tPBin

User's Guide

May 3, 2010

Gregg A Tavares & Dan Chang

♻ Printed on recycled paper (contains 50% waste paper including 10% post consumer)

tPBin

tPBin is a *tUMEPack* designed for programmers that want some kind of “raw” or “binary” dump of the tUME data so that they may create their own tUMEPACK-like programs that process these "raw" dump files instead of directly reading tUME files. When creating a tUMEPack to generate data for you games, we recommend that you modify the code for DUMPTUME to create your own tools, as it contains routines to read tUME maps directly; however, if you are not comfortable with modifying DUMPTUME or one of the other versions of tUMEPACK, then here is another solution you might be more comfortable with.

tPBin outputs following files:

<*RoomName*>.R*nn*, one for each layer *nn* in <*RoomName*>: two-dimensional array of tile indexes words
<*RoomName*>.F*nn*, one for each layer *nn* in <*RoomName*>: two-dimensional array of tile flag bytes
<*RoomName*>.PAL, one for each room: color information for this room
<*TilesetName*>.T*nn*, one for each tileset type: actual graphic images (one byte per pixel) of each tile

tPBin works as follows:

1. Read in all maps.

2. Find all tilesets with the same UserType, and "merge" them together. E.g., if you have two tilesets, SPLOON.LBM with 25 tiles and SPLAT.LBM with 17 then when they are merged SPLOON.LBM tiles will be numbered from 1 to 25 and SPLAT.LBM tile will be numbered 26 to 42.

3. Write out the tile graphics for each merged tileset to a binary file "*.T??".

4. For each room, write a file "*.PAL" with the palette of 256 colors.

4. For each room, sort tiles to make all tile of user type *n* appear on layer *x*. *n* and *x* are specified using the -m switch.

5. For each layer in each room, write a file "*.R??", which is a two-dimensional array of words, where each word is an index into the merged tileset.

6. For each layer in each room, write a file "*.F??", which is a two-dimensional array of bytes, where each byte is the tile flag byte at that location. Tile flags bits are defined in the tUME.INI file.

tPBin Command-Line Switches

Single letter switches with no arguments can be turned on with a '+', and turned off with a '-'. In the following list, the '-' or '+' in front of the switch specifies the default setting of that switch, thus debugging (-d) is off by default, while (+d) would turn it on.

-d	Disable debugging messages. +d to enable debugging messages.
-f<n>	First Tile. When tiles are renumbered after like tilesets have been merged the first tile is usually numbered 0. NULL tiles show up in the .R?? files as zeros (0). This option lets your specify the number of the first tile in the merged tilesets. If you use ‘-f0’ then you won’t be able to tell the difference between the first tile and NULL tiles; however, this may be what you want.

-g	Write Room Width/Height .R?? files. +g to write width and height information into the files.
-k	Generate only tiles used in rooms. Normally all tiles are generated from the tilesets whether or not they are actually used in any room. +k will make *tPBin* generate only those tiles actually used and will renumber the tiles in the .R?? files so they point to the correct image.
-m<tileMap>	**This option MUST BE SPECIFIED!** This option specifies which tiles should be put in which layers (or which .R?? files) You should account for all types of tiles that will appear in your map otherwise you will get a warning that a tile is being discarded. The mapping is specified by entering a tileset UserType followed by an '=' followed by a layer number. Layers start at layer 0 (zero). For example: -m0=0,1=1,3=2,2=3. This would mean tiles of UserType 0 are written to layer 0, tiles of usertype 1 to layer 1, tile of type 3 to layer 2 and tiles of type 2 to layer 3
-p	Not implemented yet.

tPBin File Formats

NOTE: All words are in Little Endian format (low byte first)

.T?? files (Tileset Graphics)

A binary .T?? file, where ?? is the tileset UserType in hex, is generated for each 'merged' tileset (see above). The filename will be the name of the first tileset of this specific UserType tUME finds in the map. The data is stored 1 byte per pixel. If the tileset used 8x8 pixel tileset then the first tile will be the first 64 bytes of the file and the second tile will be the second 64 bytes. The size of the tiles is not encoded in this file.

.R??

A binary .R?? file, where ?? is the number of the layer in hex, is generated for each layer in a room. The data is stored as an array of words. The array is <room width> by <room height> of words. Each word indexes the corresponding merged tileset.

The file is a two-dimensional array of words:

WORD map_index[{room height}][{room width}]

A word value of zero indicates the absence of any tiles at that location. By default, tiles are numbered starting from 0, but you may use the -f switch to specify a different starting number.

If the +g option is specified then the file is preceded by the size of the room in blocks.

WORD room_width
WORD room_height

.F??

A BINARY .F?? file, where ?? is the number of the layer in hex, is generated for each layer in a room. The data is stored as an array of bytes. The array is <room width> by <room height> of words. Each bytes is the tileflags for the corresponding tile in the .R?? file. Tileflags are interpreted as specified in the tUME.INI file; the defaults are:

BITS	0-2	Colorset of tile
BIT	3	Unused
BIT	4	Tile has priority set.
BIT	5	Tile is flipped vertically
BIT	6	Tile is flipped horizontally
BIT	7	Colorset information (bits 0-2) are used

The file is a two-dimensional array of bytes:

UBYTE tileflags[{room height}][{room width}]

.PAL files (Palettes)

A Binary .PAL file is generated for each room. There are 256 entries each consisting on three bytes (red, green and blue). The red, green and blue values range from 0 to 255 with 255,255,255 being white.

PALETTE[256] (formatted as follows):

PALETTE consists of:
- **UBYTE** Red color information
- **UBYTE** Green color information
- **UBYTE** Blue color information