Object file format in The Sims

Jamie Doornbos, MAXIS, January 2000.

An object in The Sims is a collection of resources, stored in the object's IFF and SPF files. For an overview of resources, see the separate document "Resource file Overview".

Object resources

OBJD – Object Definition

Struct ObjDefinition defines the layout of this resource directly (unlike most others, which use a stream to layout the resource). Every type of object (including multi-tile parts) has exactly one definition at runtime. Almost every aspect of the object type stems from this resource. For example, the C++ class to be constructed for an instance is determined by the "type" member. Also, the object's GUID is stored here. The values in the struct are set by various pieces of code. The GUID is set by class StubObject when duplicating an object or an object file (called by Edith, TDSB, and create-a-character screen). Other fields, such as multi-tile coordinates, are typed in by a programmer in the "Definition Editor" in edith. Still other fields, such as price and depreciation, are set by TDSB when reading a catalog OMK. Typically a field in this struct defines an aspect of the object that needs to be known before or during initialization of an instance.

OBJf – Object function table

Class ObjFnTable defines the layout of this resource, using a ReconBuffer stream. The class defines an API for accessing tree entry points, enumerated in enum ObjEntryPoint. For example, the first tree to run when an instance is created is the "init tree," and is specified here. The entry points are edited only in edith, in the object function editor. An OBJf resource is associated with an object through its resource id, equal to the id of the OBJD. Every type of object has an object function table at runtime. However, some older objects still define entry points the old way in the definition (fields with suffix TreeID_unused). When these objects are loaded, an ObjFnTable in constructed and converted from the definition. Once the functions are edited and saved, a resource will exist from there on.

CTSS – Catalog strings

Class StringSet determines the layout of this resource (through class CatalogResource). In it's definition, a single tile object or the lead piece of a multi-tile object specifies the id of the CTSS to use for its catalog strings. Typically the cataog id is 2000 and higher, sequentially. CTSS resources may be created, deleted and edited in an object file's "Behavior Strings", accessed through the behavior browser in edith. Existing CTSS may be edited by Slang.

Object simulation resources

The components of the behaviors for an object are stored in various resources. Some are used by the simulation, and others only by editors.

BHAV – behavior trees

Struct BehaviorTree defines the layout of this resource directly. Each BHAV is potentially an object tree entry point, or a callee of another behavior tree. One BHAV calls another by resource id. The resource is written mainly by class cTree, an editing utility class used by the tree editor in edith, and the trace window in the game. The simulation accesses BHAV read-only, through class Behavior, which implements file scoping using resource id ranges (the scopes are: private, global and "semi-global", i.e. shared but not global).

TREE – behavior tree programming data

The layout of this resource is defined by class cTree using a stream. In the visual editor, a BHAV would be a mess without a TREE. A BHAV is expanded for editing using the TREE of the same id. It holds all the comments and node positions that a programmer inputs to the tree editor. TREE resources can be removed without affecting the game.

POSI – old version of TREE

This resource is read in by cTree, but no longer written out. It is automatically removed when saving a TREE resource of the same id.

TPRP – tree properties

The layout of this resource is defined by class TreeProperties. It contains entry information for a BHAV of the same id, and is edited only in edith. For example, the names of the local variables and parameters to a behavior tree are stored here. The reason this is separate from class cTree is for efficiency in drawing call nodes (TREE can be large, TPRP is small). TPRP may be removed without affecting the game.

BCON – behavior constants

Struct BehaviorConstants defines the layout of this resource directly. The structure is simply an array of constants. Behaviors refer to constants for some otherwise literal values so that they may be used in multiple places and easily changed. The resource is saved out by the “Constants Editor” in edith, but can also be created from scratch by class ObjectFolder when reading tuning.txt.

TRCN – Tree constants

The layout of this resource is defined by class cTreeConstants using a stream. Following the pattern of BHAV/TREE, the resource holds the names and other editing data for the constants. TRCN may be removed without affecting the game.

SLOT – object slots

Class SlotLoader , class Slot and subclasses of Slot define the layout of this resource using a stream. An object's definition specifies the id of the SLOT resource to use for the object's slots. The resource contains information for containment slots, routing slots, and sprite slots. The resource is created and modified only by the “-slots” mode of TDSB.

FWAV – object sound event

Class SoundInfo defines the layout of this resource. It is just a c-string. Object behaviors specify sounds to be played by resource id. When executing a PlaySound primitive, the sim loads the FWAV and calls the sound system with the resulting name. FWAV resources are created only by the “-sounds” mode of TDSB.

GLOB – Semi global file

This resource is a character string that determines which semi-global file should be used for all objects in the file. When the objects are initialized, semi-global files are set up and handed off to class Behavior. The resource is optional and only a few object types use it. The first resource of this type that is found in the file is the one that gets used, so the id is not important. GLOB resources are set using “resource setstring … GLOB”.

TTAB – tree table

Class TreeTable defines the layout of this resource using a stream. Most parts of the structure are created and edited only by edith, but certain tuning data is modified by class ObjectFolder if tuning.txt is being used. Also, tuning modifications may be written back out by the “save_tuning” cheat. Essentially the class provides all the information necessary for a sim to begin interacting with an object, including tree entry points and advertisements. An object specifies the resource id of the TTAB to use in it’s definition (objects

in the same file can share). The resource may reside in the object's file or in the semi-global file. Both user driven and autonomous interactions are initiated using this class.

TTAs – tree table strings

Class StringSet defines layout of this resource. It is loaded and saved by class TreeTable, and by Slang. It holds the names of all the defined interactions. The names end up being used by the pie menu and in the action queue.

Various STR# - strings

Class StringSet defines the layout of STR# resources. For objects, these may be viewed and edited directly in edith using the "Behavior strings" editor. Depending on the id, some are used only by the simulation, or only by the editor, or are displayed in the UI.

ID 129-133, animation tables

Used by the animate primitive to invoke a skill by name. The 4 different id's are for the 4 adult/child combinations: a2o, c2o, a2c, c2a. The primitive dialog embeds a string index in the primitive to indicate which name to use. The strings are typically modified by the "-skills" mode of TDSB.

ID 200 – body strings.

For people only, this resource is wrapped by class BodyStrings, and determines the look of a person. It is edited by the game in various places, including the create-a-character screen, and in the skin proxy code. It may also be edited in edith to setup NPCs and other shipped characters.

ID 256 – attribute lables

Used only to show the names of an object's attributes in various places in the behavior editor. May be removed without affecting the game.

ID 257 – slots labels

Output by "-slots" mode of TDSB, indicates a name for each slot of an object, such as "on top" or "route in front of". Used in various places in edith, and may be removed without affecting the game.

ID 258 – relationships labels

Used to display the names of the fields of an object's relationships in various places. May be removed without affecting the game.

ID 259 – type attribute labels

For objects that use global attributes (currently only help system), this resource is used to label them. May be removed without affecting the game.

ID 300 – strings used by primitive ShowString

This primitive is not used anymore.

ID 301 – strings used by dialog primitive

Initially, the strings are added by the primitive dialog in edith when a tree is programmed to show a dialog. The 301 strings are the sole source for object dialog text (except for default text), and are later edited by Slang for writing and translation.

ID 302 – strings used by MakeActionString primitive

Initially, the strings are added by the primitive dialog in edith when programming an interaction that changes it's name. Later edited by Slang for translation.

ID 303 – strings used by CallNamedTree primitive

The primitive allows an object's behaviors to call the behaviors in other object by name. The strings are initially created in the primitive dialog. Used only internally by the simulation.

ID 304 – strings used by ChangeSuit primitive

The primitive allows an object behavior to add an accessory by name to a person during an interaction. The strings are initially created by the primitive dialog.

ID 305 – strings used by UserEvent primitive

The UserEvent primitive uses the resource to display a message or specify a default photo caption. The string is initially entered in the primitive dialog, but is later edited by Slang.

Object graphics

Most of the following resources are generated by TDSB from an OMK and a set of graphics files (BMP and TGA). They are used to render the object in the game. Some BMP_ resources are generated by the game upon request.

DGRP – Draw group

Class DrawGroup defines the layout of this resource using a stream. A draw group specifies the graphical state for a single tile of an object. Object behaviors refer to the states starting at graphic #0. The resource id corresponding to state 0 is stored in the "base graphic" field of the object definition. The definition also specifies the number of graphical states, that is, the number of consecutive DGRP resources expected to be found. The current draw group of an object (id = current graphic state + base graphic) drives the rendering process. DGRP resources are edited only by the "-graphics" mode of TDSB, and by SprMaker.

SPR2 – 8 bit Sprites with Alpha

Struct Sprite2 defines the layout of this resource directly. The resource has a list of sprite graphics, typically consisting of a single item in different zooms and rotations. The resource is handled mainly by class cRenderer, which will draw a sprite into a buffer given an id and index into the list. The resource is written by various TDSB options, such as "-graphics" and "-floors", as well as by SprMaker.

SPR# - 8 bit RLE Sprites with no alpha

This resource is the predecessor of SPR2, but it still used when alpha is not needed. It is also made up of a list of sprites. Struct cRenderer::NewSPRList defines the layout directly. It is also handled mainly by cRenderer, which ignores SPR2 resource that have the same id as a SPR#. Written only by TDSB and SprMaker.

PALT – 8 bit to 24 bit color palette

Class PaletteResource defines the layout directly. Class cRenderer loads and maintains palettes, which are specified by id from the SPR# or SPR2 that uses it. Sprites can share a palette. Written only by TDSB and SprMaker.

BMP_ - Bitmap

The format of these resources is exactly the BMP file format. They are added to object files using the "-bmps" option of TDSB, and by the game, depending on the id.

Catalog Thumbnails

The catalog thumbnail for an object is stored in the BMP_ with id equal to the catalog id of the object. The resource is saved out by the game if necessary and/or specified. For objects that do not have a graphical state (like the destination object), this BMP_ may be saved out by TDSB.

Speech balloon, medium

Same as catalog thumbnail, but with id equal to the catalog id plus 2000. This graphic is put inside a thought balloon when a Sim thinks of the object in medium zoom.

Speech balloon, large

Same as catalog thumbnail, but with id equal to the catalog id plus 4000.

Indexed dialog icons

BMP_ with id >=5000 and <5256 may be referenced by index from the dialog primitive. They are added only by TDSB.

Other dialog icons

The dialog primitive may also reference icons with arbitrary id using a dialog string.

Person-specific bitmaps

The following BMP_ are created in class PersonFinder on demand, and cached by the game in the IFF files for people.

ID 2002, “faces”

Used to create the person’s face button in the family panel.

ID 2003, “rel. images”

Used to create the person’s face button in the relationship panel.

ID 2004, “thumbnail”

Used by the neighborhood screen to show a tiny version of the person’s face.

ID 2005, “speech_med”

This is the person’s face in a thought balloon. Shown in the game’s medium view over the head of a sim when requested.

ID 2006, “speech_large”

Same as 2005, but shown in the large view.

Other

Some types may still be found in objects, but are not used.

CATS – old style catalog strings

This resource has been replaced by CTSS. Translated objects should never have this.

_tmp –a stray test resource.

TMPL – macintosh resource template.

Layout defined by class TemplateParser, this is way old. But it was pretty cool because the Mac had a program that would allow TMPL resources to be edited and then used to parse other resources.