();
                        activeScreen = actionScreen;
                        activeScreen.Show();
                        break;
                }
            }
        }

        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've added in four **PopUpScreens** to the game. Each of the pop ups will corrosponde to the menu choices for changing the gender, name, class and difficulty level of the player. To use these screens you will have to add four images to the **GUI** sub-folder in the **Content** folder. You can find all of the graphics in the Graphics.zip file on my web site.

The **LoadContent** method is where I create the **PopUpScreens**. For each screen, I load in the back ground for the screen. Then I create the screen. After creating the screen, I set the menu items. Then I add the new screen to the list of components in the game.

The next big change is in the **Update** method. In the **Update** method I have added in **else if** statements to handle the input for the different screens if they are the active screen.

The **HandleCreatePCScreenInput** method went through many changes. It still checks to see if the enter or space keys have been pressed. If the selected index of the menu for the **CreatePCScreen** menu is 0, the change name option, it disables **CreatePCScreen**. It then sets the menu items for the menu of the **NamePopUpScreen** to the names for the current gender. The active screen is set the the **NamePopUpScreen** and the screen is shown.

For the other three cases the the **CreatePCScreen** is disabled, the active screen is set to the pop up screen for the menu item and the menu item is shown.

The methods to handle the input for the different screens are basically the same code, except for the **HandleGenderPopUpScreenInput** method. In the **HandleNamePopUpScreenInput** method, the method checks to see if the enter or space key have been pressed. If they have I call the **ChangeName** method to change the name. I hide the screen, set the active screen to the **CreatePCScreen** and show the **CreatePCScreen**. I did pretty much the same thing in the **HandleDifficultyPopUpScreenInput** and **HandleClassPopUpScreenInput** methods. I just called the appropriate method to change the difficulty and class.

The **HandleGenderPopUpScreenInput** method is pretty much the same. It checks to see if the enter or space keys have been pressed. If they have it goes to a switch statement. In the switch statement it checks the selected index. If the selected index is 0, the index for female, it calls the **ChangeGender** method with true as the parameter, the value for female. It then hides the screen, sets the active screen to the **CreatePCScreen** and shows the active screen. The other case is identicle, except instead of passing true, it passes false.

Well, that is all for this tutorial. I will get to writing another one soon. Keep on coming back to the blog or the web site and I will try and have new stuff on both of them.