Eyes of the Dragon Tutorials
Part 38
Loading Maps

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 the last tutorial I covered adding animated tiles to the map in the map editor and painting collisions. Also, I covered saving the map to disk. In this tutorial I will cover loading levels in the editor and in the game.

First, I need to address a bug that I found in the level editor. When creating a new map the animated tile layer is null. Update the animatedLayer field to the following.

```
public static AnimatedTileLayer animatedLayer = new AnimatedTileLayer();
```

In the FormMain in the XLevelEditor you will want to update the openLevelToolstripMenuItem_Click event handler. Replace that method with the following code.

```
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;

    try
    {
        newLevel = XnaSerializer.Deserialize<LevelData>(ofDialog.FileName);
        mapData = XnaSerializer.Deserialize<MapData>(path + @"\Maps\" + newLevel.MapName +
            ".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();

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);
    }

    tilesetToolStripMenuItem.Enabled = true;
    mapLayerToolStripMenuItem.Enabled = true;
    charactersToolStripMenuItem.Enabled = true;
    chestsToolStripMenuItem.Enabled = true;
    keysToolStripMenuItem.Enabled = true;
}
```

Like before I create a new OpenFileDialog, display it and capture the result. If the result was not OK I exit the method. I then get the path to the level. I initialize several local variables to hold the level being read in. I then try to read in the level and the map. If an exception occurs I display the message in a message box and exit the method.

Now I clear the tileSetImages, tileSetData, tileSets, layers, lbTileset.Items, clbLayers.Items, collisonLayer and animatedLayer collections. I set the levelData field to the level that was read in. I now loop over all of the TileSetData objects in the Tilesets property of the mapData. I set the texture for the tileset to null. I add the current TilesetData to the list of TilesetData objects. I then add the name of the tile set to the list of tile sets. I load the Bitmap for the tile set into a local variable and add it to the tileSetImages field.

I open a stream to the image so that I can open the images as a Texture2D using the FromStream method. I add a tile set to the list of tile sets.

Now here is the new code that is similar to loading a normal tile set. I open a stream to the image. I load the image using the FromStream method. I then create an AnimatedTileSet. I also set the animateSetData field to the AnimateSetData property of the map.

The method now flows as before. I loop over all of the MapLayerData objects in the Layers collection.

I add the layer to the clbLayers.Items and I add the layer to the layers field. I set the indices of the selections to 0 as well as the Value of the nudCurrentTile. I also set the Maximum property of the sbAnimatedTile to 0. In theory you should check to make sure that there is at least one animated tile set before doing this but we should be fine. I now create a new map using the tile sets, animated tile set and the base layer. The tile sets are added to the map as are the layer. The collisions are now added to the collision layer as well as the animated tiles to the animated tile layer. The menu items are now enabled.

Before I get to loading levels into the game I'm going to share a level that I created and add it to the content of the game. Download it from https://cynthiamcmahon.ca/blog/downloads/game.zip. It has all of the content that I have created for my game so far. Extract it to a folder. Open the MonoGame Pipeline Tool. Right click the Game folder, select add and then New Folder. Name this new folder Levels. Right click the Levels folder, select Add and then New Folder. Name this new folder Maps. Right click the Levels folder, select Add and then Existing Item. Navigate to the folder you extracted my content to and drill down to the Levels folder. Select the Starting Level.xml file and add it to the folder. Right click the Maps folder, select Add and then Existing Item. Navigate to the Maps folder in your download and add the Village.xml file.

In order to read in maps in the game I added a static method to the TileMap class FromMapData. It takes as parameters a TileMapData for the map data and a ContentManager for reading in the textures. Add the following method to the TileMap class. Also, add this private constructor.

```
private TileMap()
{
    tilesets = new List<Tileset>();

    collisionLayer = new CollisionLayer();
    animatedTileLayer = new AnimatedTileLayer();
    mapLayers = new List<ILayer>();
}

public static TileMap FromMapData(MapData mapData, ContentManager content)
{
    TileMap map = new TileMap();

    foreach (RpgLibrary.WorldClasses.TilesetData data in mapData.Tilesets)
    {
        Texture2D texture = content.Load<Texture2D>(@"Tilesets\" +
            Path.GetFileNameWithoutExtension(data.TilesetImageName));

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

    Texture2D aniamtedTexture = content.Load<Texture2D>(@"Tilesets\" +
        Path.GetFileNameWithoutExtension(mapData.AnimatedTileset.TilesetImageName));

    map.animatedSet = new AnimatedTileset(
```

```
            aniamtedTexture,
            mapData.AnimatedTileset.FramesAcross,
            mapData.AnimatedTileset.TilesHigh,
            mapData.AnimatedTileset.TileWidthInPixels,
            mapData.AnimatedTileset.TileHeightInPixels);

        mapWidth = mapData.Layers[0].Width;
        mapHeight = mapData.Layers[0].Height;

        foreach (MapLayerData data in mapData.Layers)
        {
            map.AddLayer(MapLayer.FromMapLayerData(data));
        }

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

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

        return map;
    }
```

The constructor just initializes the collection fields to new instances. The FromMapData method works much the same way as the openLevelToolstripMenuItem_Click method from FormMain of the level editor.

It creates a local variable of type TileMap. It then iterates over all of the tile set data items in the mapData passed in. It loads the texture for the tileset. It then adds a new tile set using the tile set data for that item. Next it loads the texture for the animated tile set. It sets the animated tile set for the map to a new animated tile set. It sets the width and height of the map to the width and height of the first layer. It then loops over all of the layers and adds them to layers of the map using the FromMapLayerData method. Now it loops over the collisions in the map layer data and adds them to the collision layer. It then loops over the animated tiles and adds then the the animated tiles on the map. Finally it returns the map.

In the CharacterGeneratorScreen instead of calling CreateWorld I call a new method LoadWorld that will load in first map in the world. Update the LinkLabel1_Selected method to the following and add this new LoadWorld method.

```
void LinkLabel1_Selected(object sender, EventArgs e)
{
    InputHandler.Flush();

    CreatePlayer();
    LoadWorld();

    GameRef.SkillScreen.SkillPoints = 10;

    Transition(ChangeType.Change, GameRef.SkillScreen);
```

```
    GameRef.SkillScreen.SetTarget(GamePlayScreen.Player.Character);
}

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(5 * Engine.TileWidth, 5 * Engine.TileHeight);

    EntityData ed = new EntityData("Eliza", 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 LoadWorld method uses the ContentManager to load the starting level and then load its map. It then creates a TileMap using the FromMapData method that I added earlier. The method now flows much the same as the old CreateWorld method. It creates a Level using the map. It creates a chest and adds it to the map. It creates a World object and adds the level then sets the current level to zero. I then create an AnimatedSprite and position it at tile (0, 5). I then create the NPC as before and call the CreateConversation method.

So, I'm going to wrap up the tutorial here because I don't want to start a new topic at this point. 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