Storm3D Object Definition (SOD) File Format

==========

Author: Steve Williams
Storm3D Graphics Engine Lead.

Modifications: Fixed node specification error.

Audience

========

This document is intended for use by experienced 3D tools programmers for the purposes of writing exporters, importers and conversion tools to/from the .SOD format. A good understanding of real time 3D graphics principles is assumed. The reader is expected to be familiar with real time 3D geometry concepts such as lighting, animation & scene graph hierarchies.

Introduction

============

The SOD file format is a binary file format describing the 3D directed scene graph hierarchies used by the Storm3D rendering engine. Each .SOD file describes one such hierarchy.

The SOD file format has evolved through several versions. This document describes the latest format, 1.8. Documentation of previous formats is not available at this time.

Datatypes used in this document

==========

UINT8 unsigned 8 bit integer
UINT16 unsigned 16 bit integer
UINT32 unsigned 32 bit integer
FLOAT floating point (4 byte) value
VECTOR2 {FLOAT u, FLOAT v}
VECTOR3 {FLOAT x, FLOAT y, FLOAT z}
MATRIX34 { VECTOR3 RIGHT, UP, FRONT, POSITION } MATRICES MUST BE ORTHOGONAL.
COLOUR { FLOAT red, FLOAT green, FLOAT blue } Component range 0.0 - 1.0

Other local datatypes are defined where appropriate.

Additional Syntax

==========

TYPE ARRAY(nentries) - A contiguous array of nentries of type TYPE

Identifiers

==========

IDENTIFIER
{
UINT16 strlen(string), string (8 bit ascii values) including terminating '\0'
OR
UINT16 0 - Indicates null string.
}

File Structure
==========

Section 1 : File Header
Section 2 : Lighting Materials
Section 3 : Nodes - Written recursively from the root.
Section 4 : Animation Channels
Section 5 : Animation References

Section Description
==========

Section 1 : File Header
==========

HEADER Storm3D_SW File identification header (8 bit ascii values) - no strlen or terminating '\0'.
FLOAT version Current version is 1.8, older formats are not described at this time.

Section 2 : Lighting Materials
==========
Defines the characteristics of the vertex lighting materials defined in this .SOD file.

UINT16 count - The number of lighting materials defined in this file.

LIGHTING_MATERIAL ARRAY(count) Array of lighting materials.

LIGHTING_MATERIAL
{
IDENTIFIER identifier Name of the lighting material.
COLOUR ambient Real time lighting ambient component
COLOUR diffuse Real time lighting diffue component
COLOUR specular Real time lighting specular component (only used by the phong illumination model)
FLOAT specular power Specular exponent, used to determine the 'shininess' of material using the phong illumination model.
UINT8 lighting model (constant=0, lambert=1, phong=2)
}

Section 3 : Nodes
==========

The nodes consist of 5 types NULL, LOD_CONTROL, SPRITE, MESH and EMITTER which together form a scene graph which describes the object's hierarchy.

UINT16 count - The number of nodes in the hierarchy.

<Update: Note node_type & identifier order has been switched to correct error in initial specification.>

NODE
{
UINT16 node_type (0 - null, 1- mesh, 3 - sprite, 11 - LOD control node, 12 - emitter) DO NOT USE OTHER VALUES.
IDENTIFIER identifier
IDENTIFIER parent (which will be null for root node)
MATRIX34 local transform
TYPE_SPECIFIC_DATA<node_type> Type specific data field as defined below.
}

Null Nodes
==========

TYPE_SPECIFIC_DATA<NULL>
{
No addtional data required.
}

Null nodes are used for two purposes :
1. As 'glue' to stick the rest of the hierarchy together
2. To mark specific locations in the hierachy, for example, hardpoints.

LOD Control Nodes
==========

TYPE_SPECIFIC_DATA<LOD_CONTROL>
{
No addtional data required.
}

Storm3D uses discrete (rather than dynamic) LODs for level of detail control. Each child of an LOD control node indicates a discrete LOD that the graphics engine may use when rendering this object. LOD selection is based on visible on-screen area.

Sprite Nodes
============

TYPE_SPECIFIC_DATA<SPRITE_NODE>
{
None: The appropriate sprite node definition to use is determined from the
identifier. The sprite node definition is defined in the .spr files.
}

Examples of sprite node usage include running lights in ST:Armada.

TYPE_SPECIFIC_DATA<PARTICLE_EMITTER>
{
IDENTIFIER Emitter used by this node as defined by an @emitter description in the .spr files.
}

Polygon Mesh Nodes
==================

TYPE_SPECIFIC_DATA<MESH>
{
IDENTIFIER texture material (0 for default) - Defines the TEXTURE_MATERIAL to be used by this mesh.
IDENTIFIER texture (0 if untextured)
UINT16 nvertices : Number of vertices
UINT16 number of texture coordinates (ntexcoords)
UINT16 number of vertex lighting groups (ngroups)

VECTOR3 ARRAY vertex positions (nvertices entries)
VECTOR2 ARRAY texture coordinates (ntexcoords entries)

VERTEX_LIGHTING_GROUP ARRAY (ngroups entries)

UINT8 cull type (0 - no cull, 1- (backface cull)

UINT16 0 - unused must be 0.
}

VERTEX_LIGHTING_GROUP
{
UINT16 num_faces (all faces are triangles)
IDENTIFIER lighting_material (0=default)
FACE ARRAY (num_faces entries)
}

FACE_VERTEX
{
UINT16 index into mesh vertex positions array
UINT16 index into mesh texture coordinate array

}

FACE
{
FACE_VERTEX ARRAY(3) 3 entries describing a triangular face.
}

Section 4 : Animation Channels (Defines transform animation)
==========

UINT16 count // Number of animation channels

ANIMATION_CHANNEL ARRAY(count) Array of animation channels.

ANIMATION_CHANNEL
{
IDENTIFIER node : The node to which this animation channel refers.
UINT16 nkeyframes : The number of keyframes used by this channel.
FLOAT channel_period : The length of time one loop of this channel lasts.
UINT16 0 : Not currently used. Must be 0.
MATRIX34 ARRAY(nkeyframes) keyframe_data : The actual animation transforms, evenly spaced over time 'channel_period'.
}

Section 5 : Animation References (Defines texture animation)
==========

Animation references are a way of linking texture (flipbook) animations defined in the .spr files to the geometry of a .SOD mesh node. An example of their usage is the flipbook animation applied to the geometry for the various shield effects in Armada.

UINT16 num_animation_references

ANIMATION_REFERENCE ARRAY(num_animation_references)

ANIMATION_REFERENCE
{
UINT8 type : Must be 4
IDENTIFIER node : The node to which this animation applies.
IDENTIFIER anim : The animation (as defined in .spr files) that is to be applied to this node.
FLOAT playback_offset : Time offset in seconds to be applied to this animation reference.
}

Additional Information
==========

Vertex Lighting Material Sharing

Vertex lighting materials are shared between objects, when parsing a .SOD file, Storm3D searches for a match in all previously loaded files. If a match is found, that material is used. This prevents artists from having to ensure the material characteristics of commonly used materials are correct in each file, and also saves memory.

A 'palette' of commonly used materials can be found in materials.sod
In Armada, this file is loaded prior to most other SOD files & so defines the characteristics of many common materials.

Hierarchy Structure (Armada Specific)

Armada uses various nodes in a Storm3D hierarchy for special purposes.
These include hardpoints, damage nodes, running lights, borgification.
When generating new artwork, the artist must pay careful attention to the structure of the hierarchy for the new object to function correctly in Armada.
A definition of the hierarchy structure required by Armada is beyond the scope of this document. For the time being, existing artwork can be used as a reference.

TEXTURE_MATERIAL Definition

A texture material defines the characteristics of the polygon rasterizer used to render the polygons in a given mesh. The texture materials are currently fixed and defined within the executable.

Useful values are :

default - Standard material
additive - Use additive blending
translucent - Semi transparent
alphathreshold - Use for objects using alpha channel 'cut outs' - alpha channels will have hard edged 'threshold' but objects will be drawn quickly.
alpha - Uses entire alpha channel. Object will require sorting, so will have performance implications.
wireframe - Use wireframe graphics.