Creating a Role Playing Game with XNA Game Studio 3.0
Part 2
Adding a New Screen

In this tutorial I am going to show you how easy it is to extend the basic project and add a new screen and how easy it is to make changes to the project. If you are interested in downloading the project, you can find the latest version of the project on my web site at this link: New 2D RPG

To follow along with this tutorial you will have to have finished the previous tutorials. You can find the home page for this project here: http://www.jtmbooks.com/rpgtutorials/index.html In these tutorials I will have a link to the last version of the project. The previous version for this tutorial can be found at this link: New 2D RPG Part 1

So, let's get started. The first thing you will want to do is open the last version of the project. I will start with a few changes to **StartScreen Game Component**. All I am going to do is modify the menu a little. I am going to change it so that there are four options. One to start a new game, one to load an old game, one to view the help screen and one to exit the game. They way I designed the game, this is going to be super simple. All you have to do is change the menu items in the constructor. This is the new constructor:

```
public StartScreen(Game game, SpriteFont spriteFont, Texture2D background)
    : base(game)
{
    Components.Add(new BackgroundComponent(game, background));

    string[] items = { "The Story Begins", "The Story Continues",
                       "Help", "Quit" };
    menu = new MenuComponent(game, spriteFont);
    menu.SetMenuItems(items);
    Components.Add(menu);
}
```

That is all you need to do to update the **StartScreen Game Component**. In the **Game1** class you will have to do a little more work, but not much. I will get to that a little later.

To add a new screen to the game all you have to do is create a new class that will inherit from **GameScreen**. Right click the project and add a new class. Call it **CreatePCScreen**. As you can see from the name I will be using this class to create the character. For now it will be a little simple but it will easily be upgraded later. For now when you can cycle through five names for both male and female characters. You can switch the gender between male and female. You can also choose from one of four difficult levels. You can also return to the start menu and start the game. I will always give you the code and then explain why I did what I did. For changing the gender, I have two strings **Make Him A Woman** and **Make Her a Man**. When that menu item is selected, I just switch the string and toggle the gender between male and female.

```
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 CreatePCScreen : GameScreen
    {
        MenuComponent menu;
        int name = 0;
        bool gender = false;
        int difficultyLevel = 1;

        string[] menuItems = {
                  "Select New Name",
                  "Make Him a Woman",
                  "Change Difficulty Level",
                  "Return to Start Screen",
                  "Begin the Adventure" };

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

        string[] maleNames = {
                    "Aris",
                    "Barton",
                    "Evander",
                    "Kalven",
                    "Llelwyn" };

        string[] femaleNames = {
                   "Anwyn",
                   "Cantrinia",
                   "Julia",
                   "Lucy",
                   "Zoey" };

        SpriteFont spriteFont;
        SpriteBatch spriteBatch;

        public CreatePCScreen(Game game, SpriteFont spriteFont)
            : base(game)
        {
            spriteBatch =
                (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));
            this.spriteFont = spriteFont;
            menu = new MenuComponent(game, spriteFont);
            menu.SetMenuItems(menuItems);
            Components.Add(menu);
        }

        public int SelectedIndex
```

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

public int Name
{
    get { return name; }
}

public int DifficultyLevel
{
    get { return difficultyLevel; }
}
public bool Gender
{
    get { return gender; }
}

public void ChangeName()
{
    Random random = new Random();

    name++;

    if (name == femaleNames.GetLength(0))
        name = 0;
}

public void ChangeGender()
{
    if (gender)
        menuItems[1] = "Make Him a Woman";
    else
        menuItems[1] = "Make Her a Man";

    menu.SetMenuItems(menuItems);
    gender = !gender;
}

public void ChangeDifficulty()
{
    difficultyLevel++;
    if (difficultyLevel == difficultyLevels.GetLength(0))
        difficultyLevel = 0;
}

public override void Show()
{
    menu.Position = new Vector2((Game.Window.ClientBounds.Width -
                                 menu.Width) / 2, 200);
    base.Show();
}

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

```
            position = new Vector2(200, 100 - spriteFont.LineSpacing - 5);
            spriteBatch.DrawString(spriteFont,
                "Name",
                position,
                Color.Yellow);

            position.X = 340;
            spriteBatch.DrawString(spriteFont,
                "Gender",
                position,
                Color.Yellow);

            position.X = 500;
            spriteBatch.DrawString(spriteFont,
                "Game Mode",
                position,
                Color.Yellow);

            position = new Vector2(200, 100);

            if (gender)
            {
                spriteBatch.DrawString(spriteFont,
                    femaleNames[name],
                    position,
                    Color.White);

                position.X = 340;
                spriteBatch.DrawString(spriteFont,
                    "Female",
                    position,
                    Color.White);
            }
            else
            {
                spriteBatch.DrawString(spriteFont,
                    maleNames[name],
                    position,
                    Color.White);

                position.X = 340;
                spriteBatch.DrawString(spriteFont,
                    "Male",
                    position,
                    Color.White);
            }

            position.X = 500;
            spriteBatch.DrawString(spriteFont,
                difficultyLevels[difficultyLevel],
                position,
                Color.White);

            base.Draw(gameTime);
        }
    }
}
```

The first thing I had to do is add two using statements because the class needs to use XNA classes. One for the framework and one for the graphics. Next I inherited the class from **GameScreen**. This screen will have a menu so I added a **MenuComponent** to the class. For now, the character that I'm creating will only have a name, gender and difficulty level. I made an **int** to hold the index of the name in the list of names. I create a **bool** to hold the gender as it can only be one or the other. I also added an **int** to hold the difficulty level, I set this to 1 to start the player off at normal difficult level.

I also created a few arrays of strings. One for the menu, that will have the menu options. One for the difficulty levels. The levels are **Easy**, **Normal**, **Hard**, and **Ultimate**. The other ones hold the names for female and male characters. If you want to modify this list, make sure that you use the same number of names for each gender. If you don't you could get an exception that will crash the program.

Since the program will be writing text I needed a **SpriteFont** and a **SpriteBatch** object. I try to organize my code in such a way that fields come first, anything that isn't public first, then public. Next I try to have any constructors. After constructors I have properties followed by methods.

The constructor for this class needs two parameters, a **Game** object and a **SpriteFont**. When you are creating inherited classes you need to make sure that when you write the constructor you make a call to the base class's constructor. The **GameScreen** class needs a **Game** object.

In the constuctor, I had get the **SpriteBatch** object by using **Game.Services.GetService**. To use this, you need to register the service you want to use in the **Game1** class. I set the **SpriteFont**, create the **MenuComponent**, set the menu items and add the menu to the list of **GameComponents** for the screen.

I created a **get** property to get the selected index of the menu. As this is still just a start I have not made properties to get the name, gender or difficulty level. Those will be added later.

I wrote a method to go to the next name in the list, **ChangeName**. All it does is increment the **name** field and if it is outside of the number of names I set it to 0, the first name.

I wrote a method to switch the gender, **ChangeGender**. In this method, I switch the menu item, update the menu and switch the gender. If you easily want to switch a boolean variable, all you have to do is use the **!** operator.

I also wrote a method to change the difficulty level, **ChangeDifficulty**. It works the same way the **ChangeName** method.

I had to override the **Show** method. I did this because I need to set the position of the menu. After setting the position of the menu, I call the **Show** method of the base class.

The last thing I did in this class is override the **Draw** method. There really isn't much here that you haven't seen before. I created a **Vector2** to hold the position of where I wanted to draw a string. Then, using that position I draw the first string, update the position of the string and move to the next string. The only thing that might be difficult is the **if** statement with the **gender**. The **gender** determines what strings are drawn. If **gender** is true I draw Female and a female name, otherwise I draw Male and a

male name. Then I call the **Draw** method of the base class.

It is time to turn your attention back to the **Game1** class. I will post the entire **Game1** class and then explain it.

```
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;
        GameScreen activeScreen;

        SpriteFont normalFont;

        Texture2D background;

        KeyboardState newState;
        KeyboardState oldState;

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }


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

        protected override void LoadContent()
        {
            spriteBatch = new SpriteBatch(GraphicsDevice);
```

```
    Services.AddService(typeof(SpriteBatch), spriteBatch);

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

    background = Content.Load<Texture2D>("gryphon");
    startScreen = new StartScreen(this, normalFont, background);
    Components.Add(startScreen);

    background = Content.Load<Texture2D>("fire-dragon");
    helpScreen = new HelpScreen(this, background);
    Components.Add(helpScreen);

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

    activeScreen = startScreen;
}

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();
    }

    oldState = newState;

    base.Update(gameTime);
}

private void HandleHelpScreenInput()
{
      if (CheckKey(Keys.Space) || CheckKey(Keys.Enter) ||
          CheckKey(Keys.Escape))
    {
        activeScreen.Hide();
```

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

    private void HandleStartScreenInput()
    {
        if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
        {
            switch (startScreen.SelectedIndex)
            {
                case 0:
                    activeScreen.Hide();
                    activeScreen = createPCScreen;
                    activeScreen.Show();
                    break;
                case 2:
                    activeScreen.Hide();
                    activeScreen = helpScreen;
                    activeScreen.Show();
                    break;
                case 3:
                    Exit();
                    break;
            }
        }
    }

    private void HandleCreatePCScreenInput()
    {
        if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
        {
            switch (createPCScreen.SelectedIndex)
            {
                case 0:
                    createPCScreen.ChangeName();
                    break;
                case 1:
                    createPCScreen.ChangeGender();
                    break;
                case 2:
                    createPCScreen.ChangeDifficulty();
                    break;
                case 3:
                    activeScreen.Hide();
                    activeScreen = startScreen;
                    activeScreen.Show();
                    break;
                case 4:
                    activeScreen.Hide();
                    activeScreen = startScreen;
                    activeScreen.Show();
                    break;
            }
        }
    }
```

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

        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

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

The first change was to add a variable for the new screen, called **createPCScreen**. In the **LoadContent** method, after loading in the font I created the **createPCScreen** and added it to the list of **Game Compnents**. The next change is in the **Update** method. I added an **else if** to call a method to handle the input for the **createPCScreen** called **HandleCreatePCScreenInput**.

I had to change the **HandleStartScreenInput** method as I changed the menu. In the case that the menu index is one, I hide the active screen make the active screen to the **createPCScreen** and show the new active screen. I made sure to update the other cases to match the new menu.

The last new thing in this part is the **HandleCreatePCScreenInput** method. In this method I check to see if the **space** or **enter** key has been pressed. Then I use a **switch** statement on the **SelectedIndex** of the menu. For 0 I call the **ChangeName** method, 1 the **ChangeGender** method, 2 the **ChangeDifficulty** method, 3 I change back to the **StartScreen** and 4 I would call the next screen but I just went back to the **StartScreen**.

That is all for this tutorial. Check back again soon and I will try and have another ready soon.