Eyes of the Dragon Tutorials
Part 39
Characters Revisited

I'm writing these tutorials for the MonoGame 3.8 framework using Visual Studio 2019. The tutorials will make more sense if they are read in order. You can find the list of tutorials on the Eyes of the Dragon page of my web blog. I will be making each version of the project available on GitHub here. It will be included on the page that links to the tutorials.

In this tutorial I will be revisiting characters. I will be adding a new layer to the map, a character layer, adding them in the editor, saving them and loading them back into the editor and into the game. To get started right click the TileEngine folder in the MGRpgLibrary project, select Add and then Class. Name this new class CharacterLayer. Here is the code for that class.

```
using MGRpgLibrary.CharacterClasses;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using System;
using System.Collections.Generic;
using System.Text;

namespace MGRpgLibrary.TileEngine
{
    public class CharacterLayer : ILayer
    {
        public Dictionary<Point, Character> Characters { get; private set; } = new
Dictionary<Point, Character>();

        public CharacterLayer()
        {
        }

        public void Draw(SpriteBatch spriteBatch, Camera camera, List<Tileset> tilesets)
        {
            foreach (Character c in Characters.Values)
            {
                c.Draw(new GameTime(), spriteBatch);
            }
        }

        public void Update(GameTime gameTime)
        {
            foreach (Character c in Characters.Values)
            {
                c.Update(gameTime);
            }
        }
    }
}
```

A very simple class. It implements the ILayer interface so that it can be added to the list of map layers. It has a single property, Characters, that is a Dictionary<Point, Character> that holds the characters on the map. Point is the tile the character is on and Character is of course the actual character. Originally I preferred having fields and exposing their values via properties. Working with MVC has me using properties more and more often. In this case the set is private because you don't want the field set outside of the class.

It has the two required methods of the ILayer interface. In the Draw method I iterate over all of the values in the dictionary. I call the Draw method of the character passing in a new instance of GameTime, because it is required by the interface, and the SpriteBatch object. In the Update method I again iterate over all of the values in the dictionary. I call the Update method of the character passing in the GameTime parameter passed to the method.

Before I get to creating characters I want to make a change to the EntityData class. What I need to do is add a level to the constructor and calculate the hit points, magic points and stamina points. Update the EntityData class to the following.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RpgLibrary.CharacterClasses
{
    public class EntityData
    {
        #region Field Region

        public string EntityName;
        public int Level;
        public int Strength;
        public int Dexterity;
        public int Cunning;
        public int Willpower;
        public int Magic;
        public int Constitution;
        public string HealthFormula;
        public string StaminaFormula;
        public string MagicFormula;

        #endregion

        #region Constructor Region

        private EntityData()
        {
        }

        public EntityData(
            string entityName,
            int level,
            int strength,
            int dexterity,
            int cunning,
            int willpower,
            int magic,
            int constitution,
            string health,
            string stamina,
            string mana)
        {
            EntityName = entityName;
            Level = level;
            Strength = strength;
            Dexterity = dexterity;
            Cunning = cunning;
            Willpower = willpower;
            Cunning = cunning;
            Willpower = willpower;
            Magic = magic;
            Constitution = constitution;
            HealthFormula = health;
            StaminaFormula = stamina;
            MagicFormula = mana;
        }

        #endregion

        #region Method Region

        public override string ToString()
        {
            string toString = EntityName + ", ";

            toString += Level.ToString() + ", ";
            toString += Strength.ToString() + ", ";
            toString += Dexterity.ToString() + ", ";
            toString += Cunning.ToString() + ", ";
            toString += Willpower.ToString() + ", ";
            toString += Magic.ToString() + ", ";
            toString += Constitution.ToString() + ", ";
            toString += HealthFormula + ", ";
            toString += StaminaFormula + ", ";
            toString += MagicFormula;

            return toString;
        }

        public object Clone()
        {
            EntityData data = new EntityData
            {
                EntityName = EntityName,
                Level = Level,
                Strength = Strength,
                Dexterity = Dexterity,
                Cunning = Cunning,
                Willpower = Willpower,
                Magic = Magic,
                Constitution = Constitution,
                HealthFormula = HealthFormula,
                StaminaFormula = StaminaFormula,
                MagicFormula = MagicFormula
            };

            return data;
        }

        #endregion
```

```
    }
}
```

All that I did was add a new field Level to the fields. I added an argument to the constructor for the level of the entity. I updated the ToString override to append the level after the name. In the Clone method I set the level of the new entity to the level of the original entity.

This change breaks the LoadGameScreen and the CharacterGeneratorScreen because creating the entity now requires a level argument. First, change the CreateWorld method of the LoadGameScreen to the following.

```
        private void CreateWorld()
        {
            Texture2D tilesetTexture = Game.Content.Load<Texture2D>(@"Tilesets\tileset1");
            Tileset tileset1 = new Tileset(tilesetTexture, 8, 8, 32, 32);

            tilesetTexture = Game.Content.Load<Texture2D>(@"Tilesets\tileset2");

            Tileset tileset2 = new Tileset(tilesetTexture, 8, 8, 32, 32);

            tilesetTexture = Game.Content.Load<Texture2D>(@"Tilesets\fire-tiles");

            AnimatedTileset animatedSet = new AnimatedTileset(tilesetTexture, 8, 1, 64, 64);
            MapLayer layer = new MapLayer(100, 100);

            for (int y = 0; y < layer.Height; y++)
            {
                for (int x = 0; x < layer.Width; x++)
                {
                    Tile tile = new Tile(0, 0);
                    layer.SetTile(x, y, tile);
                }
            }

            MapLayer splatter = new MapLayer(100, 100);
            Random random = new Random();

            for (int i = 0; i < 100; i++)
            {
                int x = random.Next(0, 100);
                int y = random.Next(0, 100);
                int index = random.Next(2, 14);

                Tile tile = new Tile(index, 0);

                splatter.SetTile(x, y, tile);
            }

            splatter.SetTile(5, 0, new Tile(14, 0));
            splatter.SetTile(1, 0, new Tile(0, 1));
            splatter.SetTile(2, 0, new Tile(2, 1));
            splatter.SetTile(3, 0, new Tile(0, 1));

            TileMap map = new TileMap(tileset1, animatedSet, layer);

            map.AddTileset(tileset2);
            map.AddLayer(splatter);
            map.CollisionLayer.Collisions.Add(new Point(1, 0), CollisionType.Impassable);
            map.CollisionLayer.Collisions.Add(new Point(3, 0), CollisionType.Impassable);
            map.AnimatedTileLayer.AnimatedTiles.Add(new Point(5, 0), new AnimatedTile(0, 8));

            Level level = new Level(map);

            ChestData chestData = Game.Content.Load<ChestData>(@"Game\Chests\Plain Chest");

            Chest chest = new Chest(chestData);

            BaseSprite chestSprite = new BaseSprite(
                containers,
                new Rectangle(0, 0, 32, 32),
                new Point(10, 10));
            ItemSprite itemSprite = new ItemSprite(
            chest,
            chestSprite);

            level.Chests.Add(itemSprite);

            World world = new World(GameRef, GameRef.ScreenRectangle);

            world.Levels.Add(level);
            world.CurrentLevel = 0;

            AnimatedSprite s = new AnimatedSprite(

            GameRef.Content.Load<Texture2D>(@"SpriteSheets\Eliza"),

            AnimationManager.Instance.Animations);

            s.Position = new Vector2(5 * Engine.TileWidth, 5 * Engine.TileHeight);

            EntityData ed = new EntityData("Eliza", 1, 10, 10, 10, 10, 10, 10, "20|CON|12",
"16|WIL|16",
                "0|0|0");

            Entity e = new Entity("Eliza", ed, EntityGender.Female, EntityType.NPC);

            NonPlayerCharacter npc = new NonPlayerCharacter(e, s);

            npc.SetConversation("eliza1");
            world.Levels[world.CurrentLevel].Characters.Add(npc);

            GamePlayScreen.World = world;

            CreateConversation();


((NonPlayerCharacter)world.Levels[world.CurrentLevel].Characters[0]).SetConversation("eliza1");
        }
```

It just passes level one in as a parameter to the call to the constructor of EntityData. I do the same thing in the LoadWorld method of the CharacterGeneratorScreen. Update that method to the following.

```
private void LoadWorld()
{
    RpgLibrary.WorldClasses.LevelData levelData =
        Game.Content.Load<RpgLibrary.WorldClasses.LevelData>(@"Game\Levels\Starting Level");
    RpgLibrary.WorldClasses.MapData mapData =
        Game.Content.Load<RpgLibrary.WorldClasses.MapData>(@"Game\Levels\Maps\" +
levelData.MapName);

    TileMap map = TileMap.FromMapData(mapData, Game.Content);

    Level level = new Level(map);

    ChestData chestData = Game.Content.Load<ChestData>(@"Game\Chests\Plain Chest");

    Chest chest = new Chest(chestData);

    BaseSprite chestSprite = new BaseSprite(
        containers,
        new Rectangle(0, 0, 32, 32),
        new Point(10, 10));

    ItemSprite itemSprite = new ItemSprite(
        chest,
        chestSprite);

    level.Chests.Add(itemSprite);

    World world = new World(GameRef, GameRef.ScreenRectangle);

    world.Levels.Add(level);
    world.CurrentLevel = 0;

    AnimatedSprite s = new AnimatedSprite(
        GameRef.Content.Load<Texture2D>(@"SpriteSheets\Eliza"),
        AnimationManager.Instance.Animations);

    s.Position = new Vector2(0 * Engine.TileWidth, 5 * Engine.TileHeight);

    EntityData ed = new EntityData("Eliza", 1, 10, 10, 10, 10, 10, 10, "20|CON|12", "16|WIL|
16",
        "0|0|0");

    Entity e = new Entity("Eliza", ed, EntityGender.Female, EntityType.NPC);

    NonPlayerCharacter npc = new NonPlayerCharacter(e, s);

    npc.SetConversation("eliza1");
    world.Levels[world.CurrentLevel].Characters.Add(npc);

    GamePlayScreen.World = world;

    CreateConversation();


((NonPlayerCharacter)world.Levels[world.CurrentLevel].Characters[0]).SetConversation("eliza1");
}
```

I update the constructor for the Entity class and added a new method to calculate the hit points, mana and stamina of the entity. What I did in the constructor is set the level of the character. Then in the CalculateAttribute method on the health, mana and stamina fields. In the CalculateAttribute method I split the formula into its parts on the pipe character. There is a local variable that holds the base attribute. There is then a loop that loops over the number of levels minus one and updates the attribute. At this point you should apply bonuses for a high attribute but I will be doing that in another tutorial once I've balanced attributes and levelling up. Update the public constructor to the following and add the CalculateAttribute method.

```
public Entity(
    string name,
    EntityData entityData,
    EntityGender gender,
    EntityType entityType) : this()
{
    EntityName = name;
    Level = entityData.Level;
    EntityClass = entityData.EntityName;
    Gender = gender;
    EntityType = entityType;
    Strength = entityData.Strength;
    Dexterity = entityData.Dexterity;
    Cunning = entityData.Cunning;
    Willpower = entityData.Willpower;
    Magic = entityData.Magic;
    Constitution = entityData.Constitution;
    health = new AttributePair(CalculateAttribute(entityData.HealthFormula));
    stamina = new AttributePair(CalculateAttribute(entityData.StaminaFormula));
    mana = new AttributePair(CalculateAttribute(entityData.MagicFormula));
}

public int CalculateAttribute(string formula)
{
    int value = 0;

    string[] parts = formula.Split('|');

    value = int.Parse(parts[0]);

    for (int i = 1; i < level; i++)
    {
        value += int.Parse(parts[2]);
    }

    return value;
}
```

I'm now going to add creating characters to the map editor. First, we need to add a CharacterLayer field to hold the characters. Add the following field to the FormMain class of the level editor.

```
public static CharacterLayer characters = new CharacterLayer();
```

Next thing I added was a form for creating and editing characters. Right click the XlevelEditor project, select Add and then Form (Windows Forms). Name this new form FormNewCharacter. Instead of going over designing the form manually I am just going to give you the for the designer. Replace the code in FormNewCharacter.Designer.cs with the following.

```
namespace XLevelEditor
{
    partial class FormNewCharacter
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.tbName = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.mtbLevel = new System.Windows.Forms.MaskedTextBox();
            this.mtbStrength = new System.Windows.Forms.MaskedTextBox();
            this.mtbDexterity = new System.Windows.Forms.MaskedTextBox();
            this.mtbCunning = new System.Windows.Forms.MaskedTextBox();
            this.mtbMagic = new System.Windows.Forms.MaskedTextBox();
            this.mtbWillPower = new System.Windows.Forms.MaskedTextBox();
            this.mtbConstitution = new System.Windows.Forms.MaskedTextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.btnOK = new System.Windows.Forms.Button();
            this.btnCancel = new System.Windows.Forms.Button();
            this.label10 = new System.Windows.Forms.Label();
            this.tbFilePath = new System.Windows.Forms.TextBox();
            this.btnBrowse = new System.Windows.Forms.Button();
            this.tbHitPoints = new System.Windows.Forms.TextBox();
            this.tbMagicPoints = new System.Windows.Forms.TextBox();
            this.tbStamina = new System.Windows.Forms.TextBox();
            this.label9 = new System.Windows.Forms.Label();
            this.label11 = new System.Windows.Forms.Label();
            this.label12 = new System.Windows.Forms.Label();
            this.label13 = new System.Windows.Forms.Label();
            this.mtbX = new System.Windows.Forms.MaskedTextBox();
            this.mtbY = new System.Windows.Forms.MaskedTextBox();
            this.label14 = new System.Windows.Forms.Label();
            this.label15 = new System.Windows.Forms.Label();
            this.cboGender = new System.Windows.Forms.ComboBox();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(52, 9);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(38, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Name:";
            //
            // tbName
            //
            this.tbName.Location = new System.Drawing.Point(96, 6);
            this.tbName.Name = "tbName";
            this.tbName.Size = new System.Drawing.Size(188, 20);
            this.tbName.TabIndex = 1;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(54, 66);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(36, 13);
            this.label2.TabIndex = 4;
            this.label2.Text = "Level:";
            //
            // mtbLevel
            //
            this.mtbLevel.Location = new System.Drawing.Point(96, 63);
            this.mtbLevel.Mask = "000";
            this.mtbLevel.Name = "mtbLevel";
            this.mtbLevel.Size = new System.Drawing.Size(100, 20);
            this.mtbLevel.TabIndex = 5;
            this.mtbLevel.Text = "1";
            //
            // mtbStrength
            //
            this.mtbStrength.Location = new System.Drawing.Point(96, 89);
            this.mtbStrength.Mask = "000";
            this.mtbStrength.Name = "mtbStrength";
            this.mtbStrength.Size = new System.Drawing.Size(100, 20);
            this.mtbStrength.TabIndex = 7;
            this.mtbStrength.Text = "10";
            //
            // mtbDexterity
            //
            this.mtbDexterity.Location = new System.Drawing.Point(96, 115);
            this.mtbDexterity.Mask = "000";
            this.mtbDexterity.Name = "mtbDexterity";
            this.mtbDexterity.Size = new System.Drawing.Size(100, 20);
            this.mtbDexterity.TabIndex = 9;
            this.mtbDexterity.Text = "10";
            //
            // mtbCunning
            //
            this.mtbCunning.Location = new System.Drawing.Point(96, 141);
            this.mtbCunning.Mask = "000";
            this.mtbCunning.Name = "mtbCunning";
            this.mtbCunning.Size = new System.Drawing.Size(100, 20);
            this.mtbCunning.TabIndex = 11;
            this.mtbCunning.Text = "10";
            //
            // mtbMagic
            //
            this.mtbMagic.Location = new System.Drawing.Point(96, 167);
            this.mtbMagic.Mask = "000";
            this.mtbMagic.Name = "mtbMagic";
            this.mtbMagic.Size = new System.Drawing.Size(100, 20);
            this.mtbMagic.TabIndex = 13;
            this.mtbMagic.Text = "10";
            //
            // mtbWillPower
            //
            this.mtbWillPower.Location = new System.Drawing.Point(96, 193);
            this.mtbWillPower.Mask = "000";
            this.mtbWillPower.Name = "mtbWillPower";
            this.mtbWillPower.Size = new System.Drawing.Size(100, 20);
            this.mtbWillPower.TabIndex = 15;
            this.mtbWillPower.Text = "10";
            //
            // mtbConstitution
            //
            this.mtbConstitution.Location = new System.Drawing.Point(96, 219);
            this.mtbConstitution.Mask = "000";
            this.mtbConstitution.Name = "mtbConstitution";
            this.mtbConstitution.Size = new System.Drawing.Size(100, 20);
            this.mtbConstitution.TabIndex = 17;
            this.mtbConstitution.Text = "10";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(40, 92);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(50, 13);
            this.label3.TabIndex = 6;
            this.label3.Text = "Strength:";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(39, 118);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(51, 13);
            this.label4.TabIndex = 8;
            this.label4.Text = "Dexterity:";
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(41, 144);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(49, 13);
            this.label5.TabIndex = 10;
            this.label5.Text = "Cunning:";
            //
            // label6
            //
```

```
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(51, 170);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(39, 13);
this.label6.TabIndex = 12;
this.label6.Text = "Magic:";
//
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(30, 196);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(60, 13);
this.label7.TabIndex = 14;
this.label7.Text = "Will Power:";
//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(25, 222);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(65, 13);
this.label8.TabIndex = 16;
this.label8.Text = "Constitution:";
//
// btnOK
//
this.btnOK.Location = new System.Drawing.Point(290, 4);
this.btnOK.Name = "btnOK";
this.btnOK.Size = new System.Drawing.Size(75, 23);
this.btnOK.TabIndex = 31;
this.btnOK.Text = "OK";
this.btnOK.UseVisualStyleBackColor = true;
//
// btnCancel
//
this.btnCancel.Location = new System.Drawing.Point(290, 35);
this.btnCancel.Name = "btnCancel";
this.btnCancel.Size = new System.Drawing.Size(75, 23);
this.btnCancel.TabIndex = 32;
this.btnCancel.Text = "Cancel";
this.btnCancel.UseVisualStyleBackColor = true;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(22, 328);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(68, 13);
this.label10.TabIndex = 24;
this.label10.Text = "Sprite Sheet:";
//
// tbFilePath
//
this.tbFilePath.Location = new System.Drawing.Point(96, 323);
this.tbFilePath.Name = "tbFilePath";
this.tbFilePath.Size = new System.Drawing.Size(188, 20);
this.tbFilePath.TabIndex = 25;
//
// btnBrowse
```

```
//
this.btnBrowse.Location = new System.Drawing.Point(290, 323);
this.btnBrowse.Name = "btnBrowse";
this.btnBrowse.Size = new System.Drawing.Size(75, 23);
this.btnBrowse.TabIndex = 26;
this.btnBrowse.Text = "...";
this.btnBrowse.UseVisualStyleBackColor = true;
//
// tbHitPoints
//
this.tbHitPoints.Location = new System.Drawing.Point(96, 245);
this.tbHitPoints.Name = "tbHitPoints";
this.tbHitPoints.Size = new System.Drawing.Size(100, 20);
this.tbHitPoints.TabIndex = 19;
//
// tbMagicPoints
//
this.tbMagicPoints.Location = new System.Drawing.Point(96, 271);
this.tbMagicPoints.Name = "tbMagicPoints";
this.tbMagicPoints.Size = new System.Drawing.Size(100, 20);
this.tbMagicPoints.TabIndex = 21;
//
// tbStamina
//
this.tbStamina.Location = new System.Drawing.Point(96, 297);
this.tbStamina.Name = "tbStamina";
this.tbStamina.Size = new System.Drawing.Size(100, 20);
this.tbStamina.TabIndex = 23;
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(35, 248);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(55, 13);
this.label9.TabIndex = 18;
this.label9.Text = "Hit Points:";
//
// label11
//
this.label11.AutoSize = true;
this.label11.Location = new System.Drawing.Point(19, 274);
this.label11.Name = "label11";
this.label11.Size = new System.Drawing.Size(71, 13);
this.label11.TabIndex = 20;
this.label11.Text = "Magic Points:";
//
// label12
//
this.label12.AutoSize = true;
this.label12.Location = new System.Drawing.Point(10, 300);
this.label12.Name = "label12";
this.label12.Size = new System.Drawing.Size(80, 13);
this.label12.TabIndex = 22;
this.label12.Text = "Stamina Points:";
//
// label13
//
this.label13.AutoSize = true;
this.label13.Location = new System.Drawing.Point(202, 66);
this.label13.Name = "label13";
this.label13.Size = new System.Drawing.Size(17, 13);
this.label13.TabIndex = 27;
this.label13.Text = "X:";
//
// mtbX
//
this.mtbX.Location = new System.Drawing.Point(220, 63);
this.mtbX.Mask = "000";
this.mtbX.Name = "mtbX";
this.mtbX.Size = new System.Drawing.Size(64, 20);
this.mtbX.TabIndex = 28;
this.mtbX.Text = "1";
//
// mtbY
//
this.mtbY.Location = new System.Drawing.Point(220, 89);
this.mtbY.Mask = "000";
this.mtbY.Name = "mtbY";
this.mtbY.Size = new System.Drawing.Size(64, 20);
this.mtbY.TabIndex = 30;
this.mtbY.Text = "1";
//
// label14
//
this.label14.AutoSize = true;
this.label14.Location = new System.Drawing.Point(202, 92);
this.label14.Name = "label14";
this.label14.Size = new System.Drawing.Size(17, 13);
this.label14.TabIndex = 29;
this.label14.Text = "Y:";
//
// label15
//
this.label15.AutoSize = true;
this.label15.Location = new System.Drawing.Point(39, 35);
this.label15.Name = "label15";
this.label15.Size = new System.Drawing.Size(45, 13);
this.label15.TabIndex = 2;
this.label15.Text = "Gender:";
//
// cboGender
//
this.cboGender.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
this.cboGender.FormattingEnabled = true;
this.cboGender.Location = new System.Drawing.Point(96, 32);
this.cboGender.Name = "cboGender";
this.cboGender.Size = new System.Drawing.Size(100, 21);
this.cboGender.TabIndex = 3;
//
// FormNewCharacter
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(377, 355);
this.Controls.Add(this.cboGender);
this.Controls.Add(this.label15);
this.Controls.Add(this.label14);
this.Controls.Add(this.label13);
this.Controls.Add(this.label12);
```

```
        this.Controls.Add(this.label11);
        this.Controls.Add(this.label9);
        this.Controls.Add(this.tbStamina);
        this.Controls.Add(this.tbMagicPoints);
        this.Controls.Add(this.tbHitPoints);
        this.Controls.Add(this.btnBrowse);
        this.Controls.Add(this.tbFilePath);
        this.Controls.Add(this.label10);
        this.Controls.Add(this.btnCancel);
        this.Controls.Add(this.btnOK);
        this.Controls.Add(this.mtbConstitution);
        this.Controls.Add(this.mtbWillPower);
        this.Controls.Add(this.mtbMagic);
        this.Controls.Add(this.mtbCunning);
        this.Controls.Add(this.mtbDexterity);
        this.Controls.Add(this.mtbY);
        this.Controls.Add(this.mtbX);
        this.Controls.Add(this.mtbStrength);
        this.Controls.Add(this.mtbLevel);
        this.Controls.Add(this.label8);
        this.Controls.Add(this.label7);
        this.Controls.Add(this.label6);
        this.Controls.Add(this.label5);
        this.Controls.Add(this.label4);
        this.Controls.Add(this.label3);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.tbName);
        this.Controls.Add(this.label1);
        this.Name = "FormNewCharacter";
        this.Text = "FormNewCharacter";
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.TextBox tbName;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.MaskedTextBox mtbLevel;
    private System.Windows.Forms.MaskedTextBox mtbStrength;
    private System.Windows.Forms.MaskedTextBox mtbDexterity;
    private System.Windows.Forms.MaskedTextBox mtbCunning;
    private System.Windows.Forms.MaskedTextBox mtbMagic;
    private System.Windows.Forms.MaskedTextBox mtbWillPower;
    private System.Windows.Forms.MaskedTextBox mtbConstitution;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Label label5;
    private System.Windows.Forms.Label label6;
    private System.Windows.Forms.Label label7;
    private System.Windows.Forms.Label label8;
    private System.Windows.Forms.Button btnOK;
    private System.Windows.Forms.Button btnCancel;
    private System.Windows.Forms.Label label10;
    private System.Windows.Forms.TextBox tbFilePath;
    private System.Windows.Forms.Button btnBrowse;
    private System.Windows.Forms.TextBox tbHitPoints;
    private System.Windows.Forms.TextBox tbMagicPoints;
```

```
        private System.Windows.Forms.TextBox tbStamina;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.Label label12;
        private System.Windows.Forms.Label label13;
        private System.Windows.Forms.MaskedTextBox mtbX;
        private System.Windows.Forms.MaskedTextBox mtbY;
        private System.Windows.Forms.Label label14;
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.ComboBox cboGender;
    }
}
```

The way the form works is that you have the name of the character and their level. Next is their attributes: Strength, Dexterity, Cunning, Magic, Will Power and Constitution. Following that are the formulas for their hit points, magic points and stamina points. Now you browse for the sprite sheet for the character. I'm assuming that all sprites have the same sprite sheet format. In the case of our game one row for each animation and 3 columns for each animation. You also enter the X and Y coordinates of the sprite. Once all of the content is good and the user clicks the OK button the form will close.

Let's add the logic to the form. Right click the FormNewCharacter form in the Solution Explorer and select View Code. Replace the boiler plate code with the following.

```
using MGRpgLibrary.CharacterClasses;
using MGRpgLibrary.SpriteClasses;
using Microsoft.Xna.Framework.Graphics;
using RpgLibrary.CharacterClasses;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace XLevelEditor
{
    public partial class FormNewCharacter : Form
    {
        public static Dictionary<AnimationKey, Animation> Animations =
            new Dictionary<AnimationKey, Animation>();
        public Character Character { get; private set; }
        public Microsoft.Xna.Framework.Point Tile { get; private set; }
        public bool OKPressed { get; private set; }
        public GraphicsDevice GraphicsDevice { get; private set; }

        public FormNewCharacter(GraphicsDevice graphicsDevice)
        {
            InitializeComponent();

            foreach (var gender in Enum.GetValues(typeof(EntityGender)))
            {
                cboGender.Items.Add(gender);
            }
            Animation animation = new Animation(3, 32, 32, 0, 0);
            Animations.Add(AnimationKey.Down, animation);

            animation = new Animation(3, 32, 32, 0, 32);
            Animations.Add(AnimationKey.Left, animation);

            animation = new Animation(3, 32, 32, 0, 64);
            Animations.Add(AnimationKey.Right, animation);

            animation = new Animation(3, 32, 32, 0, 96);
            Animations.Add(AnimationKey.Up, animation);

            GraphicsDevice = graphicsDevice;

            btnBrowse.Click += BtnBrowse_Click;
            btnCancel.Click += BtnCancel_Click;
            btnOK.Click += BtnOK_Click;
        }

        private void BtnOK_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(tbName.Text))
            {
                MessageBox.Show("You must enter the name of the character");
                return;
            }

            if (!int.TryParse(mtbLevel.Text, out int level))
            {
                MessageBox.Show("Level must be numeric.");
                return;
            }

            if (!int.TryParse(mtbStrength.Text, out int Strength))
            {
                MessageBox.Show("Strength must be numeric.");
                return;
            }

            if (!int.TryParse(mtbDexterity.Text, out int Dexterity))
            {
                MessageBox.Show("Dexterity must be numeric.");
                return;
            }

            if (!int.TryParse(mtbCunning.Text, out int Cunning))
            {
                MessageBox.Show("Cunning must be numeric.");
                return;
            }

            if (!int.TryParse(mtbMagic.Text, out int Magic))
            {
                MessageBox.Show("Magic must be numeric.");
                return;
            }

            if (!int.TryParse(mtbWillPower.Text, out int WillPower))
            {
```

```
                    MessageBox.Show("WillPower must be numeric.");
                    return;
                }

                if (!int.TryParse(mtbConstitution.Text, out int Constitution))
                {
                    MessageBox.Show("Constitution must be numeric.");
                    return;
                }

                if (string.IsNullOrWhiteSpace(tbHitPoints.Text))
                {
                    MessageBox.Show("Hit Point forumla cannot be empty");
                    return;
                }

                if (string.IsNullOrWhiteSpace(tbMagicPoints.Text))
                {
                    MessageBox.Show("Magic Point forumla cannot be empty");
                    return;
                }

                if (string.IsNullOrWhiteSpace(tbStamina.Text))
                {
                    MessageBox.Show("Stamina Point forumla cannot be empty");
                    return;
                }

                if (!int.TryParse(mtbX.Text, out int X))
                {
                    MessageBox.Show("X must be numeric.");
                    return;
                }

                if (!int.TryParse(mtbY.Text, out int Y))
                {
                    MessageBox.Show("Y must be numeric.");
                    return;
                }

                EntityData entity = new EntityData(
                    tbName.Text,
                    level,
                    Strength,
                    Dexterity,
                    Cunning,
                    Magic,
                    WillPower,
                    Constitution,
                    tbHitPoints.Text,
                    tbMagicPoints.Text,
                    tbStamina.Text);

                Texture2D texture;

                using (Stream stream = new FileStream(tbFilePath.Text, FileMode.Open,
FileAccess.Read))
                {
                    texture = Texture2D.FromStream(GraphicsDevice, stream);
                }
```

```
            AnimatedSprite sprite = new AnimatedSprite(
                texture,
                Animations);

            Character = new NonPlayerCharacter(
                new Entity(
                    tbName.Text,
                    entity,
                    (EntityGender)Enum.Parse(
                        typeof(EntityGender),
                        cboGender.SelectedItem.ToString(),
                        true),
                    EntityType.NPC),
                sprite);
            Tile = new Microsoft.Xna.Framework.Point(X, Y);
            OKPressed = true;

            Close();
        }

        private void BtnCancel_Click(object sender, EventArgs e)
        {
            Character = null;
            OKPressed = false;
            Close();
        }

        private void BtnBrowse_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog()
            {
                Filter = "Image Files (*.jpg, *.png, *.gif, *.bmp)|*.jpg;*.png;*.gif;*.bmp",
            };

            DialogResult result = ofd.ShowDialog();

            if (result != DialogResult.OK)
            {
                return;
            }

            tbFilePath.Text = ofd.FileName;
        }
    }
}
```

So, there is a static field Animations that holds the animations for our sprite. As I mentioned this relies on all characters having the same sprite sheet layout. There is a property that exposes the created character. There is a property that tells how the form was closed. We need a graphics device to read in the texture so there is a property for that. There is also a property for the tile that the character is on.

I updated the constructor to require a GraphicsDevice argument. In a foreach loop I loop over the gender values and add them to the combo box that holds the gender options. I then create the animations and add them to the dictionary of animations like I've done before. I set the GraphicsDevice property and wire event handlers for the Click event of the three buttons on the form. In the event handler for the Click event of the OK button I validate the form. Basically I check to see that the inputs are valid and if they are not return an error message and exit the method. I then create an EntityData object using the values from the form. I then create a stream to use in opening the Texture2D for the sprite. I now create the animated sprite. I create a new NonPlayerCharacter next using an Entity and the sprite. The Entity is created using the EntityData and form fields. OKPressed is set to true and the form is closed.

The Click event handler for the close button is trivial. It sets Character to null, OKPressed to false and closes the form.

The Click event handler for the browse button is very straight forward. It creates an OpenFileDialog setting its Filter property to images. It shows the dialog. If the result of the dialog is not OK it returns out of the method. It then sets the Text property of tbFilePath to the file name selected in the dialog.

Now it is time to turn our attention to FormMain. First we need to design our form. In the tab control navigate to the Characters tab. Drag a list box onto the tab. Set its Name property to lbCharacters. Set its Dock property so that it fills the tab.

The next step will be to display the new form to create a character. I did that by wiring an event handler in the constructor of the form. Replace the constructor of FormMain with the following and add this new event handler.

```
public FormMain()
{
    InitializeComponent();

    this.Load += new EventHandler(FormMain_Load);
    this.FormClosing += new FormClosingEventHandler(FormMain_FormClosing);

    tilesetToolStripMenuItem.Enabled = false;
    mapLayerToolStripMenuItem.Enabled = false;
    charactersToolStripMenuItem.Enabled = false;
    chestsToolStripMenuItem.Enabled = false;
    keysToolStripMenuItem.Enabled = false;
    chkPaintAnimated.Enabled = false;
    chkPaintCollision.Enabled = false;

    newLevelToolStripMenuItem.Click += new EventHandler(newLevelToolStripMenuItem_Click);
    newTilesetToolStripMenuItem.Click += new EventHandler(newTilesetToolStripMenuItem_Click);
    newLayerToolStripMenuItem.Click += new EventHandler(newLayerToolStripMenuItem_Click);
    saveLevelToolStripMenuItem.Click += new EventHandler(saveLevelToolStripMenuItem_Click);
    openLevelToolStripMenuItem.Click += new EventHandler(openLevelToolStripMenuItem_Click);

    x1ToolStripMenuItem.Click += new EventHandler(x1ToolStripMenuItem_Click);
    x2ToolStripMenuItem.Click += new EventHandler(x2ToolStripMenuItem_Click);
    x4ToolStripMenuItem.Click += new EventHandler(x4ToolStripMenuItem_Click);
    x8ToolStripMenuItem.Click += new EventHandler(x8ToolStripMenuItem_Click);

    blackToolStripMenuItem.Click += new EventHandler(blackToolStripMenuItem_Click);
    blueToolStripMenuItem.Click += new EventHandler(blueToolStripMenuItem_Click);
    redToolStripMenuItem.Click += new EventHandler(redToolStripMenuItem_Click);
    greenToolStripMenuItem.Click += new EventHandler(greenToolStripMenuItem_Click);
    yellowToolStripMenuItem.Click += new EventHandler(yellowToolStripMenuItem_Click);
    whiteToolStripMenuItem.Click += new EventHandler(whiteToolStripMenuItem_Click);
    saveTilesetToolStripMenuItem.Click += new EventHandler(saveTilesetToolStripMenuItem_Click);
    saveLayerToolStripMenuItem.Click += new EventHandler(saveLayerToolStripMenuItem_Click);

    openTilesetToolStripMenuItem.Click += new EventHandler(openTilesetToolStripMenuItem_Click);
    openLayerToolStripMenuItem.Click += new EventHandler(openLayerToolStripMenuItem_Click);

    animatedTIlesetToolStripMenuItem.Click += new
EventHandler(animatedTIlesetToolStripMenuItem_Click);

    chkPaintAnimated.CheckedChanged += new EventHandler(chkPaintAnimated_CheckChanged);
    chkPaintCollision.CheckedChanged += new EventHandler(chkPaintCollision_CheckChanged);

    mapDisplay.MouseEnter += mapDisplay_MouseEnter;
    mapDisplay.MouseDown += mapDisplay_MouseDown;
    mapDisplay.MouseMove += mapDisplay_MouseMove;
    mapDisplay.MouseUp += mapDisplay_MouseUp;
    mapDisplay.MouseLeave += mapDisplay_MouseLeave;

    charactersToolStripMenuItem.Click += CharactersToolStripMenuItem_Click;
}

private void CharactersToolStripMenuItem_Click(object sender, EventArgs e)
{
    FormNewCharacter form = new FormNewCharacter(mapDisplay.GraphicsDevice);
    form.ShowDialog();

    if (!form.OKPressed)
    {
        return;
    }

    characters.Characters.Add(form.Tile, form.Character);

    form.Character.Sprite.Position = new Vector2(
        form.Tile.X * Engine.TileWidth,
        form.Tile.Y * Engine.TileHeight);
}
```

In the event handler I create a new FormNewCharacter passing in the GraphicsDevice property of the MapDisplay instance. I call the ShowDialog method of the form to display it. I then check to see if the OK button was pressed. If it wasn't I exit out of the method. I then add the character to the Dictionary of characters. Finally I set the position of the Sprite to the Tile property of the form.

Now it is time to turn our attention to the MapDisplay to draw the characters. Update the Draw method to the following.

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

    shadowPosition = new Vector2(mouseState.Position.X, mouseState.Position.Y) +
        FormMain.camera.Position;
    Point p = Engine.VectorToCell(shadowPosition);

    for (int i = 0; i < FormMain.layers.Count; i++)
    {
        Editor.spriteBatch.Begin(
            SpriteSortMode.Deferred,
            BlendState.AlphaBlend,
            SamplerState.PointClamp,
            null,
            null,
            null,
            FormMain.camera.Transformation);

        FormMain.layers[i].Draw(Editor.spriteBatch, FormMain.camera, FormMain.tileSets);

        Rectangle destination = new Rectangle(
            (int)p.X * Engine.TileWidth,
            (int)p.Y * Engine.TileHeight,
            FormMain.brushWidth * Engine.TileWidth,
            FormMain.brushWidth * Engine.TileHeight);


        Color tint = Color.White;
        tint.A = 1;
        Editor.spriteBatch.Draw(shadow, destination, tint);
        Editor.spriteBatch.End();
    }

    Editor.spriteBatch.Begin(
        SpriteSortMode.Deferred,
        BlendState.AlphaBlend,
        SamplerState.PointClamp,
        null,
        null,
        null,
        FormMain.camera.Transformation);

    if (FormMain.animatedSet != null && FormMain.animatedLayer.AnimatedTiles.Count > 0)
        FormMain.animatedLayer.Draw(Editor.spriteBatch, FormMain.animatedSet);

    foreach (var c in FormMain.characters.Characters.Values)
    {
        c.Draw(Editor.GameTime, Editor.spriteBatch);
    }

    if (FormMain.collisionLayer.Collisions.Count > 0)
        DrawCollisions();

    Editor.spriteBatch.End();

    DrawDisplay();
}
```

All the new code does is iterate over all of the values in the Characters property of the character layer. Inside of the loop it renders the character.

That leaves saving and loading the character layer. Unfortunately it is going to take a fair bit of work. For saving we are going to need some data classes like we have for the tile map. First, lets add in a class that represents a game item. Right click the ItemClasses folder in the MGRpgLibrary project, select Add and then Class. Name this new class GameItemData.

```
using Microsoft.Xna.Framework;
using RpgLibrary.ItemClasses;
using System;
using System.Collections.Generic;
using System.Text;

namespace MGRpgLibrary.ItemClasses
{
    public class GameItemData
    {
        public Vector2 Position { get; set; }
        public string TextureName { get; set; }
        public Rectangle? SourceRectangle { get; set; }
        public string BaseItem { get; set; }
        public string Type { get; set; }
    }
}
```

Just a simple class with five properties to define a game item. They map to the GameItem class' fields. Position is the position of the item. TextureName is the name of the texture for the item. SourceRectangle is a nullable that is the source rectangle of the item in the texture. If it is null the entire texture will be used as the source rectangle. Next is BaseItem which is a string and is the name of the item. Finally there is Type and that is the type of item.

Now I will add a class for a character. Right click the CharacterClasses folder in the MGRpgLibrary, select Add and then Class. Name this new class CharacterData. Here is the code.

```
using MGRpgLibrary.ItemClasses;
using RpgLibrary.CharacterClasses;
using System;
using System.Collections.Generic;
using System.Text;

namespace MGRpgLibrary.CharacterClasses
{
    public class CharacterData
    {
        public string Name { get; set; }
        public EntityGender Gender { get; set; }
        public EntityData EntityData { get; set; }
        public string TextureName { get; set; }
        public GameItemData Head { get; set; }
        public GameItemData Torso { get; set; }
        public GameItemData Hands { get; set; }
        public GameItemData Feet { get; set; }
        public GameItemData MainHand { get; set; }
        public GameItemData OffHand { get; set; }
    }
}
```

Another simple class made up entirely of properties. First there are properties for the name and gender. There is an EntityData property that defines the entity that makes up the character. The TextureName property is the name of the texture. Next are properties for the items that are equipped.

We need another data class for NonPlayerCharacters. Right click the CharacterClasses folder, select Add and then Class. Name this new class NonPlayerCharacterData. This is the code for that class.

```
using RpgLibrary.QuestClasses;
using System;
using System.Collections.Generic;
using System.Text;

namespace MGRpgLibrary.CharacterClasses
{
    public class NonPlayerCharacterData : CharacterData
    {
        public List<Quest> Quests { get; set; } = new List<Quest>();
        public string CurrentConversation { get; set; }
        public string CurrentQuest { get; set; }
    }
}
```

Another simple class made up of properties. It does inherit from CharacterData so it will have access to all of its public and protected members. It initializes the List<Quest> property to a new list of quests. Unfortunately Quest is still not defined but I'm not going into quests in this tutorial as they deserve a tutorial to themself. It then has string properties for the current conversation and quest.

Now we need a data class for a character layer. Right click the TileEngine folder in the MGRpgLibrary project, select Add and then Class. Name this new class CharacterLayerData. Here is the code for that class.

```
using MGRpgLibrary.CharacterClasses;
using Microsoft.Xna.Framework;
using System;
using System.Collections.Generic;
using System.Text;

namespace MGRpgLibrary.TileEngine
{
    public class CharacterLayerData
    {
        public Dictionary<Point, CharacterData> Characters { get; set; } = new
Dictionary<Point, CharacterData>();
    }
}
```

Just a single property in this class. It is a Dictionary<Point, CharacterData> that holds the characters on the map like in the CharacterLayer.

I had to add another constructor to the EntityData class and some new fields. I also updated the first constructor. It will impact the Entity class when we go to load the characters in and I will cross that bridge then. For now, replace the code in the EntityData class with the following.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RpgLibrary.CharacterClasses
{
    public class EntityData
    {
        #region Field Region
```

```csharp
public string EntityName;
public int Level;
public int Strength;
public int Dexterity;
public int Cunning;
public int Willpower;
public int Magic;
public int Constitution;
public string HealthFormula;
public string StaminaFormula;
public string MagicFormula;
public int MaximumHealth;
public int MaximumStamina;
public int MaximumMana;

#endregion

#region Constructor Region

private EntityData()
{
}

public EntityData(
    string entityName,
    int level,
    int strength,
    int dexterity,
    int cunning,
    int willpower,
    int magic,
    int constitution,
    string health,
    string stamina,
    string mana)
{
    EntityName = entityName;
    Level = level;
    Strength = strength;
    Dexterity = dexterity;
    Cunning = cunning;
    Willpower = willpower;
    Cunning = cunning;
    Willpower = willpower;
    Magic = magic;
    Constitution = constitution;
    HealthFormula = health;
    StaminaFormula = stamina;
    MagicFormula = mana;
}

public EntityData(
    string entityName,
    int level,
    int strength,
    int dexterity,
    int cunning,
    int willpower,
    int magic,
    int constitution,
    int maximumHealth,
    int maximumStamina,
    int maximumMana)
{
    EntityName = entityName;
    Level = level;
    Strength = strength;
    Dexterity = dexterity;
    Cunning = cunning;
    Willpower = willpower;
    Magic = magic;
    Constitution = constitution;

    HealthFormula = "";
    MaximumHealth = maximumHealth;

    MagicFormula = "";
    MaximumMana = maximumMana;

    StaminaFormula = "";
    MaximumStamina = maximumStamina;
}

#endregion

#region Method Region

public override string ToString()
{
    string toString = EntityName + ", ";

    toString += Level.ToString() + ", ";
    toString += Strength.ToString() + ", ";
    toString += Dexterity.ToString() + ", ";
    toString += Cunning.ToString() + ", ";
    toString += Willpower.ToString() + ", ";
    toString += Magic.ToString() + ", ";
    toString += Constitution.ToString() + ", ";
    toString += MaximumHealth.ToString() + ", ";
    toString += MaximumMana.ToString() + ", ";
    toString += MaximumStamina.ToString() + ", ";
    toString += HealthFormula + ", ";
    toString += StaminaFormula + ", ";
    toString += MagicFormula;

    return toString;
}

public object Clone()
{
    EntityData data = new EntityData
    {
        EntityName = EntityName,
        Level = Level,
        Strength = Strength,
        Dexterity = Dexterity,
        Cunning = Cunning,
        Willpower = Willpower,
        Magic = Magic,
        Constitution = Constitution,
        HealthFormula = HealthFormula,
        StaminaFormula = StaminaFormula,
        MagicFormula = MagicFormula
    };

    return data;
}

#endregion
}
}
```

So, I added fields MaximumHealth, MaximumMana and MaximumStamina. As the names imply the hold the maximum health, mana and stamina of a character. The new constructor takes three additional integer parameter: maximumHealth, maximumStamina and maximumMana. They are the maximum health, stamina and mana of the character. In the constructor I set the HealthFormula to the empty string and the MaximumHealth field to the parameter passed in. I do the same thing for mana and stamina. I update the ToString method as well. Before writing the formulas I write the new maximum fields.

That is all the classes and changes that we need to save characters. In FormMain for the level editor open the saveLevelToolStripMenuItem_Click method and replace it with the following code.

```
void saveLevelToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (map == null)
        return;

    List<MapLayerData> mapLayerData = new List<MapLayerData>();

    for (int i = 0; i < clbLayers.Items.Count; i++)
    {
        if (layers[i] is MapLayer)
        {
            MapLayerData data = new MapLayerData(
                clbLayers.Items[i].ToString(),
                ((MapLayer)layers[i]).Width,
                ((MapLayer)layers[i]).Height);

            for (int y = 0; y < ((MapLayer)layers[i]).Height; y++)
            {
                for (int x = 0; x < ((MapLayer)layers[i]).Width; x++)
                {
                    data.SetTile(
                        x,
                        y,
                        ((MapLayer)layers[i]).GetTile(x, y).TileIndex,
                        ((MapLayer)layers[i]).GetTile(x, y).Tileset);
                }
            }

            mapLayerData.Add(data);
        }
    }

    MapData mapData = new MapData(levelData.MapName, tileSetData, animatedSetData,
        mapLayerData, collisionLayer, animatedLayer);

    CharacterLayerData charData = new CharacterLayerData();

    foreach (var c in FormMain.characters.Characters)
    {
        CharacterData character;

        if (c.Value is NonPlayerCharacter)
        {
            character = new NonPlayerCharacterData();
            ((NonPlayerCharacterData)character).Quests = ((NonPlayerCharacter)c.Value).Quests;
            ((NonPlayerCharacterData)character).CurrentConversation =
((NonPlayerCharacter)c.Value).CurrentConversation;
            ((NonPlayerCharacterData)character).CurrentQuest =
((NonPlayerCharacter)c.Value).CurrentQuest;
        }
        else
        {
            character = new CharacterData();
        }

        character.Name = c.Value.Entity.EntityName;
        character.Gender = c.Value.Entity.Gender;
        character.TextureName = c.Value.Sprite.Texture.Name;
        character.Head = new GameItemData();
        character.Torso = new GameItemData();
        character.Feet = new GameItemData();
        character.Hands = new GameItemData();
        character.MainHand = new GameItemData();
        character.OffHand = new GameItemData();
        character.EntityData = new EntityData(
            c.Value.Entity.EntityName,
            c.Value.Entity.Level,
            c.Value.Entity.Strength,
            c.Value.Entity.Dexterity,
            c.Value.Entity.Cunning,
            c.Value.Entity.Willpower,
            c.Value.Entity.Magic,
            c.Value.Entity.Constitution,
            c.Value.Entity.Health.MaximumValue,
            c.Value.Entity.Stamina.MaximumValue,
            c.Value.Entity.Mana.MaximumValue);

        charData.Characters.Add(c.Key, character);
    }

    FolderBrowserDialog fbDialog = new FolderBrowserDialog
    {
        Description = "Select Game Folder",
        SelectedPath = Application.StartupPath
    };

    DialogResult result = fbDialog.ShowDialog();

    if (result == DialogResult.OK)
    {
        if (!File.Exists(fbDialog.SelectedPath + @"\Game.xml"))
        {
            MessageBox.Show("Game not found", "Error");
            return;
        }

        string LevelPath = Path.Combine(fbDialog.SelectedPath, @"Levels\");
        string MapPath = Path.Combine(LevelPath, @"Maps\");
        string CharPath = Path.Combine(LevelPath, @"Chars\");

        if (!Directory.Exists(LevelPath))
        {
            Directory.CreateDirectory(LevelPath);
        }

        if (!Directory.Exists(MapPath))
        {
            Directory.CreateDirectory(MapPath);
        }

        if (!Directory.Exists(CharPath))
        {
            Directory.CreateDirectory(CharPath);
        }

        XnaSerializer.Serialize<LevelData>(
            LevelPath + levelData.LevelName + ".xml",
            levelData);

        XnaSerializer.Serialize<MapData>(
            MapPath + mapData.MapName + ".xml",
            mapData);

        XnaSerializer.Serialize<CharacterLayerData>(
            CharPath + levelData.LevelName + ".xml",
            charData);
    }
}
```

What the new code does is create a new CharacterLayerData object for writing to disk. It then iterates over all of the characters on the map. It has a local CharacterData field. If the character is an NPC it creates a NonPlayerCharacterData object, otherwise it creates a CharacterData object. If it is an NPC it sets the Quests, CurrrentConversation and CurrentQuests properties. It sets the TextureName of the character data to the Name property of the texture of the sprite. For now it creates new instances of GameItemData for the equipped items. In another tutorial I will add equipped items to characters. It then creates the EntityData using the properties of the Entity. Finally the character is added to the Dictionary of characters.

To save the characters I create a new folder in the LevelPath folder Chars. If the path does not exist I create it. I then use the XnaSerializer to serialize the layer.

There are two fixes that I need to make to creating the character. First, I need to add a property to the AnimatedSprite class to expose the Texture2D. Second, I need to set the Name property of the texture for the Sprite when I create it. So, add this property to the AnimatedSprite class.

```
public Texture2D Texture
{
    get { return texture; }
}
```

Now, update the BtnOK_Click method of FormNewCharacter to the following.

```
private void BtnOK_Click(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(tbName.Text))
    {
        MessageBox.Show("You must enter the name of the character");
        return;
    }

    if (!int.TryParse(mtbLevel.Text, out int level))
    {
        MessageBox.Show("Level must be numeric.");
        return;
    }

    if (!int.TryParse(mtbStrength.Text, out int Strength))
    {
        MessageBox.Show("Strength must be numeric.");
        return;
    }

    if (!int.TryParse(mtbDexterity.Text, out int Dexterity))
    {
        MessageBox.Show("Dexterity must be numeric.");
        return;
    }

    if (!int.TryParse(mtbCunning.Text, out int Cunning))
    {
        MessageBox.Show("Cunning must be numeric.");
        return;
    }

    if (!int.TryParse(mtbMagic.Text, out int Magic))
    {
        MessageBox.Show("Magic must be numeric.");
        return;
    }

    if (!int.TryParse(mtbWillPower.Text, out int WillPower))
    {
        MessageBox.Show("WillPower must be numeric.");
        return;
    }

    if (!int.TryParse(mtbConstitution.Text, out int Constitution))
    {
        MessageBox.Show("Constitution must be numeric.");
        return;
    }

    if (string.IsNullOrWhiteSpace(tbHitPoints.Text))
    {
        MessageBox.Show("Hit Point forumla cannot be empty");
        return;
    }
```

```csharp
if (string.IsNullOrWhiteSpace(tbMagicPoints.Text))
{
    MessageBox.Show("Magic Point forumla cannot be empty");
    return;
}

if (string.IsNullOrWhiteSpace(tbStamina.Text))
{
    MessageBox.Show("Stamina Point forumla cannot be empty");
    return;
}

if (!int.TryParse(mtbX.Text, out int X))
{
    MessageBox.Show("X must be numeric.");
    return;
}

if (!int.TryParse(mtbY.Text, out int Y))
{
    MessageBox.Show("Y must be numeric.");
    return;
}

EntityData entity = new EntityData(
    tbName.Text,
    level,
    Strength,
    Dexterity,
    Cunning,
    Magic,
    WillPower,
    Constitution,
    tbHitPoints.Text,
    tbMagicPoints.Text,
    tbStamina.Text);

Texture2D texture;

using (Stream stream = new FileStream(tbFilePath.Text, FileMode.Open, FileAccess.Read))
{
    texture = Texture2D.FromStream(GraphicsDevice, stream);
}

AnimatedSprite sprite = new AnimatedSprite(
    texture,
    Animations);

sprite.Texture.Name = tbFilePath.Text;

Character = new NonPlayerCharacter(
    new Entity(
        tbName.Text,
        entity,
        (EntityGender)Enum.Parse(
            typeof(EntityGender),
            cboGender.SelectedItem.ToString(),
            true),
        EntityType.NPC),
        sprite);
    Tile = new Microsoft.Xna.Framework.Point(X, Y);
    OKPressed = true;

    Close();
}
```

If you build and run now you can add characters to the map and save them to disk.I know that the tutorial is getting on the long side but I want to tackle loading maps back in now instead of doing it in a separate tutorial.

One important note. It is important that you save your maps before updating the loading code. If you don't the loading code will error out. Change the openLevelToolStripMenuItem_Click method to the following.

```
void openLevelToolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog ofDialog = new OpenFileDialog();

    ofDialog.Filter = "Level Files (*.xml)|*.xml";
    ofDialog.CheckFileExists = true;
    ofDialog.CheckPathExists = true;

    DialogResult result = ofDialog.ShowDialog();

    if (result != DialogResult.OK)
    {
        return;
    }

    string path = Path.GetDirectoryName(ofDialog.FileName);

    LevelData newLevel = null;
    MapData mapData = null;
    CharacterLayerData charData = null;

    try
    {
        newLevel = XnaSerializer.Deserialize<LevelData>(ofDialog.FileName);
        mapData = XnaSerializer.Deserialize<MapData>(path + @"\Maps\" + newLevel.MapName +
            ".xml");
        charData = XnaSerializer.Deserialize<CharacterLayerData>(path + @"\Chars\" +
newLevel.LevelName + ".xml");
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message, "Error reading level");
        return;
    }

    tileSetImages.Clear();
    tileSetData.Clear();
    tileSets.Clear();

    layers.Clear();

    lbTileset.Items.Clear();
    clbLayers.Items.Clear();
```

```
collisionLayer.Collisions.Clear();
animatedLayer.AnimatedTiles.Clear();
characters.Characters.Clear();

levelData = newLevel;

foreach (TilesetData data in mapData.Tilesets)
{
    Texture2D texture = null;

    tileSetData.Add(data);
    lbTileset.Items.Add(data.TilesetName);

    GDIImage image = (GDIImage)GDIBitmap.FromFile(data.TilesetImageName);
    tileSetImages.Add(image);
    using (Stream stream = new FileStream(data.TilesetImageName, FileMode.Open,
        FileAccess.Read))
    {
        texture = Texture2D.FromStream(GraphicsDevice, stream);

        tileSets.Add(
            new Tileset(
            texture,
            data.TilesWide,
            data.TilesHigh,
            data.TileWidthInPixels,
            data.TileHeightInPixels));
    }
}

using (Stream textureStream = new FileStream(mapData.AnimatedTileset.TilesetImageName,
    FileMode.Open, FileAccess.Read))
{
    Texture2D aniamtedTexture = Texture2D.FromStream(GraphicsDevice, textureStream);

    animatedSet = new AnimatedTileset(
        aniamtedTexture,
        mapData.AnimatedTileset.FramesAcross,
        mapData.AnimatedTileset.TilesHigh,
        mapData.AnimatedTileset.TileWidthInPixels,
        mapData.AnimatedTileset.TileHeightInPixels);
    animatedSetData = mapData.AnimatedTileset;
}

foreach (MapLayerData data in mapData.Layers)
{
    clbLayers.Items.Add(data.MapLayerName, true);
    layers.Add(MapLayer.FromMapLayerData(data));
}

lbTileset.SelectedIndex = 0;
clbLayers.SelectedIndex = 0;
nudCurrentTile.Value = 0;
sbAnimatedTile.Maximum = 0;

map = new TileMap(tileSets[0], animatedSet, (MapLayer)layers[0]);

for (int i = 1; i < tileSets.Count; i++)
{
```

```
            map.AddTileset(tileSets[i]);
        }

        for (int i = 1; i < layers.Count; i++)
        {
            map.AddLayer(layers[i]);
        }

        foreach (var collision in mapData.Collisions.Collisions)
        {
            collisionLayer.Collisions.Add(collision.Key, collision.Value);
        }

        foreach (var tile in mapData.AnimatedTiles.AnimatedTiles)
        {
            animatedLayer.AnimatedTiles.Add(tile.Key, tile.Value);
        }

        foreach (var c in charData.Characters)
        {
            Character character;

            if (c.Value is NonPlayerCharacterData)
            {
                Entity entity = new Entity(c.Value.Name, c.Value.EntityData, c.Value.Gender,
EntityType.NPC);

                using (Stream stream = new FileStream(c.Value.TextureName, FileMode.Open,
FileAccess.Read))
                {
                    Texture2D texture = Texture2D.FromStream(GraphicsDevice, stream);
                    AnimatedSprite sprite = new AnimatedSprite(texture, Animations);
                    sprite.Position = new Vector2(c.Key.X * Engine.TileWidth, c.Key.Y *
Engine.TileHeight);
                    character = new NonPlayerCharacter(entity, sprite);

                    ((NonPlayerCharacter)character).SetConversation(
                        ((NonPlayerCharacterData)c.Value).CurrentConversation);

                    characters.Characters.Add(c.Key, character);
                }
            }
        }
        tilesetToolStripMenuItem.Enabled = true;
        mapLayerToolStripMenuItem.Enabled = true;
        charactersToolStripMenuItem.Enabled = true;
        chestsToolStripMenuItem.Enabled = true;
        keysToolStripMenuItem.Enabled = true;
    }
```

What is new here is that I included a CharacterLayerData local variable and read it in using the XnaSerializer class. Then after creating the other map data I create a new CharacterLayer. I then loop over all of the characters on the CharacterLayerData. If the character is a NonPlayerCharacter I create a new Entity passing in the parameters from the character and EntityType.NPC for the entity type. Next is a using statement where I open the stream to the texture for the sprite. I read the texture using the FromStream method of the Texture2D class. I create the sprite and set its position to the tile times the width and height of the tiles on the screen. I create the character next. I call the SetConversation method of the NonPlayerCharacter class passing in the conversation. This is currently the empty string but we will be adding it in when we do the conversation editor. The character is then added to the character layer.

If you build and run now you can add characters to the map, save the map and then load them back in again. I'm going to wrap up the tutorial here because I don't want to start loading into that game at this point. There is more than enough to think about and I want to keep the tutorials at a reasonable length. So, I encourage you to visit my blog at, https://cynthiamcmahon.ca/blog/, for the latest news on my tutorials and other goodness.

Good luck in your Game Programming Adventures!

Cynthia