SurvivorSurvivor Helicopter!

Level Data Format

Open Game Developers

[Version: 0.0.0.1] [Date: 13|08|2013]

SurvivorSurvivor Helicopter!

Change Log

Version	Author	Changes
0.0.0.1	El-Rico	A first-adjustment pass of the level file format

SurvivorSurvivor Helicopter!

Table of Contents

SurvivorSurvivor Helicopter!

Preface

This document exists to detail the particulars of the level format used for SurvivorSurvivor Helicopter! Level formations will not be described in this document, see the Level Designs Document for more information on the levels used in the game.

SurvivorSurvivor Helicopter!

Anatomy of a Level

Components
Tiles are assumed to be meshes in the ZED mesh file format.

Tile ID
A 16-bit value which provides more than enough variety in tiles for a single level.

Tile Flags
A 16-bit value, which indicates whether this tile is a spawn point or if it is a helicopter landing zone (additional flag

File Layout

Type		Count	Name	Description
char		4	ID	Contains “SSHL” (SurvivorSurvivor Helicopter! Level)
char		256	Path	Path to the tile set
uint16		2	Dimensions	The width and depth of the level, respectively
*for width*height*				
	uint16	1	Tile ID	The tile to use (tile sets contain meshes with names starting at zero, zero and one are reserved for the spawn point tile and the helicopter landing zone tile, respectively)
	uint16	1	Tile Flags	Sixteen OR-ed flags which determines the type of tile this is
end for				

Loading a Level
Levels are stored in a .ssh file, which reference 3D tiles. 3D tiles are named “xxxxx.zed”, using the ZED model format, each tile can have animation data as well as geometry data. Tile names are numbers in the range of 0-65535, any file with less than five digits will be left-padded with zeros, such as; “00001.zed” or “00014.zed”. The path for the tiles is relative to the tile data file, not the executable.

Pseudocode

```
read file_header;((char*4)+(char*256)+(uint16*2))
if compare( file_header.id, “SSHL” ) not successful
        return;
if failed( check_tile0_and_tile1_are_present( file_header.path ) )
        return;
if failed( check_all_tiles_in_file_exist( file_header.path ) )
```

```
        return;
store_tile_ids_in_order_with_index( )    ;Takes the tile IDs from the previous check and
                                                ;stores them in a list so that the renderer
                                                ;doesn't need to keep switching meshes and
                                                ;textures as often
```

Rendering a Level

Levels are rendered so that they are always centred. Using the bounding box for the level, this is easily determined and allows for both odd- and even-numbered rows and columns to exist. As levels can be rotated Pi/2 radians, this helps with ensuring the levels are centred correctly.

Pseudocode

```
for( x = 0; x < tile_id; ++x )
        itr =  tile_id_position_map[ x ].begin   ;tile_id_position_map = map< int, list > - The list contains
                                                 ; all of the positions for the tile
        while( itr != tile_id_position_map[ x ].end )
                render_tile( x, itr.position )
```