ZED

File Formats

Open Game Developers

[Version: 0.0.0.0] [Date: 23|02|2013]

Change Log

Version	**Author**	**Changes**
0.0.0.0	Rico	Initial revision

ZED

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.

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: • 'M' == Model • 'A' == Animation • 'F' == Font
Version	char[3]	3-dotted decimal: Byte Distribution 0 | 1 | 2 Major | Minor | Revision
Flags	int32	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.
File Size	int32	The size (in bytes) of the rest of the file after the header

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	int16	A token to identify the following data
Chunk Size	int32	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.