Creating a Role Playing Game with XNA Game Studio 3.0
Part 18
Improving the Player Character Class
Changing the HUD

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

In this tutorial I will making a few changes to the **PlayerCharacter** class. I am going to make it a **DrawableGameComponent** and have the **PlayerCharacter** responsible for drawing itself. I still have a small HUD for the game, the height of one tile to display short messages to the player. I will be creating a pop up window later on in the game for displaying larger messages.

To get started today you will need to download the new graphics from the Graphics.zip file. I made four new images. The new short HUD(**characterhud.png**), two colored textures for the hit points(**redbar.jpg**) and spell points(**bluebar.jpg**) and a texture to frame the hit points and spell points(**hpspborder.png**). Add the HUD texture to the **Backgrounds** folder in the **Content** folder. The other three are added to the **GUI** folder. I also made a small change to the size of the font I added to the game last time. I changed it to 10.

```
    <Size>10</Size>
```

First I will deal with the change that I had to make to the tile engine. It was pretty simple. I just had to change the height of the view port.

```
static int viewPortHeight = 720;
```

Like I said I decided to make the **PlayerCharacter** class a **DrawableGameComponent** and have it draw itself instead of having the **ActionScreen**. I will give you the code for the new **PlayerCharacter** class then explain the new code.

```
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};

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

        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;

    spriteBatch.Draw(hpspDisplay, hpspPosition, Color.White);

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

```
            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, Color.White);

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

            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, Color.White);
        }
    }
}
```

I added in using statements for all of the XNA framework components. I probably didn't need them all but I thought they might come in handy later on. To make this a game component I inherited the class form **DrawableGameComponent**.

There are quite a few new protected variables in the class. Since I will be drawing things I needed a **SpriteBatch** object, **spriteBatch**. Instead of passing in textures and fonts I decided to let the component load in it's own content so I needed a **ContentManager** object, **Content**. There are three **Texture2D** variables in the class. There is one for the spell points bar, **blueBar**, on for the hit points bar, **redBar**, and one for the border of the hit points and spell points, **hpspDisplay**. There is a **SpriteFont** object, **spriteFont** and a **Vector2** called **hpspPosition**. This variable will be used later on in the game. I am going to give the player the ability to drag things like the hit point and spell point display around the screen. There is one other variable that I decided to add to the class. I thought it would be nice to have the **Game** object, **game**.

I had to add a constructor for the class. Like all other **DrawableGameComponents** you need to pass in the **Game** object that you are using, because the parent class needs the **Game** object as a parameter. Inside the constructor I set a few of the values of the class. I use the **GetServices** method to get the **SpriteBatch** and **ContentManager** objects that I added to the list of services in the main game class. I used the **ContentManager** object to load in the content needed for the class. I also set the position of the hit point and spell point display. I just picked the position of the top of my head as a nice place.

I added in two new methods, like all of the other **DrawableGameComponents**, to show and hide the component called **Show** and **Hide**. I also added in an override of the **Draw** method. All this method does is call another method to draw the player character's hit points and spell points, **DrawHitPointsSpellPoints**.

In the **DrawHitPointsSpellPoints** method. Like in the last tutorial this is just positioning things on the screen. First I draw the border for the hit points and spell points. Positioning the first bar is relatively easy. I created the texture so that the bar would start 5 pixels in and down. I take the position of the border and add in **Vector2.One** times five. **Vector2.One** has the values (1.0f, 1.0f) so multiplying it by 5 your will get (5.0f, 5.0f). Then I draw the red bar that represents hit points.

To find out where to draw the text for the hit points I take position of the bar and subtract 5 from the Y value. I then set a string to what I am going to draw. Instead of just drawing the number I decided to draw the current value and then maximum value, 10/20 for example. Then I center the text like I did in the last tutorial and draw the string.

Drawing the spell points is the same. To find where to draw the bar I added 19 to the Y value of the bar, because of the way I created the texture. Then it is just a matter of position the text.

Since I changed the **PlayerCharacter** class I also had to change the other classes that were inherited from it. I simply added in the using statements for all the XNA framework namespaces and then changed the constructors to take as a parameter the **Game** object and add a call to the base class' constructor using the **Game** object. Since the classes are still short I will give you the code for all of the classes. You will also notice that I gave the fighter and thief characters spell points. While they will not be able to cast spells I will be giving them special abilities that will use spell points. I am thinking of renaming spell points to mana points.

```
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 FighterCharacter : PlayerCharacter
```

```
    {
        const int startingHitPoints = 20;
        const int startingSpellPoints = 5;

        const int startingStrength = 16;
        const int startingStamina = 14;
        const int startingAgility = 12;
        const int startingSpeed = 10;
        const int startingIntellect = 10;
        const int startingLuck = 10;

        public FighterCharacter(string name, bool gender, Game game)
            : base(game)
        {
            this.className = "Fighter";
            this.name = name;
            this.gender = gender;

            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
            abilities.Luck = startingLuck;
        }

    }
}

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 PriestCharacter : PlayerCharacter
    {
        const int startingHitPoints = 15;
        const int startingSpellPoints = 15;


        const int startingStrength = 12;
        const int startingStamina = 12;
        const int startingAgility = 10;
```

```
        const int startingSpeed = 10;
        const int startingIntellect = 12;
        const int startingLuck = 12;

        public PriestCharacter(string name, bool gender, Game game)
            : base(game)
        {
            this.className = "Priest";
            this.name = name;
            this.gender = gender;
            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
            abilities.Luck = startingLuck;
        }
    }
}

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 ThiefCharacter : PlayerCharacter
    {
        const int startingHitPoints = 15;
        const int startingSpellPoints = 5;

        const int startingStrength = 12;
        const int startingStamina = 10;
        const int startingAgility = 14;
        const int startingSpeed = 12;
        const int startingIntellect = 10;
        const int startingLuck = 12;

        public ThiefCharacter(string name, bool gender, Game game)
            : base(game)
        {
            this.className = "Thief";
            this.name = name;
            this.gender = gender;
```

```
            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
            abilities.Luck = startingLuck;
        }
    }
}

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 WizardCharacter : PlayerCharacter
    {
        const int startingHitPoints = 10;
        const int startingSpellPoints = 20;

        const int startingStrength = 10;
        const int startingStamina = 12;
        const int startingAgility = 12;
        const int startingSpeed = 12;
        const int startingIntellect = 16;
        const int startingLuck = 10;

        public WizardCharacter(string name, bool gender, Game game)
            : base(game)
        {
            this.className = "Wizard";
            this.name = name;
            this.gender = gender;
            this.hitPoints[0] = startingHitPoints;
            this.hitPoints[1] = startingHitPoints;
            this.spellPoints[0] = startingSpellPoints;
            this.spellPoints[1] = startingSpellPoints;

            abilities.Strength = startingStrength;
            abilities.Stamina = startingStamina;
            abilities.Agility = startingAgility;
            abilities.Speed = startingSpeed;
            abilities.Intellect = startingIntellect;
```

```
            abilities.Luck = startingLuck;
        }

    }
}
```

I had to make a few changes to the **ActionScreen** class as well. There were quite a few so I will give you the entire new class. As I always do, I will explain the changes after I give you the code. This is the new code for the **ActionScreen** 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;
using Microsoft.Xna.Framework.Content;
using New2DRPG.SpriteClasses;

namespace New2DRPG
{
    class ActionScreen : GameScreen
    {
        SpriteFont gameFont;
        SpriteFont interfaceFont;
        Texture2D chest;
        Texture2D tilesetTexture;
        Texture2D characterHUDTexture;

        ContentManager Content;
        SpriteBatch spriteBatch;

        string tilesetName;
        TileEngine tileEngine;
        Tileset tileset;
        PlayerCharacter playerCharacter;
        CharacterAbilities abilityNames = new CharacterAbilities();

        int viewportWidth;
        int viewportHeight;
        int screenWidth;
        int screenHeight;

        public ActionScreen(Game game, SpriteFont gameFont, string tilesetName)
            : base(game)
        {
            playerCharacter = new PlayerCharacter(game);
            this.gameFont = gameFont;
            Content =
                (ContentManager)Game.Services.GetService(typeof(ContentManager));

            spriteBatch =
                (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

            this.tilesetName = tilesetName;
            LoadContent();

            tileset = new Tileset(tilesetTexture, 128, 128, 4, 4);
```

```
            tileEngine = new TileEngine(game, this.tileset, 50, 50);
            Components.Add(tileEngine);
            tileEngine.Show();

            viewportWidth = TileEngine.ViewPortWidth;
            viewportHeight = TileEngine.ViewPortHeight;
            screenWidth = game.Window.ClientBounds.Width;
            screenHeight = game.Window.ClientBounds.Height;
        }

        protected override void LoadContent()
        {
            base.LoadContent();
            tilesetTexture = Content.Load<Texture2D>(@"Tilesets\" + tilesetName);
            chest = Content.Load<Texture2D>(@"Items\chest");
            characterHUDTexture =
                  Content.Load<Texture2D>(@"Backgrounds\characterhud");
            this.interfaceFont = Content.Load<SpriteFont>("smallFont");
        }

        public override void Update(GameTime gameTime)
        {
            base.Update(gameTime);
        }

        public override void Draw(GameTime gameTime)
        {
            base.Draw(gameTime);

            Vector2 position = new Vector2(0, 0);
            position.Y = viewportHeight;
            spriteBatch.Draw(characterHUDTexture, position, Color.White);
            playerCharacter.Draw(gameTime);
        }

        public override void Show()
        {
            base.Show();
            Enabled = true;
            Visible = true;
            playerCharacter.Show();
        }

        public override void Hide()
        {
            base.Hide();
            Enabled = false;
            Visible = false;
            playerCharacter.Hide();
        }

        public void SetPlayerCharacter(PlayerCharacter playerCharacter)
        {
            this.playerCharacter = playerCharacter;
        }
    }
}
```

I removed everything that had to do with drawing the player's hit points and spell points, except

for the **SpriteFont** object. I thought I would use it later so I kept it in. In the constructor I create a **PlayerCharacter** object using the **Game** object.

In the **Draw** method, after drawing everything else I call the **Draw** method of the player character. In the **Hide** method I call the **Hide** method of the player character and in the **Show** method I call the **Show** method of the player character.

That just leaves the **Game1** class. I only had to make two changes to this class. The first was in the **HandleStartScreenInput** method. I needed to pass the **Game** object as a parameter to the **FighterCharacter** constructor. This is the updated method.

```
private void HandleStartScreenInput()
{
    if (CheckKey(Keys.Enter) || CheckKey(Keys.Space))
    {
        switch (startScreen.SelectedIndex)
        {
            case 0:
                activeScreen.Hide();
                activeScreen = createPCScreen;
                activeScreen.Show();
                break;
            case 1:
                activeScreen.Hide();
                playerCharacter = new FighterCharacter("Evander", false, 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;
        }
    }
}
```

The only other change that I had to make was in the **CreatePlayerCharacter** method. Where I had to pass in the **Game** object as well when I called the different constructors. This is the updated method.

```
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);
    }
```

Well, that is it for another tutorial. I will be working on more so I encourage you to keep either visiting this site or my blog, http://xna-rpg.blogspot.com for the latest news on these tutorials.