# Tiled Map XML Loading in Unity

Now it is time to talk about the actual loading of our maps in Unity. These concepts can be used in other C# projects that you have; Not just in Unity.

For serializing and deserializing xml files we need the using System.Xml.Serialization; and for file stream we need using System.IO; using statements.

To deserialize our XML files we will use something like this

XmlSerializer serializer = new XmlSerializer(typeof(ClassRepresentingXMLObject));

using(FileStream stream = new FileStream(PathToXMLFile, FileMode.Open))

map = serializer.Deserialize(stream) as ClassRepresentingXMLObject;

XML documents are set up with things called nodes or elements. These Elements can have attributes and can contain child elements. Every xml document will start with <?xml …> the next line is called the root element. Every xml document will contain a root element, all of the elements under the root element are called child elements. If you open your map with a program that can read xml files like Visual Studio, MonoDevelop, NotePad, or Internet Explorer (just to name a few) you will see something like

<?xml version="1.0" encoding="UTF-8"?>

<map version="1.0" orientation="orthogonal" ... nextobjectid="1">

<properties>

<property name="boolPropertyFalse" type="bool" value="false"/>

...

<property name="stringProperty" value="Hello"/>

</properties>

...

<layer name="Tile Layer 1" width="5" height="5">

<data>

<tile gid="0"/>

...

<tile gid="0"/>

</data>

</layer>

</map>

The root element is map. Map has several attributes (version, orientation, nextobjectid) and several child elements (properties, layer). Some of the child elements have their own attributes like layer(name, width, height) and some have no attributes at all like properties. Some of the child elements have child elements like layer (data which has a child element of tile) and some do not like tile. The Xml Serialization has some useful tags that we can add to our classes and variables to tell the serializer what to serialize and what to serialize them as.

[XmlRoot("elementName")]

Tells the serializer that this class is the root element with the name elementName.

[XmlIgnore]

public string stringToIgnore = "The seralizer will not write this out to your xml document";

[XmlEnum("orthogonal")]

Orthogonal

This tells the serializer deserializer that the Enum value Orthogonal is orthogonal in the xml document

[XmlElement("elementName")]

This allows you to have a variable with one name and different name in the xml document, it is also needed for list.

[XmlAttribute("attributeName")]

This is how the serializer deserializer know that the variable is an attribute in the xml document.

As we go through each section of the TMX file format documentation you will see how to use these tags in your code and why they are important.

It is now time to write some code to actually do something useful. In Unity double click on the TMX script, this will cause the script to open up in MonoDevelop; or whatever editor that you are using, I am using Visual Studio 2015.

The TMX script is going to be responsible for loading our map using xml serialization. We need to make sure that we have the correct using statements

using System.Xml.Serialization;

using System.IO;

Now we need a variable to hold our map

public TMXMap map;

We need a method to load the map. The Load method takes in a file to load and uses the xml serializer to load that file into our map variable.

public void Load (string tmxFilePath)

{

// Load the map from the xml file at tmxFilePath

XmlSerializer serializer = new XmlSerializer(typeof(TMXMap));

using(FileStream stream = new FileStream(tmxFilePath, FileMode.Open))

map = serializer.Deserialize(stream) as TMXMap;

}

Now we have a way to load in our map using xml serialization.

### TMX.cs

using System.Xml.Serialization;

using System.IO;

namespace TileMapXML

{

/// <summary>

/// Loads in a map created in Tiled using xml serialization.

/// Stores the map in a variable called map

/// </summary>

public class TMX

{

/// <summary>

/// The tmx map

/// </summary>

public TMXMap map;

/// <summary>

/// Loads A TMX file from xml using serialization

/// </summary>

/// <param name="tmxFilePath">The path of the file to load</param>

public void Load(string tmxFilePath)

{

// Load the map from the xml file at tmxFilePath

XmlSerializer serializer = new XmlSerializer(typeof(TMXMap));

using(FileStream stream = new FileStream(tmxFilePath, FileMode.Open))

map = serializer.Deserialize(stream) as TMXMap;

}//public void Load

}//public class TMX

}//namespace TileMapXML