Eyes of the Dragon Tutorials
Part 40
Characters Revisited – Part 2

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.

So, I ended up breaking a few things that I didn't fix in the last tutorial. Most are related to the changes that I made to EntityData. The first is that the classes need a Level attribute associated with them. The second is they are expecting MaximumHealth, MaximumMana and MaximumStamina. I ended up adding a Level attribute to the classes. For the maximum attributes I ended up adding in an attribute to those fields in the EntityData class to have those fields ignored.

First, replace the Fighter.xml, Priest.xml, Rogue.xml and Wizard.xml files with the following versions.

```
<?xml version="1.0" encoding="utf-8"?>
<XnaContent xmlns:CharacterClasses="RpgLibrary.CharacterClasses">
  <Asset Type="CharacterClasses:EntityData">
    <EntityName>Fighter</EntityName>
       <Level>1</Level>
    <Strength>14</Strength>
    <Dexterity>12</Dexterity>
    <Cunning>10</Cunning>
    <Willpower>12</Willpower>
    <Magic>10</Magic>
    <Constitution>12</Constitution>
    <HealthFormula>20|CON|12</HealthFormula>
    <StaminaFormula>12|WIL|12</StaminaFormula>
    <MagicFormula>0|0|0</MagicFormula>
  </Asset>
</XnaContent>

<?xml version="1.0" encoding="utf-8"?>
<XnaContent xmlns:CharacterClasses="RpgLibrary.CharacterClasses">
  <Asset Type="CharacterClasses:EntityData">
    <EntityName>Priest</EntityName>
       <Level>1</Level>
    <Strength>12</Strength>
    <Dexterity>10</Dexterity>
    <Cunning>12</Cunning>
    <Willpower>12</Willpower>
    <Magic>14</Magic>
    <Constitution>10</Constitution>
    <HealthFormula>12|CON|12</HealthFormula>
    <StaminaFormula>0|0|0</StaminaFormula>
    <MagicFormula>12|WIL|12</MagicFormula>
  </Asset>
</XnaContent>

<?xml version="1.0" encoding="utf-8"?>
<XnaContent xmlns:CharacterClasses="RpgLibrary.CharacterClasses">
```

```
  <Asset Type="CharacterClasses:EntityData">
    <EntityName>Rogue</EntityName>
       <Level>1</Level>
    <Strength>10</Strength>
    <Dexterity>14</Dexterity>
    <Cunning>14</Cunning>
    <Willpower>12</Willpower>
    <Magic>10</Magic>
    <Constitution>10</Constitution>
    <HealthFormula>10|CON|8</HealthFormula>
    <StaminaFormula>12|WIL|12</StaminaFormula>
    <MagicFormula>0|0|0</MagicFormula>
  </Asset>
</XnaContent>

<?xml version="1.0" encoding="utf-8"?>
<XnaContent xmlns:CharacterClasses="RpgLibrary.CharacterClasses">
  <Asset Type="CharacterClasses:EntityData">
    <EntityName>Wizard</EntityName>
       <Level>1</Level>
    <Strength>10</Strength>
    <Dexterity>10</Dexterity>
    <Cunning>12</Cunning>
    <Willpower>14</Willpower>
    <Magic>14</Magic>
    <Constitution>10</Constitution>
    <HealthFormula>10|CON|8</HealthFormula>
    <StaminaFormula>0|0|0</StaminaFormula>
    <MagicFormula>20|WIL|20</MagicFormula>
  </Asset>
</XnaContent>
```

Now, update the EntityData class to the following.

```
using Microsoft.Xna.Framework.Content;
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;
        [ContentSerializerIgnore]
        public int MaximumHealth;
```

```
[ContentSerializerIgnore]
public int MaximumStamina;
[ContentSerializerIgnore]
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
    }
```

```
}
```

In order to have characters render automatically as a layer I updated the TileMap class. What I did was in the Draw method instead of iterating over MapLayers I iterate over Ilayers. Replace the Draw method of the TileMap class with the following.

```
public void Draw(SpriteBatch spriteBatch, Camera camera)
{
    foreach (ILayer layer in mapLayers)
    {
        layer.Draw(spriteBatch, camera, tilesets);
    }

    if (animatedSet != null)
        animatedTileLayer.Draw(spriteBatch, animatedSet);
}
```

Now before we can render the layer we need the content. You can download my content from the following link: https://cynthiamcmahon.ca/blog/downloads/game40.zip. Download that and add it to the Content folder of the EyesOfTheDragon project. Open the MonoGame Pipeline Tool. Expand the Game folder and right click the Levels folder, select Add and then Existing Folder. Add the Chars folder to the project. Repeat the process for the Maps folder. Right click the Levels folder, select Add and then Existing Item. Add the Starting Level.xml file. Save, build and close the tool.

Open the CharacterGeneratorScreen and replace the LoadWorld method with 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);

    CharacterLayerData charData =
        Game.Content.Load<CharacterLayerData>(@"Game\Levels\Chars\Starting Level");
    CharacterLayer characterLayer = new CharacterLayer();

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

    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,
AnimationManager.Instance.Animations)
```

```
                {
                    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);
            }

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

    map.AddLayer(characterLayer);

    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");
}
```

The new code works much like the code from the editor. Instead of using the XnaSerliazer to load the character layer data I use the ContentManager. After creating the map I iterate over the Characters in the layer. There is a local variable to hold the character. If the character being read is a non-player character I create an entity using the values from the data. I could have used the ContentManager to load the texture for the sprite but instead I used the code from the editor. I create the sprite and set its position. I then create the character. I also set the conversation for the character. Finally the character is added to the character layer. I then add the layer to the map. The rest of the method flows as before.

If you build and run now you can see the characters on the map. The one that was loaded and the one that we add manually. There is a problem though. You can walk over the one that we loaded through the ContentManager. It is because instead of using the CharacterLayer field that we added we are using an ILayer for rendering the layer. Second, you would not be able to start a conversation with the character if they had one assigned to them. To start resolving this issue add the following property to the TileMap class to expose the list of layers.

```
public List<ILayer> Layers
{
    get { return mapLayers; }
}
```

Now, in the Player.cs class in the EyesOfTheDragon project replace the UpdatePosition method with the following.

```
private void UpdatePosition(GameTime gameTime, Vector2 motion)
{
    Sprite.IsAnimating = true;
    motion.Normalize();

    Vector2 distance = motion * Sprite.Speed *
        (float)gameTime.ElapsedGameTime.TotalSeconds;

    Vector2 next = distance + Sprite.Position;

    Rectangle playerRect = new Rectangle(
        (int)next.X + CollisionLayer.CollisionRadius,
        (int)next.Y + CollisionLayer.CollisionRadius,
        Engine.TileWidth - CollisionLayer.CollisionRadius,
        Engine.TileHeight - CollisionLayer.CollisionRadius);
    foreach (Point p in
        GamePlayScreen.World.CurrentMap.CollisionLayer.Collisions.Keys)
    {
        Rectangle r = new Rectangle(
            p.X * Engine.TileWidth + CollisionLayer.CollisionRadius,
            p.Y * Engine.TileHeight + CollisionLayer.CollisionRadius,
            Engine.TileWidth - CollisionLayer.CollisionRadius,
            Engine.TileHeight - CollisionLayer.CollisionRadius);

        if (r.Intersects(playerRect))
        {
```

```
            return;
        }
    }

    foreach (ILayer layer in
GamePlayScreen.World.Levels[GamePlayScreen.World.CurrentLevel].Map.Layers)
    {
        if (layer is CharacterLayer)
        {
            foreach (Character c in ((CharacterLayer)layer).Characters.Values)
            {
                Rectangle r = new Rectangle(
                    (int)c.Sprite.Position.X + Character.CollisionRadius,
                    (int)c.Sprite.Position.Y + Character.CollisionRadius,
                    Engine.TileWidth - Character.CollisionRadius,
                    Engine.TileHeight - Character.CollisionRadius);

                if (r.Intersects(playerRect))
                {
                    return;
                }
            }
        }
    }

    Sprite.Position = next;
    Sprite.LockToMap();

    if (camera.CameraMode == CameraMode.Follow)
    {
        camera.LockToSprite(Sprite);
    }
}
```

What the new code does is iterate over all of the layers on the map. The count should be low so it is efficient enough. You could also try LINQ here to find the character layer. Inside the loop there is a check to see if the current layer is a CharacterLayer. If it is it iterates over all of the values in the Characters property of the layer. From here it flows the same way as it did before loading the map dynamically.

I did something similar in the GamePlayScreen when checking for a conversation between the player and a character. Replace the Update method of the GamePlayScreen to the following.

```
public override void Update(GameTime gameTime)
{
    world.Update(gameTime);
    player.Update(gameTime);
    player.Camera.LockToSprite(player.Sprite);

    if (InputHandler.KeyReleased(Keys.Space) ||
        InputHandler.ButtonReleased(Buttons.A, PlayerIndex.One))
    {
        foreach (ILayer layer in World.Levels[World.CurrentLevel].Map.Layers)
        {
            if (layer is CharacterLayer)
            {
                foreach (Character c in ((CharacterLayer)layer).Characters.Values)
```

```
                {
                    float distance = Vector2.Distance(
                        player.Sprite.Center,
                        c.Sprite.Center);

                    if (distance < Character.SpeakingRadius && c is NonPlayerCharacter)
                    {
                        NonPlayerCharacter npc = (NonPlayerCharacter)c;

                        if (npc.HasConversation)
                        {
                            StateManager.PushState(GameRef.ConversationScreen);

                            GameRef.ConversationScreen.SetConversation(
                                player,
                                npc,
                                npc.CurrentConversation);

                            GameRef.ConversationScreen.StartConversation();
                        }
                    }
                }
            }
        }
    }

    base.Update(gameTime);
}
```

Pretty much the same as before. Loop over all of the layers on the map. Check if the current layer is a CharacterLayer. If it is iterate over all of the characters on the map and check if their distance is less than the speaking distance. If it is check for a conversation and start the conversation.

The last thing I'm going to tackle in this tutorial is bringing the LoadGameState inline with the CharacterGeneratorState. First, I need to fix a bug in the ListBox class. In the constructor I assign the image field twice instead of setting the image and cursor fields. Replace the constructor of the ListBox class with the following.

```
public ListBox(Texture2D background, Texture2D cursor)
    : base()
{
    hasFocus = false;
    tabStop = false;
    this.image = background;
    this.cursor = cursor;
    this.Size = new Vector2(image.Width, image.Height);
    lineCount = image.Height / SpriteFont.LineSpacing;
    startItem = 0;
    Color = Color.Black;
}
```

Now, add this using statement to the LoadGameScreen class for the System.IO namespace.

```
using System.IO;
```

Now, add the LoadWorld method to the LoadGameScreen class.

```
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);

    CharacterLayerData charData =
        Game.Content.Load<CharacterLayerData>(@"Game\Levels\Chars\Starting Level");
    CharacterLayer characterLayer = new CharacterLayer();

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

    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,
AnimationManager.Instance.Animations)
                {
                    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);
            }

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

    map.AddLayer(characterLayer);

    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");
}
```

Delete the old CreateWorld method and update the loadListBox_Selected method to the following.

```
void loadListBox_Selected(object sender, EventArgs e)
{
    loadLinkLabel.HasFocus = true;
    ControlManager.AcceptInput = true;

    Transition(ChangeType.Change, GameRef.GamePlayScreen);

    CreatePlayer();
    LoadWorld();
}
```

If you build and run now you can load the map from either the create character or load game screens. You can also run around the map and not walk over the character that I created. I'm going to wrap up the tutorial here because I don't want to something new at this point. 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