Tyler Wolverton Static Mesh (TWSM)

File Format Specification (SMU Guildhall, SD4, Tyler’s Engine)

*Tyler Wolverton, Mar 2021*

# Overview

.twsm is a custom “cooked” binary file format used to save and load 3D static mesh objects.

The purpose/intent of the .TWSM is that it can load much faster than .OBJ files, for several reasons, primarily:

* No text parsing
* No load-time triangulation of quads, etc.
* No manual building/massaging of vertex/indexing scheme
* Matches exact in-memory format of Vertex structure; ready to memcpy and submit directly as VBO/IBO

### Structure

* File Header (19 bytes)
* Vertex Data
* Index Data

Endianness

The file uses little endian storage throughout.

### Data Types

All data in this file - including all chunk data - consists only of the following primitive data elements:

* byte (8 bit unsigned char / uint8\_t)
* int (32 bit signed / int32\_t)
* float (32 bit IEEE 754 standard)

# File Format

***File Header****: 19 bytes total*

* (4 byte array) **FourCC** byte sequence: ‘G’ ‘H’ ‘S’ ‘M’ (hex: 0x47, 0x48, 0x53, 0x4D) (dec: 71, 72, 83, 77)
* (1 byte) **Major file version** number: currently 29 (increases only)
* (1 byte) **Minor file version** number: currently 1 (increases, resets within each Major version number)
* (1 byte) **Layout Type**: byte corresponding to one of the defined vertex struct layouts listed below
* (4 bytes, int32) **sizeof(Vertex)**: should match the sizeof(Vertex) you expect to load this file into
* (4 bytes, int32) **NumVertexes (N)**:(can be zero if mesh is empty)
* (4 bytes, int32) **NumIndexes (I)**:(can be zero if mesh is empty OR verts are provided non-indexed)

***Vertex Data***

* Array of [N] vertex structures, exactly as they will exist in memory (and on the GPU)
* Array of [I] int32 index into the above vertex list

*(end of file)*

# Layout Types

The layout of a vertex structure is given as a byte value corresponding to one of the following types.

* 0: Reserved
* 1: PCU (not yet implemented)
* 2: PCUTBN

The alphabetic letters each represent a traditional vertex attribute, as such:

* ‘P’ = Vec3 **P**osition (12B: float x, float y, float z)
* ‘C’ = Rgba8 **C**olor/tint (4B: byte red, byte green, byte blue, byte alpha)
* ‘U’ = Vec2 **U**VTexCoords (8B: float u, float v)
* ‘T’ = Vec3 **T**angent (12B: float x, float y, float z)
* ‘B’ = Vec3 **B**itangent (12B: float x, float y, float z)
* ‘N’ = Vec3 **N**ormal (12B: float x, float y, float z)

Reasons for this approach:

* 1 byte is the minimum amount of data that could be used to save the information in the header, saving some space, while allowing for expansion for up to 254 types of vertex layouts which should be more than sufficient.
* Since vertex types are defined in the engine, if new types are required there will need to be a larger, intentional effort to implement that support and updating this type table can easily be a part of that process.
* Objs can be used as a universal file sharing format, which is general enough to cover most use cases.