Jefferson TDR

Section 17.4.1 : TDSBuildObject
Jamie Doornbos
10/13/97
Revision 0.

(Note: this document is numbered from 1 instead of 17.4.1 since the numbers may change and the other tools document (behavior editor) has no numbers.)

TDSBuildObject

TDSBuildObject (tdsb.exe) is a command line tool for generating new objects and modifying existing ones. It operates on "OMK" files. OMK is an abbreviation for "object make". Depending on the parameters passed to tdsb, it can copy an object from an object template file, build a set of sprites from a targa file, build draw group resources, build slot resources and labels, create sound string resources, and make object attribute labels.

This is the usage string provided by the executable when just the name is typed at the command line:

```
#######################################################
# usage: tdsbuildobject (dest-object-file)
# [-tmpl (template-object-file)]
# [-spec (omk-attribute-file)]
# [-wall (tga-file) (palette-file) ((style number)|(style number)c)]
# [-headline (tga-file) (palette-file)]
# [-dgrp (tga-file) (palette-file)]
# [-dgrp2 (tga-directory) (palette-file)]
# [-sounds]
# [-slot]
# [-attr]
#######################################################
```

1. OMK Files

An OMK file is a tagged text file. TDSBuildObject extracts information from the OMK file by looking for *tags.* Tags have this form:

<name0[=value] [name1[=value1]] …>

A tag must have at least one name. The first name is the name of the tag. All values are optional. Both names and values may be a sequence of non-special characters, or a sequence of characters inside quotes. Special characters are: white space (tab, space, return), and '='. Unquoted names and values are automatically converted to lowercase as they are read in. Quoted names and values may not contain quotes. The parser may accept things other than this, but its best to keep it simple. Here are some sample tags that pass the parser:

<sprite id=129 foo=bar who me>
<sound id=347 name = asound>
<attribute index = 4 "dance command"
= "shake your booty">

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

2. TGA files and palette files

Several options to tdsb require a TGA file and a palette file for compression of graphics into sprite list resources.

In addition to other properties for specific options, TGA files are assumed to have certain overall properties:

- Must have 3 or 4 channels (24 or 32 bit). If it has 4 channels, the 4th channel, the "alpha" channel, is used as a z buffer, and the final sprites produced will have z buffer information.
- Must consist of a number of rows of graphics, all of the same height.
- There is no vertical space between rows and the top edge of the top row is the same as the top edge of the file.
- Each row consists of a sequence of large graphics, then the same number of medium graphics, then the same number of small graphics. The valid numbers of graphics are 1,2, and 4.
- The top of each graphic is the same as the top of the row it is in.
- There is no horizontal space between graphics. The leftmost large graphic in each row has its left edge at the left edge of the file.

The palette file is expected to be in Photoshop's "ACT" format (a simple raw 3 bytes-per-color format) and have 256 colors. The palette is used to map 24 bit colors in the TGA to 8-bit colors in the final sprite. Thus the palette must be present to decompress the graphic. Currently sprites are all rendered with the same palette, so this argument should always be set to the game palette. TBD: how to embed or refer to the palette from the sprite or object—will probably change the sprite format. TBD: to implement lighting effects, there may be no intermediate TGA files, or an associated lighting map may be optionally searched for each TGA.

3. Compression

The compressor classes from the Jefferson code base use a tokenized 8 bit compression format. TBD: should details of format appear here?

3.1 Output: Sprite Lists

A sprite list is an 'SPR#' resource. It contains a number of individual sprites, indexed from zero. A unique sprite is specified by id and index. The header for the resource contains the number of sprites, some version information, and a table of offsets to the compressed data. Currently, the version determines if a sprite list has z buffer information. Such a list has the same format as other lists, but the last half of the sprites are interpreted as z buffers. TBD TODO: whether or not a sprite list has z buffer information is better off in a flag field, and version should refer to the format of the resource.

3.2 Input: Compression parameters

Certain parameters to the compressor determine when and what tokens are used:

- Run tokens: sequences of the same color longer than a specified minimum value are compressed into a single run token. In TDSB, this feature is turned on and the minimum run value is set to 16.
- Clear tokens: all runs of any length of a specified clear color are compressed into clear tokens. In TDSB, this is always turned on and the clear color is the first color appearing in the rectangle to be compressed (before shrinking).
- Color Masking: forces colors generated to have values between a lowest color and a highest color. TDSB uses a lowest color of 10 and a highest of 246. TBD: this constraint is necessary for 8-bit graphics but not for 16.
- Pattern tokens: with the pattern feature turned on, the compressor searches for special pattern colors: light, medium, and dark. Each color gets compressed into a light, medium, or dark token, which is replaced by a pixel from a pattern structure at decompression. TDSB does not currently use this option. TBD: revival of pattern tokens for wall paper.

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

- Shading tokens: with the shading feature turned on, the compressor makes shade tokens for colors within a certain range. The values indicated by shade tokens are relative to the base color, so at decompression, the base color can be swapped, and lighting effects maintained. TDSB does not use the shading option.

3.3 Pre-processing: Shrink-wrap

Shrink-wrap refers to the process of reducing the size of a rectangle to be compressed as much as possible without losing non-clear pixels. This has the effect of making the bounding box of the final sprite small, which optimizes clipping tests in decoding. Shrink-wrapping works by altering each side of the rectangle (towards minimum total area) until a non-clear pixel is crossed. Shrink-wrap takes place before compression, and can be optionally limited to a collection of edges. In TDSB, shrink-wrap is turned on for all edges during draw group generation, but can optionally be turned completely off on a per-sprite-list basis. For wall and headline compression, it is turned on only for the top edge.

4. TDSBuildObject options

Each option executes a different function in tdsb.

4.1 -tmpl option

`-tmpl (template-object-file)`

Template file option. This option has one argument: the name of an object file to be copied to the destination file. All resources in the source object file are copied to the destination file. Resources of type OBJD are modified to have a new guid field (see objects chapter, 3.2.1.14) and a new name that is the same as the destination file name. A copied object is therefore ready to be plugged into the game without conflicts with other objects. TDB: currently, in copying an object file, all objects in the resulting file will have the same name. It may not be worth fixing this, since the names can be edited in edith.

4.2 -spec option

`-spec (omk-attribute-file)`

Attribute file option. The argument is the name of the OMK file to use in processing most of the other options. It is only not required for -tmpl and -wall.

4.3 -wall option

`-wall (tga-file) (palette-file) ((style number)|(style number)c)`

The wall option is used to create wall sprites from source TGA graphics. The TGA file is expected to have the default layout generated by the sprite exporter (see Sprite Exporter section, ???). TBD: this is somewhat rigid—walls should probably use an OMK file. Currently the -wall and -spec are not allowed together. 3 rows of graphics are expected: one showing a corner section in the first rotation; one showing a left edge in the first rotation and a right edge in the second rotation; and one showing a left-to-right section in the first rotation and an top-to-bottom section in the second rotation. Other sprites are ignored. These are demonstrated here with grayed boxes to show ignored areas:

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

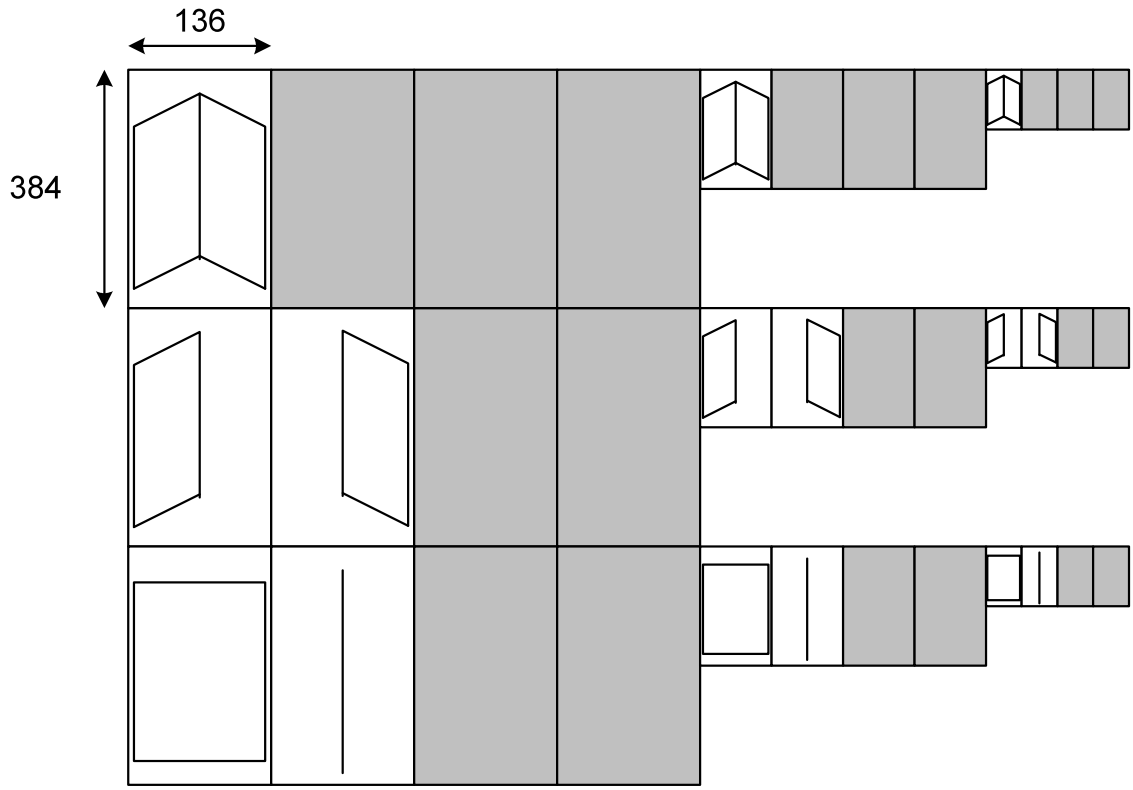


Figure: which graphics are expected by the -wall option. The sizes of the smaller areas are half and quarter of the largest size, both horizontally and vertically.

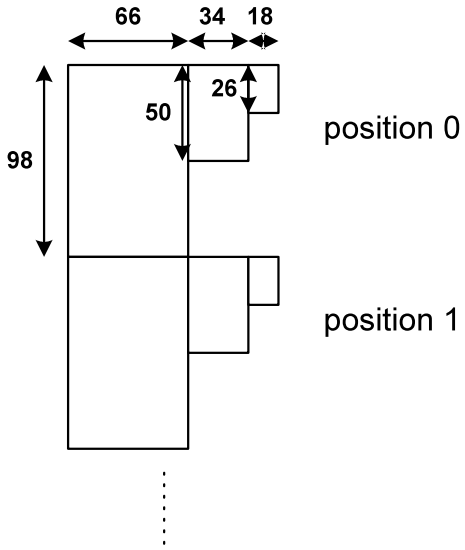
TBD: the wall compression process may need to include a scheme for representing wall paper, such as a key color for different facing directions.

4.4 -headline option

`-headline (tga-file) (palette-file)`

The headline option uses certain compression options to make headline sprites. The -spec option is required, and "headline" tags are the only ones used from the OMK file. The TGA file is expected to have 3 graphics on each row: large, medium, and small. Each graphic has a one pixel border which is ignored. Each row is flush against the row above it and the top row is flush against the top, left of the whole image. This is demonstrated here, with pixel widths, including borders:

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.



The headline tags in the OMK file determine what gets compressed where. The parameters of the tag are interpreted as follows:

- id: id of the resulting sprite list resource. This is a required parameter.
- position: which row of graphics to compress. Default value is the last position value + 1, or zero if there are no previous headline tags.
- frames: how many consecutive rows to compress into the sprite list. Default value is 1. Frames are sequenced automatically when the headlines are rendered.

4.5 -dgrp and -dgrp2 options

```
-dgrp (tga-file) (palette-file)
-dgrp2 (tga-directory) (palette-file)
```

These options use the tags in the OMK file specified by the required -spec option to generate the draw groups and sprites for an object. The necessary tags are “sizes”, “sprite”, “drawgroup”, and “item”:

- sizes: describes the dimensions of each size graphic (large, medium, small) in a TGA file.
- sprite: describes the layout of a row in the TGA file and a destination sprite resource, and possibly a TGA file.
- drawgroup: describes how to begin building a ‘DGRP’ resource.
- item: describes a single element in a draw group.

The -dgrp option specifies a single TGA file to use for all sprites being generated. The -dgrp2 option specifies a directory and the sprite tags individually specify the file relative to this directory.

4.5.1 Scale and Rotation

For these options, the graphics in the TGA file, in addition to the layout assumptions described above, also have scale and rotation assumptions. The large, medium, and small graphics correspond to the three zoom levels of tds. The columns of each size correspond to the rotations of the view (or object). The first column is rotation value 0 and the graphic should be “facing” back and to the right. The second , third, and fourth columns are rotation values 2, 4, and 6 and the facing is rotated clockwise by 90 degrees for each column. See also below: columns field in sprite tag. TBD: what about objects that use the full 8 directions?

4.5.2 Sizes Tag

Within each large, medium, and small graphic, the sizes tag describes the area to be compressed and where the origin of the tile is. The position of the tile origin is important in determining the draw group sprite

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

offsets, which are expressed as pixel offsets from the tile origin. The “shrink wrapping” process makes these offsets non-constant and pixel-dependent.

The parameters of the sizes tag, their interpretations, and their default values are as follows (refer to the figure below):

- slopwidth: the width of the border around the “cage” which may contain pixels but is not strictly part of the tile. The value of this parameter is for the largest graphic. The medium and small graphics take half and quarter of the value for their slopwidths. If “slopwidth” and “uniform slopwdith” are both specified in a sizes tag, the program produces an error. Default value is zero.
- “uniform slopwidth”: same as slopwidth, except the value is not divided for the lesser sized graphics. Default value is zero.
- border: the width of the area around each graphic which is not considered for compression. The same value is used for all 3 sizes. Default value is zero. Essential for hand edited graphics so the area to be compressed is obvious.
- lwidth: The width of the largest graphic, including slop and border. This value is required.
- mwidth: same for medium graphic.
- swidth: same for small graphic.
- lheight: The height of the largest graphic, including slop and border. This value is required.
- mheight: same for medium graphic.
- sheight: same for small graphic.
- lxorigin: horizontal pixels to the center of tile from the left edge of the large sprite box. Default value is lwidth/2.
- mxorigin: same for medium box. Default value is mwidth/2.
- sxorigin: same for small box. Default value is swidth/2.
- lyorigin: vertical pixels to the center of tile from the top edge of the large sprite box. Default value is lheight - half large tile height - large slop width - border width.
- myorigin: same for medium box. Default value is mheight - half medium tile height - medium slop width - border width.
- syorigin: same for small box. Default value is sheight - half small tile height - small slop width - border width.

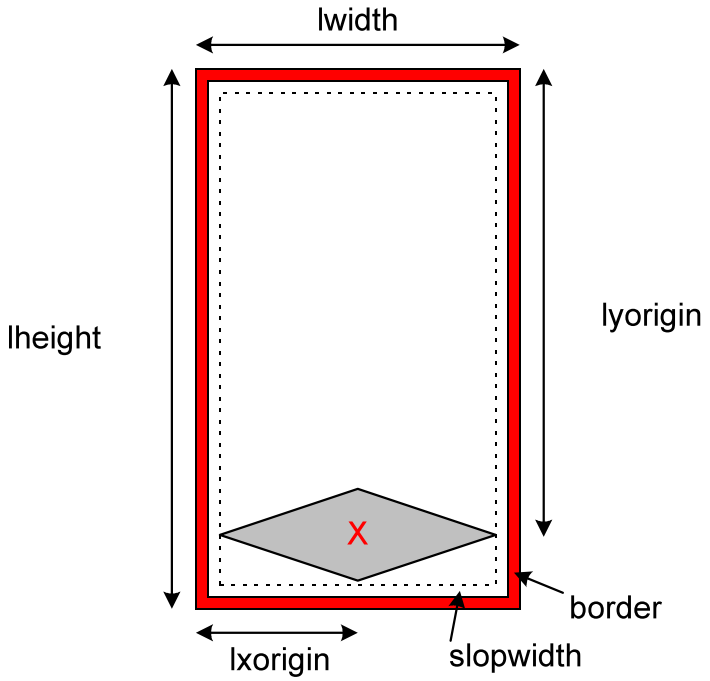


Figure: meaning of values in sizes tag for a large graphic. The X is the center of tile point.

Currently the graphics generated by the sprite exporter require the following sizes tag:

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

<sizes lheight=384 lwidth=136 mheight=192 mwidth=68 sheight=96 swidth=34 slopwidth=4>

4.5.3 Sprite Tag

The sprite tag specifies a row of graphics to compress, some layout parameters for the row, and a destination sprite list resource. The dimensions of the boxes in the row are those of the last sizes tag encountered. The fields of the sprite tag are interpreted as follows (see figure):

- position: the source row in the TGA. Default value is the position of the last sprite tag +1, or zero if there are no previous sprite tags.
- id: the id of the destination sprite list resource. This parameter is required.
- name: the name of the destination sprite list resource. Default value is an empty string.
- columns: the number of columns for each size graphic. Must be either 1, 2, or 4. Default value is 4. If less than 4, the draw group building process assumes the mapping from rotations to sprite index just wraps around for each scale, and that the graphics are not to be flipped for wrapped rotations. This option is currently only used for the trees, which were hand edited.
- shrink: whether or not to "shrink wrap" the sprite (see above). Default value is true.
- overwrite: whether or not to overwrite the old offsets for this sprite in the draw group. Default value is false.
- reverse: true if the indices of the final sprites should be reversed. Default value is false. The option is currently used only for floors, which are expected by the game rendering code to have small sprites first in the list.
- tgafile: for use only with the dgrp2 option. Specifies the TGA file to use for the sprite, relative to the TGA directory.
- TBD TODO: symmetry: describes which columns are just flipped versions of other columns. Reduces the number of sprites in the final list, and causes draw group flip values to get set for adjacent rotations. Possible values are "none", "lateral", and "radial". Default value is "none".

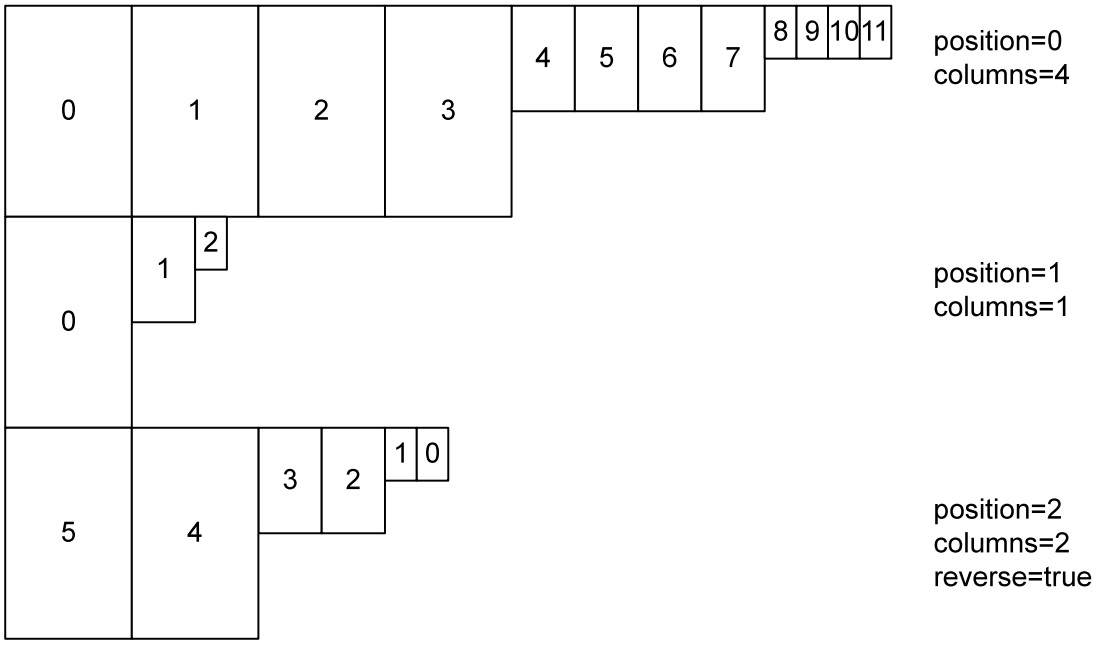


Figure: Sample TGA file layout and sprite tag parameters. The number in each graphic box is the index of that graphic in the final sprite list.

In the current set of OMK files, sprite tags referring to TGA files output by the sprite exporter do not refer to "columns" or "reverse", thus taking on the default values of "columns = 4" and "reverse = false".

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

4.5.4 DrawGroup Tag

The drawgroup tag begins the expression of a draw group resource, 'DGRP'. It specifies the id and name of the resulting resource. A draw group is made up of draw lists, one list for each rotation and zoom. A draw list is made up of items, one for each item that composes the graphic of the object at that rotation and zoom. Subsequent item tags (until the next draw group tag or the end of file) are incorporated into the items of each draw list. The parameters and meanings of the draw group tag are as follows:

- id: the id of the destination draw group resource. Required parameter.
- name: the name of the destination draw group resource. Default value is "".
- overwrite: true if the old draw group should be replaced. False if new items should be appended to or swapped in to the old draw group. Default value is false. This parameter needs to be true when an item is being removed from the draw group. Usually it is only set to true temporarily, because otherwise the -dgrp and the -slot options, when run separately, will overwrite each other's draw items.
- ordering: "fixed" or "obscured". The "obscured" value causes items to be inserted in reverse order for rotations that face "away from the camera", useful for component objects that have sprites with no z buffer. The default is "fixed" and no OMK files currently specify anything else. TBD: this will probably never be used since we now have z buffers.

4.5.5 Item Tag

Item tags are used to specify the drawing sequence of sprites, "top objects", and slots. For each item tag, an entry is created in each draw list, automatically modifying anything that depends on rotation or zoom. The first name after the tag name, "item" is an indicator of the item's type: "sprite", "object", or "slot".

4.5.5.1 Sprite items

A sprite item denotes a sprite that should be drawn. The id field is required and specifies the sprite list that this sprite item comes from. The item inserted into each draw list indicates the appropriate graphic for the draw list's direction and zoom, and the correct offset of the top left of the sprite from the center of tile position. As an optimization, the draw code may attempt to draw the first sprite encountered without reading from the z buffer. Thus the first sprite item in each draw group should be the biggest sprite (most pixels) of the group.

4.5.5.2 Top Object Items (obsolete)

A top object item used to be a sign to draw the contained object (target). It also specified the pixel offsets. This type of item is not referenced from the draw code anymore, and should not be used. The more versatile slot items have replaced top object items.

4.5.5.3 Slot Items

A slot item indicates a particular slot object to be drawn and where to draw it. The slot item should only be used if the coordinates of a slot in the current draw group are different than the default coordinates found in the slot itself, because the '-slot' option inserts draw group items as well (see "-slot" option, section 4.7 of this document). For backwards compatibility in the drawing code, slots are drawn independently of the draw group if the draw group has no slot items.

The parameters for the slot item tag are as follows:

- index: the index of the slot to draw. Object slots are indexed from zero. Required.
- x: the x offset of where the slot object's center of tile should be placed when drawing.
- y: the y offset for the same.
- alt: the altitude offset for the same.

The coordinates in the slot item are in slot coordinates, described in section 2.1. (Floating point text representation automatically converted to appropriate fixed point.) When the item is added to each draw list, the coordinates are rotated to the direction for that list.

4.5.5.4 TBD: Handle items.

For objects that may be carried and whose handle coordinates change with the frame of their graphic, the draw group would be the most appropriate place to specify the handle coordinates.

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

4.5.5.5 TBD: Headline items.

For objects that have headlines whose coordinates change with the frame of their graphic, the draw group would be the most appropriate place to specify the headline coordinates.

4.6 -sounds option

`-sounds`

The sounds option reads “sound” tags from the OMK file specified by the required -spec option and creates an ‘FWAV’ resource in the destination file for each one. ‘FWAV’ resources are used by the game sound player to allow clients to refer to WAV files by id. A sound tag has only an id and a name. The id is the id of the destination FWAV resource, which is used to play the sound by game sound clients. The name is the name of the WAV file to be played for that id, and is specified relative to the Sounds directory (TDSScen/Sounds). TBD: How to have plug-in sounds. Currently the main OMK file for the game sounds is “$/TDS/TDSScen/sounds/sounds.omk”.

4.7 -slot option

`-slot`

The -slot option reads in “slots” and “slot” tags from the OMK file specified by the required -spec option. It uses the information from the tags to create ‘SLOT’ resources, create a string list resource of slot labels, and add slot items to draw group resources. Only one “slots” tag is allowed per OMK file and has the following fields:

- id: the id of the destination ‘SLOT’ resource. Required.
- name: the name of the slots. Used to name the saved string resource. Default is “”.

Each slot tag builds on the current set of slots since the last “slots” tag and has the following fields:

- type: one of “headline”, “handle”, or “contained object”. This field specifies what type of slot is to be created and the default is “contained object”.
- name: specifies the label of the slot to be used in the editor or in inspector windows. Default is “”.
- x, y, alt: the coordinates of the slot. The defaults are 0,0,0. If any one is specified, the slot is added to each draw group specified by the OMK file. TBD: this limits OMK files to one-per-object.

‘SLOT’ resource

The destination ‘SLOT’ resource has no name and takes the id specified in the “slots” tag. It includes a binary representation of each slot tag. This resource is referred to by id in the object definition and is loaded in when an object is initialized.

Slot labels

The string list resource of slot labels is used to display the slot names in the editor and in inspector windows. It takes the name specified in the “slots” tag and the default id for object slot labels (constant XObjLang :: kSlotNamesStringsID = 257). TBD: this is sloppy since the objects in the same file do not necessarily share slots. Should be in the slot resource itself, or in another resource with the same id. If a label is not present, the previous label from the resource is left as-is. Thus the label must be specified as “” to get rid of a label.

Slot Items in draw group

If coordinates are found in a contained object slot tag, the slot is added to each draw group specified in the OMK file as described in the “-dgrp” option, section 4.5 in this document. TBD TODO: currently this is overwriting existing slot items in the draw groups, which is a bug since the draw groups may already have frame-specific coordinates attached to the slot items. The slot coordinates should be placed in the draw group all at once prior to the frame-specific coordinates.

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.

4.8 -attr option

`-attr`

The "-attr" option reads "attributes" and "attribute" tags from the OMK file specified by the required -spec option. It uses the tags to create labels for an object's attributes (see Object attributes, section 3.3.2.6). The attributes tag specifies the id and name of the destination resource. The id, however, is currently not stored anywhere else, and is assumed by the editor to have a constant value (XObjLang :: kAttrStringsID = 256). TBD: the attribute labels id should be specified by the object definition so different objects in the same file can have different attribute labels. Each subsequent "attribute" tag, until the next "attributes" tag, adds a label to the string list. Each attribute tag specifies the following fields:

- index: which attribute is being labeled. Required.
- label: the label for the attribute. Required.

5. Tag Formats Appendix

This is just a list of valid tag formats. Where an integer is expected, "integer" is used. Likewise, "string" and "float" denote where strings and floating point numbers are expected. "boolean" denotes a value which is expected to be "true" or "false". Optional parameters to a tag are shown with brackets, "[ ]".

Sprite

<sprite id = integer [name = string] [position = integer] [columns = integer] [shrink = boolean] [overwrite = boolean] [reverse = boolean] [tgafile = string]>

DrawGroup

<drawgroup id = integer [name = string] [overwrite = boolean] [ordering = string]>

Item

<item sprite id = integer>
<item object>
<item slot index = integer [x = float] [y = float] [alt = float] >

Sizes

<sizes lwidth = integer mwidth = integer swidth = integer lheight = integer mheight = integer sheight = integer [slopwidth = integer] ["uniform slopwidth" = integer] [border = integer] [lxorigin = integer] [mxorigin = integer] [sxorigin = integer] [lyorigin = integer] [myorigin = integer] [syorigin = integer]>

Sound

<sound id = integer name = string >

Slots

<slots [name = string]>
<slot type = string [name = string] [x = float] [y = float] [alt = float]>

Headline

<headline id = integer [position = integer] [frames = integer]>

Attributes

<attributes id = integer [name = string]>
<attribute index = integer label = string>

Confidential and Proprietary Information
May not be reproduced or distributed without the express permission of Maxis, Inc.