TML: AnaGame’s native Markup Language

Real quick, what does TML stand for? *Tormonscript[[1]](#footnote-1) Markup Language*

I designed TML as an easy to parse mark-up language managing Object creation and attribute management. It’s easy to parse because each new object declaration and attribute declaration is on a new line. You can also declare child objects for a given parent object in ways exemplified in the generic example below

->TML # Header for TML file (note # begins a comment)

-|Type:TML\_type # Purpose of THIS tml file

-|Version:0.0.1 # Version of TML type (used to flag incompatibility issues)

-/ # Ending of header

->ObjectType # Declares parent object

-|Attribute1: Value1 # First attribute of the object

-|Attribute2: Value2 # Second attribute of the object

-->ObjectType # Child Object declared

--|childAttribute: childValue

This is a generic TML file and likely won’t be used in real life. However, it does provide a visual example as to what a TML file looks like.

The first four lines are the *header* of the TML file. It’s purpose is to establish itself as a TML file, allow the reader to check to see if the parser and TML type match up, and check versioning (since new standards can introduce new features that old parsers might not support)

Objects are declared via the “->” operator followed by a token representing the object type. Acceptable tokens depend on your TML use/implementation. (“->” <object type>)

Attributes are set via the “-|” operator, followed by an attribute name, “:”, and the attribute’s value (“-|” <attribute name> “:” <attribute value>)

Comments are signified by “#”. Anything to the right of “#” is considered a comment and ignored by the TML reader.

Parent/child relationships are managed by the *number* of dashes “-“ before each operator. If you look at the example above, the *parent* object uses “->” and “-|” which include **1**dash. The Child Object used “-->” and “--|”, which have **2** dashes.

Note: TML is sensitive to the number of dashes before the operators and should only *increase* by 1. You can *decrease* (i.e. go up the tree) as many times as available.

->Object1

--> Object2 # Okay, dash count was incremented by only 1

---->Object 3 # Error, you can only increment by 1, this increments by 2

# Uses in AnaGame

|  |  |  |
| --- | --- | --- |
| Files to look at | | |
| File (h/cpp) | Library | Description |
| Parser\_ | TrecLib | “Abstract” class for responding to calls from the Reader |
| ParseReader\_ | TrecLib | “Abstract” class for reading the file |
| HTML\_Reader | TrecLib | Reader for HTML files |
| TML\_Reader | TrecLib | Reader for TML files |
| ShaderParser | VideoGraphics | Parser for Shader TML files |
| AnafaceParser | Anaface | Parser for Anaface UI TML files |
| HTMLParser | Anaface | Parses HTML tags, attributes and content |

So far, AnaGame has two notable area’s where TML is used: Shader Parsing and Anaface UI parsing.

The following is an example of a Shader TML file:

->TML

-|Type:ShaderSource

-|Version:0.0.1

-/

-|InputSlot:0

-|BufferPurpose:Position

-|BufferWidth:4

-|BufferSize:3

->Buff

-|PixelFile:sColor.fx

-|PixelFunction:PS\_Main

-|VertexFile:sColor.fx

-|VertexFunction:VS\_Main

-|ConstantSize:64

-|ConstantShaderPhase:Vertex

-|ConstantSlot:0

-|ConstantPurpose:Model

->Const

# This snippet was cut short

Notice how the attribute operators tend to come before the object operator. This is okay as long as your TML implementation manages this properly. I advise only doing this to hold single layer data (meaning no children, only single dashes for each line).

If you’re interested in what this file is doing, it is used to help the VideoGraphics library understand the shader[[2]](#footnote-2) it is expected to work with. If you’re familiar with 3D programming, you might be able to deduce what the attributes/objects actually do. However, there will be documentation provided soon.

It is also used heavily by Anaface to produce User Interfaces. The following snippet is an example of an Anaface TML file. Notice how in the child controls, the objects are given parameters. These parameters serve as metadata about the object that may not make sense as the object’s attribute. In this case, the metadata represents where in the parent object the child object belongs.

->TML

-|Type:Anaface\_UI

-|Version:0.0.1

-/

->TGrid

-|ColumnWidth:500

-|ColumnWidth:500

-|RowHeight:300

-|RowHeight:500

-|BorderColor:0.1,0.9,0.2,0.8

-|BorderThickness:1.5

-->TControl(0,0)

--|Caption:ChildControl

--|FontColor:0.1,0.1,1.0,1.0

--|BorderColor:0.1,1.0,0.1,1.0

--/

-->TControl(1,0)

--|BorderColor:0.5,0.9,0.0,1.0

--|BorderThickness:0.2

--|ContentColor:0.3,0.3,0.3,0.7

--/

The metadata could also be used to manage abstraction. For instance, if you have an abstract class with common attributes, this metadata could help you decide which subclass you want to use.

There is more to the Anaface TML that will be discussed in the Document on Anaface (once such a document is produced).

# Parser vs. Reader

AnaGame parses Mark-up files using two different objects that work together. The Reader is an abstract class that actually reads the file and processes it’s contents. The parser is an abstract class that is called by the reader depending on the contents of the file.

AnaGame currently has two specific readers: TML and HTML. Hopefully, an XML reader and a JSON parser are provided by the time AnaGame reaches version 1.0.

There are also three major parsers implemented within the libraries: AnafaceParser (Anaface), ShaderParser (VideoGraphics), and HTMLParser (Anaface). Parsers are not meant to care about what type of reader is calling them. Currently, the HTMLParser is called by the HTML Reader while the other two are used by the TML Reader.

# To-Dos:

1. Add XML and JSON reader to TrecLib
2. Improve the HTML Parser (when it comes out, see the Anaface Documentation for more details

## Eventual To-Do:

1. Create Generic Parser for non-native code to handle

1. Tormonscript: the name of a company I hope to found one day [↑](#footnote-ref-1)
2. A Shader is a small program that runs on the GPU (rather than the CPU) that helps render 3D models [↑](#footnote-ref-2)