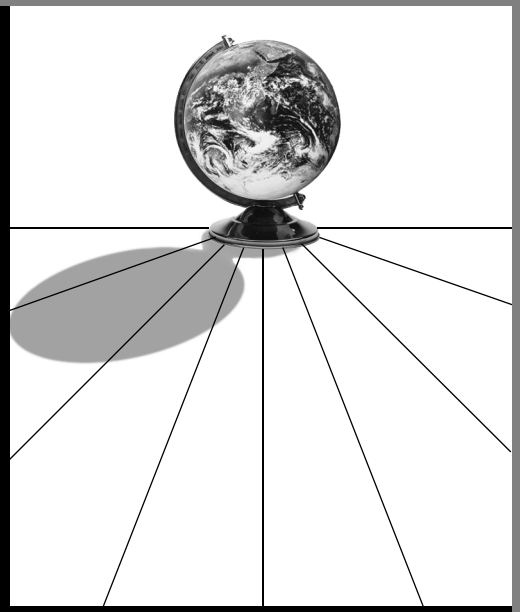

QUICK REFERENCE GUIDE

Alphabetical summary

of all WorldToolKit functions,

macros, and constants

WorldToolKit® Release 9

Current version: **April 1999**

Engineering Animation, Inc.
SENSE8 Product Line
100 Shoreline Highway, Suite 282
Mill Valley, CA 94941 USA

Telephone: (415) 339-3200
Facsimile: (415) 339-3201
Web site: www.sense8.com

This book was printed in the United States of America.

INTRODUCTION

This Quick Reference Guide summarizes the functions, macros, and constants described in the WorldToolKit Reference Manual (Release 9). They are arranged here alphabetically, with a brief description of their actions.

Each has a corresponding page number listed, so you can read about it in more detail in the WorldToolKit Reference Manual.

Function Name	Returns	Parameters	Page	Description
WTanchornode_getlocation	char *	WTnode *node	4-63	Returns the anchor string (URL) of the specified anchor node
WTanchornode_new	WTnode *	WTnode *parent	4-40	Creates an anchor node and adds it to the scene graph after the last child of the specified parent
WTanchornode_setlocation	FLAG	WTnode *node char *url	4-63	Replaces the anchor string (URL) of an anchor node with the new string
WTbaron_update	void	WTsensor *sensor	13-76	Updates a Logitech 3D Mouse's location
WTbase_addparent	void	WTbase *object WTbase *parent	3-8	Adds the specified parent WTbase object as a new parent of object
WTbase_delete	void	void *object	3-11	Deletes an object
WTbase_deleteproperties	FLAG	void *object	3-13	Deletes all user-defined properties from the specified object
WTbase_find	void*	int objtype const char *name	3-13	Finds an object of the specified objtype by name
WTbase_findchild	WTbase*	WTbase *object const char *name	3-10	Returns a pointer to the WTbase object which is a first generation child of the specified WTbase object and whose name matches the name parameter
WTbase_getchild	WTbase*	WTbase *object int childnum	3-10	Returns a pointer to the childnum'th child of the specified WTbase object
WTbase_getdata	void*	void *object	3-11	Returns the user-defined data field from an object
WTbase_getname	char*	void *object	3-12	Returns the name of an obejct
WTbase_parent	WTbase*	WTbase *object int parentnum	3-9	Returns a pointer to the parentnum'th parent of a WTbase object
WTbase_getproperty	char*	void *objecty int propnum	3-12	Returns the propnum'th property of the specified object
WTbase_gettype	int	void *object	3-11	Returns the type of the object passed in (WTBASE, WTNODE, WTWINDOW, WTVIEWPOINT, WTSENSOR, or WTPATH)
WTbase_ischild	FLAG	WTbase *parent WTbase *child	3-10	Returns TRUE if child is a first generation child of parent

Function Name	Returns	Parameters	Page	Description
WTbase_new	WTbase*	WTbase *parent	3-8	Creates a new WTbase object as a child of parent and returns a pointer to it
WTbase_next	WTbase*	WTbase *object	3-8	Returns the next WTbase object in the universe's list of WTbase objects
WTbase_nfind	void*	int objtype const char *name int ntocmp	3-13	Finds an object of the specified objtype whose name's first ntocmp characters matches the first ntocmp characters of name
WTbase_nfindproperty	char*	void *object const char *propname int ntocmp	3-13	Returns the full property name of the first property of an object whose first ntocmp characters of the property name match the first ntocmp characters of propname
WTbase_numchildren	int	WTbase *object	3-9	Returns the number of children of the specified WTbase object
WTbase_numparents	int	WTbase *object	3-9	Returns the number of parents of the specified WTbase object
WTbase_numproperties	int	void *object	3-12	Returns the number of properties associated with an object
WTbase_print	void	void *object	3-11	Prints information about an object
WTbase_removeallhandlers	FLAG	void *object	3-26	Removes all handlers for the specified object's propname property
WTbase_removeparent	void	WTbase *object WTbase *parent	3-9	Removes the specified parent WTbase object as a parent of object so that the object WTbase object is no longer a child of parent
WTbase_setdata	void	void *object void *data	3-11	Sets the user-defined data field in an object
WTbase_setname	void	void *object const char *name	3-12	Sets the name of an object
WTbase_unshare	FLAG	void *object	21-11	Unshares all of an object's properties
WTBIRD_AFT **WTBIRD_FORWARD** **WTBIRD_LEFT** **WTBIRD_LOWER** **WTBIRD_RIGHT** **WTBIRD_UPPER**	n/a	n/a	13-43	Constants used with the function WTbird_sethemisphere to tell the Ascension Bird sensor in which hemisphere the receiver will be navigating

Function Name	Returns	Parameters	Page	Description
WTBIRD_LEFTBUTTON **WTBIRD_MIDDLEBUTTON** **WTBIRD_RIGHTBUTTON**	n/a	n/a	13-43	Constants used to retrieve event data from an Ascension Bird device
WTbird_autohemishere	void	WTsensor *sensor	13-44	Automatically sets the hemisphere for the bird receiver based on its current location
WTbird_getabsoluterecord	int	WTsensor *sensor WTp3 p WTq q	13-41	Obtains the most recent absolute position and orientation record for the specified bird sensor in relation to the bird receiver
WTbird_gethemisphere	int	WTsensor *sensor	13-43	Returns the hemisphere currently set for the specified Bird sensor
WTbird_sethemisphere	void	WTsensor *sensor int hemisphere	13-43	Tells a Bird sensor in which hemisphere the Bird's receiver will be navigating
WTbird_setsync	void	WTsensor *bird short synctype	13-42	Sets a Bird to synchronize its scanning frequency with that of a CRT
WTbird_update	void	WTsensor *sensor	13-42	Updates a Bird’s location
WTBOOM_DOWN **WTBOOM_LEFT** **WTBOOM_LEFTBUTTON** **WTBOOM_RESET** **WTBOOM_RIGHT** **WTBOOM_RIGHTBUTTON** **WTBOOM_UP**	n/a	n/a	13-57	Set of constants used with the function WTsensor_getmiscdata to determine the state of a BOOM joystick
WTboom_update	void	WTsensor *sensor	13-57	Updates a BOOM device
WTcalloc	void	size_t num size_t size	24-9	Similar to calloc; used to allocate shared memory *Note: Required for SGI MP/MP*
WTconnection_addcallback	void	WTconnection *c WTconncb cb	21-31	Adds a callback to a connection
WTconnection_connect	FLAG	WTconnection *c	21-28	Connects the World2World Server Manager and port represented by the specified connection object
WTconnection_delete	void	WTconnection *c	21-27	Disconnects and deletes a connection
WTconnection_deleteallenumtrees	void	WTconnection *connection	21-36	Deletes all of the enumeration trees that are currently stored with the specified WTconnection object

Function Name	Returns	Parameters	Page	Description
WTconnection_deleteenumtreebyid	void	WTconnection *connection unsigned int enumid	21-36	Deletes the specified enumeration tree from a WTconnection object
WTconnection_disconnect	FLAG	WTconnection *c	21-28	Disconnects from the World2World Server Manager and port represented by the specified connection object
WTconnection_getcallback	WTconncb	WTconnection *c int conncbnum	21-32	Returns a numbered callback on a connection
WTconnection_getclockdiff	double	WTconnection *c	21-30	Returns the time, in seconds, that the local and World2World Simulation Server clocks differ
WTconnection_getdata	void*	WTconnection *c	21-27	Returns the user-defined data field on a connection
WTconnection_getenumtree	WTbase *	WTconnection *connection unsigned int nenumtree	21-37	Returns the root WTbase object for an enumeration tree of the specified connection by index
WTconnection_getenumtreebyid	WTbase *	WTconnection *connection unsigned int enumid	21-37	Returns the root WTbase object of the enumeration tree whose enumeration tree id is enumid
WTconnection_getenumtreeid	unsigned int	WTconnection *connection unsigned int nenumtree	21-38	Returns the enumeration tree id for an enumeration tree of the specified connection by index
WTconnection_getlatency	double	WTconnection *c	21-30	Returns the latency associated with a connection
WTconnection_getmyid	unsigned int	WTconnection *c	21-28	Returns the local client's id for the specified connection
WTconnection_getmyname	const char*	WTconnection *c	21-29	Returns the local client's name for the specified connection
WTconnection_getroot	WTsharegroup*	WTconnection *c	21-32	Returns the root sharegroup for a connection
WTconnection_getstatus	int	WTconnection *c	21-29	Returns the current status of a connection
WTconnection_getupdaterate	unsigned short	WTconnection *c	21-31	Returns the number of times per second that data packets are sent for a connection
WTconnection_getuserid	unsigned int	WTconnection *c unsigned int usernum	21-33	Returns a numbered user's id
WTconnection_getuseridbyname	unsigned int	WTconnection *c const char*username	21-33	Returns a user's id
WTconnection_getusername	const char*	WTconnection *c unsigned int usernum	21-33	Returns a numbered user's name
WTconnection_getusernamebyid	const char*	WTconnection *c unsigned int userid	21-33	Returns a user's name

Function Name	Returns	Parameters	Page	Description
WTconnection_issynchronous	FLAG	WTconnection *c	21-31	Returns TRUE if the operating mode of a connection is synchronous, returns FALSE otherwise
WTconnection_new	WTconnection*	const char *host unsigned short port const char *usrname const char *passwrd	21-26	Defines a new connection to the World2World Server Manager at a specified port
WTconnection_next	WTconnection*	WTconnection *c	21-28	Returns the next connection in the local client's list of connections
WTconnection_numcallbacks	int	WTconnection *c	21-32	Returns the number of callbacks on a connection
WTconnection_numenumtrees	unsigned int	WTconnection *connection	21-37	Returns the number of enumeration trees currently stored with a WTconnection object
WTconnection_numusers	unsigned int	WTconnection *c	21-32	Returns the number of users on a connection
WTconnection_print	void	WTconnection *c	21-29	Prints information about a connection
WTconnection_removecallback	void	WTconnection *c WTconncb cb	21-32	Removes a callback from a connection
WTconnection_setdata	void	WTconnection *c void *data	21-27	Sets the user-defined data field on a connection
WTconnection_setsynchronous	void	WTconnection *c FLAG synchronous	21-30	Sets the operating mode of a connection to synchronous if the synchronous parameter is TRUE. If FALSE, the operating mode will be asynchronous
WTconnection_setupdaterate	void	WTconnection *c unsigned short fps	21-31	Sets the rate at which data packets are sent over the connection
WTconnection_synch	FLAG	WTconnection *c	21-30	Waits for pending requests to be fulfilled
WTconnection_update	void	WTconnection *c	21-29	Updates a connection (send and receive packets)
WTCONSTRAIN_X **WTCONSTRAIN_Y** **WTCONSTRAIN_Z** **WTCONSTRAIN_XROT** **WTCONSTRAIN_YROT** **WTCONSTRAIN_ZROT**	n/a	n/a	14-20	Constants used with the function WTpath_setconstraints to constrain translation along, or rotation about, the specified axis
WTcrystaleyesVR_update	void	WTsensor *sensor	13-110	Updates the CrystalEyes VR device

Function Name	Returns	Parameters	Page	Description
WTcybermaxx2_rawupdate	void	WTsensor *sensor	13-121	Reads the tracker input and puts it in the raw data structure as an absolute euler rotation
WTcybermaxx2_update	void	WTsensor *sensor	13-120	Updates the Cybermaxx2 device's location
WTcybglove_deletehandmodel	void	WTcybglove *glove	13-129	Deletes the hand model associated with the Cyberglove
WTcybglove_getanglearray	float *	WTcybglove *glove	13-132	Returns a 6x4 array of bend angles in radians
WTcybglove_getfingers	WTnode **	WTcybglove *glove	13-131	Returns a 5x3 array of WTnode pointers representing the fingers of the Cyberglove
WTcybglove_getforearm	WTnode *	WTcybglove *glove	13-130	Returns a pointer to the node representing the forearm of the Cyberglove
Wtcybglove_getpalm	WTnode *	WTcybglove *glove		Returns a pointer to the node representing the palm of the Cyberglove
WTcybglove_new	WTcybglove *	int baud char *device char *calibrationfilename	13-124	Creates a Cyberglove sensor object
WTcybglove_setvisibility	void	WTcybglove *glove FLAG visible	13-130	Sets the visibility of all of the objects in the hand model associated with the Cyberglove
WTcybglove_showcalibrationpanel	void	FLAG on	13-126	Controls whether the calibration panel of the Cyberglove is displayed or not
WTcybglove_usehandmodel	WTnode *	Wtcybglove *glove char *handmodelname float scale WTnode *parent	13-127	Builds a hand model from a multi-object NFF file
WTdir_2q	void	WTp3 dir WTq q	25-30	Converts a unit vector to a quaternion. In generating q, it's assumed there is no twist about the direction vector
WTdirandtwist_2q	void	WTp3 dir float twist WTq q	25-31	Converts a vector dir to a quaternion having a twist as specified in the argument "twist"
WTDIRECTION_BACKWARD **WTDIRECTION_FORWARD**	n/a	n/a	14-20	Constants used with the WTpath_setdirection function to set a path's play direction backward or forward
WTdirectory_close	void	WTdirectory *dir	24-5	Closes an open directory
WTdirectory_getentry	char *	WTdirectory *dir	24-4	Returns the next entry from an opened directory

Function Name	Returns	Parameters	Page	Description
WTdirectory_open	WTdirectory *	char *path	24-4	Opens a directory
WTDISPLAY_CYRSTALEYES **WTDISPLAY_DEFAULT** **WTDISPLAY_NEEDSTENCIL** **WTDISPLAY_NOWINDOW** **WTDISPLAY_STEREO**	n/a	n/a	2-2	Constants used with the WTuniverse_new function to specify a display type **CYRSTALEYES** = a CyrstalEyes display **DEFAULT** = a single window display **NEEDSTENCIL** = request stencil buffer **NOWINDOW** = no display **STEREO** = a two window display (for legacy code)
WTDRAW2D_HOLLOW **WTDRAW2D_SOLID**	n/a	n/a	C-2	Constants used to define the drawing style for 2D drawing functions
WTerror	void	char *format **....**	24-8	Displays an error messages and immediately exits application
WTeuler_2m3	void	float wx float wy float wz WTm3	25-27	Constructs a 3D rotation matrix from the given euler angles (specified in radians)
WTeuler_2q	void	float wx float wy float wz WTq q	25-27	Converts euler angles, specified in radians, to a unit quaternion
WTEVENT_ACTIONS **WTEVENT_OBJECTSENSOR** **WTEVENT_PATHS** **WTEVENT_TASKS**	n/a	n/a	2-9	Constants used with the WTuniverse_seteventorder function to change the order of events in the simulation loop
WTEYE_LEFT **WTEYE_RIGHT**	n/a	n/a	C-3	Constants used with the WTwindow_geteye and WTwindow_seteye functions, which indicate whether the window's viewpoint is displaying the left or right eye's view
WTfastrak_afilter	void	WTsensor *ftrak float sensitivity float flow float fhigh float factor	13-95	Controls the amount of adaptive filtering applied to the orientation or altitude values returned by the Fastrak device

Function Name	Returns	Parameters	Page	Description
WTfastrak_afilteroff	void	WTsensor *ftrak	13-95	Turns off the filtering of orientation or altitude values set by WTfastrak_afilter
WTfastrak_pfilter	void	WTsensor *ftrak float sensitivity float flow float fhigh float factor	13-95	Controls the amount of adaptive filtering applied to the position values returned by the Fastrak device
WTfastrak_pfilteroff	void	WTsensor *ftrak	13-95	Turns off the filtering of position values previously set by WTfastrak_pfilter
WTFASTRAK_STYLUSBUTTON_DOWN	n/a	n/a	C-15	Constant used with the function WTsensor_getmiscdata to determine the state of the Polhemus Stylus
WTfastrak_update	void	WTsensor *sensor	13-94	Updates the Fastrak sensor device's location
WTFILE_DELIM	n/a	n/a	C-21	Constant that represents a forward slash (/) on Unix or a backslash (\) on MS-DOS
WTFILE_PATHDELIM	n/a	n/a	C-21	Constant that represents the maximum length of a filename (which includes the full path name)
WTFILETYPE_BFF **WTFILETYPE_DXF** **WTFILETYPE_NFF**	n/a	n/a	4-48	Constants used with the functions WTgeometry_save and WTnode_save to save a geometry or a geometry node to the binary file format (p.D-1), Autocad file format, or neutral file format
WTFILETYPE_WRL	n/a	n/a	4-48	Constant used with the function WTnode_save to save a node to the VRML format
WTFILTER_NEAREST **WTFILTER_LINEAR** **WTFILTER_NEARESTMIPMAP NEAREST** **WTFILTER_LINEARMIPMAP NEAREST** **WTFILTER_NEARESTMIPMAP LINEAR** **WTFILTER_LINEARMIPMAPLINEAR**	n/a	n/a	10-26	Arguments used with the function WTtexture_setfilter
WTflock_close	void	WTsensor *sensor	13-46	Corresponds to the WTbird_close function

Function Name	Returns	Parameters	Page	Description
WTflock_deviceopen	WTserial *	char *device int baud char parity int databits int stopbits int buffersize	13-45	Needed only when not using the default WTflock_new
WTflock_getabsmat	FLAG	WTsensor *sensor WTm3 mat	13-50	Retrieves the FOB's current absolute position/ orientation matrix and stores it in mat
WTflock_getabsoluterecord	FLAG	WTsensor *sensor WTp3 p WTq q	13-50	Retrieves the last absolute, unreletavized, unscaled position and orientation values stored in the sensor object's record
WTflock_getabspos	FLAG	WTsensor*sensor WTp3 p	13-50	Obtains the FOB's current unreletavized, absolute, and unscaled position and store it in p
WTflock_getcrtsyncdata	FLAG	WTsensor *sensor float *voltage float *rate	13-47	Returns success in retrieving the current CRT synchronization settings
WTflock_getdefaulthemisphere	int	void	13-46	Returns the FOB startup default hemisphere
WTflock_gethemisphere	int	WTsensor *sensor	13-47	Returns the current hemisphere setting for a sensor
WTflock_getlastmat	FLAG	WTsensor *sensor WTm3 mat	13-49	Copies the FOB's last stored absolute sensor matrix into mat
WTflock_getlastpos	FLAG	WTsensor *sensor WTp3 p	13-49	Retrieves the FOB's last stored absolute position and places it in p
WTflock_getorgmat	FLAG	WTsensor *sensor WTm3 mat	13-49	Retrieves the FOB sensor's original position/ orientation matrix and places it in mat
WTflock_open	int	WTsensor *sensor	13-45	Corresponds to the WTbird_open function
WTflock_resetorigin	FLAG	WTsensor *sensor	13-48	Returns TRUE if the FOB's current position and orientation values have been set to the FOB's original position and orientation values
WTflock_setcrtsync	FLAG	WTsensor *sensor int mode	13-48	Corresponds to the WTbird_setsync function
WTflock_setdefaulthemisphere	FLAG	int hemisphere	13-46	Returns success in setting the default startup hemisphere (useful with multiple FOB configurations)

Function Name	Returns	Parameters	Page	Description
WTflock_sethemisphere	FLAG	WTsensor *sensor int hemisphere	13-47	Returns success in setting the current hemisphere for a sensor
WTflock_update	void	WTsensor *sensor	13-48	Relativizes the current absolute position of the FOB
WTFOG_EXP **WTFOG_EXPSQUARED** **WTFOG_LINEAR** **WTGOG_NONE**	n/a	n/a	4-66	Constants used with WTfognode_setmode to set the mode of the specified fog node
WTfognode_getcolor	FLAG	WTnode *node float *red float *grn float *blue float *alpha	4-65	Retrieves the fog color of a fog node
WTfognode_getlinearstart	float	WTnode *node	4-67	Returns the linear start distance of the specified fog node
WTfognode_getmode	int	WTnode *node	4-66	Returns the mode of the specified fog node
WTfognode_getrange	float	WTnode *node	4-66	Returns the range (distance) of the specified fog node
WTfognode_new	WTnode *	WTnode *parent	4-45	Creates a fog node and puts it in the scene graph after the last child of the specified parent
WTfognode_setcolor	FLAG	WTnode *node float red float grn float blue float alpha	4-65	Sets the fog color of a fog node (default color is black)
WTfognode_setlinearstart	FLAG	WTnode *node float start	4-66	Specifies the starting distance where the fog color will affect the appearance of objects (default is 0.0)
WTfognode_setmode	FLAG	WTnode *node int mode	4-66	Sets the mode of the specified fog node (default is WTFOG_LINEAR)
WTfognode_setrange	FLAG	WTnode *node float range	4-65	Specifies the distance where all objects are blended into the fog (default is 0.0)
WTfont3d_charexists	FLAG	WTfont3d *font char character	9-5	Returns TRUE if the character is defined in the font
WTfont3d_delete	void	WTfont3d *font	9-3	Frees the memory used by a WTfont3d structure

Function Name	Returns	Parameters	Page	Description
WTfont3d_getextents	void	WTfont3d *font WTp3 extents[2]	9-4	Gets the 3D extents box for the specified 3D font
WTfont3d_getspacing	float	WTfont3d *font	9-3	Returns the current spacing value for the specified 3D font
WTfont3d_load	WTfont3d *	char *filename	9-2	Loads a 3D font file into memory and returns a pointer to a structure used to refer to that font
WTfont3d_setspacing	void	WTfont3d *font float spacing	9-3	Set the spacing for a font (default is max width + 10%)
WTformula_drive	void	WTsensor *sensor	13-112	Obtains the raw wheel and pedal information and applies any scale factors that may have been set with WTsensor_setsensitivity and WTsensor_setangularrate
WTformula_rawupdate	void	WTsensor *sensor	13-113	Obtains the raw wheel and pedal information
WTFRAME_LOCAL **WTFRAME_PARENT** **WTFRAME_VPOINT** **WTFRAME_WORLD**	n/a	n/a	C-4	Constants for the reference frame in which motion is to occur
WTfree	void	void *pointer	24-10	Frees memory allocated by WTmalloc, WTcalloc, WTrealloc *Note: Required for SGI MP/MP*
WTFUZZ	n/a	n/a	C-6	Defined constant that equals 0.004
WTGEOBALL_LEFTBUTTON **WTGEOBALL_RIGHTBUTTON**	n/a	n/a	13-55	Constants used to retrieve event data from the Geometry Ball, Jr.
WTgeoball_present	FLAG	WTserial *serial	13-55	checks whether there is a live Geometry Ball, Jr. sensor object at the particular serial port corresponding to the serial port object
WTgeoball_update	void	WTsensor *sensor	13-54	Updates a Geometry Ball, Jr. device's location
WTgeometry_begin	WTgeometry *	void	6-22	Begins construction of a new geometry
WTgeometry_beginedit	FLAG	WTgeometry *geom	6-42	Lets WTK know you are going to edit a geometry

Function Name	Returns	Parameters	Page	Description
WTgeometry_beginpoly	WTpoly *	WTgeometry *geom	6-23	Returns a new empty polygon assigned to the specified geometry (default color is white with backface rejected)
WTgeometry_changetexture	FLAG	WTgeometry *geom char *bitmap FLAG shaded FLAG transparent	10-16	Changes all polygons of the specified geometry that have texture to use the new texture
WTgeometry_close	FLAG	WTgeometry *geom	6-23	Finishes the definition of a geometry (no more vertices or polygons can be added)
WTgeometry_closesmooth	FLAG	WTgeometry *geom	6-25	Finishes the definition of a geometry and computes the vertex normals for each of the geometry's vertices
WTgeometry_computevertexnormal	FLAG	WTgeometry *geom WTvertex *v	6-46	Automatically computes the normal of the specified vertex in the specified geometry. The vertex's normal is computed as the average of the surrounding polygon normals. It returns TRUE if the vertex normal could be computed, and FALSE otherwise
WTgeometry_copy	WTgeometry *	WTgeometry *geom	6-26	Creates and returns a new geometry by copying an existing geometry
WTgeometry_delete	int	WTgeometry *geom	6-26	Deletes a geometry (or reduces the number of instances of the geometry by one, if multiple instances exist) and returns the number of remaining instances
WTgeometry_deleteprebuild	FLAG	WTgeometry *geom	6-40	Deletes the optimized data structures created when WTgeometry_prebuild was called
WTgeometry_deletetexture	void	WTgeometry *geom	10-23	Removes all the textures from a geometry's surfaces
WTgeometry_endedit	FLAG	WTgeometry *geom	6-43	Call this function when you are finished editing the specified geometry
WTgeometry_getextents	void	WTgeometry *geom WTp3 extents	6-29	Retrieves the extents of the geometry's axis-aligned bounding box (X,Y,Z dimensions)
WTgeometry_getmidpoint	void	WTgeometry *geom WTp3 p	6-28	Obtains the midpoint of a geometry's extents box in local coordinates
WTgeometry_getmtable	WTmtable *	WTgeometry *geom	6-31	Returns a pointer to the material table referenced by the geometry

Function Name	Returns	Parameters	Page	Description
WTgeometry_getname	char *	WTgeometry *geom	6-30	Returns the name of the specified geometry
WTgeometry_getpolys	WTpoly *	WTgeometry *geom	6-32	Returns a pointer to the first polygon in the geometry
WTgeomety_getradius	float	WTgeometry *geom	6-29	Returns the radius of the specified geometry
WTgeometry_getrenderingstyle	int	WTgeometry *geom	6-35	Returns the current rendering style for a geometry
WTgeometry_getvertexmatid	int	WTgeometry *geom WTvertex *vertex	6-48	Returns the material ID (index) for the specified vertex of the geometry
WTgeometry_getvertexnormal	FLAG	WTgeometry *geom WTvertex *vertex WTp3 normal	6-46	If a normal has been set for the specified vertex, then its value is returned in the “normal” argument and TRUE is returned, FALSE otherwise
WTgeometry_getvertexposition	void	WTgeometry *geom WTvertex *vertex WTp3 pos	6-44	Obtains the specified vertex’s position in the local coordinate system of the geometry
WTgeometry_getvertexrgb	FLAG	WTgeometry *geom WTvertex *vertex unsigned character *red unsigned character *green unsigned character *blue	6-47	Retrieves the specified vertex’s color
WTgeometry_getvertices	WTvertex *	WTgeometry *geom	6-33	Returns a pointer to the first vertex in the geometry
WTgeomety_id2poly	WTpoly *	WTgeometry *geom short id	6-33	Returns a pointer to the specified polygon in the geometry
WTgeometry_merge	WTpoly *	WTgeometry *geometry1 WTgeometry *geometry2	6-27	Merges geometry2 into geometry1 and returns a pointer to the merged geometry's first polygon. A new merged material table is also created
WTgeometry_newblock	WTgeometry *	float lx, float ly, float lz, FLAG bothsides	6-15	Creates and returns a pointer to a new block geometry, with the X, Y, and Z dimensions specified by the lx, ly, and lz arguments (bothsides indicates whether backfaces are visible)
WTgeometry_newcone	WTgeometry *	float height float radius int tess FLAG bothsides	6-16	Creates and a returns a pointer to a new cone-shaped geometry. The height and radius parameters specify size, tess gives the number of polygons to use, bothsides indicates if backfaces are visible

Function Name	Returns	Parameters	Page	Description
WTgeometry_newcylinder	WTgeometry *	float height float radius int tess FLAG bothsides FLAG gouraud	6-15	Creates and returns a pointer to a new cylinder geometry. The height and radius parameters specify size, tess gives the number of polygons to use, bothsides indicates if backfaces are visible. If the gouraud flag is TRUE, outward-pointing vertex normals parallel to the cylinder base are defined
WTgeometry_newextrusion	WTgeometry *	WTp2points[] int numpts float height FLAG bothsides FLAG gouraud	6-19	Makes a new geometry by extruding a given contour a specified distance (in the parameter height). The number of points must be between 3 and 256, bothsides indicate if backfaces are visible. If gouraud is TRUE, outward-pointing normals for the side polygons are defined
WTgeometry_newhemisphere	WTgeometry *	float radius int nlat int nlong FLAG bothsides FLAG gouraud	6-17	Creates and returns a pointer to a new hemisphere (exactly like WTgeometry_newsphere except only the top half of the sphere is created)
WTgeometry_newrectangle	WTgeometry *	float height float width FLAG bothsides	6-17	Constructs a geometry composed of a single rectangle (height is theY dimension, width is X), bothsides indicates if backfaces are visible
WTgeometry_newsphere	WTgeometry *	float radius int nlat int nlong FLAG bothsides FLAG gouraud	6-16	Creates and returns a pointer to a new sphere geometry (radius is the sphere's size, nlat and nlong are the number of latitude and longitude subdivisions to use), bothsides indicate if backfaces are visible If gouraud is TRUE, then outward-pointing vertex normals are defined
WTgeometry_newtext3d	WTgeometry *	WTfont3d *font char *string	6-20	Takes a character string and creates a geometry using the specified 3D font
WTgeometry_newtruncone	WTgeometry *	float height float toprad float baserad int tess FLAG bothsides FLAG gouraud	6-18	Creates a new geometry consisting of a single truncated cone (height gives the cone's height, toprad gives the top radius, baserad gives the bottom radius, tess gives the number of polygons to use to approximate the cone), bothsides indicate if backfaces are visible. If gouraud is TRUE, the outward-pointing normals are defined

Function Name	Returns	Parameters	Page	Description
WTgeometry_newvertex	WTvertex *	WTgeometry *geom WTp3 p	6-22	Adds the vertex with coordinates (X,Y,Z) in the WTp3 structure to a geometry
WTgeometry_numpolys	int	WTgeometry *geom	6-32	Returns the total number of polygons in the specified geometry
WTgeometry_prebuild	FLAG	WTgeometry *geom	6-40	Optimizes geometry data structures so they render faster
WTgeometry_prebuildreflectmap	FLAG	WTgeometry *geom char *texmap	6-41	Optimizes geometry data structures so they render faster while also applying a reflection map to the geometry.
WTgeometry_recomputestats	FLAG	WTgeometry *geom FLAG clearverts	6-44	Recomputes the specified geometry's statistics (extents, midpoint, radius, and bounding box) based on the locations of the geometry's vertices. If clearverts is TRUE, it removes unused vertices from the geometry
WTgeometry_save	FLAG	WTgeometry *geom char *filename WTpq *pq int filetype	6-26	Saves a geometry to a file using the specified filename
WTgeometry_scale	void	WTgeometry *geom float factor WTp3 center	6-38	Scales a geometry by a specified factor about a specified point in the local coordinate system
WTgeometry_setmatid	FLAG	WTgeometry *geom int id	6-31	Changes the material table index of all the polygons in the specified geometry to the specified index value
WTgeometry_setmtable	void	WTgeometry *geometry WTmtable *mtable	6-30	Causes the indicated geometry to reference the specified material table
WTgeometry_setname	void	WTgeometry *geom char *name	6-29	Assigns a name to a geometry
WTgeometry_setrenderingstyle	FLAG	WTgeometry *geom int modes int style	6-33	Sets the rendering style for a geometry (modes is a bitmask of the rendering flags to change, style is the new value of the rendering flags)

Function Name	Returns	Parameters	Page	Description
WTgeometry_setrgb	void	WTgeometry *geom unsigned char red unsigned char green unsigned char blue	6-31	Specifies the 24-bit color value of a geometry (red, green, blue), valid range is 0 to 255
WTgeometry_settexture	FLAG	WTgeometry *geom char *bitmap FLAG shaded FLAG transparent	10-12	Applies a texture bitmap to each polygon surface of a geometry, “bitmap” specifies the bitmap filename. If shaded = TRUE, then the texture elements are affected by lighting. If transparent = TRUE, then black pixels in the texture are rendered transparent on the geometry
WTgeometry_settextureuv	FLAG	WTgeometry *geom char *bitmap float (*fu)(WTp3) float (*fv)(WTp3) FLAG shaded FLAG transparent	10-15	Drapes a texture onto a geometry (fu and fv specify how a texture is mapped onto the geometry by taking a 3D point, a WTp3, as an argument and return a floating point value)
WTgeometry_setuv	FLAG	WTgeometry *geom float (*fu)(WTp3) float (*fv)(WTp3)	10-32	Changes the way textures are mapped onto a geometry’s polygon
WTgeometry_setvertexmatid	FLAG	WTgeometry *geom WTvertex *vertex int id	6-48	Sets a vertex’s material table index
WTgeometry_setvertexnormal	FLAG	WTgeometry *geom WTvertex *vertex WTp3 normal	6-45	Sets a vertex normal for a geometry
WTgeometry_setvertexposition	FLAG	WTgeometry *geom WTvertex *vertex WTp3 pos	6-44	Sets a vertex position specified in world coordinates
WTgeometry_setvertexrgb	FLAG	WTgeometry *geom WTvertex *vertex unsigned character red unsigned character green unsigned character blue	6-47	Sets the vertex’s color to the specified red, green, and blue components
WTgeometry_stretch	void	WTgeometry *geom WTp3 factors WTp3 center	6-37	Stretches a geometry in its local coordinate frame by applying a different scale factor in each of the three coordinate dimensions

Function Name	Returns	Parameters	Page	Description
WTgeometry_transform	void	WTgeometry *geom WTpq *pq	6-39	Transforms an object in its local coordinate frame by the specified position and orientation
WTgeometry_translate	void	WTgeometry *geom WTp3 offset	6-38	Translates a geometry in its local coordinate frame by adding the specified offset to each of the geometry's vertices
WTgeometrynode_load	WTnode *	WTnode *parent char *filename float scale	4-46	Creates a single geometry node from the data read in from a file and adds the newly created node to the scene graph after the last child of the specified parent
WTgeometrynode_new	WTnode *	WTnode *parent WTgeometry *geom	4-44	Creates a geometry node with the specified geometry and adds it to the scene graph after the last child of the specified parent
WTglnode_getflags	int	WTnode *node	4-72	Returns the status of a Open GL node's flags
WTglnode_new	WTnode *	WTnode *parent void *GLCallbackFunction int flags	4-70	Creates an OpenGL callback node
WTglnode_replacecallback	FLAG	WTnode *node void *GLCallbackFunction	4-71	Replaces the currently assigned callback function in an OpenGL callback node with a new function
WTglnode_setcullingbox	FLAG	WTnode *node WTp3 midpoint WTp3 extents	4-71	Enables culling of the OpenGL callback node
WTglnode_setflags	FLAG	WTnode *node int flags	4-71	Replaces the flag settings of an OpenGL callback node
WTGLOVE5DT_ALL **WTGLOVE5DT_CLOSED** **WTGLOVE5DT_INDEX** **WTGLOVE5DT_MIDDLE** **WTGLOVE5DT_OPEN** **WTGLOVE5DT_PINKY** **WTGLOVE5DT_RING** **WTGLOVE5DT_THUMB**	n/a	n/a	13-65	Constants used with the 5DT Glove functions
WTglove5dt_calibrateclosed	void	WTsensor *sensor	13-66	Sets the state and orientation of the glove, which makes up the "closed" state
WTglove5dt_calibrateopen	int	WTsensor *sensor	13-66	Sets the state and orientation of the glove, which makes up the "open" state

Function Name	Returns	Parameters	Page	Description
WTglove5dt_loadhandmodel	int	WTsensor *sensor char *filename float scale	13-66	Changes the hand model while a simulation is running
WTglove5dt_rawupdate	int	WTsensor *sensor	13-65	Obtains the absolute pitch and roll data and the finger flex information
WTglove5dt_update	void	WTsensor *sensor	13-64	Obtains the absolute orientation and the current state of the fingers in the hand model, applies any rotational contraints to the record, and finally relativizes the record and stores it with the sensor
WTglove5dt_updatefingers	void	WTsensor *sensor	13-64	Updates the current state of the fingers in the hand model independently of the model's orientation
WTgroupnode_new	WTnode *	WTnode *parent	4-40	Creates a group node and adds it to the scene graph after the last child of the specified parent
WTiglasses_rawupdate	void	WTsensor *sensor	13-123	Obtains the tracker input and puts it in the raw data structure as an absolute euler rotation
WTiglasses_update	void	WTsensor *sensor	13-122	Updates the i-glasses! device
WTIMAGES	n/a	n/a	C-20	Environment variable to set additional file paths for images (see hardware guide for details)
WTinit_defaults	FLAG	int *argc char **argv	2-32	Creates an X Resource Database that overrides WTuniverse structure values. Unix platforms only
WTinit_setimages	void	const char *paths	2-33	Sets the path to the images directories
WTinit_setmodels	void	const char *paths	2-33	Sets the path to the models directories
WTinit_usewindow	void	(platform specific) *parent	17-6	Integrates WTK windows with host-specific windows *Note: On Windows platforms, parent is a HWND * On Unix platforms, parent is a Widget **
WTinlinenode_getlocation	char *	WTnode *node	4-64	Returns the inline string (URL) reference of the specified inline node
WTinlinenode_new	WTnode *	WTnode *parent	4-40	Creates an inline node and adds it to the scene graph after the last child of the specified parent
WTinlinenode_setlocation	FLAG	WTnode *node char *url	4-63	Replaces the inline string (URL) reference of the specified inline node with the new character string

Function Name	Returns	Parameters	Page	Description
WTinsidetrak_update	void	WTsensor *sensor	13-91	Packages the translation and rotation record from the device into the sensor object's record after relativizing it and then applying any translational scale factor that may have been set with WTsensor_setsensitivity
WTisotrak2_update	void	WTsensor *sensor	13-90	Updates an Isotrak II device's location
WTJOYSERIAL_BOTTOMDOWN **WTJOYSERIAL_HATDOWN** **WTJOYSERIAL_HATLEFT** **WTJOYSERIAL_HATRIGHT** **WTJOYSERIAL_HATUP** **WTJOYSERIAL_SIDEDOWN** **WTJOYSERIAL_TOPDOWN** **WTJOYSERIAL_TRIGGERDOWN** **WTJOYSERIAL_WCS1 (through 7)** **WTJOYSERIAL_WCSTOGGLEA** **WTJOYSERIAL_WCSTOGGLEB**	n/a	n/a	13-117	Constants to retrieve event data from a joystick
WTjoyserial_fly	void	WTsensor *sensor	13-116	Update function that moves sensor forward along the Z axis at a constant velocity
WTjoyserial_getdrift	float	WTsensor *sensor	13-119	Returns the drift amount of the specified joystick sensor
WTjoyserial_getrange	void	WTsensor *sensor WTp2 range	13-118	Returns the maximum (divided by 2) X and Y values that can be attained by the joystick
WTjoyserial_rawupdate	void	WTsensor *sensor	13-117	Reads in the raw data from the joystick and saves it in the sensor's raw data structure
WTjoyserial_readcalibrationfile	void	void	13-119	Reads the joystick's calibration file so the joystick can be re-initialized
WTjoyserial_setdrift	void	WTsensor *sensor float drift_per	13-118	Sets the joystick's drift amount to the specified percentage of the joystick's range
WTjoyserial_walk	void	WTsensor *sensor	13-116	Initializes joystick to move in a "walkthrough" mode
WTjoyserial_walk2	void	WTsensor *sensor	13-116	Initializes joystick to move in a second "walkthrough" mode
WTjoystick_fly	void	WTsensor *sensor	13-70	Gameport Joystick: Update function that moves sensor forward along the Z axis at a constant velocity

Function Name	Returns	Parameters	Page	Description
WTjoystick_getdrift	float	WTsensor *sensor	13-73	Gameport Joystick: Returns the drift amount of the specified joystick sensor
WTjoystick_getrange	void	WTsensor *sensor WTp2 range	13-72	Gameport Joystick: Returns the maximum (divided by 2) X and Y values that can be attained by the joystick
WTjoystick_rawupdate	void	WTsensor *sensor	13-71	Gameport Joystick: Reads in the raw data from the joystick and saves it in the sensor's raw data structure
WTjoystick_readcalibrationfile	void	void	13-73	Gameport Joystick: Reads the joystick's calibration file so the joystick can be re-initialized
WTjoystick_setdrift	void	WTsensor *sensor float drift_per	13-72	Gameport Joystick: Sets the joystick's drift amount to the specified percentage of the joystick's range
WTjoystick_walk	void	WTsensor *sensor	13-70	Gameport Joystick: Initializes joystick to move in a "walkthrough" mode
WTjoystick_walk2	void	WTsensor *sensor	13-70	Gameport Joystick: Initializes joystick to move in a second "walkthrough" mode
WTKCODES	n/a	n/a	D-8, HG	Environment variable specifying file path to WTKCODES file and also filename of file that contains the unlock code for WTK (see installation/ hardware guide for details)
WTKEY_DOWNARROW **WTKEY_END** **WTKEY_ESC** **WTKEY_F1 (through F12)** **WTKEY_HOME** **WTKEY_LEFTARROW** **WTKEY_MIDDLE** **WTKEY_PAGEDOWN** **WTKEY_PAGEUP** **WTKEY_RIGHTARROW** **WTKEY_UPARROW**	n/a	n/a	C-4	Keyboard constants returned by WTkeyboard_getkey and WTkeyboard_getlastkey
WTkeyboard_close	void	void	24-3	Closes a keyboard previously opened with WTkeyboard_open
WTkeyboard_getkey	short	void	24-2	Returns the next key from the keyboard input buffer
WTkeyboard_getlastkey	short	void	24-2	Returns the most recently pressed key from the keyboard input buffer, discards any others

Function Name	Returns	Parameters	Page	Description
WTkeyboard_open	void	void	24-1	Initializes a keyboard
~~**WTlight_add**~~	void	WTlight *light WTobject *object	F-9	**OBSOLETE** Adds the specified light to the default scene graph
~~**WTlight_remove**~~	void	WTlight *light	F-10	**OBSOLETE** Removes the WTlightnode associated with the specified WTlight from the scene graph, leaving the light parentless
WTlightnode_getambient	void	WTnode *light float *r float *g float *b	12-15	Obtains the ambient red, green, and blue components of a light's color (each component ranges from 0.0 to 1.0; when all three equal 1.0, the light is white)
WTlightnode_getangle	float	WTnode *light	12-19	Returns the half-angle of a spot light's cone
WTlightnode_getattenuation	void	WTnode *light float *atten0 float *atten1 float *atten2	12-18	Obtains a point or spot light's attenuation coefficients
WTlightnode_getdiffuse	void	WTnode *light float *r float *g float *b	12-16	Obtains the diffuse red, green, and blue components of a light's color (each component ranges from 0.0 to 1.0; when all three equal 1.0, the light is white)
WTlightnode_getdirection	void	WTnode *light WTp3 dir	12-13	Obtains the direction of a light (this direction vector is normalized to have a length equal to 1.0)
WTlightnode_getexponent	float	WTnode *light	12-20	Returns the exponent of a spot light
WTlightnode_getintensity	float	WTnode *light	12-14	Returns a light's intensity value
WTlightnode_getposition	FLAG	WTnode *light WTp3 p	12-12	Obtains the position of a light node
WTlightnode_getspecular	void	WTnode *light float *r float *g float *b	12-16	Obtains the specular red, green, and blue components of a light's color (each component ranges from 0.0 to 1.0; when all three equal 1.0, the light is white)
WTlightnode_gettype	int	WTnode *light	12-18	Determines which light constructor function was used to create the light

Function Name	Returns	Parameters	Page	Description
WTlightnode_load	FLAG	WTnode *parent char *filename	12-9	Reads a WTK light format file and creates spot, point, directed or ambient light nodes as indicated in the file
WTlightnode_newambient	WTnode *	WTnode *parent	12-5	Creates an ambient light node and adds it to the scene graph after the last child of the specified parent
WTlightnode_newdirected	WTnode *	WTnode *parent	12-6	Creates a directed light node and adds it to the scene graph after the last child of the specified parent
WTlightnode_newpoint	WTnode *	WTnode *parent	12-7	Creates a point light node and adds it to the scene graph after the last child of the specified parent
WTlightnode_newspot	WTnode *	WTnode *parent	12-8	Creates a spot light node and adds it to the scene graph after the last child of the specified parent
WTlightnode_save	FLAG	WTnode *light char *filename	12-11	Saves out an ambient, directed, point, or spot light to a file with the specified name
WTlightnode_setambient	void	WTnode *light float r float g float b	12-14	Sets the ambient component of a light's color
WTlightnode_setangle	void	WTnode *light float angle	12-19	Sets the radian value for spot lights (range of values from 0.0 to PI/2), default value is PI/8 (22.5 degrees)
WTlightnode_setattenuation	void	WTnode *light float atten0 float atten1 float atten2	12-17	Sets the attenuation value for point and spot lights (the rate at which light falls off as distance from the light increases). By default, no attenuation is set
WTlightnode_setdiffuse	void	WTnode *light float r float g float b	12-15	Set the diffuse component of a light's color (directed, point, and spot lights only)
WTlightnode_setdirection	void	WTnode *light WTp3 dir	12-13	Sets the direction of a light to the vector passed in (spot and directed lights only)
WTlightnode_setexponent	void	WTnode *light float val	12-19	Specifies how the intensity of a spot light falls off within the spot light's cone (1.0 to 128.0 are valid), default is 0.0
WTlightnode_setintensity	void	WTnode *light float x	12-14	Sets the intensity of a light to x (valid values are between 0.0 and 1.0)

Function Name	Returns	Parameters	Page	Description
WTlightnode_setposition	FLAG	WTnode *light WTp3 p	12-12	Sets the 3D position of a light to the point passed in (point and spot lights only). Default position is 0,0,0
WTlightnode_setspecular	void	WTnode *light float r float g float b	12-15	Set the specular component of a light's color (directed, point, and spot lights only)
WTLIGHTTYPE_AMBIENT **WTLIGHTTYPE_DIRECTED** **WTLIGHTTYPE_POINT** **WTLIGHTTYPE_SPOT**	n/a	n/a	C-5	Constants returned by the function WTlight_gettype
WTlodnode_getcenter	FLAG	WTnode *node WTp3 center	4-56	Returns the center position of an LOD node
WTlodnode_getrange	FLAG	WTnode *node float *range int num	4-55	Returns the array of floats specifying the switch-out distances for the children of the specified LOD nodes
WTlodnode_new	WTnode *	WTnode *parent	4-41	Creates a LOD node and adds it to the scene graph after the last child of the specified parent
WTlodnode_numranges	int	WTnode *node	4-56	Returns the number of range entries in the LOD node
WTlodnode_setcenter	FLAG	WTnode *node WTp3 center	4-56	The center used by an LOD node to compute distance from the viewpoint (default is 0.0, 0.0, 0.0 in world coordinates)
WTlodnode_setrange	FLAG	WTnode *node float *range int num	4-55	Sets the array of floats specifying the switch-out distances for the children of the specified LOD node
WTLOGITECH_FLAGBIT **WTLOGITECH_FLYING** **WTLOGITECH_LEFTBUTTON** **WTLOGITECH_MIDDLEBUTTON** **WTLOGITECH_OUTBIT** **WTLOGITECH_PEDESTALBUTTON** **WTLOGITECH_RIGHTBUTTON** **WTLOGITECH_SUSPENDBUTTON**	n/a	n/a	12-74	Constants used to retrieve event data from the Logitech 3D Mouse device
WTlogitech_update	void	WTsensor *sensor	12-80	Updates a Logitech head tracker device

Function Name	Returns	Parameters	Page	Description
WTm3_2euler	void	WTm3 m WTp3 first WTp3 second	25-27	Extracts euler angles, specified in radians, from the specified rotation matrix
WTm3_2eulernear	void	WTm3 m WTp3 nearp WTp3 euler	25-32	Returns a euler that is closest to a specified euler, corresponding to the specified WTm3 (a 3 x 3 matrix)
WTm3_2q	void	WTm3 m WTq q	25-26	Converts a 3D rotation matrix to a quaternion
WTm3_copy	[MACRO]	WTm3 min WTm3 mout	25-22	Copies min into mout
WTm3_init	[MACRO]	WTm3 m	25-21	Sets a matrix equal to the identity matrix
WTm3_multm3	void	WTm3 m1 WTm3 m2 WTm3 mout	25-22	Performs the matrix multiplication mout = m1 * m2
WTm3_transpose	void	WTm3 min WTm3 mout	25-22	Puts the transpose of min into mout
WTm4_2pq	void	WTm4 m WTpq *pq	25-31	Converts a WTm4 matrix (a 4 x 4 matrix) into a WTpq structure that holds position (p) and orientation (q) values
WTm4_copy	[MACRO]	WTm4 min WTm4 mout	25-23	Copies min into mout
WTm4_init	[MACRO]	WTm4 m	25-23	Sets a matrix equal to the identity matrix
WTm4_invert	int	WTm4 src WTm4 dst	25-24	Inverts a WTm4 matrix (a 4 x 4 matrix)
WTm4_multm4	void	WTm4 m1 WTm4 m2 WTm4 mout	25-24	Performs the matrix multiplication mout = m1 * m2
WTm4_rotatep3	void	WTm4 m4 WTp3 pin WTp3 pout	25-25	Rotates a vector pin by a matrix, m4 and returns the result in pout. It ignores the translational component of m4, and applies only the rotational part to the WTp3 vector
WTm4_transpose	void	WTm4 min WTm4 mout	25-23	Puts the transpose of min into mout

Function Name	Returns	Parameters	Page	Description
WTm4_xformp3	void	WTm4 m4 WTp3 pin WTp3 pout	25-24	Transforms a point pin by the transform matrix in m4 and returns the result in pout
WTmalloc	void	size_t size	24-9	Similar to malloc; used to allocate shared memory. *Required for SGI MP/MP*
WTMAT_AMBIENT **WTMAT_AMBIENTDIFFUSE** **WTMAT_DIFFUSE** **WTMAT_EMISSION** **WTMAT_OPACITY** **WTMAT_SHININESS** **WTMAT_SPECULAR**	n/a	n/a	C-5	Constant used with material table functions **AMBIENT** = controls color reflected from a material in ambient white light (specified in RGB floats ranging from 0.0 to 1.0) **AMBIENTDIFFUSE** = controls color reflected from a material in ambient or diffuse white light (specified in RGB floats ranging from 0.0 to 1.0) **DIFFUSE** = controls color reflected from a material in diffuse white light (specified in RGB floats ranging from 0.0 to 1.0) **EMISSION** = controls the color of light *produced* (not reflected) by a material (specified in RGB floats ranging from 0.0 to 1.0) **OPACITY** = controls the extent to which the color value of a pixel is combined with the color value behind it (specified in values from 0.0 to 1.0, where 0 is invisible and 1 is opaque) **SHININESS** = controls the narrowness of focus of specular highlights (specified in values ranging from 0 to 128) **SPECULAR** = controls color reflected from a material in specular white light (specified in RGB floats ranging from 0.0 to 1.0)
WTmessage	void	char *format	24-5	Similar to printf; prints messages from your application to output
WTMESSAGE_ERROR **WTMESSAGE_USER** **WTMESSAGE_WARNING**	n/a	n/a	24-6	Constants that indicate the type of message (error, user, or warning) sent by an application

Function Name	Returns	Parameters	Page	Description
WTMESSAGE_TOCALLBACK **WTMESSAGE_TOCONSOLE** **WTMESSAGE_TOFILE** **WTMESSAGE_TONOWHERE**	n/a	n/a	24-6	Constants that indicate where to direct the message (file, console, etc.) sent by an application
WTmessage_sendto	void	int type int where FILE *output_file	24-6	Redirects messages, warnings, and errors from your application to a file, console, or callback function
WTmessage_setcallback	void	void (*callback function) (char *str)	24-7	Sets the destination for the message callbacks
WTMODELS	n/a	n/a	B-2	Environment variable to set additional file paths for models (see hardware guide for details)
WTmotionlink_addconstraint	FLAG	WTmotionlink *link int dof float min float max	15-11	Adds a constraint to a motion link so the position and/or the orientation of the motion link's target is constrained (dof is the degree of freedom)
WTmotionlink_delete	void	WTmotionlink *link	15-5	Deletes the specified motion link from the universe's list of motion links and releases all memory used by it
WTmotionlink_enable	void	WTmotionlink *link FLAG flag	15-5	If the flag is TRUE (default), the specified motion link is enabled and has an effect on its target
WTmotionlink_getconstraintframe	int	WTmotionlink *link	15-11	Returns the frame in which the constraints on the specified motion link are applied
WTmotionlink_getdata	void	WTmotionlink *link	15-6	Retrieves the user-defined data field for the specified motion link
WTmotionlink_getreference frame	int	WTmotionlink *link	15-9	Returns a WTframe value in which the indicated motion link is applied
WTmotionlink_getsource	FLAG	WTmotionlink *link void **source int *type	15-6	Retrieves the source and source type for the specified motion link (returns TRUE if successful)
WTmotionlink_gettarget	FLAG	WTmotionlink *link void **target int *type	15-6	Retrieves the target and target type for the specified motion link (returns TRUE if successful)
WTmotionlink_isenabled	FLAG	WTmotionlink *link	15-5	Returns TRUE if the motion link is enabled (active), or FALSE if the motion link is disabled

Function Name	Returns	Parameters	Page	Description
WTmotionlink_new	WTmotionlink *	void *source void *target int from_type int to_type	15-3	Creates a new motion link whose source will affect the position and orientation of the target relative to a particular reference frame
WTmotionlink_next	WTmotionlink *	WTmotionlink *link	15-8	Returns the next motion link from the universe's list of motion links
WTmotionlink_removeconstraint	FLAG	WTmotionlink *link int dof	15-12	Removes a specified constraint from the specified motion link (returns TRUE if successful)
WTmotionlink_setconstraintframe	FLAG	WTmotionlink *link int constraintframe	15-10	Sets the constraint frame for a motion link
WTmotionlink_setdata	void	WTmotionlink *link void *data	15-6	Sets the user-defined data field for the specified motion link
WTmotionlink_setreferenceframe	FLAG	WTmotionlink *link int frame WTviewpoint *vpoint	15-8	Sets the reference frame in which the indicated motion link will operate
WTMOUSE_LEFTBUTTON **WTMOUSE_LEFTDBLCLK** **WTMOUSE_LEFTDOWN** **WTMOUSE_LEFTUP** **WTMOUSE_MIDDLEBUTTON** **WTMOUSE_MIDDLEDBLCLK** **WTMOUSE_MIDDLEDOWN** **WTMOUSE_MIDDLEUP** **WTMOUSE_RIGHTBUTTON** **WTMOUSE_RIGHTDBLCLK** **WTMOUSE_RIGHTDOWN** **WTMOUSE_RIGHTUP**	n/a	n/a	13-33	Constants used to retrieve event data from mouse buttons
WTmouse_drawcursor	void	WTsensor *sensor	13-28	Update function that does not allow any movement (neither translational nor rotational)
WTmouse_gettrackballdrift	float	WTsensor *sensor	13-37	Returns the current drift value of the trackball
WTmouse_gettrackballsnap	float	WTsensor *sensor	13-39	Returns the current snap value of the trackball
WTmouse_gettrackballsnapangle	float	WTsensor *sensor	13-38	Returns the current snap angle of the trackball
WTmouse_inwindow	FLAG	WTsensor *mouse WTwindow *w	13-35	Determines whether the mouse is within a WTK window

Function Name	Returns	Parameters	Page	Description
WTmouse_move2D	void	WTsensor *sensor	13-29	Mouse function that allows Z translation and Y rotation
WTmouse_moveview1	void	WTsensor *sensor	13-29	A mouse update function that operates in 3D without pitch or roll
WTmouse_moveview2	void	WTsensor *sensor	13-31	A mouse update function that supports 6 DOF
WTmouse_settrackballdrift	void	WTsensor *sensor float drift	13-37	Sets the drift of the trackball
WTmouse_settrackballsnap	void	WTsensor *sensor float snap	13-38	Sets the snap value of the trackball
WTmouse_settrackballsnapangle	void	WTsensor *sensor float snapangle	13-38	Sets the snap angle of the trackball
WTmouse_trackball	void	WTsensor *sensor	13-36	Update function for using the mouse as a trackball; use if connecting the trackball to the object
WTmouse_trackballreset	void	WTsensor *sensor	13-39	Stops the rotation of a continuously rotating object
WTmouse_trackballvpoint	void	WTsensor *sensor	13-37	Update function for using the mouse as a trackball; use if connecting the trackball to the viewpoint
WTmouse_whichwindow	WTwindow	WTsensor *mouse	13-35	Determines which WTK window (if any) the mouse is in and returns a pointer to that window
WTmouse_rawupdate	void	WTsensor *sensor	13-32	Obtains the X,Y screen location of the mouse and stores it in the sensor's raw data structure
WTmovgeometrynode_new	WTnode *	WTnode *parent WTgeometry *geom	5-3	Creates a movable geometry node from the existing geometry and adds it to the scene graph after the last child of the specified parent
WTmovlightnode_newdirected	WTnode *	WTnode *parent	5-4	Creates a movable directed light node and adds it to the scene graph after the last child of the specified parent
WTmovlightnode_newpoint	WTnode *	WTnode *parent	5-3	Creates a movable point light node and adds it to the scene graph after the last child of the specified parent
WTmovlightnode_newspot	WTnode *	WTnode *parent	5-4	Creates a movable spot light node and adds it to the scene graph after the last child of the specified parent
WTmovlodnode_new	WTnode *	WTnode *parent	5-5	Creates a movable LOD node and adds it to the scene graph after the last child of the specified parent

Function Name	Returns	Parameters	Page	Description
WTmovnode_alignaxis	FLAG	WTnode *movnode int axis WTp3 dir	5-8	Rotates the movable node about its midpoint so that the specified axis of the movable aligns with (i.e., points in the same direction) as the direction vector
WTmovnode_attach	FLAG	WTnode *parent WTnode *child int attachmentnum	5-11	Attaches the child node to the parent node (movable nodes only), where int is the value of the attachment. If int =2, then the child node is the third attachment
WTmovnode_axisrotation	FLAG	WTnode *movnode int axis float angle	5-8	Rotates a movable node in its local frame (i.e., it rotates around itself)
WTmovnode_deleteattachment	FLAG	WTnode *parent int attachmentnum	5-12	Detaches the child node, whose attachment number is specified in int, from the parent node (movable nodes only) and deletes all nodes of the sub-tree beginning with that attachment node (if they have no parents in other branches of any scene graph)
WTmovnode_detach	FLAG	WTnode *parent int attachmentnum	5-12	Detaches the child node, whose attachment number is specified in int, from the parent node (movable nodes only), possibly leaving the detached node with no parents
WTmovnode_getattachment	WTnode *	WTnode *node int attachmentnum	5-13	Returns a pointer to the node whose attachment number is specified in attachmentnum
WTmovnode_instance	WTnode *	WTnode *parent WTnode *movable	5-13	Creates a separate instance of a movable node and adds it to the scene graph after the last child of the specified parent
WTmovnode_load	WTnode *	WTnode *parent char *fname float scale	5-5	Creates a movable node from data read in from a file and adds it to the scene graph after the last child of the specified parent
WTmovnode_numattachments	int	WTnode *node	5-13	Returns the movable node's number of attachments
WTmovsepnode_new	WTnode *	WTnode *parent	5-4	Creates a movable separator node and adds it to the scene graph after the last child of the specified parent
WTmovswitchnode_new	WTnode *	WTnode *parent	5-5	Creates a movable switch node and adds it to the scene graph after the last child of the specified parent
WTmsleep	void	int msec	24-8	Causes the application to "sleep" for a given amount of time in milliseconds

Function Name	Returns	Parameters	Page	Description
WTmtable_copyentry	int	WTmtable *from int matid WTmtable *to	8-14	Copies an entry from one material table to another (or duplicates an entry within the same material table)
WTmtable_delete	FLAG	WTmtable *mtable	8-8	Deletes the specified material table
WTmtable_getbyname	WTmtable *	char *name	8-13	Returns a pointer to the named material table
WTmtable_getdata	void	WTmtable *mtable	8-13	Retrieves the user-defined data stored within a material table
WTmtable_getentrybyname	int	WTmtable *mtable char *name	8-17	Returns the index of the specified entry in the specified material table
WTmtable_getentryname	char *	WTmtable *mtable int matid	8-16	Returns the name of an entry in the specified material table
WTmtable_getname	char *	WTmtable *mtable	8-13	Returns the name of the specified material table
WTmtable_getnumentries	int	WTmtable *mtable	8-8	Returns the number of table entries in the specified material table
WTmtable_getproperties	int	WTmtable *mtable	8-11	Returns a bitwise combination of defined properties for the specified material table
WTmtable_getvalue	FLAG	WTmtable *mtable int matid float *value int propertybit	8-15	Queries the characteristics of an entry in the specified material table
WTmtable_load	WTmtable *	char *filename	8-11	Reads a material table from the specified filename
WTmtable_merge	WTmtable *	WTmtable *table1 WTmtable *table2	8-8	Merges two material tables and returns a new material table that contains the materials from both tables
WTmtable_new	WTmtable *	int defineprops int estimatedentries char *name	8-7	Creates a new material table (must have a unique name)
WTmtable_newentry	int	WTmtable *mtable	8-14	Creates a new entry in the specified material table with all the defined fields set to the default values and returns the index value which corresponds to the new material table entry
WTmtable_save	FLAG	WTmtable *table	8-12	Writes a material table to a file in the current directory (overwrites existing file with the same name)

Function Name	Returns	Parameters	Page	Description
WTmtable_setdata	void	WTmtable *mtable void *data	8-13	Sets a user-defined data field in a material table
WTmtable_setentryname	FLAG	WTmtable *mtable int matid char *name	8-16	Assigns a name to the specified entry in the specified material table
WTmtable_setname	FLAG	WTmtable *mtable char *name	8-12	Sets the name of the specified material table
WTmtable_setproperties	WTmtable *	WTmtable *mtable int definedprops	8-9	Adds or removes properties defined in a material table
WTmtable_setvalue	FLAG	WTmtable *mtable int matid float *value int propertybit	8-15	Alters the characteristics of an entry in a material table
WTnet_additem	FLAG	void *item int len int type int tag	22-9	Adds an item to the packet sent over the network (for distributed simulations)
WTnet_addstring	FLAG	void *string int len int type int tag	22-10	Adds a string item to the packet sent over a network (for distributed simulations)
WTnet_close	void	void	22-9	Closes the network and deletes private data structures
WTnet_flush	void	void	22-13	Insures that all message items previously added by WTnet_additem and WTnet_addstring are sent out on the network
WTnet_getport	unsigned short	void	22-13	Returns the current port value
WTnet_getrange	FLAG	void	22-13	Returns the current range (Time to Live) value
WTnet_next	int	int *tag int *retlen	22-11	Finds what is the next available item
WTnet_open	FLAG	char *group unsigned short port unsigned char range	22-7	Opens the network

Function Name	Returns	Parameters	Page	Description
WTnet_removeitem	int	void *item int len int *tag int *retlen	22-11	Copies the next available item into the specified buffer
WTnet_removestring	int	void *item int len int *tag int *retlen	22-12	Same as WTnet_removeitem but used to get data known to be a string
WTnet_skip	void	void	22-13	Removes the next available item from the input queue
WTNODE_ANCHOR **WTNODE_FOG** **WTNODE_GEOMETRY** **WTNODE_GROUP** **WTNODE_ILLEGAL** **WTNODE_INLINE** **WTNODE_LOD** **WTNODE_LIGHT** **WTNODE_MGEOMETRY** **WTNODE_MLOD** **WTNODE_MSEP** **WTNODE_MSWT** **WTNODE_ROOT** **WTNODE_SEP** **WTNODE_SWT** **WTNODE_WTOBJECT** **WTNODE_XFORM** **WTNODE_XFORMSEP**	n/a	n/a	4-50	Constants returned by the function WTnode_gettype that indicates the specified node's type: **ANCHOR** = an anchor node **FOG** = a fog node **GEOMETRY** = a geometry node **GROUP** = a group node **ILLEGAL** = an illegal node **INLINE** = an inline node **LOD** = an level of detail node **LIGHT** = a light node **MGEOMETRY** = a movable geometry node **MLOD** = a movable level of detail node **MSEP** = a movable separator node **MSWT** = a movable switch node **ROOT** = a root node **SEP** = a separator node **SWT** = a switch node **WTOBJECT**= corresponds to a WTK 2.1 object **XFORM** = a transform node **XFORMSEP** = a transform separator node
WTnode_addchild	FLAG	WTnode *parentnode WTnode *child	4-74	Adds the child to the scene graph after the last child of the specified parent node
WTnode_addsensor	motionlink *	WTnode *node WTsensor *sensor	4-92	Attaches a sensor to a transform node

Function Name	Returns	Parameters	Page	Description
WTnode_axisrotation	FLAG	WTnode *node int axis float angle int frame	4-62	Creates an incremental rotation to the existing transform either in the local or parent frame around the specified axis (X, Y, or Z axis can be specified in radians)
WTnode_boundingbox	FLAG	WTnode *node FLAG onoff	4-73	Enable a bounding box for the specified node (TRUE = enabled)
WTnode_canaddchild	FLAG	WTnode *parent WTnode *child	4-51	Tests to see if the specified node can be added to the scene graph as the child of the specified parent node without creating a cycle in the scene graph (TRUE = yes)
WTNODECULL_OFF **WTNODECULL_ON**	n/a	n/a	C-7	Constants used with the WTsepnode_setcullmode function to set a separator node's culling mode
WTnode_delete	FLAG	WTnode *node	4-75	Detaches all occurrences of the specified node from their parent nodes. All nodes in the sub-tree beginning with this node will have the specified child nodes deleted (if they have no other parent in the scene graph)
WTnode_deletechild	FLAG	WTnode *parent int childnum	4-75	Detaches the numbered occurrence of the specified node from its parent node. All nodes in the sub-tree beginning with this node will be deleted (if they have no other parent in the scene graph)
WTnode_enable	FLAG	WTnode *node FLAG flag	4-49	Enables or disables the specified node during a rendering or picking traversal of the scene graph
WTnode_getchild	WTnode *	WTnode *parentnode int childnum	4-77	Returns the numbered child of the specified parent node
WTnode_getdata	void	WTnode *node	4-51	Retrieves the user-defined data stored within a node
WTnode_getextents	FLAG	WTnode *node WTp3 extents	4-53	Obtains the extents of a specified node (including the node's sub-tree)
WTnode_getgeometry	WTgeometry *	WTnode *node	4-45	Returns a pointer to the WTgeometry referenced by the specified geometry node
WTnode_getmidpoint	FLAG	WTnode *node WTp3 p	4-54	Obtains the midpoint of the specified node's extents box

Function Name	Returns	Parameters	Page	Description
WTnode_getname	char *	WTnode *node	4-49	Returns the name of the specified node
~~**WTnode_getobject**~~	WTobject *	WTnode *node	G-35	**OBSOLETE** Returns a pointer to a WTobject associated with the specified node
WTnode_getorientation	FLAG	WTnode *node WTq q	4-60	Returns the rotational component of the specified node's transformation matrix (using a quaternion input parameter)
WTnode getparent	WTnode *	WTnode *node int num	4-78	Returns the numbered parent of the specified node
WTnode_getradius	float	WTnode *node	4-54	Obtains the distance from the midpoint of the specified node's extents box to the corner of the box
WTnode_getrotation	FLAG	WTnode *node WTm3 m	4-60	Returns the rotational component of the specified node's transformation matrix (using a 3 x 3 matrix output parameter)
WTnode_gettransform	FLAG	WTnode *node WTm4 m	4-58	Returns the transformation matrix of the specified node
WTnode_gettranslation	FLAG	WTnode *node WTp3 p	4-59	Returns the translation component of the specified node's transformation matrix
WTnode_gettype	int	WTnode *node	4-50	Returns the type of the specified node
WTnode_hasboundingbox	FLAG	WTnode *node	4-73	If the node's bounding box has been enabled with WTnode_boundingbox, returns TRUE
WTnode_insertchild	FLAG	WTnode *parentnode WTnode *child int childnum	4-74	Adds the child node as the numbered node of the specified parent
WTnode_isenabled	FLAG	WTnode *node	4-50	Indicates whether the specified node is enabled (or disabled) for rendering and picking traversals
WTnode_ismovable	FLAG	WTnode *node	4-50	Returns TRUE if the specified node is a movable node
WTnode_load	WTnode *	WTnode *parent char *filename float scale	4-46	Creates one or more nodes from the data read in from a file and adds them to the scene graph after the last child of the specified parent
WTnode_numchildren	int	WTnode *node	4-77	Returns the number of children of the specified node
WTnode_numparents	int	WTnode *node	4-77	Returns the number of parents of the specified node

Function Name	Returns	Parameters	Page	Description
WTnode_numpolys	int	WTnode *node	4-78	Returns the number of polygons contained in the specified node's sub-tree
WTnode_print	void	WTnode *node	4-76	Prints formatted text, which represents the sub-tree starting at the specified node, to a standard I/O device
WTnode_rayintersect	WTpoly *	WTnode *node WTp3 ray WTp3 origin float *distance WTnodepath **nodepath	4-88	Obtains the frontmost intersected polygon along a specified ray contained in any geometry node in the sub-tree, beginning with the specified node (the ray is defined by the ray and origin arguments in the same coordinate system as the specified node)
WTnode_remove	FLAG	WTnode *node	4-75	Removes a specified node from all its parent nodes, disconnecting it from the scene graph
WTnode_removechild	FLAG	WTnode *parentnode int childnum	4-74	Removes the numbered child and its sub-tree from the parent node, possibly leaving it an orphan
WTnode_removesensor	void	WTnode *node WTsensor *sensor	4-92	Detaches the specified sensor from the specified node
WTnode_rotatem3	FLAG	WTnode *node WTm3 m3 int frame	4-61	Creates an incremental rotation to the existing transform either in the local or parent frame by the amount specified by the 3 x 3 matrix
WTnode_rotatem4	FLAG	WTnode *node WTm4 m4 int frame	4-62	Creates an incremental rotation to the existing transform either in the local or parent frame by the amount specified by the 4 x 4 matrix
WTnode_rotateq	FLAG	WTnode *node WTq q int frame	4-61	Creates an incremental rotation to the existing transform either in the local or parent frame by the amount specified by the quaternion
WTnode_rotation	FLAG	WTnode *node float y float x float z int frame	4-61	Creates an incremental rotation to the existing transform in either the local or parent frame (unlike WTnode_setrotation, which replaces the rotation value)

Function Name	Returns	Parameters	Page	Description
WTnode_save	FLAG	WTnode *node char *filename WTviewpoint *view int filetype int options	4-48	Saves the specified node (and its sub-tree for VRML only) to a file, whose name and type are specified by the variables filename and filetype
WTnode_setdata	void	WTnode *node void *data	4-51	Sets the user-defined data field in a node
WTnode_setname	FLAG	WTnode *node char *name	4-49	Sets the node's name (more than one node can have the same name)
WTnode_setorientation	FLAG	WTnode *node WTq q	4-60	Replaces the rotational component of the specified node's transformation matrix using a quaternion as an input parameter
WTnode_setrotation	FLAG	WTnode *node WTm3 m	4-60	Replaces the rotational component of the specified node's transformation matrix
WTnode_settransform	FLAG	WTnode *node WTm4 m	4-58	Replaces the transformation matrix of the specified node
WTnode_settranslation	FLAG	WTnode *node WTp3 p	4-59	Replaces the translation component of the transformation matrix of the specified node
WTnode_translate	FLAG	WTnode *node WTp3 pos int frame	4-59	Creates an incremental translation to the existing transform either in the local or parent frame, unlike WTnode_settranslation, which replaces the translation value
WTnode_vacuum	void	void	4-76	Deletes all non-root nodes from all the scene graphs in the universe
WTnodepath_addsensor	motionlink *	WTnodepath *nodepath WTsensor *sensor int frame	4-93	Allows you to attach a sensor to a node path (if the bottom-most node on the node path is a transform node)
WTnodepath_delete	FLAG	WTnodepath *nodepath	4-82	Deletes a node path
WTnodepath_getextents	FLAG	WTnodepath *nodepath float ext[2][3]	4-83	Places the center and extents of the node path in the specified floating point array
WTnodepath_getnode	WTnode *	WTnodepath *nodepath int num	4-82	Obtains the specified node in the specified node path

Function Name	Returns	Parameters	Page	Description
WTnodepath_getorientation	FLAG	WTnodepath *nodepath WTq q	4-85	Returns the rotational component of the transformation matrix that would be applied to the leaf node of the node path
WTnodepath_gettransform	FLAG	WTnodepath *nodepath WTm4 m4	4-84	Returns the transformation matrix that would be applied to the leaf node of the node path
WTnodepath_gettranslation	FLAG	WTnodepath *nodepath WTp3 p	4-84	Returns the translational component of the transformation matrix that would be applied to the leaf node of the node path
WTnodepath_gettraversal	int	WTnodepath *nodepath int *numarray int maxsize	4-83	Obtains a description of the specified node path relative to the entire scene graph
WTnodepath_intersectbbox	FLAG	WTnodepath *n1 WTnodepath *n2	4-87	Test for the intersection of two node paths, n1 and n2, based on their bounding boxes
WTnodepath_intersectnode	WTnodepath *	WTnodepath *nodepath WTnode *node int which	4-87	Performs a bounding box intersection test between the specified node path and the numbered occurrence of the specified node (and its sub-tree)
WTnodepath_intersectpoly	FLAG	WTnodepath *nodepoly1 WTnodepath *nodepoly2	4-86	Test for the intersection of any polygons in two node paths, n1 and n2, and their sub-trees
WTnodepath_new	WTnodepath *	WTnode *node WTnode *ancestor int which	4-81	Creates a new node path
WTnodepath_numnodes	int	WTnodepath *nodepath	4-82	Obtains the number of nodes in the specified node path
WTnodepath_removesensor	void	WTnode *nodepath WTsensor *sensor	4-93	Detaches the specified sensor from the specified node path's leaf node
WTnormal2slope	float	WTp3 normal	25-33	Takes a normal and returns the corresponding slope in radians
~~**WTobject_add**~~	void	WTobject *object	F-11	**OBSOLETE** Adds the specified object to the default scene graph
~~**WTobject_getnode**~~	void	WTobject *obj	G-35	**OBSOLETE** Returns the (automatically created) node associated with the specified object

Function Name	Returns	Parameters	Page	Description
~~**WTobject_remove**~~	void	WTobject *object	F-14	**OBSOLETE** Removes the WTobject node associated with the specified WTobject from the scene graph, making it an orphan
WTOPTION_3DSCHGTEXEXT **WTOPTION_MGENREADVCOLOR** **WTOPTION_NFFWRITE12** **WTOPTION_NFFWRITEV21** **WTOPTION_NFFWRITEUV** **WTOPTION_OLD3DS** **WTOPTION_OLDTEXTROT** **WTOPTION_OLDWFRONT** **WTOPTION_USEWTPUMP** **WTOPTION_VERTWARN**	n/a	n/a	2-24	Constants to set global parameters with the function WTuniverse_setoption.
WTp2_copy	[MACRO]	WTp2 pin WTp2 pout	25-4	Copies pin into pout
WTp2_dot	[MACRO]	WTp2 a WTp2 b	25-5	Returns the dot product of a and b
WTp2_init	[MACRO]	WTp2 p	25-4	Initializes a 2D vector so that p[X] = p [Y] = 0.0
WTp2_mag	[MACRO]	WTp2 p	25-4	Returns the magnitude of p
WTp2_norm	[MACRO]	WTp2 p	25-4	Normalizes p (scales it to unit length)
WTp2_subtract	[MACRO]	WTp2 p1 WTp2 p2 WTp2 pout	25-5	Subtracts vector p2 from p1 and puts the results in pout
WTp3_add	[MACRO]	WTp3 p1 WTp3 p2 WTp3 pout	25-7	Adds vectors p1 and p2 and puts the result in pout
WTp3_coplanar	short	WTp3 *verts int nverts WTp3 normal	25-11	Tests whether a set of 3D points are coplanar to within the WTK fuzz value (WTFUZZ)
WTp3_copy	[MACRO]	WTp3 pin WTp3 pout	25-6	Copies pin into pout
WTp3_cross	void	WTp3 p1 WTp3 p2 WTp3 pout	25-8	Finds the cross product of vectors p1 and p2 and puts the result into pout
WTp3_distance	float	WTp3 p1 WTp3 p2	25-11	Returns the distance between points p1 and p2

Function Name	Returns	Parameters	Page	Description
WTp3_disttovector	float	WTp3 pt WTp3 ptondir WTp3 dir	25-11	Returns the perpendicular distance from a point to a vector in 3D space
WTp3_dot	[MACRO]	WTp3 p1 WTp3 p2	25-7	Returns the dot product of p1 and p2
WTp3_equal	[MACRO]	WTp3 p1 WTp3 p2	25-8	Tests two 3D vectors for equality (returns TRUE, if equal)
WTp3_exact	[MACRO]	WTp3 p1 WTp3 p2	25-9	Tests two 3D vectors for equality
WTp3_frame2frame	void	WTp3 pin WTpq *frame1 WTpq *frame2 WTp3 pout	25-34	Takes frame1's position (specified in pin) and determines its position relative to frame2, then places the result in pout
WTp3_init	[MACRO]	WTp3 p	25-5	Initializes a 3D vector so that p[X] = p[Y] = p[Z] = 0.0
WTp3_invert	[MACRO]	WTp3 pin WTp3 pout	25-6	Negates a 3D vector
WTp3_local2worldframe	void	WTp3 pin WTpq *frame WTp3 pout	25-34	Takes the frame's position (specified in pin) and determines its position relative to the world coordinate frame, then places the result in pout
WTp3_mag	[MACRO]	WTp3 p	25-6	Returns the length of a 3D vector
WTp3_multm3	void	WTp3 v1 WTm3 m WTp3 v2	25-10	Multiplies a direction vector by a WTm3 matrix and places the result in v2
WTp3_multm4	void	WTp3 v1 WTm4 m WTp3 v2	25-10	Multiplies a direction vector by a WTm4 matrix and places the result in v2
WTp3_mults	[MACRO]	WTp3 p float s	25-10	Multiplies the vector p by the scalar (floating point value) s and places the result in p
WTp3_norm	[MACRO]	WTp3 p	25-6	Normalizes a 3D vector
WTp3_print	void	WTp3 pos char *string	25-12	Prints to WTMESSAGE_USER the specified string followed by the value of each element of the WTp3 structure

Function Name	Returns	Parameters	Page	Description
WTp3_rotate	void	WTp3 pin WTq q WTp3 pout	25-9	Rotates a 3D vector (specified in pin) through the rotation represented by q and puts the result into pout
WTp3_rotatept	void	WTp3 pin WTq q WTp3 pout WTp3 point	25-9	Rotates a 3D vector (specified in pin) around a 3D point by the rotation represented by a quaternion q and puts the result into pout
WTp3_subtract	[MACRO]	WTp3 p1 WTp3 p2 WTp3 pout	25-7	Subtracts vector p2 from p1 and puts the result into pout
WTp3_world2localframe	void	WTp3 pin WTpq *frame WTp3 pout	25-34	Takes the position of the world coordinate frame (specified in pin) and determines its position relative to the specified local frame, then places the result into pout
WTp3_xform	void	WTp3 pin WTpq *pq WTp3 pout	25-10	Transforms a point (specified in pin) from world coordinates to a local reference frame (specified in pq) and places it into pout
WTpath_appendelement	void	WTpath *path WTpathelement *element	14-27	Appends a path element onto a specified path's list of elements, making it the last element on the path
WTPATH_BEZIER **WTPATH_BSPLINE** **WTPATH_LINEAR**	n/a	n/a	14-6	Constants used with the function WTpath_interpolate for the method used to generate the positions of interpolated points
WTPATH_CURRENT **WTPATH_FIRST** **WTPATH_LAST**	n/a	n/a	14-17	Constants used with the function WTpath_seek to specify the starting point from which to specify the offset
WTpath_copy	WTpath *	WTpath *path	14-5	Creates a new path with a sequence of path elements with the same position and orientation values as the original path
WTpath_delete	void	WTpath *path	14-5	Deletes the specified path
WTpath_getconstraints	short	WTpath *path	14-21	Returns a path's constraints, as set by WTpath_setconstraints
WTpath_getcurrentelement	WTpathelement *	WTpath *path	14-18	Returns a path's current element
WTpath_getdata	void *	WTpath *path	14-30	Retrieves user-defined data stored within a path
WTpath_getdirection	FLAG	WTpath *path	14-20	Returns a path's play direction

Function Name	Returns	Parameters	Page	Description
WTpath_getelements	WTpathelement *	WTpath *path	14-10	Returns a pointer to the first element in the list of path elements
WTpath_getmarker	WTgeometry *	WTpath *path	14-10	Returns a pointer to the geometry that is currently being used to represent path elements when the path is made visible
WTpath_getmode	short	WTpath *path	14-22	Returns a path's play mode, as set by WTpath_setmode
WTpath_getname	const char *	WTpath *path	14-29	Returns the name of the specified path
WTpath_getplayspeed	int	WTpath *path	14-23	Returns the playback speed of a path (default is 1)
WTpath_getsamples	int	WTpath *path	14-23	Returns the sample rate of a path (default is 1)
WTpath_getvisibility	FLAG	WTpath *path	14-9	Returns TRUE if a path is currently visible
WTpath_insertelement	void	WTpath *path WTpathelement *element	14-27	Inserts a path element into the path's list of elements at the path's current position (immediately after the path's current element)
WTpath_interpolate	WTpath *	WTpath *path int npoints int method	14-6	Creates a new path by interpolating between the elements of the specified path
WTpath_isplaying	FLAG	WTpath *path	14-16	Returns TRUE if the specified path is currently playing
WTpath_isrecording	FLAG	WTpath *path	14-16	Returns TRUE is the specified path is currently being recorded
WTPATHLEN	n/a	n/a	C-22	Constant to specify the maximum length of a filename (which includes the full path name)
WTpath_load	WTpath *	char *filename NULL	14-11	Creates a new path consisting of one element per position and orientation record in the specified file
WTpath_new	WTpath *	NULL	14-4	Creates and returns a pointer to a new path
WTpath_next	WTpath *	WTpath *path	14-10	Iterates through the universe's list of paths
WTpath_numelements	int	WTpath *path	14-10	Returns the number of elements in a path's element list
WTpath_play	void	WTpath *path	14-14	Begins the playback of the specified path starting from the path's current element

Function Name	Returns	Parameters	Page	Description
WTpath_play1	void	WTpath *path	14-14	Begins the playback of the specified path starting from the path's current element, but only plays one frame
WTpath_record	FLAG	WTpath *path	14-14	Starts recording the position and orientation of the current viewpoint (default) or the target of a motion link
WTpath_record1	FLAG	WTpath *path	14-15	Starts recording the position and orientation of the current viewpoint (default) or the target of a motion link, but only one frame is recorded
WTpath_rewind	void	WTpath *path	14-16	Sets a path's current pointer to the path's first element
WTpath_save	FLAG	WTpath *path char *filename	14-12	Saves a path to the specified file (TRUE = success)
WTpath_seek	FLAG	WTpath *path int offset int where	14-18	Moves a path's current element pointer forward or backward in the path's element list
WTpath_setconstraints	void	WTpath *path short constraints	14-20	Constrains the position and orientation information played back by a path
WTpath_setcurrentelement	FLAG	WTpath *path WTpathelement *setelement	14-17	Sets the current element of a path (the element from which play begins)
WTpath_setdata	void	WTpath *path void *data	14-29	Sets the user-defined data field in a path
WTpath_setdirection	void	WTpath *path FLAG flag	14-19	Sets the play direction of a path
WTpath_setmarker	void	WTpath *path WTgeometry *marker	14-9	Sets the geometry that will be used to display a path element (if the path is made visible)
WTpath_setmode	void	WTpath *path short mode	14-21	Sets a path's playback mode
WTpath_setname	void	WTpath *path const char *name	14-29	Sets the name of the specified path
WTpath_setplayspeed	void	WTpath *path int speed	14-22	Sets a path's playback speed (default is 1)
WTpath_setrecordlink	FLAG	WTpath *path WTmotionlink *link	14-15	Records the motion of a target of a motion link

Function Name	Returns	Parameters	Page	Description
WTpath_setsamples	void	WTpath *path int frames_per_element	14-23	Sets a path's sample rate (default is 1)
WTpath_setvisibility	void	WTpath *path FLAG flag	14-8	Toggles the visibility of a path's graphical representation (TRUE = visible)
WTpath_showcurrentelement	void	WTpath *path	14-17	Moves the viewpoint to the position and orientation of the path's current element
WTpath_stop	void	WTpath *path	14-16	Stops a path that is playing or recording
WTpathelement_copy	WTpathelement *	WTpathelement *element	14-25	Creates a copy of the specified path element, with the same position and orientation information as the original
WTpathelement_delete	void	WTpathelement *element	14-24	Deletes a path element and frees the memory used by it (if the path element is a member of a path, it is removed from the path and then deleted)
WTpathelement_getorientation	void	WTpathelement *element WTq q	14-26	Retrieves the orientation of a single path element and places it in q
WTpathelement_getpath	WTpath *	WTpathelement *element	14-26	Returns a pointer to the path to which a path element belongs
WTpathelement_getposition	void	WTpathelement *element WTp3 pos	14-25	Retrieves the position of the specified path element and places it in pos
WTpathelement_new	WTpathelement *	WTpq *location	14-24	Creates and returns a pointer to a new path element, which is initialized to the specified position and orientation
WTpathelement_next	WTpathelement *	WTpathelement *element	14-26	Returns the next element in a list of path elements
WTpathelement_remove	void	WTpathelement *element	14-25	Removes a path element from the path that references it, but does not delete it
WTpathelement_setorientation	void	WTpathelement *element WTq q	14-26	Sets the orientation of a single path element to the specified orientation
WTpathelement_setposition	void	WTpathelement *element WTp3 pos	14-25	Sets the position of a single path element to the specified location

Function Name	Returns	Parameters	Page	Description
WTPINCH_LINDEX **WTPINCH_LMIDDLE** **WTPINCH_LPINKIE** **WTPINCH_LRING** **WTPINCH_LTHUMB** **WTPINCH_NOTOUCH** **WTPINCH_RINDEX** **WTPINCH_RMIDDLE** **WTPINCH_RPINKIE** **WTPINCH_RRING** **WTPINCH_RTHUMB**	n/a	n/a	13-62	Set of constants to retrieve contact information from a Fakespace PINCH Glove
WTpinch_update	void	WTsensor *sensor	13-61	Updates a Fakespace PINCH Glove's contact information
WTPLAY_CONTINUOUS **WTPLAY_OSCILLATE** **WTPLAY_TOEND**	n/a	n/a	14-21	Constants used with the function WTpath_setmode to set a path's playback mode
WTpolhemus_new	WTsensor *	port	16-6	Constructor macro for a Polhemus Isotrak
WTpoly_addvertex	FLAG	WTpoly *poly int vindex	7-10	Adds a vertex to the polygon under construction by referencing a vertex in the geometry to which the polygon belongs
WTpoly_addvertexptr	FLAG	WTpoly *p WTvertex *v	7-10	Adds a vertex to a polygon under construction by passing in a pointer to the vertex
WTpoly_close	FLAG	WTpoly *poly	7-12	Finishes the definition of a polygon
WTpoly_delete	FLAG	WTpoly *poly	7-12	Deletes the specified polygon
WTpoly_deletetexture	void	WTpoly *poly	10-23	Removes a texture from a polygon that has previously been textured
WTpoly_getbothsides	FLAG	WTpoly *poly	7-4	Returns the polygon's backface rejection status (TRUE means the polygon can be viewed from both sides)
WTpoly_getcg	void	WTpoly *poly WTp3 cg	7-5	Retrieves the center of gravity for the specified polygon and places it in cg
WTpoly_getgeometry	WTgeometry *	WTpoly *poly	7-7	Returns the geometry that contains the specified polygon
WTpoly_getid	short	WTpoly *poly	7-7	Returns the value of the specified polygon's ID (default is 0)

Function Name	Returns	Parameters	Page	Description
WTpoly_getmatid	int	WTpoly *poly	7-3	Retrieves the specified polygon's index into the material table
WTpoly_getnormal	void	WTpoly *poly WTp3 normal	7-4	Retrieves the normal vector of the specified polygon
WTpoly_getrgb	void	WTpoly *poly unsigned char *r unsigned char *g unsigned char *b	7-2	Retrieves the values of the red, green, and blue components of the polygon's color
WTpoly_gettextureinfo	FLAG	WTpoly *poly WTtextureinfo *info	10-31	Retrieves texture information for a specified polygon and places it in the specified WTtextureinfo structure
WTpoly_gettexturestyle	FLAG	WTpoly *poly FLAG *shaded FLAG *transparent FLAG *blended	10-24	Retrieves the shading, transparency, and blending settings of a texture that has been applied to a polygon
WTpoly_getuv	FLAG	WTpoly *poly float *uarray float *varray	10-32	Places the uv coordinates of a polygon's texture into the specified arrays
WTpoly_getvertex	WTvertex *	WTpoly *poly short index	7-8	Returns the specified vertex pointer for the specified polygon
WTpoly_intersectbbox	FLAG	WTpoly *poly1 WTnodepath *nodepath1 WTnodepath *nodepath2	4-86	Tests whether a polygon instance (nodepath1) intersects any part of the bounding box of the scene graph sub-tree whose start node is the bottom-most node of the node path (nodepath2)
WTpoly_intersectnode	FLAG	WTpoly *poly1 WTnodepath *nodepath1 WTnodepath *nodepath2	4-86	Tests whether a polygon instance (nodepath1) intersects any polygons in the scene graph sub-tree whose start node is the bottom-most node of the node path (nodepath2)
WTpoly_intersectpolygon	FLAG	WTpoly *poly1 WTnodepath *nodepath1 WTpoly *poly2 WTnodepath *nodepath2	4-85	Tests whether two polygons intersect (TRUE means yes). In addition to the polygons, you must specify the node path of the specific instance of each polygon
WTpoly_mirrortexture	void	WTpoly *poly	10-29	Flips an applied texture in 3D about the v axis of texture space

Function Name	Returns	Parameters	Page	Description
WTpoly_next	WTpoly *	WTpoly *poly	7-8	Returns the next polygon in the linked list of polygons associated with geometries
WTpoly_numvertices	short	WTpoly *poly	7-9	Returns the number of vertices in a polygon
WTpoly_rayintersect	FLAG	WTpoly *poly WTnodepath *npath WTp3 direction WTp3 origin float *dist	4-88	Tests whether a ray specified by an origin and a direction vector (in world coordinates) intersects a specified polygon. If the ray intersects the polygon, TRUE is returned
WTpoly_rotatetexture	void	WTpoly *poly float angle	10-27	Rotates the texture on a polygon in 2D (in texture space)
WTpoly_scaletexture	void	WTpoly *poly float factor	10-28	Scales textures applied to a polygon
WTpoly_setbothsides	void	WTpoly *poly FLAG flag	7-4	Specifies whether both sides of a polygon are visible (TRUE means both sides are visible, FALSE is default)
WTpoly_setid	void	WTpoly *poly short id	7-6	Sets the value of a polygon's ID (default is 0)
WTpoly_setmatid	FLAG	WTpoly *poly int id	7-3	Changes the polygon's material table index
WTpoly_setrgb	void	WTpoly *poly unsigned char r unsigned char g unsigned char b	7-2	Specifies the 24-bit color value of a polygon (0 to 255 are valid). Default color is white (255, 255, 255)
WTpoly_settexture	FLAG	WTpoly *poly char *bitmap FLAG shaded FLAG transparent	10-11	Applies a texture bitmap stored in the specified file to the specified polygon
WTpoly_settexturestyle	FLAG	WTpoly *poly FLAG shaded FLAG transparent FLAG blended	10-23	Changes the shading, transparency, and blending settings of a texture that has already been applied to a polygon

Function Name	Returns	Parameters	Page	Description
WTpoly_settextureuv	FLAG	WTpoly *poly char *bitmap float *uarray float *varray FLAG shaded FLAG transparent	10-13	Applies a texture bitmap stored in the specified file to the specified polygon and allows you to specify the way the texture is mapped onto the polygon
WTpoly_setuv	FLAG	WTpoly *poly float *uarray float *varray	10-32	Changes the way a texture is applied to a polygon's vertices (on polygons that already have a texture applied)
WTpoly_stretchtexture	void	WTpoly *poly float u float v	10-30	Stretches a polygon's texture, with separate scale factors for u and v (horizonal and vertical) texture coordinates
WTpoly_translatetexture	void	WTpoly *poly WTp2 displacement	10-29	Shifts the origin of the texture bitmap on the polygon surface to "slide" the texture around
WTpq_2m4	void	WTpq *pq WTm4 m	25-31	Converts a WTpq structure with position and orientation fields into a WTm4 matrix (a 4 x 4 matrix)
WTpq_copy	[MACRO]	WTpq *pqin WTpq *pqout	25-20	Copies the contents of pqin into pqout
WTpq_frame2frame	void	WTpq *pqin WTpq *frame1 WPpq *frame2 WTpq *pqout	25-36	Takes a specified position and orientation for frame1 and determines the corresponding position and orientation relative to frame2, then places the result in pqout
WTpq_init	[MACRO]	WTpq *pq	25-20	Initializes the specified WTpq structure
WTpq_local2worldframe	void	WTpq *pqin WTpq *frame WTpq *pqout	25-36	Takes the specified local frame's position and orientation, determines its position and orientation relative to the world coordinate frame, then places the result in pqout
WTpq_print	void	WTpq *pq char *string	25-20	Prints to WTMESSAGE_USER the value of the string followed by the value of each element of the WTpq structure

Function Name	Returns	Parameters	Page	Description
WTpq_world2localframe	void	WTpq *pqin WTpq *frame WTpq *pqout	25-36	Takes the specified position and orientation structure specified in relation to the world coordinate frame, determines the corresponding position and orientation relative to the local frame, then places the result into pqout
WTprecision_rawupdate	void	WTsensor *sensor	13-97	Reads the input and puts it in the raw data structure as an absolute euler rotation
WTprecision_update	void	WTsensor *sensor	13-97	Updates the raw data structure, converts it to a quaternion, and relativizes it with the previous record
WTPROJECTION_ASYMMETRIC **WTPROJECTION_GENERAL** **WTPROJECTION_ORTHOGRAPHIC** **WTPROJECTION_SYMMETRIC**	n/a	n/a	17-16	Constants used with WTwindow_setprojection to set the window's projection type
WTproperty_addhandler	FLAG	void *object const char *propname WTeventhandler eh	3-25	Adds an event handler callback to the specified object's propname property
WTproperty_delete	FLAG	void *object const char *propname	3-15	Deletes the user-defined property whose name is propname from a specified object
WTproperty_exists	FLAG	void *object const char *propname	3-16	Returns TRUE if the property whose name is propname exists on a specified object
WTproperty_get	FLAG	void *object const char *propname void *value	3-20	Retrieves the specified object's propname property value
WTproperty_getasstring	char*	void *object const char *propname	3-22	Returns the specified object's propname property value as a string
WTproperty_getd	double	void *object const char *propname	3-21	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_getdata	void*	void *object const char *propname	3-16	Returns the user-defined data field for a property
WTproperty_getdatatype	WTdatatype	void *object const char *propname	3-16	Returns the datatype of the specified object's propname property
WTproperty_getf	float	void *object const char *propname	3-21	Type-specific function used as an alternative to the WTproperty_get function

Function Name	Returns	Parameters	Page	Description
WTproperty_gethandler	WTeventhandler	void *object const char *propname int handlernum	3-26	Returns the handlenum'th handler for the specified object's propname property
WTproperty_geti	int	void *object const char *propname	3-21	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_getp	void*	void *object	3-22	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_getp2	FLAG	void *object const char *propname WTp2 value	3-21	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_getp3	FLAG	void *object const char *propname WTp3 value	3-22	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_getq	FLAG	void *object const char *propname WTq value	3-22	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_gets	char*	void *object const char *propname	3-22	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_getsharegroup	WTsharegroup*	void *object const char *propname int nshare	21-8	Returns the nshare'th sharegroup in which an object's property is shared
WTproperty_getsizeofdata	unsigned int	void *object const char *propname	3-17	Returns the number of bytes used by the specified object's propname property value
WTproperty_gettimesensitive	FLAG	void *object const char *propname	21-9	Returns TRUE if the specified object's property is time sensitive, otherwise it returns FALSE
WTproperty_getui	unsigned int	void *object const char *propname	3-21	Type-specific function used as an alternative to the WTproperty_get function
WTproperty_getupdatefreq	double	void *object const char *propname	21-8	Returns the frequency with which data updates for an object's property occur
WTproperty_islocked	unsigned int	void *object const char *propname	21-10	Returns the id of the client who has a lock on the specified object's propname property, or 0 if the property isn't locked

Function Name	Returns	Parameters	Page	Description
WTproperty_islockedbyme	FLAG	void *object const char *propname	21-11	Returns TRUE if the specified object's propname property is locked by the local client, otherwise it returns FALSE
WTproperty_isshared	FLAG	void *object const char *propname	21-7	Returns TRUE if the object's property is currently shared, otherwise it returns FALSE
WTproperty_lock	FLAG	void *object const char *propname	21-10	Requests a property lock for the local client so that other clients cannot modify the specified object's propname property
WTproperty_new	FLAG	void *object const char *propname WTdatatype dtype	3-15	Creates a new user-defined property whose name is propname and whose data type is dtype for the specified object
WTproperty_numhandlers	int	void *object const char *propname	3-25	Returns the number of handlers assigned to the specified object's propname property
WTproperty_numshares	int	void *object const char *propname	21-7	Returns the number of times an object's property is shared
WTproperty_removeallhandlers	void	void *object const char *propname	3-26	Removes all handlers for the specified object's propname property
WTproperty_removehandler	FLAG	void *object const char *propname WTeventhandler eh	3-25	Removes an event handler callback from the specified object's propname property
WTproperty_sendupdate	void	void *object const char *propname	21-9	Manually queues an update for the specified object's property
WTproperty_set	FLAG	void *object const char *propname void *value	3-17	Sets the specified object's propname property's value to value
WTproperty_setat	FLAG	void *object const char *propname void *value double time	3-19	Sets the specified object's propname property's value at a specified time
WTproperty_setd	FLAG	void *object const char *propname double value	3-18	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_setdata	void	void *object const char *propname void *data	3-16	Sets the user-defined data field for a property

Function Name	Returns	Parameters	Page	Description
WTproperty_setf	FLAG	void *object const char *propname float value	3-18	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_seti	FLAG	void *object const char *propname int value	3-18	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_setp	FLAG	void *object const char *propname void *value	3-19	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_setp2	FLAG	void *object const char *propname WTp2 value	3-18	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_setp3	FLAG	void *object const char *propname WTp3 value	3-19	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_setq	FLAG	void *object const char *propname WTq value	3-19	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_sets	FLAG	void *object const char *propname const char *value	3-19	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_settimesensitive	void	void *object const char *propname FLAG timesensitive	21-9	Makes an object's property time sensitive if the timesensitve parameter is set to TRUE and makes the object's property non-time sensitive if timesensitive is set to FALSE
WTproperty_setui	FLAG	void *object const char *propname unsigned int value	3-18	Type-specific function to use as an alternative to WTproperty_set function
WTproperty_setupdatefreq	void	void *object const char *propname double frequency	21-8	Sets the frequency with which data updates for an object's property will be queued for transmission to a Simulation Server

Function Name	Returns	Parameters	Page	Description
WTproperty_share	FLAG	void *object const char *propname WTsharegroup *shgrp int shareflags	21-5	Shares an object's property under a sharegroup of a Simulation Server
WTproperty_unlock	FLAG	void *object const char *propname	21-10	Requests that a property that is locked by the local client be unlocked
WTproperty_unshare	FLAG	void *object const char *propname WTsharegroup *shgrp FLAG forcedelete	21-7	Unshares an object's property from the specified sharegroup
WTq_2dir	void	WTq q WTp3 dir	25-30	Converts a quaternion into a direction vector
WTq_2dirandtwist	void	WTq q WTp3 dir float *twist	25-30	Converts a quaternion into a direction vector and gets the twist factor from the quaternion apart from the direction
WTq_2euler	void	WTq q WTp3 first WTp3 second	25-29	Extracts euler angles, specified in radians, from the specified unit quaternion
WTq_2eulernear	void	WTq q WTp3 nearp WTp3 euler	25-32	Returns the euler that is closest to the specified euler, corresponding to the specified WTq
WTq_2m3	void	WTq q WTm3 m	25-26	Converts a quaternion to a 3D matrix
WTq_2m4	void	WTq q WTm4 m	25-32	Converts a quaternion to a 4 x4 matrix and places the result in m
WTq_construct	void	WTp3 vec float ang WTq qout	25-17	Composes a quaternion qout from the vector vec and angle ang specified in radians.
WTq_copy	[MACRO]	WTq qin WTq qout	25-15	Copies qin into qout
WTq_dot	[MACRO]	WTq q1 WTq q2	25-18	Returns the dot product of q1 and q2
WTq_equal	[MACRO]	WTq q1 WTq q2	25-16	Tests two quaternions for equality

Function Name	Returns	Parameters	Page	Description
WTq_exact	[MACRO]	WTq q1 WTq q2	25-16	Tests two quaternions for equality
WTq_frame2frame	void	WTq qin WTpq *frame1 WTpq *frame2 WTq qout	25-35	Takes the specified orientation qin relative to frame1 and determines its orientation relative to frame 2, placing the result in quot
WTq_getangle	float	WTq q	25-17	Returns the angle sweep of the quaternion (in radians)
WTq_getvector	void	WTq q WTp3 vec	25-16	Returns the unit vector that is rotated around by the quaternion
WTq_init	[MACRO]	WTq q	25-14	Initializes a quaternion so that q[X] = q[Y] = q[Z] = 0.0; q[W] = 1.0
WTq_interpolate	void	WTq q1 WTq q2 float u WTq qout	25-18	Finds the spherical linear interpolation using the shortest path between quaternion q1 and q2 and places the result in qout
WTq_invert	[MACRO]	WTq qin WTq qout	25-15	Inverts a quaternion
WTq_local2worldframe	void	WTq qin WTpq *frame WTq qout	25-35	Takes a specified orientation for the local frame and determines its orientation relative to the world coordinate frame, then places the result in qout
WTq_mag	[MACRO]	WTq q	25-15	Returns the length of a quaternion vector
WTq_mult	void	WTq q1 WTq q2 WTq qout	25-17	Multiplies quaternion q2 into q1 and puts the result into qout
WTq_multinv	void	WTq q1 WTq q2 WTq qout	25-18	Multiplies q2 into the inverse of q1 and puts the result into qout
WTq_norm	[MACRO]	WTq q	25-16	Normalizes a quaternion (scales each component of the quaternion so the sum of the squares of the components equals one)

Function Name	Returns	Parameters	Page	Description
WTq_print	void	WTq orientation char *string	25-19	Prints to WTMESSAGE_USER the value of the string followed by the value of each element of the WTq structure
WTq_scale	void	WTq qin WTq qout float scale	25-17	Controls the angle sweep or the amount of rotation in a quaternion
WTq_world2localframe	void	WTq qin WTpq *frame WTq qout	25-35	Takes a specified orientation for the world coordinate frame and determines its orientation relative to the local frame, then places the result in qout
WTrealloc	void	void *old size_t newsize	24-10	Similar to realloc; used to allocate shared memory. *Required for SGI MP/MP*
WTRENDER_ALLMODES **WTRENDER_ANTIALIAS** **WTRENDER_BEST** **WTRENDER_DEFAULT** **WTRENDER_GOURAUD** **WTRENDER_LIGHTING** **WTRENDER_NOSHADE** **WTRENDER_PERSPECTIVE** **WTRENDER_SMOOTH** **WTRENDER_TEXTURED** **WTRENDER_WIREFRAME**	n/a	n/a	6-34	Constants used with the functions WTgeometry_setrenderingstyle and WTuniverse_setrenderingstyle to set the rendering style of a geometry or the universe
WTrootnode_new	WTnode *	void	4-39	Creates a new root node (and therefore a new scene graph)
WTrootnode_next	WTnode *	WTnode *rootnode	4-77	Returns a pointer to the next root node in the universe's list of root nodes
WTscreen_getyblank	int	void	2-21	Returns the current value of the screen blanking interval used for field-sequential devices
WTscreen_load	FLAG	char *filename	10-33	Loads an image file to each WTK window
WTscreen_pickpoly	WTpoly *	int numscreen WTp2 pt WTnodepath **nodepath WTp3 p	4-91	Obtains a pointer to the frontmost polygon rendered at the specified 2D screen point of the specified screen

Function Name	Returns	Parameters	Page	Description
WTscreen_setyblank	void	int distance	2-20	Allows you to adjust the vertical blanking interval between the left and right eye images (measured in pixels)
WTSENSOR_DEFAULT	n/a	n/a	13-6	Constant used for sensor driver construction
WTsensor_delete	void	WTsensor *sensor	13-10	Removes a sensor object from the universe's list of sensors; detaches sensor from viewpoints, lights, or objects; calls the sensors close function, deletes the sensors serial port object (if applicable); and frees the memory used by the sensor object.
WTsensor_getangularrate	float	WTsensor *sensor	13-13	Returns the maximum angular rate of change around each axis for a given sensor
WTsensor_getconstraints	short	WTsensor *sensor	13-22	Returns a short describing the constraints currently imposed on the values returned by the sensor
WTsensor_getdata	void	WTsensor *sensor	13-24	Retrieves user-defined data stored within a sensor
WTsensor_getlastrecord	void	WTsensor *sensor WTp3 absolute_p WTq absolute_q	13-25	Retrieves the position and orientation record most recently set with the function WTsensor_setlastrecord and stores them in absolute_p and absolute_q
WTsensor_getmiscdata	int	WTsensor *sensor	13-15	Returns an integer value in which sensor data is stored (such as button press events); defined constants are used to interpret the return value
WTsensor_getname	const char*	WTsensor *sensor	13-23	Returns the name of the specified sensor
WTsensor_getrawdata	void	WTsensor *sensor	13-15	Returns the sensor-specific raw data structure
WTsensor_getrotation	void	WTsensor *sensor WTq rotation	13-14	Retrieves the current rotation record from the sensor and stores it as a quaternion
WTsensor_getsensitivity	float	WTsensor *sensor	13-12	Returns the sensor's sensitivity value
WTsensor_getserial	WTserial *	WTsensor *sensor	13-16	Returns the serial port object associated with a sensor
WTsensor_gettranslation	void	WTsensor *sensor WTp3 translation	13-13	Retrieves the current translation record from the sensor
WTsensor_getunit	short	WTsensor *sensor	13-16	Retrieves the unit number of the specified sensor

Function Name	Returns	Parameters	Page	Description
WTsensor_new	WTsensor *	int (*openfn)(WTsensor*) void (*closefn)(WTsensor*) void (*updatefn)(WTsensor*) WTserial *serial short unit short location	13-7	Creates a new sensor object and adds it to the universe
WTsensor_next	WTsensor *	WTsensor *sensor	13-10	Returns the next sensor object in the list of sensors maintained by the universe
WTsensor_relativizerecord	void	WTsensor *sensor WTp3 absolute_p WTq absolute_q WTp3 relative_p WTq relative_q	13-24	For sensors with absolute position/orientation records, generates corresponding relative records. Returns the change in position and orientation since the last time through the simulation loop
WTsensor_rotate	void	WTsensor *sensor WTq rotation	13-20	Rotates a sensor's coordinate frame
WTsensor_setangularrate	void	WTsensor *sensor float scale	13-12	Sets the scale factor for a sensor's rotation records
WTsensor_setconstraints	void	WTsensor *sensor short c	13-21	Constrains the values returned by a sensor
WTsensor_setdata	void	WTsensor *sensor void *data	13-23	Sets the user-defined data field in a sensor
WTsensor_setlastrecord	void	WTsensor *sensor WTp3 absolute_p WTq absolute_q	13-25	For sensors with absolute position/orientation records, sets the absolute record
WTsensor_setmiscdata	void	WTsensor *sensor int data	13-25	Stores miscellaneous sensor data, like button press events, with the sensor object
WTsensor_setname	void	WTsensor *sensor const char *name	13-23	Sets the name of the specified sensor
WTsensor_setrawdata	void	WTsensor *sensor void *dataptr	13-26	Stores raw sensor data with the sensor object
WTsensor_setrecord	void	WTsensor *sensor WTp3 p WTq q	13-24	Stores the current relative position and orientation record with the sensor
WTsensor_setsensitivity	void	WTsensor *sensor float sensitivity	13-11	Sets the sensitivity value for the sensor

Function Name	Returns	Parameters	Page	Description
WTsensor_setupdatefn	void	WTsensor *sensor void (*updatefn)(WTsensor*))	13-10	Allows you to change a sensor's update function. A sensor object's update function is initially set in the sensor constructor function
WTsepnode_getcullmode	int	WTnode *node	4-57	Returns the specified separator node's culling mode
WTsepnode_new	WTnode *	WTnode *parent	4-41	Creates a separator node and adds it to the scene graph after the last child of the specified parent
WTsepnode_setcullmode	FLAG	WTnode *node int mode	4-57	Sets the specified separator node's culling mode (default is on, WTCULLMODE_ON)
WTserial_delete	void	WTserial *serial	23-2	Frees a serial port object
WTserial_new	WTserial *	char *port int baud char parity int databits int stopbits int buffersize	23-1	Creates a serial port object *Note: parity, databits, stopbits, and buffersize arguments apply to the Windows NT/95 platforms only*
WTserial_ntoread	int	WTserial *serial	23-4	Determines how many characters are waiting to be read at the serial port
WTserial_read	short	WTserial *serial char *data int length FLAG retry	23-3	Reads a string of a specified length (in bytes) from the serial port and returns the number of characters that were actually read
WTserial_write	short	WTserial *serial char *buffer int length	23-4	Writes a string of a given length to a serial port and returns the number of characters that were successfully written
WTsharegroup_delete	FLAG	WTsharegroup *shgrp FLAG forcedelete	21-16	Deletes the specified sharegroup
WTsharegroup_enumerate	unsigned int	WTsharegroup *parent FLAG recursive FLAG properties	21-36	Requests an enumeration of the specified parent sharegroup
WTsharegroup_findchildbyname	WTsharegroup *	WTsharegroup *group const char *name	21-21	Finds an immediate child of group matching name
WTsharegroup_getchild	WTsharegroup*	WTsharegroup *shgrp int childnum	21-20	Returns the childnum'th child sharegroup of the specified sharegroup

Function Name	Returns	Parameters	Page	Description
WTsharegroup_getconnection	WTconnection*	WTsharegroup *shgrp	21-18	Returns the WTconnection on which the specified sharegroup exists
WTsharegroup_getdata	void*	WTsharegroup *shgrp	21-18	Returns the user-defined data field on a sharegroup
WTsharegroup_getname	char*	WTsharegroup *shgrp	21-18	Returns the name of the specified sharegroup
WTsharegroup_getparent	WTsharegroup*	WTsharegroup *shgrp	21-20	Returns the parent sharegroup of the specified sharegroup
WTsharegroup_getproperty	char*	WTsharegroup *shgrp int propertynum void **object	21-21	Returns the name of the propertynum'th property of a sharegroup
WTsharegroup_islocked	unsigned int	WTsharegroup *shgrp	21-19	Returns the id of the client which has a lock on the specified sharegroup, or 0 if the sharegroup isn't locked
WTsharegroup_islockedbyme	FLAG	WTsharegroup *shgrp	21-19	Returns TRUE if the specified sharegroup is locked by the local client, otherwise it returns FALSE
WTsharegroup_isshared	FLAG	WTsharegroup *group	21-17	Returns the share status of the WTsharegroup group
WTsharegroup_lock	FLAG	WTsharegroup *shgrp	21-19	Requests a sharegroup lock for the local client so that other clients cannot modify this sharegroup's subtree
WTsharegroup_new	WTsharegroup*	const char *name WTsharegroup *parent int shareflags	21-15	Creates a new sharegroup named name as a child of the specified parent sharegroup
WTsharegroup_numchildren	int	WTsharegroup *shgrp	21-20	Returns the number of sharegroups that are direct children of the specified sharegroup
WTsharegroup_numproperties	int	WTsharegroup *shgrp	21-21	Returns the number of properties of the specified sharegroup
WTsharegroup_print	void	WTsharegroup *shgrp FLAG children FLAG properties	21-18	Prints information about the specified sharegroup
WTsharegroup_registerinterest	void	WTsharegroup *shgrp FLAG interested	21-20	Registers or unregisters interest in the specified sharegroup for the local client
WTsharegroup_setdata	void	WTsharegroup *shgrp void *data	21-17	Sets the user-defined data field on a sharegroup
WTsharegroup_share	FLAG	WTsharegroup *group int shareflags	21-17	Attempts to share an unshared WTsharegroup

Function Name	Returns	Parameters	Page	Description
WTsharegroup_unlock	FLAG	WTsharegroup *shgrp	21-19	Requests a sharegroup lock for the local client so that other clients cannot modify this sharegroup's subtree
WTSOUND_DOPPLER **WTSOUND_FBPAN** **WTSOUND_LOOPS** **WTSOUND_LRPAN** **WTSOUND_OFF** **WTSOUND_ON** **WTSOUND_PITCH** **WTSOUND_PLAYRATE** **WTSOUND_PRIORITY** **WTSOUND_SPATIALIZE** **WTSOUND_VOLUME**	n/a	n/a	20-14	Constants used with WTsound_setparam to set the parameters for a sound
WTsound_delete	FLAG	WTsound *sound	20-10	Deletes a sound (returns TRUE if successful)
WTsound_getdata	void	WTsound *sound	20-16	Retrieves the user-defined data from a sound
WTsound_getdonefn	PFVS	WTsound *sound	20-17	Retrieves the function that will be called when the sound is done playing
WTsound_getname	char *	WTsound *sound	20-15	Returns a pointer to the filename of a sound
WTsound_getnodepath	WTnodepath *	WTsound *sound	20-18	Retrieves the source of a sound
WTsound_getparam	float	WTsound *sound int param	20-15	Retrieves the parameters for a sound
WTsound_getposition	void	WTsound *sound WTp3 position	20-17	Returns the current position setting of a sound
WTsound_isplaying	FLAG	WTsound *sound	20-12	Determine if a sound is currently playing
WTsound_load	WTsound *	WTsounddevice *device char *source	20-10	Creates a new sound from a source and returns a pointer to the new sound
WTsound_next	WTsound *	WTsound *sound	20-11	Iterates through a list of sounds currently loaded by a device
WTsound_play	FLAG	WTsound *sound	20-11	Cues a sound to begin playing
WTsound_setdata	void	WTsound *sound void *data	20-15	Attaches user-definable data to a sound
WTsound_setdonefn	void	WTsound *sound PFVS done	20-16	Sets a function to call when the sound is done playing

Function Name	Returns	Parameters	Page	Description
WTsound_setnodepath	FLAG	WTsound *sound WTnodepath *npath	20-18	Assigns a sound to a source specified by a node path
WTsound_setparam	void	WTsound *sound int param float value	20-12	Sets various parameters for a sound
WTsound_setposition	void	WTsound *sound WTp3 position	20-17	Sets a sound's position in 3D space
WTsound_stop	void	WTsound *sound	20-10	Stops a currently playing sound
WTSOUNDDEVICE_ABSORBDIST **WTSOUNDDEVICE_OFF** **WTSOUNDDEVICE_ON** **WTSOUNDDEVICE_OUTPUT** **WTSOUNDDEVICE_ROLLOFF** **WTSOUNDDEVICE_ROLLOFFEXP** **WTSOUNDDEVICE_SPATIALIZE**	n/a	n/a	C-19	Constants used with WTsounddevice_setparam to set the parameters for a sound device
WTsounddevice_close	FLAG	WTsounddevice *device	20-3	Closes an audio device and deletes all sounds associated with it
WTsounddevice_getdata	void	WTsounddevice *device	20-9	Retrieves the user-defined data from a sound device
WTsounddevice_getlistener	WTviewpoint *	WTsounddevice *device	20-9	Gets the viewpoint that is being used as a listener by a device
WTsounddevice_numplayable	int	WTsounddevice *device	20-4	Gets the number of sources available for a device (the number of sounds that can be played simultaneously)
WTsounddevice_getparam	float	WTsounddevice *device int param	20-8	Returns a parameter for a sound device
WTsounddevice_getsounds	WTsound *	WTsounddevice *device	20-4	Gets a pointer to a list of sounds currently loaded by a device
WTsounddevice_name2sound	WTsound *	WTsounddevice *device char *name	20-5	Gets a sound by its name
WTsounddevice_open	WTsounddevice *	int type int nplayable WTviewpoint *listener	20-3	Opens an audio device and assigns the specified viewpoint as the listener
WTsounddevice_setdata	void	WTsounddevice *device void *data	20-8	Attaches user-defined data to a sound device

Function Name	Returns	Parameters	Page	Description
WTsounddevice_setlistener	FLAG	WTsounddevice *device WTviewpoint *viewpoint	20-9	Specifies a viewpoint as a listener
WTsounddevice_setparam	void	WTsounddevice *device int param float value	20-5	Sets various parameters for a sound device
WTsounddevice_update	void	WTsounddevice *device	20-4	Updates listener and sound positions/orientations
WTSOURCE_PATH **WTSOURCE_SENSOR**	n/a	n/a	C-7	Constants used with motion link functions representing a source type for the motion links
WTSPACEBALL_BUTTON1 (through 8) **WTSPACEBALL_PICKBUTTON** **WTSPACEBALL_BUTTONS** **WTSPACEBALL_BUTTON1_DOWN (through 8)** **WTSPACEBALL_PICKBUTTON_DOWN** **WTSPACEBALL_BUTTONS_DOWN** **WTSPACEBALL_BUTTON1_UP (through 8)** **WTSPACEBALL_PICKBUTTON_UP** **WTSPACEBALL_BUTTONS_UP**		n/a	C-16	Constants to retrieve event data from a Spaceball device
WTspaceball_dominant	void	WTsensor *sensor	13-102	Restricts Spaceball movement to only one axis
WTspaceball_new	WTsensor *	port	13-13	Constructor macro for a Spaceball device
WTspaceball_rezero	void	WTsensor *spaceball	13-103	Redefines Spaceball's center value at its current position
WTspaceball_update	void	WTsensor *sensor	13-102	Updates a Spaceball device's location
WTSPACEBALLSC_BUTTONS **WTSPACEBALLSC_BUTTONx**	n/a	n/a	13-103	Constants used with the Spaceball Spacecontrol device
WTspaceballSC_dominant	void	WTsensor *sensor	13-106	Restricts Spaceball Spacecontrol movement to only one axis
WTspaceballSC_rezero	void	WTsensor *spaceball	13-107	Redefines Spaceball Spacecontrol's center value at its current position
WTspaceballSC_update	void	WTsensor *sensor	13-106	Updates a Spaceball Spacecontrol device's location
WTSPACECONTROL_BUTTON1 (through 8) **WTSPACECONTROL_BUTTONA**		n/a	13-84	Set of constants to retrieve event data from a Space Control Mouse (Logitech Magellan) device

Function Name	Returns	Parameters	Page	Description
WTspacecontrol_rawupdate	void	WTsensor *sensor	13-83	Obtains raw position, orientation, and button-press data for a Space Control Mouse (Logitech Magellan) device
WTspacecontrol_update	void	WTsensor *sensor	13-83	Updates a Space Control Mouse (Logitech Magellan) device's location
WTswitchnode_getwhichchild	int	WTnode *node	4-58	Returns the index of the current child being processed
WTswitchnode_new	WTnode *	WTnode *parent	4-42	Creates a switch node and adds it to the scene graph after the last child of the specified parent
WTswitchnode_setwhichchild	FLAG	WTnode *node int which	4-57	Allows you to specify which child of a switch node is processed (default is none)
WTTARGET_MOVABLE **WTTARGET_NODEPATH** **WTTARGET_TRANSFORM** **WTTARGET_VIEWPOINT**	n/a	n/a	C-7	Constants used with motion link functions representing the type of targets for the motion link
WTtask_add	FLAG	WTtask *task	11-4	Adds a task back to the simulation
WTtask_delete	FLAG	WTtask *task	11-4	Deletes a task and frees the memory associated with the WTtask object
WTtask_getfunction	WTtask_function	WTtask *task	11-5	Returns a task's function
WTtask_getpriority	float	WTtask *task	11-5	Returns the priority of a task
WTtask_new	WTtask *	void *objptr WTtask_function fptr float priority	11-2	Creates a new WTtask and activates it, so it is automatically executed as the simulation runs
WTtask_remove	FLAG	WTtask *task	11-4	Removes a task from a simulation without deleting the WTtask
WTtask_setpriority	FLAG	WTtask *task float priority	11-5	Sets the priority of a task (lower numbered tasks are executed first)
WTtexture_cache	FLAG	char *bitmap FLAG enable	10-17	Controls the caching of a texture
WTtexture_getfilter	FLAG	char *bitmap int *magfilter int *minfilter	10-27	Returns the magnification and minification filter values of the specified texture bitmap
WTtexture_getmemory	int	void	10-18	Returns the amount of texture memory (in bytes) used by the application

Function Name	Returns	Parameters	Page	Description
WTtexture_iscached	FLAG	char *bitmap	10-18	Returns the caching state of a texture
WTtexture_load	unsigned char *	char *bitmap int *width int *height	10-17	Reads in a texture bitmap and returns both a pointer to the image file and the height and width of the texture
WTtexture_replace	FLAG	char *bitmap int format int width int height unsigned char *image	10-16	Dynamically replaces the image associated with a texture bitmap used for texturing polygons
WTtexture_setfilter	FLAG	char *bitmap int magfilter int minfilter	10-25	Sets the magnification and minification filters of the texture bitmap
WTtime_getcurrent	double	void	3-27	Returns the current Greenwich mean time (GMT) in seconds
WTtime_getcurrentlocal	double	void	3-27	Returns the current (local timezone) time in seconds
WTtime_getcurrentmsec	unsigned short	void	3-28	Returns the number of milliseconds beyond the current second in Greenwich mean time (GMT)
WTtime_getcurrentmseclocal	unsigned short	void	3-28	Returns the number of milliseconds beyond the current second in local time
WTtime_getcurrentsec	int	void	3-27	Returns the whole number of seconds from 01-01-70 Greenwich mean time (GMT)
WTtime_getcurrentseclocal	int	void	3-27	Returns the whole number of seconds form 01-01-70 in the local timezone
WTtime_getdouble	double	int sec unsigned short msec	3-28	Returns the seconds part of a ‘double’ time value
WTtime_getmsec	unsigned short	double time	3-28	Returns the milliseconds part of a ‘double’ time value
WTtime_getsec	int	double time	3-28	Returns the seconds part of a ‘double’ time value
WTtime_update	void	void	3-27	Used to update the time value returned from WTtime_getcurrent if not in a WTuniverse_go loop
WTui_check	int	WTui *ui int flag	18-35	Enables or disables the checkmark on a checkbutton or menu item

Function Name	Returns	Parameters	Page	Description
WTuicheckbutton_new	WTui*	WTui *parent char *label	18-16	Creates a checkbutton object and returns a pointer to it
WTui_delete	void	WTui *ui	18-40	Deletes the UI object and its children
WTui_deleteitem	void	WTui *ui int item	18-33	Deletes a text string from an existing scrolled list object
WTUI_EDITABLE **WTUI_NOTEDITABLE**	n/a	n/a	C-20	Constants used with WTuiscrolledtext_new
WTui_enable	FLAG	WTui *ui FLAG flag	18-34	Enables or disables (dims or undims) the specified menu item or radiobox
WTUI_FILE **WTUI_TEXT**	n/a	n/a	C-20	Constants used with WTuilabel_new
WTuifileselection_new	WTui *	WTui *parent char *title char *file char *pat	18-13	Creates a file selection box (modal) and returns a pointer to it
WTuiform_new	WTui *	WTui *parent char *title ...	18-6	Creates a form (container) object and returns a pointer to it
WTui_getcallback	UICBP *	WTui *ui int eventtype	18-41	Returns a pointer to the callback function associated with the specified UI object
WTui_getid	WTwinidtype	WTui *ui	18-37	Returns a platform-specific ID. On Windows platforms, the return type is HWND. On Unix platforms, the return type is Widget
WTui_getitemtext	const char*	WTui *ui int item	18-32	Returns the text string associated with the item numbered element of the specified scrolled list
WTui_getnumitems	int	WTui *ui	18-32	Returns the number of items contained in the specified scrolled list or radiobox UI object
WTui_getparent	WTui *	WTui *ui	18-40	Returns a pointer to the UI object's parent
WTui_getposition	void	WTui *ui int *left int *top int *width int *height	18-36	Returns the scaled position and size of a UI object

Function Name	Returns	Parameters	Page	Description
WTui_getscalefactor	void	float *x float *y	18-29	Retrieves the scaling factors used by WTK to position UI objects
WTui_getselecteditem	int	WTui *ui	18-31	Retrieves the position of the selected text string in a scrolled list object or the position of the selected togglebutton in a radiobox object
WTui_gettext	char *	WTui *ui	18-30	Returns the text of the specified UI object or NULL if the UI object is not a text UI
WTui_go	void	WTui *toplevel FLAG startwtk	18-12	Enters the main platform-specific application loop and continuously processes events and messages (if startwtk is TRUE, the WTK simulation loop is automatically started)
WTui_init	WTui *	int *argc char **argv	18-5	Performs platform-specific initialization and creates the top level application shell
WTui_ischecked	int	WTui *ui	18-35	Returns TRUE if the specified checkbutton's menu item's checkmark is checked and FALSE if it is not checked
WTui_isconsolevisible	int	void	18-41	Returns TRUE if the console window is visible, and FALSE if it is not visible
WTui_isenabled	FLAG	WTui *ui	18-34	Returns TRUE if the menu item or radiobox is enabled, or FALSE otherwise
WTui_insertitem	void	WTui *ui int index char *text	18-33	Inserts a new text string into an existing scrolled list object
WTui_iswtkrunning	FLAG	void	18-38	Returns TRUE if WTK is running
WTuilabel_new	WTui *	WTui *parent char *label FLAG labeltype ...	18-17	Creates a simple, static text label (non-editable) and returns a pointer to it
WTui_manage	void	WTui *ui	18-11	Manages a form or frame object (after all the children of the object have been created)
WTuimenubar_new	WTui *	WTui *parent	18-24	Creates a new menu bar and returns a pointer to it

Function Name	Returns	Parameters	Page	Description
WTuimenuitem_new	WTui *	WTui *parent char *label	18-26	Creates a new menu item button and returns a pointer to it
WTuimenupopup_new	WTui *	WTui *parent char *label	18-25	Creates a new menu pop-up button and returns a pointer to it
WTuimessagebox_new	WTui *	WTui *parent char *message char *title	18-15	Creates a modal message box and returns a pointer to it
WTuipushbutton_new	WTui *	WTui *parent char *label ...	18-18	Creates a pushbutton object and returns a pointer to it
WTuiradiobox_new	WTui *	WTui *parent int num char **labels ...	18-18	Creates a radiobox object and returns a pointer to it
WTuiscale_new	WTui *	WTui *parent char *label int minimum int maximum int decimal_points int value ...	18-19	Creates a scale object and returns a pointer to it
WTuiscrolledlist_new	WTui *	WTui *parent char *label char *items[] int nitems ...	18-21	Creates a scrolled list of strings and returns a pointer to it
WTuiscrolledtext_new	WTui *	WTui *parent char *text FLAG editable ...	18-23	Creates an editable text box with a scroll bar and returns a pointer to it
WTui_setcallback	void	WTui *ui int eventtype UICBP cb void *cbdata	18-8	Sets the callback handler function for a UI object
WTui_setitemtext	void	WTui *ui char *text int item	18-32	Assigns the text string to the item numbered element of the specified scroll list

Function Name	Returns	Parameters	Page	Description
WTui_setposition	void	WTui *ui int left int top int width int height	18-36	Sets the scaled position and size of the UI object
WTui_setscalefactor	void	float x float y	18-29	Adjusts the scaling factors that WTK uses when positioning a UI object
WTui_setselecteditem	void	WTui *ui int item	18-31	Selects the item numbered text string in a scrolled list object or the item numbered togglebutton in a radiobox object
WTui_settext	FLAG	WTui *ui char *text	18-29	Sets the text of a menu item, scrolled list, scrolled text, text input dialog box, or text field
WTui_settoolbarcallback	void	WTui *ui int id int eventtype UICBP cb void *cbdata	18-9	Sets a callback handler function for a particular button on the tool bar
WTui_showconsole	void	int flag	18-41	Used to show or hide the console window
WTuitextfield_new	WTui *	WTui *parent char *text ...	18-24	Creates a new editable text-field object and returns a pointer to it
WTuitextinput_new	WTui *	WTui *parent char *title	18-15	Creates a simple input box composed of label and text fields and returns a pointer to it
WTuitoolbar_new	WTui *	WTui *parent int items char **bitmap_files	18-28	Creates a new tool bar and returns a pointer to it
WTui_unmanage	void	WTui *ui	18-40	Hides a UI object
WTui_wtkstart	void	void	18-38	Starts the WTK simulation loop from a callback handler function
WTui_wtkstop	void	void	18-38	Stops the WTK simulation loop from a callback handler function or from the universe action function

Function Name	Returns	Parameters	Page	Description
WTuiwtkwindow_new	WTwindow *	WTui *form int window_config	18-11	Integrates a WTK rendering window with a user-defined GUI window
WTuniverse_avgframerate	float	int samples	2-24	Returns the number of frames per second, averaged over a specified number of updates
WTuniverse_delete	void	void	2-5	Frees all the objects in the universe
WTuniverse_deleteconnections	void	void	21-28	Deletes all connections made by the local client
WTuniverse_deletelink	void	void *source void *target	2-17	Deletes any motion link connecting the specified source and target objects
WTuniverse_findnodebyname	WTnode *	char *name int num	4-49	Finds the numbered occurrence of a specified node
WTuniverse_framecount	int	void	2-23	Returns the number of frames drawn since the last clock reset or since WTuniverse_new was called
WTuniverse_framerate	float	void	2-23	Returns the number of frames per second at which the universe is currently running
WTuniverse_getbases	WTbase*	void	3-8	Returns a pointer to the first WTbase object in the universe's list of WTbase objects
WTuniverse_getbgrgb	void	unsigned char *r unsigned char *g unsigned char *b	2-21	Returns the current background color of the universe
WTuniverse_getconnections	WTconnection*	void	21-27	Returns a pointer to the first connection in the universe's list of connections for the local client
WTuniverse_getcurrscridx	int	void	2-14	Returns the number of the screen that has input focus
WTuniverse_getcurrwindow	WTwindow *	void	2-14	Returns a pointer to the window that has input focus
WTuniverse_getcurrwinidx	[platform specific]	void	2-14	Returns a system-specific window ID. On Windows platform, the return type is HWND. On Unix platforms, the return type is Window
WTuniverse_geteventorder	short *	void	2-10	Returns the order of events currently set to occur in the simulation loop
WTuniverse_getinitialview	void	WTpq *position	2-16	Extracts the position and orientation information from the most recently loaded model and places it in pq
WTuniverse_getmotionlinks	WTmotionlink *	void	2-17	Returns a pointer to the first motion link in a universe's list of motion links

Function Name	Returns	Parameters	Page	Description
WTuniverse_getoption	int	int option	2-27	Returns the value of the specified option
WTuniverse_getpaths	WTpath *	void	2-13	Returns a pointer to a list of all paths in the universe
WTuniverse_getrendering	FLAG	void	2-20	Returns the current value of the universe's rendering style
WTuniverse_getrootnodes	WTnode *	void	2-17	Returns a pointer to the first node in the universe's list of root nodes
WTuniverse_getsensors	WTsensor *	void	2-13	Returns a pointer to a list of all sensors currently in the universe
WTuniverse_getsubfaceoffset	float	void	2-22	Returns the value of the subface offset
WTuniverse_gettaskbypointer	WTtask *	void *pointer int numtask	11-6	Obtains the WTtask associated with an object pointer
WTuniverse_getviewpoints	WTviewpoint *	void	2-15	Returns a pointer to a list of all viewpoints in the universe
WTuniverse_getwindows	WTwindow *	void	2-13	Returns a pointer to a list of all windows currently in the universe
WTuniverse_go	void	void	2-7	Starts the main simulation loop
WTuniverse_go1	void	void	2-7	Starts the main simulation loop for one loop only
WTuniverse_new	void	int display_config int window_config	2-2	Initializes the universe state and the graphics device used to view the simulation
WTuniverse_processevents	void	void		Processes all events in the universe
WTuniverse_ready	void	void	2-6	Prepares the application for entry into the main simulation loop
WTuniverse_resetframecount	void	void	2-23	Resets the universe frame count
WTuniverse_resettime	void	void	2-23	Resets the universe time
WTuniverse_setactions	void	void (*actionfn)(void)	2-12	Sets the universe action function
WTuniverse_setbboxrgb	void	float r float g float b	2-21	Sets the color of all active bounding boxes in the universe (default is white)
WTuniverse_setbgrgb	void	unsigned char r unsigned char g unsigned char b	2-21	Sets the background color of the universe (0 to 255 are valid values)

Function Name	Returns	Parameters	Page	Description
WTuniverse_seteventorder	FLAG	short nevents short *events	2-9	Allows you to change the order of activity in the simulation loop
WTuniverse_setoption	void	int option int value	2-24	Sets certain global parameters
WTuniverse_setrendering	void	FLAG style	2-18	Specifies the universe's rendering style
WTuniverse_setsubfaceoffset	void	float val	2-22	Sets the distance by which a subface is offset from its parent polygon (default is 0.65)
WTuniverse_setviewpoint	WTnode *	WTviewpoint *viewpoint	2-15	Assigns the specified viewpoint as the universe's current viewpoint
WTuniverse_stop	void	void	2-8	Exits the main simulation loop
WTuniverse_time	float	void	2-22	Returns the number of seconds the simulation has been running since WTuniverse_new or WTuniverse_resettime was called
WTuniverse_updateconnections	void	void	21-29	Updates all connections of the local client
WTurl_download	char *	char *url char *localfile	4-47	Copies a file from an http server to a file on the local machine
WTvalue_tostring	char *	WTdatatype dtype void *value	3-23	Returns the data value of type dtype as a string
WTvertex_next	WTvertex *	WTvertex *vertex	6-33	Returns the next vertex in a geometry's list of vertices
WTviewpoint_addsensor	void	WTviewpoint *viewpoint WTsensor *sensor	16-7	Attaches a sensor to a viewpoint
WTviewpoint_alignaxis	void	WTviewpoint *viewpoint short axis WTp3 dir	16-14	Rotates the viewpoint so the specified axis aligns with the specified direction
WTviewpoint_copy	WTviewpoint *	WTviewpoint *old_viewpoint	16-5	Copies an existing viewpoint and returns a pointer to the new viewpoint
WTviewpoint_delete	void	WTviewpoint *viewpoint	16-5	Deletes the specified viewpoint
WTviewpoint_getaspect	float	WTviewpoint *viewpoint	16-19	Returns the viewpoint's current aspect ratio
WTviewpoint_getaxis	void	WTviewpoint *viewpoint short axis WTp3 vector	16-14	Returns the unit vector in the direction of the specified viewpoint's axis (X, Y, or Z)

Function Name	Returns	Parameters	Page	Description
WTviewpoint_getconvdistance	float	WTviewpoint *viewpoint	16-23	Returns the value of the viewpoint's convergence distance parameter
WTviewpoint_getconvergence	short	WTviewpoint *viewpoint	16-21	Returns the viewpoint's stereo convergence value in screen pixel units
WTviewpoint_getdata	void	WTviewpoint *viewpoint	16-26	Retrieves the user-defined data stored within a viewpoint
WTviewpoint_getdirection	void	WTviewpoint *viewpoint WTp3 dir	16-13	Returns the direction of the viewpoint
WTviewpoint_getdirectionframe	void	WTviewpoint *view WTp3 dir WTpq *frame	16-18	Returns the direction of the viewpoint relative to the specified frame
WTviewpoint_getframe	void	WTviewpoint *viewpoint WTpq *frame	16-15	Returns the specified viewpoint's position and orientation
WTviewpoint_getlastposition	void	WTviewpoint *view WTp3 pos	16-9	Gets the specified viewpoint's position in the last frame
WTviewpoint_getlastorientation	void	WTviewpoint *view WTq q	16-11	Gets a viewpoint's orientation in the last frame
WTviewpoint_getorientation	void	WTviewpoint *viewpoint WTq q	16-11	Returns the orientation of the viewpoint, specified as a quaternion
WTviewpoint_getorientationframe	void	WTviewpoint *view WTq q WTpq *frame	16-16	Returns the orientation of the viewpoint relative to the specified frame, specified as a quaternion
WTviewpoint_getparallax	float	WTviewpoint *viewpoint	16-20	Returns the viewpoint's parallax value
WTviewpoint_getposition	void	WTviewpoint *viewpoint WTp3 p	16-8	Returns the 3D position of the viewpoint
WTviewpoint_getpositionframe	void	WTviewpoint *view WTp3 pos WTpq *frame	16-15	Returns the 3D position of the viewpoint relative to the specified frame
WTviewpoint_intersectpoly	FLAG	WTviewpoint *vpoint WTpoly *poly WTnodepath *npath float distance	4-89	Tests whether the specified viewpoint intersects the specified polygon as a result of the viewpoint's motion in the current frame

Function Name	Returns	Parameters	Page	Description
WTviewpoint_local2world	void	WTviewpoint *viewpoint WTp3 pin WTp3 pout	16-24	Takes a 3D point in the coordinate frame of the specified viewpoint and determines its position relative to the world coordinate frame
WTviewpoint_move	void	WTviewpoint *viewpoint WTpq *moveby short frame	16-12	Moves a viewpoint by the specified translation and rotation values in the local, world, or viewpoint frame
WTviewpoint_moveframe	void	WTviewpoint *view WTpq *pq WTpq *frame	16-17	Moves a viewpoint by the specified translation and rotation values in the specified frame. It is like WTviewpoint_move but takes an additional argument frame, which can be any coordinate frame (i.e., the specified WTpq). This is a relative move, compared to WTmovetoframe, which is an absolute move.
WTviewpoint_moveto	void	WTviewpoint *view WTpq *newviewat	16-13	Moves a viewpoint to the specified (absolute) position and orientation. It is an absolute move, compared to WTviewpoint_move, which is a relative move.
WTviewpoint_movetoframe	void	WTviewpoint *view WTpq *pq WTpq *frame	16-17	Moves a viewpoint to the specified position and orientation in the specified frame. It is like WTviewpoint_moveto but takes an additional argument frame, which can be any coordinate frame (i.e., the specified WTpq). This is an absolute move, compared to WTmoveframe, which is a relative move.
WTviewpoint_new	WTviewpoint *	void	16-3	Creates a new viewpoint object and returns a pointer to it
WTviewpoint_next	WTviewpoint *	WTviewpoint *viewpoint	16-5	Returns the next viewpoint in the universe's list of viewpoints
WTviewpoint_removesensor	void	WTviewpoint *viewpoint WTsensor *sensor	16-8	Detaches a sensor from a viewpoint object
WTviewpoint_rotate	void	WTviewpoint *viewpoint short axis float angle short frame	16-12	Rotates a viewpoint on a specified axis around the viewpoint's position in the local, world, or viewpoint frame

Function Name	Returns	Parameters	Page	Description
WTviewpoint_rotateframe	void	WTviewpoint *view short axis float angle WTpq *frame	16-16	Rotates a viewpoint around a given axis around the viewpoint's position in the specified frame, which can be any coordinate frame (i.e., the specified WTpq)
WTviewpoint_setaspect	void	WTviewpoint *viewpoint float aspect	16-18	Sets the viewpoint's aspect ratio
WTviewpoint_setconvdistance	void	WTviewpoint *viewpoint float val	16-22	Sets the convergence distance of the specified viewpoint
WTviewpoint_setconvergence	void	WTviewpoint *viewpoint short convergence	16-21	Sets the horizonal offset (in pixels) that is applied to the left and right eye images
WTviewpoint_setdata	void	WTviewpoint *viewpoint void *data	16-25	Sets the user-defined data field in a viewpoint
WTviewpoint_setdirection	void	WTviewpoint *viewpoint WTp3 dir	16-13	Sets the viewpoint to a specified view direction
WTviewpoint_setdirectionframe	void	WTviewpoint *viewpoint WTp3 dir WTpq *frame	16-17	Rotates the viewpoint to a specified view direction in the specified frame
WTviewpoint_setorientation	void	WTviewpoint *viewpoint WTq q	16-11	Sets the viewpoint's orientation to the specified quaternion
WTviewpoint_setorientationframe	void	WTviewpoint *view WTq q WTpq *frame	16-16	Sets the viewpoint's orientation in the specified frame to the specified quaternion
WTviewpoint_setparallax	void	WTviewpoint *viewpoint float parallax	16-19	Sets the parallax value for stereo viewing
WTviewpoint_setposition	void	WTviewpoint *viewpoint WTp3 p	16-8	Moves the viewpoint to the specified 3D position
WTviewpoint_setpositionframe	void	WTviewpoint *view WTp3 pos WTpq *frame	16-15	Moves the viewpoint to the specified 3D position in the specified frame
WTviewpoint_translate	void	WTviewpoint *viewpoint WTp3 p short frame	16-9	Translates a viewpoint by the specified vector in the local, world, or viewpoint frame

Function Name	Returns	Parameters	Page	Description
WTviewpoint_translateframe	void	WTviewpoint *view WTp3 p WTpq *frame	16-15	Translates a viewpoint by the specified vector in the specified frame
WTviewpoint_world2local	void	WTviewpoint *viewpoint WTp3 pin WTp3 pout	16-24	Takes the specified 3D point in the world coordinate frame and determines its location in relation to the specified viewpoint's reference frame
WTvrml_seturl	void	char *basepath	4-62	Sets the URL base path, so that relative pathnames can be specified using WTanchornode_setlocation and WTinlinenode_setlocation
WTwarning	void	char *format	24-6	Used to print warning messages (like the C function printf) from the WTK application to the console or file
WTWINDOW_DEFAULT **WTWINDOW_INTERLACEDEVENODD** **WTWINDOW_INTERLACEDODDEVEN** **WTWINDOW_NOBORDER** **WTWINDOW_SCREENn** **WTWINDOW_RBSTEREO** **WTWINDOW_STEREO** **WTWINDOW_STEREOVSPLIT**	n/a	n/a	2-3	Constants used to set window characteristics: **DEFAULT** = window with border **INTERLACEDEVENODD** = interlaced stereo **INTERLACEDODDEVEN** = interlaced stereo **NOBORDER** = window without border **SCREENn** = screen number 1 to 8 (for MP/MP) **RBSTEREO** = red/blue stereo display **STEREO** = for stereo window **STEREOVSPLIT** = vertically split stereo window
WTwindow_delete	void	WTwindow *window	17-7	Deletes a WTK window object
WTwindow_draw2Dcircle	void	WTwindow *window float xc float yc float radius int mode	19-3	Draws a circle whose center is specified by the xc, yc parameters. The mode indicates whether the circle is filled or hollow
WTwindow_draw2Dline	void	WTwindow *window float x1 float y1 float x2 float y2	19-4	Draws a line between the point specified by x1 and y1 to the point specified by x2 and y2
WTwindow_draw2Dpoint	void	WTwindow *window float x float y	19-4	Draws a point at the coordinates specified by the x and y values

Function Name	Returns	Parameters	Page	Description
WTwindow_draw2Drectangle	void	WTwindow *window float x1 float y1 float x2 float y2 int mode	19-3	Draws a rectangle whose bottom left point is the x1, y1 values and whose upper right point is specified by the x2, y2 values. The mode parameter indicates whether the rectangle is filled or hollow
WTwindow_draw2Dtext	void	WTwindow *window float x float y char *text	19-7	Draws the specified text string at the x, y coordinates of the specified window
WTwindow_draw2Dtexture	void	WTwindow *window char *bitmapname FLAG transparent WTp2 *xyarray WTp2 *uvarray	19-5	Drapes the specified texture bitmap onto the specified 2D polygon represented by the sequence of coordinates defined by xyarray and whose texture coordinates are contained in uvarray
WTwindow_draw3Dlines	void	WTwindow *window WTp3 *pts int numpts FLAG style	19-10	Draws a set of lines between the specified points. Style indicates whether to draw line segments, a connected series of line segments, or a closed series of line segments
WTwindow_draw3Dpoints	void	WTwindow *window WTp3 *pts int numpts	19-10	Draws a set of points at the specified coordinates
WTwindow_enable	void	WTwindow *window FLAG enable	17-9	Allows you to enable or disable rendering to a specified window (by default, windows are enabled)
WTwindow_get2Dtextextents	void	WTwindow *window char *string float *width float *height	19-7	Returns the width and height of the specified text string's extents
WTwindow_getbgrgb	void	WTwindow *window unsigned char *r unsigned char *g unsigned char *b	17-22	Obtains the background color of the specified window
WTwindow_getdata	void	WTwindow *window	17-28	Retrieves a pointer to the user-defined data stored within a window

Function Name	Returns	Parameters	Page	Description
WTwindow_geteye	short	WTwindow *window	17-12	Determines which eye the window's viewpoint is set to display
WTwindow_gethithervalue	float	WTwindow *window	17-18	Returns the window's hither clipping value
WTwindow_getidx	[platform specific]	WTwindow *window	17-29	Returns the system-specific window ID for the specified window. On Windows platform, the return type is HWND. On Unix platforms, the return type is Window
WTwindow_getimage	FLAG	WTwindow *window int x int y int pixels int scanlines unsigned char *image	10-33	Gets an image from the specified window
WTwindow_getname	const char *	WTwindow *win	17-27	Returns the name of the specified window
WTwindow_getparams	void	WTwindow *window FLAG eye float *left float *right float *bottom float *top float *near float *far	17-17	Obtains the current window parameters describing the viewing frustum
WTwindow_getposition	void	WTwindow *window int *x0 int *y0 int *width int *height	17-10	Returns the location and size of a window
WTwindow_getprojection	int	WTwindow *window	17-16	Returns the projection type for the specified window
WTwindow_getray	FLAG	WTwindow *window WTp2 point WTp2 rayorigin WTp3 ray	17-21	Determines the ray that emanates from the view position, which passes through the specified point
WTwindow_getrootnode	WTnode *	WTwindow *window	17-9	Returns the root node of the scene graph associated with the specified window
WTwindow_getscreen	int	WTwindow *window	17-13	Returns the screen number upon which the window appears

Function Name	Returns	Parameters	Page	Description
WTwindow_getviewangle	float	WTwindow *window	17-20	Returns the window's view angle in radians
WTwindow_getviewpoint	WTviewpoint *	WTwindow *window	17-11	Returns the viewpoint currently set for the specified window
WTwindow_getviewpoint2	WTviewpoint *	WTwindow *window	17-13	Returns the second viewpoint associated with the specified window
WTwindow_getviewport	FLAG	WTwindow *window int *xoff int *yoff int *xsize int *ysize	17-32	Retrieves the position and size of a viewport
WTwindow_getwidget	Widget	WTwindow_getwidget WTwindow *w	17-29	Gets the X Widget that corresponds to the specified WTK window. If the pointer is not recognized to be a valid WTK window, NULL is returned. *Unix platforms only*
WTwindow_getyonvalue	float	WTwindow *window	17-19	Returns the window's yon clipping value
WTwindow_isenabled	FLAG	WTwindow *window	17-9	Returns TRUE if the specified window is enabled, FALSE otherwise
WTwindow_loadimage	FLAG	WTwindow *window char *filename float zval FLAG swapbuf FLAG bitmapdel	17-25	Loads the specified image file into the specified window so that it fills the window
WTwindow_new	WTwindow *	int x0 int y0 int xsize int ysize int flags	17-2	Creates a new window object and displays it (returns a pointer to the window object, if successful)
WTwindow_newuser	WTwindow *	(platform specific) *parent int window _config	17-7	Integrates WTK windows with host-specific windows *Note: On Windows platforms, parent is a HWND * On Unix platforms, parent is a Widget **

Function Name	Returns	Parameters	Page	Description
WTwindow_newviewport	WTwindow *	WTwindow *window int xoff int yoff int xsize int ysize	17-32	Creates a new viewport within a window
WTwindow_next	WTwindow *	WTwindow *window	17-8	Returns the next window in the universe's list of WTK window objects
WTwindow_numpolys	int	WTwindow *window	17-25	Returns the number of polygons sent to the graphics pipeline associated with the specified window
WTwindow_pickpoly	WTpoly *	WTwindow *window WTp2 point WTnodepath **nodepath WTp3 p	17-20	Returns the frontmost polygon at the specified point in the specified window
WTwindow_projectpoint	FLAG	WTwindow *window int eye WTp3 pos WTp2 point	17-21	Computes the 2D screen point relative to the window position where a 3D world coordinate projects
WTwindow_set2Dcolor	void	WTwindow *window unsigned char r unsigned char g unsigned char b	19-1	Specifies the color to be used by subsequent 2D drawing functions (default is white)
WTwindow_set2Dfont	void	WTwindow *window int fontindex	19-6	Sets the font index to be used when drawing 2D text to the specified window
WTwindow_set2Dlinestyle	void	WTwindow *window int style	19-2	Sets the 2D line style of the specified window
WTwindow_set2Dline width	void	WTwindow *window float width	19-2	Sets the 2D line width (in pixels) for the specified window
WTwindow_set3Dcolor	void	WTwindow *window unsigned char r unsigned char g unsigned char b	19-8	Specifies the color to be used by subsequent 3D drawing functions (default is white)
WTwindow__set3Dlinestyle	void	WTwindow *window int style	19-8	Sets the 3D line style of the specified window
WTwindow_set3Dlinewidth	void	WTwindow *window float width	19-9	Sets the 3D line width (in pixels) for the specified window

Function Name	Returns	Parameters	Page	Description
WTwindow_set3Dpointsize	void	WTwindow *window float size	19-9	Sets the 3D point size (in pixels) for the specified window
WTwindow_setbgrgb	void	WTwindow *window unsigned char r unsigned char g unsigned char b	17-22	Sets the 24-bit color background for the specified window (0 to 255 are valid values)
WTwindow_setdata	void	WTwindow *window void *data	17-28	Sets the user-defined data field for the specified window
WTwindow_setdrawfn	void	WTwindow *window void (*drawfn)(WTwindow *win, FLAG eye)	17-23	Specifies a function containing calls to 3D drawing routines
WTwindow_seteye	void	WTwindow *window short eye	17-12	Specifies whether the scene displayed in the specified window should be rendered from the left or right eye of the viewpoint
WTwindow_setfgactions	void	WTwindow *window void (*fgdrawfn)(WTwindow *win, FLAG eye)	17-24	Specifies a function containing calls to 2D drawing routines (drawn in the foreground)
WTwindow_sethithervalue	void	WTwindow *window float val	17-18	Sets the distance of the window's hither clip plane in front of the viewpoint
WTwindow_setname	void	WTwindow *win const char *name	17-27	Sets the name of the specified window
WTwindow_setparams	void	WTwindow *window FLAG eye float left float right float bottom float top float near float far	17-16	Specifies the parameters describing the window's viewing frustum used for the specified eye
WTwindow_setposition	void	WTwindow *window int x0 int y0 int width int height	17-10	Changes a window's size and/or location on a screen

Function Name	Returns	Parameters	Page	Description
WTwindow_setprojection	void	WTwindow *window int type	17-14	Sets a projection type for the specified window
WTwindow_setrootnode	void	WTwindow *window WTnode *rootnode	17-8	Associates a scene graph with a specified window by passing in the root node of the scene graph
WTwindow_setviewangle	void	WTwindow *window float angle	17-19	Sets the specified window's horizonal view angle (in radians)
WTwindow_setviewpoint	void	WTwindow *window WTviewpoint *view	17-11	Sets the specified viewpoint to display in the specified window
WTwindow_setviewpoint2	void	WTwindow *window WTviewpoint *view	17-12	Sets the second viewpoint to display for the specified window when using a stereo window
WTwindow_setviewport	FLAG	WTwindow *window int xoff int yoff int xsize int ysize	17-31	Sets the position and size of the default viewport of a window
WTwindow_setyonvalues	void	WTwindow *window float val	17-19	Sets the window's yon clipping value
WTwindow_zoomviewpoint	void	WTwindow *window	17-13	Zooms the viewpoint of the specified window so that all geometries in the scene graph (associated with the window) are visible
WTwindow_zoomviewtonode	void	WTwindow *window WTnode *node int which	17-14	Zooms the viewpoint of the specified window so that all geometries in the node (and node's sub-tree) are visible
WTxformnode_new	WTnode *	WTnode *parent	4-42	Creates a transform node and adds it to the scene graph after the last child of the specified parent
WTxformsepnode_new	WTnode *	WTnode *parent	4-43	Creates a transform separator node and adds it to the scene graph after the last child of the specified parent
WTzero	[MACRO]	float value	25-33	Returns TRUE if the magnitude of the specified value is less than the defined constant WTFUZZ

Notes: