The Scripted Display Tools ("sdt"/"sdt3d)"

Abstract

The Scripted Display Tools are open source software developed by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) group. "sdt" provides a simple 2D visualization capability using standard image files for a background and set of overlaid nodes. "sdt3d" provides a 3D visualization capability using NASA's World Wind 3D interactive world viewer and set of overlaid nodes. Nodes are assigned iconic images for the display sourced from standard format image files (e.g. JPEG, PNG, GIF, etc) or from 3D Model files (3ds). In "sdt" a custom coordinate system can be defined for the background and node positions can be dynamically updated to "move" their associated icons about the background. In "sdt3d" nodes are placed at geographic coordinates that can be dynamically updated to "move" their associated icons about the globe.

Displayed nodes can also be dynamically "linked" and "unlinked" with lines of user-specified color and thickness. This makes the sdt tools well-suited for one of its intended purposes which is to provide a real-time visualization of dynamic, possibly mobile data communication networks. While the sdt feature set will be centered around supporting this function, sdt capabilities will be left open to other creative uses. Here are screenshots of "sdt" and "sdt3d" (1) (2) in action.

"sdt" is written in C++ using the freely-available, cross-platform wxWidgets library for graphical user interface applications. "sdt3d" is written in Java using WorldWinds opensource JDK. Versions of the libraries are available for most Unix, MacOS, and Win32 platforms in a number of different forms.

1. Download

Source code and binary distributions (Linux, Mac OS, and Windows) are available at:

http://downloads.pf.itd.nrl.navy.mil/sdt/

The "sdt3d" source has dependencies on NASA's Worldwind JDK, joglutils (a library that provides Java bindings for OpenGL and 3d Model support), and protolib-jni (a library that provides Java native interface to NRL's protolib protoPipe implementation). "sdt" is dependent on NRL's protolib library and wxWidgets.

2. Usage

2.1. "sdt3d" Usage

The "sdt3d" program may be launched from a command-line, the sdt3d.bat windows batch file, a mac application, or the Linux shell script sdt3d.sh. To launch "sdt3d" from the command line, use the following command-line syntax:

```
java -Xmx512m -Dsun.java2d.noddraw=true -jar sdt3d.jar
```

The sdt3d.jar file used is available in the sdt3d binary distribution or may be built as described in the source code download.

The "sdt3d" application can either accept real-time commands over a command pipe, over a UDP socket, or you may use the "Open File" menu option to load an input file containing sdt commands.

2.1.1. File Menu

`Open File...`	Causes the "sdt3d" application to to parse the selected <scriptFile> containing sdt commands. This command will clear any existing application state with the exception of any loaded sprites.
`Save a Screenshot`	Cause the "sdt3d" application to save a screenshot of the current display to <fileName>

`Listen to port...`
 	
`Off <port>`
	Cause the "sdt3d" application to listen to a udp socket. A popup window will prompt for the (optional) multicast address and port in the format .
	
	`[udp,][<addr>/]<port>`
	
	For example, entering "224.0.0.1/5000" will cause "sdt3d" to join the multicast group 224.0.0.1 on port 5000 whereas "5000" or "udp,5000" will open a udp socket on port 5000.
	
	If the application is currently listening on a socket, the menu item will toggle to "Off <port>" and allow the user to stop listening to the socket.

`Exit`	Exits the "sdt3d" application

2.1.2. View Menu

`Layer Controls>`	Controls to toggle worldwind and "sdt3d" layers on and off.

`Bookmarks>`	Creates a bookmark of the current view. Bookmarks can be subsequently reloaded to return to the saved view-point.
`Status`	Toggles the status field on and off.
`Collapse Multiple Links`	Toggles the display of multiple links between two nodes on or off.

2.2. "sdt" Usage

To run "sdt" with script file "script.sdt":

```
sdt input script.sdt
```

Alternatively the input script can be loaded from the file menu after "sdt" is invoked.

By default, "sdt" monitors stdin for commands. The user may even type commands and manipulate the "sdt" display manually via sdtin, but it is generally expected that another process will control the "sdt" display, even to the point of providing very dynamic updates to produce animations. "sdt" will also accept commands sent over a protopipe named *sdt* by default.

3. Sending real time commands to sdt applications.

To send real time commands to a sdt application use the "sdtcmd" utility:

```
sdtcmd <sdt commands>
```

The "sdtcmd" utility is available in the binary distributions or can be built from within the sdt source distribution.

By default, sdt creates an input pipe that it monitors for sdt commands. The pipe is named *sdt* by default. It is generally expected that another process will control the sdt display, even to the point of providing very dynamic updates to produce animations.

To send any quoted value to an sdt application via "sdtcmd", the string to be quoted must be enclosed in single quotes. Note that nested quotes are not allowed.

For example:

```
# Update the status of an sdt display
sdtcmd status "New Status Text"

# Draw a sphere around node m1-node08
sdtcmd node m1-node08 symbol sphere
```

To send sdt commands to an sdt instance named other than the default, use the instance command:

```
# Send sdt commands to an sdt instance named sdt2
# and move m1-node03 to position X,Y and set the label color to cyan
sdtcmd instance sdt2 node m1-node03 position -77.005610,38.824472,0.000000 label cyan
```

4. sdt Scripts

sdt scripts are text files containing a sequence of commands. Comments may be embedded within sdt scripts by preceding them with a '#'. Script files can include other scripts with the INPUT command. Script files included this way will be processed "in line", meaning that when the INPUT command is encountered within a script, all commands in the included script will be processed before any subsequent commands in the original script. Note that this is in contrast with input commands received over the command pipe. In this case the files will be processed serially.

5. sdt Commands

The sdt command set is made up of key-value pairs separated by white space. Simpler commands (such as commands to control the background, or the WAIT command) are simple key-value pairs. In some cases the value of simple commands may be a comma-delimited list. More complex commands are made up of "objects" (e.g. node, link, sprite) and their "options" (e.g. position, color). "Options" may also have a comma-delimited set of attributes. Options are exclusive to an object type.

The basic grammar is as follows:

```
<key> [<value>]
```

```
<objectType> <objectName> [<option> <attributeList>]
```

An option's attribute list need not be fully qualified, but elements omitted *within* a list should be set to "X" to indicate "use the default value or the last attribute assigned". For example, the default attribute list for a symbol option has the following defaults:

```
# color = red
# outline thickness = 3
# x_radius = radius of sprite
# y_radius = radius of sprite
# opacity = 0.15
#symbol <symbolType>[,<color>,[<thickness>[,<x_radius>[,<y_radius>[,opacity]]]]
```

To draw a blue sphere centered around node and sized to the node's sprite with a 15% opacity, the following command may be used:

```
node node01 symbol sphere,blue
```

To alter the opacity of the sphere to 95% opacity, in a subsequent command:

```
node node01 symbol sphere,X,X,X,X,0.95
```

To increase the size of the sphere to 300 meters, retaining other attributes (e.g. color blue, 95% opacity):

```
node node01 symbol sphere,X,X,300,X,X
```

To turn the symbol red, 300 meters wide, 95% opacity:

```
node node01 symbol sphere,red,X,X,X,X
```

5.1. sdt Objects

The set of sdt "objects" includes the sprite, node, region, link, tile, and popup commands. The name associated with an object can be any string, with the exception of the link command which must be made up of valid node names being "linked" e.g. "link node01:node02". (See below for more detail on the command).

Some sdt "objects" may have other "objects" as options. For example:

```
# Define a sprite type
sprite truck image truck.png size 52,25

# Now define a node and associate it with the sprite "truck"
node node-01 type truck
```

If an object with the associated object name already exists, the previously defined object will be used and any attribute changes will be made to the previously defined object. For example:

```
# Define a node
node node-01 type truck label blue

# Set the position of node-01, all other attributes remain the same
node node-01 position -77.025146,38.822059
```

5.2. Background Commands

These commands control the display of the sdt background:

bgimage <imageFile>	The indicated <imageFile> is used as the background image in the sdt display. A number of standard image formats are supported. Note the use of a background image is optional. (*"sdt" only*)
bgbounds <left>,<upper>,<right>,<lower>	Sets the background boundary coordinates. In "sdt3d" the display will pan to the center of these latitude/longitude coordinates.
bgscale <factor>	The size of the background is proportionally scaled according to the given <factor> value. Any positive floating point value may be used. (*"sdt" only*)
bgsize <width>,<height>	The background image is scaled to the size specified by the <width> and <height> parameters. If one of the parameters is less than zero, the image's aspect ratio is preserved and the image is scaled to match its corresponding dimension to the non-zero parameter given. (*"sdt" only)*

5.3. Sprite Commands

These commands are used to define a set of sprites (icons) and their characteristics:

sprite <spriteName>	This creates a new sprite instance of name <spriteName> or addresses a previously created sprite instance for application of other Sprite Commands (i.e. image, scale, size). The default "size" of a new sprite is 32x32 pixels.
image <imageFile>	This assigns the <imageFile> given to be displayed for nodes of the given sprite type. (e.g. "sprite Car image car.png"). When an image for the sprite is specified, the image is scaled such that the smallest dimension (width or height) equals the minimum dimension of the sprite (32 by default). 3D Models can also be associated with sprite images. sdt will first attempt to open the imageFile as specified e.g. as fully qualified, relative to the current directory, or in the current directory. If not found, it will search for the file in any path(s) previously specified with the sdt PATH command. Finally, it will look for the file in the same directory as the current input script (if any. "sdt3d" only)
scale <factor>	The size of the sprite and its associated image (if given) is proportionally scaled according to the given <factor> value. Any positive floating point value may be used.
size <width>,<height>	This assigns width and height sizes (meters in "sdt3d", pixel units in "sdt") to the given sprite. If an image is already specified, the image is scaled to directly match the given dimensions. If one of the parameters is less than zero, the image's aspect ratio is preserved and the image is scaled to match its corresponding dimension to the non-zero parameter given.

light [on|off]	Controls whether "sdt3d" should apply lighting to a 3D sprite. The default is off. Check the 3D model's attributes to determine the correct setting.

5.4. Node Commands

These commands are used to instantiate nodes, assign an image to them, and set their position and other characteristics:

node <nodeName>	This creates a new node instance of name <nodeName> or addresses a previously created node instance for application of other Node Commands (i.e. type, position, label). Multiple Node Commands may be given following the "node <nodeName> specification. By default, the first sprite in the sdt sprite list is assigned to the node.
type <spriteName>	This assigns the sprite of the indicated <spriteName> as the image used to represent the specified node on the sdt display (e.g. "node Alpha type Car". A special reserved <spriteName> of "none" is used to indicate the specified node should not be displayed. **By default, the first sprite in the sdt sprite list is assigned to a node. Specify type "none" to disable the assignment of a default sprite type.**

position <x>,<y>[,<z>],[{msl|agl}]]	This command is used to specify the current position (and altitude if specified) of the given node. The coordinates are in units of the coordinate system defined by the bgbounds commands. The <x> attribute assigns longitude, the <y> attribute assigns latitude, and the optional <z> attribute sets altitude.
	
	If no altitude is specified, the node will be positioned at terrain elevation in "sdt3d". Altitude is not useful in "sdt" and will be ignored.
	
	Any altitude specified will position the node at AGL by default (e.g. distance above ground level) or at the default altitude assigned by the defaultAltitudeType command. The "agl" or "msl" attributes will override the default, and position the node at the given elevation above ground level or at mean sea level respectively. To retain the last assigned elevation set the altitude to "X", e.g.
	
	`# set altitude to 300 meters agl (default)`
	`node node1 position -77.005217,`
	`38.819009,300.000000`
	
	`# change the location, retain the`
	`# previously specified altitude`
	`node node1 position -77.005217,38.820009,x`
	
	`# position the node at agl`
	`node node1 position x,x,x,agl`
	
	`# Position node2 at terrain elevation (default)`
	`node node2 position -77.005217,38.820009`
	
	`or`
	
	`node node2 position -77.005217,38.820009,0`

	Note that any nodes positioned below mean sea level will be positioned at seal level for this release of "sdt3d".

label on|<color>|off,[,<text>]	This indicates whether or not a text label should be displayed for the indicated node. By default, a cyan label using the node's name is displayed below the node's sprite image.
	
	To hide the label for a node use the command
	
	`node <nodeName> label off`
	
	The label can be enabled with either the "on" command or by specifying a color, e.g.
	
	`node <nodeName> label on,<text>`
	
	`node <nodeName> label blue,<text>`
	
	Label text that contains spaces or formatting characters must be enclosed in quotes e.g.
	
	`node <nodeName> label blue,"Label text"`

symbol <symbolType>[,<color>,[<thickness>[,<x_radius>[,<y_radius>[,opacity]]]]	This is used to put a symbol around the node. Valid "sdt" symbol types are: circle, ellipse, square, rectangle, rndrectangle (rounded rectangle), rndsquare (rounded square), and none. Valid "sdt3d" symbol types are sphere and cube. By default there is no symbol associated with a node. The symbol will be sized to fit the sprite size by default. (If no sprite is associated with the node, a symbol may still be associated that will default to the default sprite size of 32x32). The default symbol color is "red". Colors are specified by name, and a large color set is supported. See the wxWidgets documentation for the "sdt" color set until this user's guide is further revised. Valid "sdt3d" colors are white, yellow, green, blue, cyan, red, pink, orange, magenta, purple, and gray. The radius parameter is only available for "sdt" circles and "sdt3d" symbols. They allow the symbol to represent a real circle of given radius in the coordinate system. This could be useful for showing radio range, for instance. This should be given in the same units as those of the coordinate system defined by the bgbounds commands or in meters in the "sdt3d" app. As such, the size of the circle will increase as you zoom in, and decrease as you zoom out. The opacity parameter is only available in "sdt3d" and will set the opacity of the interior of the symbol. By default the symbol is drawn at opacity .15 If any parameter within the parameter list is omitted, specify a "X" value to use the defaults. See the sdt Commands section for more information on attribute lists. For example to draw a sphere based on the size of the sprite at a .50 opacity:

	`symbol sphere,blue,X,X,X,0.50` In "sdt" the circle is drawn as an ellipse so that if you change the aspect ratio with scaling or Ctrl-A, it will flatten out, though it still represents a perfect circle in your coordinate system. You are also allowed to specify x and y radii, in case your coordinate system does not use the same unit on both axes (lat/lon, for instance). If only one radius is given, then the x and y radii will be the same, and the symbol will appear circular until you change one of the scales independently from the other or use Ctrl-A (zooming scales both x and y axes together, so that is not a problem).
delete node,<nodeName>	This command deletes the specified node and any links associated with it. *Note that the legacy delete <nodeName> command is still supported.*

5.5. Region Commands

These commands ("sdt3d" only) are used to instantiate regions, and set their position and other characteristics:

region <regionName>	This creates a new region of name <regionName> or addresses a previously created region for application of other region Commands (i.e. shape, position).
shape <regionShape>[,<color>[,<thickness>[,<x_radius>[,<y_radius>[,<opacity>]]]]	This assigns the indicated <regionShape> to the region (e.g "region region1 shape circle"). Valid regionShapes are circle,sphere,square,cube, rectangle, box, and none. Circle, square, and rectangle regions are "surface shapes" meaning they will overlay the terrain surface, whereas spheres, cubes, and boxes are 3gl objects that will be centered at the position set with the center attribute. The default region color is "grey", the default outline thickness is one, the default size is 300 x 300 meters, and the default opacity is 15%. Colors are specified by name, The color set includes white, yellow, green, blue, cyan, red, pink, orange, magenta, purple, and gray. The x_radius argument can be used to set the width of the region in meters. The y_radius argument sets the height in meters.
center <x>,<y>[,<alt>]	This command is used to specify the current position of the region. The coordinates are in units of the coordinate system defined by the bgbounds commands. Altitude is only relevant for spheres and cubes.
delete region,<regionName>	This command removes the specified region.

5.6. Link Commands

These commands are used to define (and undefine) "links" (drawn as lines) between pairs of previously defined nodes. Note that the deprecated link syntax is still supported in the current versions of "sdt"/"sdt3d".

link <node1>,<node2>[,<linkID|all>[,<dir,all>]]]	This indicates that a "link" (drawn line) should be managed and displayed for the given pair of nodes.

The optional linkID can be used to create multiple links between the node pairs, e.g.

```
link node-01,node-02,eth1

link node-01,node-02,wifi
```

The dir attribute will create a "uni-directional" link between the two nodes with an arrow anchored on the target node. For example:

```
# arrow anchored on node-02
link node-01,node-02,wifi,dir
```

A new bi-directional link will delete any existing uni-directional links between the two nodes that have the same link id. A new uni-directional link deletes an existing bi-directional link between the same nodes. If such a link was present, then the new command will create an additional uni-directional link in the opposite direction if so indicated. For example:

```
# Creates a uni-directional link between
# nodes 1 and 2 with an arrow anchored on
# node 2
link 1,2,wifi,dir line blue,3 linklabel on

# Creates a second uni-directional link
# between nodes 1 and 2 with an arrow
# anchored on node 1
link 2,1,wifi,dir line yellow,3 linklabel on

# Deletes existing uni-directional links
# and creates a single bi-directional
# (arrowless) link
link 2,1,wifi line red,3 linklabel on

# Deletes existing bi-directional link and
# creates a uni-directional link with an
# arrow anchored on node 2
link 1,2,wifi,dir line red,3, linklabel on
```

The optional linkID "all" keyword can be used to refer to all linkIDs between the two nodes. By default only bidirectional links will be referenced, the default when the directional keyword "dir" is omitted. To reference the set of all directed links specify the "dir" directional keyword, or the "all" keyword to reference all directed and bidirectional links. For example:

```
# reference the set of all bi-directional
# links regardless of linkId
link 1,2,all

# reference the set of all uni-directional
# links
link 1,2,all,dir

# reference the set of all bi and
# uni-directional links
link 1,2,all,all

# reference the set of all links associated
# with linkid wifi
link 1,2,wifi,all
```

	Note that linkids may not be valid sdt colors due to legacy link command support.
line color[,<thickness>]]	The color and thickness (1-8) of the line drawn can be optionally specified. The default color is "red" and the default thickness is 1. Colors are specified by name, and a large color set is supported. See the wxWidgets documentation for the "sdt" color set. Valid "sdt3d" colors are white, yellow, green, blue, cyan, red, pink, orange, magenta, purple, and gray.

linklabel on|<color>|off,[,<text>]	This indicates whether or not a text label should be displayed for the indicated link. When turned on with no associated text, a label the color of the link and containing the link's name is created and will be displayed when the link label layer is turned on (the link label layer is not initially displayed by default). If no linklabel command is specified for the link, no linklabel is created. Note that the link line color must be defined before the linklabel in order to default the label color to the link line color.
	
	To hide the label for a link use the command
	
	`link <linkName> linklabel off`
	
	Alternatively, the linklabel layer can be toggled on and off with the link labels radio button.
	
	The linklabel can be enabled with either the "on" command or by specifying a color, e.g.
	
	`link <linkName> linklabel on`
	`link <linkName> linklabel on,<text>`
	`link <linkName> linklabel blue`
	`link <linkName> linklabel blue,<text>`

delete link,<node1>,<node2>[,<linkID>]	This command causes a "link" previously specified to be no longer displayed for the given node pair. If the node pair is later again linked, any non-default color or thickness attributes will need to be re-specified. To delete all links between a node pair set the linkId to "all": `delete link,node01,node02,all` *Note that the legacy unlink command is still supported unlink <node1>,<node2>[,<linkID>]*

5.6.1. Link Commands - Deprecated

These commands are used to define (and undefine) "links" (drawn as lines) between pairs of previously defined nodes.

link <node1>,<node2>[,<color>[,<thickness>]]	This indicates that a "link" (drawn line) should be managed and displayed for the given pair of nodes. The <color> and <thickness> (1-8) of the line drawn can be optionally specified using the indicated format. The default color is "red" and the default thickness is 1. Colors are specified by name, and a large color set is supported.

	See the wxWidgets documentation for the color set until this user's guide is further revised.
unlink <node1>,<node2>	This command causes a "link" previously specified to be no longer displayed for the given node pair. If the node pair is later again linked, any non-default color or thickness attributes will need to be re-specified.

5.7. TILE Commands ("sdt3d" only)

The tile command causes the specified image to be overlaid at the specified lat/lon coordinates.

tile <tileName>	Specifies the name of the tile object.
tileImage <imageFile>	The image file to be overlaid on the terrain surface. "sdt3d" will first attempt to open the imageFile as specified e.g. as fully qualified, relative to the current directory, or in the current directory. If not found, it will search for the file in any path(s) previously specified with the sdt PATH command. Finally, it will look for the file in the same directory as the current input script (if any) ("sdt3d" only).
sector <left>,<upper>,<right>,<lower>	The surface coordinates for the image file.
delete tile,<tileName>	Removes the specified tile.

5.8. Popup Commands ("sdt" only)

These commands are used to create "popup" windows which contain specified text content. The implementation of these commands is slightly different in "sdt" and "sdt3d".

5.8.1. "sdt" popup commands

These commands are used to create, update, and destroy "popup" windows which contain specified text content. An example use of these windows is to provide some display of information upon <doubleclick> of a displayed node (Note this requires monitoring the stdout of "sdt" to learn of <doubleclick> events).

popup <windowName>	This specifies a popup window titled with the given <windowName>. The window is not displayed until its "content" is specified.
content <"contentText">	This command specifies the content of the popup window. The content of window can be changed at any time. The content text should be enclosed in quotes.
resize	The resize command can be given with or without the "content" command, and results in the window being automatically resized to fit the current text.
popdown <windowName>	This command destroys the specified popup window

The following example illustrates the use of the "sdt" "popup" commands used in an input script:

```
popup info content "The current time is 10:30AM "
wait 500
popup info content "The time is now 10:30AM plus 500 msec"
wait 5000
popdown info
```

5.8.2. "sdt3d" popup command

popup <"contentText">	This creates a popup window containing the associated text. The window will popdown when the user clicks on the windows ok button.

5.9. Miscellaneous Commands

There are some additional commands provided for sdt operation.

clear {all | nodes | sprites | symbols | links | labels | regions | tiles }	This command deletes the specified object type. *clear all* will delete all sdt elements with the exception of the sprite table (Use *delete all,all* to also delete the sprite table). *clear nodes* will delete all nodes and their associated sprites, symbols, links, and labels. See the delete command to delete individual objects.

defaultAltitudeType { msl | agl }	This command will change the default altitude type (relative to the terrain or absolute) for location altitude settings. AGL is the system wide default altitude.
	`# Set the default altitude for all location`
	`# assignments to above ground level (relative`
	`# altitude)`
	`defaultAltitudeType agl`
	
	`# Set the default altitude for all location`
	`# assignments to mean sea level (absolute`
	`# altitude)`
	`defaultAltitudeType msl`

delete <objectType|all>,<objectName|all>	Deletes the object type of the specified name. Object types that can be deleted include nodes, regions, links, and tiles. Sprites and symbols cannot be deleted by name. The all key word can be used to delete all objects of the given type. As opposed to the clear all command, "delete all,all" will delete the sprite table.
	`# Delete node "node01"`
	`delete node,node01`
	`delete node,"a node name"`
	
	`# Delete region region01`
	`delete region,region01`
	
	`# Delete tile "antenna range overlay"`
	`delete tile,"antenna range overlay"`
	
	`# Delete link node01:node02:eth0`
	`delete link,node01:node02:eth0`
	
	`# Delete all sprites from the sprite table`
	`delete sprite,all`
	
	`# Delete all object types`
	`delete all,all`

flyto <lon>,<lat>,<alt>	This will "fly" to the specified coordinate and center the view around it.
follow <on|off>	This command toggles follow node behavior on and off. It does not disable the settings for individual nodes. Use

	the "follow all,off" command to disable following on all nodes regardless of this global setting.

follow [node,]<nodeName|all>[,<on,off>]	This command will cause the view to be centered around the specified node or all nodes if the *all* keyword is specified. For example:
	
	`# center the view around node01`
	`follow node01`
	
	`# the view will encompass all nodes`
	`follow all`
	
	`# stop following all nodes`
	`follow all,off`
	
	`# stop following node01`
	`follow node01,off`

instance <instanceName>	This command will change the application's command pipe name from the default of *sdt* to <instanceName>.

input <fileName>	When specified on the command line or received over the command pipe, this command will load the specified <inputFile> and process the file in its entirety before loading any other input files. For example, all commands in the "spriteDefs" file will be loaded before the "script1" file is loaded.
	
	`sdtcmd input spriteDefs input script1`
	
	Note that *input* commands embedded *within* a script file will cause sdt to process the file in its entirety at the point the input command is encountered. For example:
	
	`<spriteDefs>`
	`def 1.1`
	`def 1.2`
	`def 1.3`
	`</spriteDefs>`
	
	`<script1>`
	
	`cmd 1.1`
	`cmd 1.2`
	
	`<script2>`
	
	`cmd 2.1`
	`cmd 2.2`
	
	`</script2>`
	
	`cmd 1.3`
	`cmd 1.4`
	
	`</script1>`
	
	Note that when input files are loaded from the file menu, all previous sdt state will be cleared while input files loaded over the command pipe or via a command within an input script will not clear sdt state.

listen [udp,][off|<addr>/]<port>	Cause the "sdt3d" (only) application to listen to a udp socket on the specified port. Optionally a multicast address may be specified.

	Entering "224.0.0.1/5000" will cause "sdt3d" to join the multicast group 224.0.0.1 on port 5000 whereas "5000" or "udp,5000" will open a udp socket on port 5000.
	
	If listening on a socket is enabled, the listen menu item will toggle to "Off <port>" and allow the user to stop listening to the socket. Alternatively the "listen off" command may be used.
	
	`# join multicast group 224.0.0.1/5000`
	`listen 224.0.0.1/5000`
	`# open a udp socket on port 5000`
	`listen udp,5000`
	`# open a udp socket on port 5000`
	`listen 5000`
	`# stop listening`
	`listen off`

log <logFile>	Turns on debug logging to <logFile>. Entering "log off" turns off file logging so that it reverts back to stderr. `log errlog.txt`
path "<directoryPath>;<directoryPath>:"	This command sets a directory path to be searched for sdt sprite images, tiles, input files, etc. Each path should be delimited by ";" in a Windows environment. Either ";" or ":" may be used in a *nix environment. The entire path must be included in quotes (") if spaces exist in the path e.g. `path "/Documents and Settings;/Documents and Settings/A dire`
off	This command will turn off listening on any open udp sockets.
status "<theStatus>"	This command sets the status content to the text enclosed in quotes. Usage: `status "A status update"`
title "<theTitle>"	This command sets the title of the main sdt window to the text enclosed in quotes. Usage: `title "A main window title"`

wait <msec>	When this command is encountered in the input, sdt will pause for the number of milliseconds indicated by <msec> before processing other commands in the input. This allows self-contained animations to be created using the sdt script format. Cheesy sdt animation script example:
	
	`bgimage roadmap.jpg`
	`bgbounds 0,0,100,100`
	`sprite Car image car.png`
	`node alpha type Car position 10,10`
	`wait 200`
	`node Car position 10,20`
	`wait 200`
	`node Car position 10,30`

6. Examples

Sample scripts and icons are available in the /sdt/examples subdirectory.

6.1. "sdt3d" Example Script

This is a simple script to illustrate the use of "sdt3d":

```
path "/Documents and Settings/nrl/My Documents/sdt3d/examples/:/cygwin/home/nrl/sdt3d/examples/:"

bgbounds -77.028633,38.828533,-77.003298,38.817720
sprite helo image helo.png size 105,43
sprite truck image truck.png size 50,25
sprite uav image uav.png size 72,20
sprite warrior image warrior/warrior.3ds size 32,32 length 32
#
region region01 position -77.025146,38.822059 symbol circle,blue,300

node m1-xcom type helo label cyan symbol sphere
node m1-node01 type truck
node m1-node02 type uav label cyan
node m1-node03 type warrior label cyan

status "GMT>15:53:57"
node m1-xcom position -77.005342,38.818870,900.000000 label cyan
node m1-node01 position -77.005620,38.825368,0.000000
node m1-node02 position -77.009610,38.828472,400.000000 label cyan
node m1-node03 position -77.019179,38.824029,0.000000 label cyan

wait 50.0

status "GMT>15:54:37"
node m1-xcom position -77.005342,38.818870,900.000000 label cyan
node m1-node01 position -77.005620,38.825368,0.000000
node m1-node02 position -77.009610,38.838472,400.000000 label cyan
node m1-node03 position -77.019179,38.834029,0.000000 label cyan
```

6.2. "sdt" Example Script

This is a simple script to illustrate the use of "sdt":

```
bgimage roadmap.jpg
bgbounds 0,0,100,100
sprite Car image car.png
node alpha type Car pos 10,10
node beta type Car pos 10,20
node gamma type Car pos 10,30
link alpha,beta
link beta,gamma
wait 500
node alpha pos 20,10
node beta pos 20,20
wait 500
node alpha pos 30,10
node beta pos 30,20
wait 500
unlink beta,gamma
```

Assuming the script is stored in a file named "script.sdt", "sdt" can be launched with either:

```
cat script.sdt | sdt
```

or

```
sdt input script.sdt
```

to execute the given set of commands. When input commands are piped into the "sdt" stdin input, the sdt wait command will generally not need to be used if the program providing the input to sdt is providing the commands on a realtime basis. This makes "sdt" appropriate for realtime visualization given an appropriate controlling program or shell script.

7. "sdt" Only Functionality

This section is relevant to "sdt" only.

7.1. Popups

When a popup window is closed by the user, the following message is printed to stdout:

```
popdown <windowName>
```

If a live program is controlling "sdt"'s input (as is often the case for popups), it should pay attention to this message, and stop sending any more "popup" commands for that window (e.g. if the window is being updated live). If this is not done properly, then a subsequent "popup" command will recreate the window that the user just closed. If a window is being updated live by a timer, etc., it's a good idea to go ahead and re-send "sdt" the "popdown" command to make sure the window stays closed in case another popup message has been issued after the user closed the popup window.

7.2. Mouse Operations

You may do a variety of mouse operations in order to output messages that may be fed into another program. First, double-clicking the left mouse button on any node will print a simple message to stdout. The message format is as follows:

```
node <node_name> doubleclick
```

Additionally, holding down the shift key while left-clicking a node or position within "sdt" will print another message to stdout, with the coordinate system position of mouse cursor, and an optional node name and position (if a node was clicked). The message format for this is as follows:

```
shiftclick position <x>,<y> [node <node_name> nodeposition <x>,<y>]
```

7.3. Background Resizing Options

Under the Options folder of the Menu bar, you can select one of two options for resizing your background image. The Auto-Size selection maximizes the background to the window while keeping the original dimensions. The Fill Window selection will stretch the background image to fully match the size of the window. You can also resize the background anytime by using the hot key commands:

CTRL-F for Fill Window (was CTRL-A in 1.0aX, but had to be changed for MacOS)

CTRL-S for Auto-Size (preserves image aspect ratio)

You may also resize the background by zooming in and out. This is done by holding the left or right mouse button down and dragging up and down.

7.4. Miscellaneous Options/Shortcuts

Holding down CTRL and clicking the mouse on the "sdt" window will center the image. CTRL-P will save the current contents of the "sdt" window in PNG format to a file called "sdt-(sequence number).png" in the directory "sdt" was started from. The sequence number in the file name is padded with zeros to 4 digits, and is reset every time "sdt" is restarted, so be careful, or you will overwrite your captured files! This feature is also available under "Capture Screen" in the Options menu.