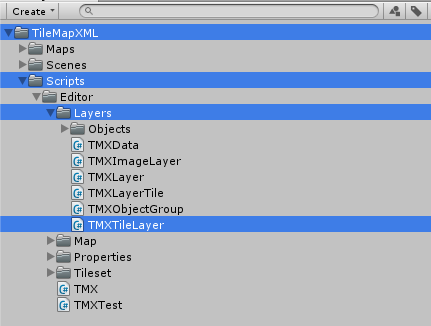
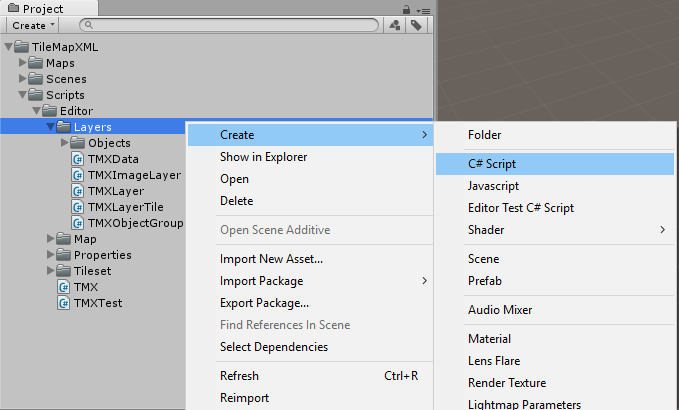
# Code Clean up

There are some things that need to be fixed for this code to look good and be useable. One of the things that needs fix is the fact that all of the layers inherit from TMXLayer. TMXLayer has data which is only used for a tile layer. To fix this we need to add a new script to Unity called TMXTileLayer, inherit from TMXLayer, move data from TMXLayer to TMXTileLayer. In TMXMap change the type from TMXLayer to TMXTileLayer. Then update the TMXTest to use our new TMXTileLayer.

In Unity create a new C# script in TileMapXML->Scripts->Editor->Layers called TMXTileLayer



## Code

### TMXTileLayer.cs

namespace TileMapXML.Layers

{

/// <summary>

/// <layer>

/// All<tileset> tags shall occur before the first<layer> tag so that parsers may rely on having the tilesets before needing to resolve tiles.

/// • name: The name of the layer.

/// • x: The x coordinate of the layer in tiles.Defaults to 0 and can no longer be changed in Tiled Qt.

/// • y: The y coordinate of the layer in tiles.Defaults to 0 and can no longer be changed in Tiled Qt.

/// • width: The width of the layer in tiles.Traditionally required, but as of Tiled Qt always the same as the map width.

/// • height: The height of the layer in tiles.Traditionally required, but as of Tiled Qt always the same as the map height.

/// • opacity: The opacity of the layer as a value from 0 to 1. Defaults to 1.

/// • visible: Whether the layer is shown (1) or hidden(0). Defaults to 1.

/// • offsetx: Rendering offset for this layer in pixels.Defaults to 0. (since 0.14)

/// • offsety: Rendering offset for this layer in pixels. Defaults to 0. (since 0.14)

///

/// Can contain: properties, data

/// </summary>

public class TMXTileLayer : TMXLayer

{

#region attributes

// All are in TMXLayer

#endregion

// Properties is in TMXLayer

/// <summary>

/// The data on the tile layer

/// </summary>

public TMXData data;

}//public class TMXImageLayer

}//namespace TileMapXML.Layers

TMXTileLayer inherits from TMXLayer

Contains the data.

### TMXLayer.cs

using System.Collections.Generic;

using System.Xml.Serialization;

namespace TileMapXML.Layers

{

/// <summary>

/// All<tileset> tags shall occur before the first<layer> tag so that parsers may rely on having the tilesets before needing to resolve tiles.

/// • name: The name of the layer.

/// • x: The x coordinate of the layer in tiles.Defaults to 0 and can no longer be changed in Tiled Qt.

/// • y: The y coordinate of the layer in tiles.Defaults to 0 and can no longer be changed in Tiled Qt.

/// • width: The width of the layer in tiles.Traditionally required, but as of Tiled Qt always the same as the map width.

/// • height: The height of the layer in tiles.Traditionally required, but as of Tiled Qt always the same as the map height.

/// • opacity: The opacity of the layer as a value from 0 to 1. Defaults to 1.

/// • visible: Whether the layer is shown (1) or hidden(0). Defaults to 1.

/// • offsetx: Rendering offset for this layer in pixels.Defaults to 0. (since 0.14)

/// • offsety: Rendering offset for this layer in pixels. Defaults to 0. (since 0.14)

///

/// Can contain: properties

/// </summary>

public class TMXLayer

{

#region attributes

/// <summary>

/// The name of the layer.

/// </summary>

[XmlAttribute]

public string name;

/// <summary>

/// The width of the layer in tiles.

/// Traditionally required, but as of Tiled Qt always the same as the map width.

/// </summary>

[XmlAttribute]

public float width;

/// <summary>

/// The height of the layer in tiles.

/// Traditionally required, but as of Tiled Qt always the same as the map height.

/// </summary>

[XmlAttribute]

public float height;

/// <summary>

/// The opacity of the layer as a value from 0 to 1.

/// Defaults to 1.

/// </summary>

[XmlAttribute]

public float opacity = 1;

/// <summary>

/// Whether the layer is shown (1) or hidden(0).

/// Defaults to 1.

/// </summary>

[XmlAttribute]

public int visible = 1;

/// <summary>

/// Rendering offset for this layer in pixels.

/// Defaults to 0. (since 0.14)

/// </summary>

[XmlAttribute]

public float offsetx = 0;

/// <summary>

/// Rendering offset for this layer in pixels.

/// Defaults to 0. (since 0.14)

/// </summary>

[XmlAttribute]

public float offsety = 0;

#endregion

/// <summary>

/// Wraps any number of custom properties.

/// </summary>

[XmlArray("properties")]

[XmlArrayItem("property")]

public List<TMXProperty> properties;

//delete this line public TMXData data;

}//public class TMXLayer

}//namespace TileMapXML.Layers

Removed the data.

### TMXMap.cs

/// <summary>

/// The Layers on this map

/// </summary>

[XmlElement("layer", typeof(TMXTileLayer))]// Changed this line from [XmlElement("layer", typeof(TMXLayer))]

[XmlElement("objectgroup", typeof(TMXObjectGroup))]

[XmlElement("imagelayer", typeof(TMXImageLayer))]

public List<TMXLayer> layers;//Changed this line from public List<object> layers;

Changed the layer type to be TMXTileLayer.

Also changed the layers list to be TMXLayer instead of object.

### TMXTest.cs

[Test]

public void TMXLayersLoaded()

{

bool hasTileLayer = false;

foreach(TMXLayer layer in tmx.map.layers)// Changed from foreach(var layer in tmx.map.layers)

{

if(layer is TMXObjectGroup)

TMXObjectGroupLoaded(layer as TMXObjectGroup);

else if(layer is TMXImageLayer)

TMXImageLayerLoaded(layer as TMXImageLayer);

else if(layer is TMXTileLayer)// Changed from else if(layer is TMXLayer)

{

hasTileLayer = true;

TMXTileLayerLoaded(layer as TMXTileLayer);// Changed from TMXTileLayerLoaded(layer as TMXLayer)

}//else if(layer is TMXTileLayer)

}//foreach(var layer in tmx.map.layers)

//There must be at least one layer

Assert.True((tmx.map.layers.Count > 0) && (hasTileLayer), "There needs to be at least on tile layer");

}//void TMXLayersLoaded()

Changed the foreach loop to use TMXLayer instead of var.

Changed the TMXLayer check to TMXTileLayer in 2 places.

[Test]

public void TMXLayersPropertiesLoaded()

{

foreach(TMXLayer layer in tmx.map.layers)

{

// If you are using properties to set a value in your tile layer that you need for use in Unity

// add a check here to make sure that it is included in your tileset

foreach(TMXProperty property in layer.properties)

{

TMXPropertyLoaded(property);

}//foreach(TMXProperty property in layer.properties)

}//foreach(TMXLayer layer in tmx.map.layers)

}//void TMXTilesetsPropertiesLoaded()

Added this test to check the properties for all layers.

public void TMXTileLayerLoaded(TMXTileLayer layer)// Changed from public void TMXTileLayerLoaded(TMXLayer layer)

{

//Name of the layer must not be null

Assert.IsNotNullOrEmpty(layer.name, "Layer must have a name");

//The data is loaded correctly

TMXLayerDataLoaded(layer.data);

}//void TMXTileLayerLoaded(TMXLayer layer)

Changed the layer type from TMXLayer to TMXTileLayer.

## Things to Consider

Tiled is constantly updating. New features get added. Like the color property and file property in Tiled version 0.17.0

If you want to make use of these properties in your code you will have to change the TMXTest. TMXPropertyLoaded to include them.

If you are wanting to use some of the other types of Tile Layer Format or external tilesets you will have to write your own parser instead of using xml deserialization.

If you are planning on deploying your final game to a mobile platform like IOS or Andriod then you need to consider all the optimizations that you need. One optimization is to put everything that relies on the System.Xml.Serialization namespace in a folder called Editor, like we did in this tutorial.