# ZED

## File Formats

**Open Game Developers**

[Version: 0.0.0.1] [Date: 07|04|2013]

# ZED

## Change Log

| Version | Author | Changes |
|---|---|---|
| 0.0.0.0 | Rico | Initial revision |
| 0.0.0.1 | Rico | Altered the mesh chunk to incorporate necessary additional data. |

# Table of Contents

# Preface

This document describes the file formats which ZED will be using for assets intended for real-time consumption.

APIs used to access file format data will not be described in this document.

# ZED

# File Containers

## How Files Are Stored
Files for ZED are stored in one of two ways; an flat file image format, individual files. Image files can be compressed. Individual files store more data to describe the file's contents.

### Image Files
ZED's image files store ZED-specific and generic files which can be managed by ZED. ZED-only files have their headers stripped and stored at the index of the image. File names are hashed for quick retrieval and smaller storage space.

### Individual Files
Asset files for fonts, 3D models and animations will need to be loaded fast and take up as little storage and memory space as possible. Textures, video, and audio files will most likely not be bespoke files, as the existing formats are optimised for the platforms which they run on. Fonts, 3D models, and animations need to be tailored for usage on each platform and features must carry across with them.

# ZED

# Individual Files

## File Format

All individual ZED files will have a header which describes what the file is representing and a series of chunks to describe the actual file's contents which will in turn be loaded by the ZED API.

## Header

Headers for individual ZED files will describe as much as is necessary to funnel an asset down through the content runtime pipeline.

| **Name** | **Type** | **Description** |
|---|---|---|
| ID | char [3] | The ID should always be “ZED” |
| Type | char | One-char identifier for the file type:<br>• 'M' == Model<br>• 'A' == Animation<br>• 'F' == Font |
| Version | char[ 3 ] | 3-dotted decimal:<br>Byte Distribution<br>0 Major<br>1 Minor<br>2 Revision |
| Flags | uint32 | 32 bits for setting flags, such as if a model has an animation, whether an animation contains a set of inverse kinematics, or if a font is vector-based, for example. |

*Table 1 | Individual File Header Format*

## Chunk

Each chunk describes the data to follow with as small a footprint as possible

| **Name** | **Type** | **Description** |
|---|---|---|
| Type | uint16 | A token to identify the following data |
| Chunk Size | uint32 | Complete size of this chunk (in bytes) after the chunk description |

Chunks are self-contained and are completely isolated. Which creates a dependency on the data being correct for all chunks and interpreted fully.

# ZED

# Models

## File Chunks

### Model Meta Data

| Name | Type | Description |
|---|---|---|
| Vertex Count | uint32 | Total vertices for the model |
| Index Count | uint32 | Total indices for the model |
| Mesh Count | uint32 | Total meshes for the model |
| Material Count | uint32 | Total materials for the model |
| Model Name | char [ 64 ] | Used primarily for debugging purposes |
| Triangle Strips | uint32 | Total amount of triangle strips |
| Triangle Strip Count | uint32 | Triangles generated via triangle strips |
| Triangle Lists | uint32 | Total amount of triangle lists |
| Triangle List Count | uint32 | Triangles generated via triangle lists |
| Triangle Fans | uint32 | Total amount of triangle fans |
| Triangle Fan Count | uint32 | Triangles generated via triangle fans |
| Version | char [3] | 3-dotted decimal: |

| Byte Distribution | | |
|---|---|---|
| 0 | 1 | 2 |
| Major | Minor | Revision |

### Meshes

Each mesh can contain multiple different types of triangle descriptors. After the mesh chunk, the triangle indices are stored declaring first the total number of indices per type then storing the actual indices afterwards.

| Name | Type | Description |
|---|---|---|
| Vertex Count | uint32 | The set of vertices for the mesh |
| Material ID | uint32 | Material ID for the mesh |
| Strips | uint32 | The amount of strips in the mesh |
| Lists | uint32 | The amount of lists in the mesh |

| | | |
|---|---|---|
| Fans | uint32 | The amount of fans in the mesh |

Mesh data is stored immediately after the mesh header chunk.  Vertices adhere to the format specified in the Vertex Data section.  Following the mesh header chunk, vertices are stored, then the strips, lists, and fans.  Before each strip, list, or fan array, the size of the array is stored.  If multiple arrays are used, the size will be extracted, memory allocated for the array, and the array will be read in.  The process repeats until the *n* arrays are read.

### Vertex Data [REFERENCE]

All mesh vertices are stored as follows

| Name | Type | Description |
|---|---|---|
| Position | float [ 3 ] | Vertex position |
| Normal | float [ 3 ] | Vertex normal |
| UV | float [ 2 ] | Texture UV |
| Joint Weight | float [ 4 ] | Range 0…1 for each joint |
| Joint Index | float [ 4 ] | Aliases the weights to the joints |

Vertices are not stored as a file chunk, this information is for reference only.