Tutorial XML laden in Unity

# Benötigte Vorkenntnisse

* Grundlagen der objektorientierten Programmierung
* Grundlagen von XML
* Grundkenntnisse von C#

# Ziele

* Eine XML-Datei auslesen zur Weiterverwendung in Unity

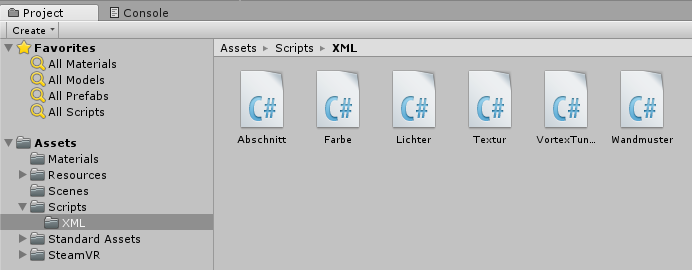
# Anleitung

1. Falls noch keine XML-Datei gegeben ist: Definiere die Struktur der XML-Datei. Das untenstehende Beispiel zeigt eine verschachtelte Struktur. Innerhalb des „Abschnitte“-Tags gibt es das sich wiederholende Element „Abschnitt“. Alle Abschnitte sollen also ausgelesen werden. Zusätzlich müssen die beiden Parameter („intro“ und „durchmesser“) auf dem Root-Element gelesen werden.



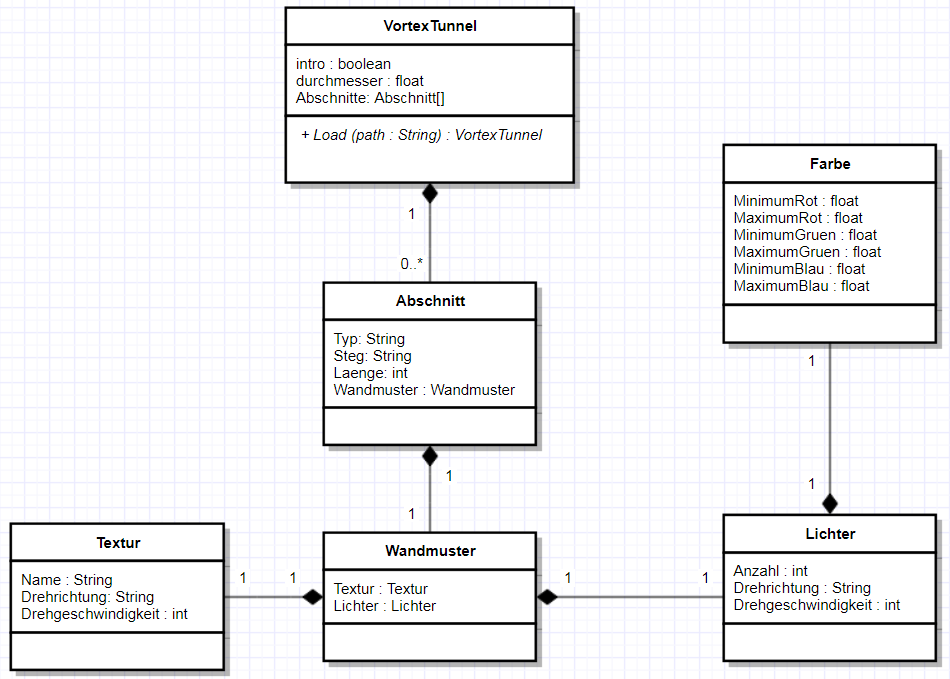
Die XML-Datei sollte im Unterordner „Assets“ des Unity-Projekts abgelegt werden. Dort kann die Datei auch nach dem Deployment noch eingesehen, geändert oder ausgetauscht werden.

1. Erstelle einen „Scripts“-Ordner im Unity-Projekt, als Unterordner von „Assets“ (falls „Scripts“-Ordner noch nicht vorhanden). Bei grösserer Anzahl an Scripts empfiehlt es sich, darin einen weiteren Unterordner „XML“ (o.ä.) zu erstellen, ausschliesslich zur Übersichtlichkeit.



1. Definiere C#-Klassen analog zur XML-Struktur.

Schlussendlich möchten wir in diesem Beispiel eine Klasse „VortexTunnel“ haben, die über eine Liste von „Abschnitt“-Objekten verfügt. Über eine statische Methode „Load“ direkt auf der VortexTunnel-Klasse soll dann ein Objekt erstellt werden, welches direkt die aus der XML-Datei geladenen Werte enthält. Folgendes Klassendiagramm soll veranschaulichen, wie die XML-Struktur aus Schritt 1 in eine Klassen-Struktur umgewandelt werden kann:



Um späteren zusätzlichen Aufwand zu minimieren empfiehlt es sich zudem, dass die Namensgebung der C# Klassen und derer Klassenvariablen mit der Namensgebung in der Xml-Datei übereinstimmen. Achtung: Dabei ist auch auf die Gross-Kleinschreibung zu achten!

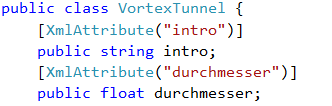
Es ist ausserdem anzumerken, dass eine C#-Klasse, die in Unity erstellt wird, standardmässig immer von MonoDevelop erbt. Für die soeben erstellten Klassen ist dies jedoch **nicht** notwendig.

1. Damit die XML-Datei später korrekt in ein Objekt „übersetzt“ wird, sind in folgenden Fällen noch Annotations im C#-Code notwendig:
   1. Wenn ein C#-Attribut auf ein XML-Element abgebildet werden soll, aber die Namensgebung in C# und XML sich unterscheiden, dann ist eine „XmlElement“-Annotation notwendig, welche den Namen des Xml-Knotens spezifiziert:



Ist die Namensgebung in C# und XML jedoch identisch, dann kann diese Annotation weggelassen werden.

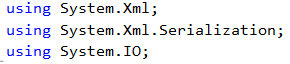
* 1. Wenn ein Attribut (nicht Element!) aus der XML-Datei auf ein Attribut der Klasse gemappt werden soll, dann muss dies über eine „XmlAttribute“-Annotation angegeben werden. In unserem Beispiel wären dies also die beiden Attribute „intro“ und „durchmesser“ auf dem VortexTunnel-Element, welche entsprechend in der VortexTunnel-Klasse vermerkt werden:



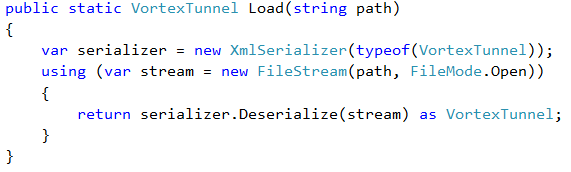
* 1. Wenn sich eine unbestimmte Anzahl an (gleich strukturierten) Elementen innerhalb eines XML-Elements befindet, dann ist eine „XmlArray“-Annotation notwendig. In unserem Beispiel ist dies der Fall beim „Abschnitte“-Element, da dieses eine beliebige Anzahl an „Abschnitt“-Elementen beinhalten kann:



1. Wenn die Klassen übereinstimmend zur XML-Struktur erstellt wurden, dann kann die XML-Datei jetzt in ein Objekt des Root-Typen, in diesem Fall ein VortexTunnel-Objekt, geladen werden. In diesem Beispiel wird dies direkt in der statischen Methode Load der VortexTunnel-Klasse gemacht. Zuerst benötigt die VortexTunnel-Klasse jedoch noch folgende Referenzen:



Danach kann die Load-Methode implementiert werden:



1. Nun muss die Load-Methode nur noch mit einem gültigen Pfad aufgerufen werden.



Das soeben erstellte Objekt „vortex“ beinhaltet nun alle Informationen aus der XML-Datei.

# Referenzen

<http://web.archive.org/web/20130921190426/http://tech.pro/tutorial/798/csharp-tutorial-xml-serialization> (01.11.2017)