Virtual Set 2000

Environment Builder
for Virtual Studio Sets Control

Script Commands Guide

Revision 1.07

DARIM

Darim Vision Co., Ltd.

Limited Warranty

Our company warrants this product against defects in materials and workmanship for a period of one year from the date of purchase. During the warranty period, products determined by us to be defective in form or function will be repaired or replaced at our option, at no charge. This warranty does not apply if the product has been damaged by accident, abuse, misuse, or as a result of service or modification other than by us.

This warranty is in lieu of any other warranty expressed or implied. In no event shall we be held liable for incidental or consequential damages, such as lost revenue or lost business opportunities arising from the purchase of this product.

Table of Contents

1 Script commands

1.1 The commands directory

The VS2000 script language gives you powerful tools for video production and design. You can utilize and control several virtual cameras in a scene, easily switching input from one camera to another. This is achieved with one line in the script: **DATA.CURRENT.CAMERA = «Camera1»,** where **«Camera1»** replaces the previous current (active) camera. Or, for example, opening of a file: **DATA.OPEN = «3d-file name»**, where **«3d-file name»** can be replaced with an actual file name when needed.

Script commands in *HotActions* work with four basic objects: **DATA**, **TRACK**, **RENDER** and **SYS**. To define actions, these objects can be combined with other actions or given certain properties, by using the dot notation. Name of object, its commands or properties are separated each with a dot (**.**) to form the needed sequence altering the state of the object. One example is given above.

Each command uses four types of data: strings, floating point numbers, integers and vectors. For simplicity of interpretation of the data in the following chapters, we shall use the Hungarian notation for the indication of the type of data for a specific parameter. For example, *iSpeed* means, that the parameter *iSpeed* is of the type «integer», and *fX* – is a parameter using a floating point number. Use of inverted commas around strings are unessential except in cases when strings with spaces are used such as file path etc. However it is always recommended to use the inverted commas.

Commands are separated by line end symbol (pressing **«Enter»**) or a semi-colon (**;**), if there is more than one command on the line.

For comments in your script you should add comments in the style of the **C++** and **C** programming languages. Summary information on the syntax for comments is discussed below.

Commands separators

{line end} .	Commands that are written on one line are ended with the Enter key,
;	If there is more than one command on a line each should end with a semi-colon.

Comments

//	These should be placed at the beginnig of a line that contains only comments. It must also end with the Enter key.
/* */	Used to surround comments embedded in a command line or when a comment requires more than one line .

Data types

String	The text lines such as «text» should be included in inverted commas; examples: **«Camera1»**, **«End of Track»**.
Float	Number using floating point, for example, 10.0.
Int	Integer, for example, 10.
Vector	Vector – a combination of three numbers using floating point, for example, (1.0, 2.0, 3.0).

1.2 *Actions* Commands

ACTION.START = «ActionName»

Executes *Action* with the name *ActionName.* This name can specify its library name as well as the action name. For exapmle: ACTION.START = «*NewSport.PlayTrack*», where *NewSport* – is the ***Action Library*** name, and *PlayTrack* – ***Action*** name.

After the execution, ***Action*** automatically executes the new event «*ACTION.ActionName*». This event can be used as parameter for the command

SYS.WAIT = «event» while waiting for the completition of an ***Action***. For example: *SYS.WAIT = Action.ShowSport*.

1.3 Data Commands

1.3.1. Common data commands

DATA.OPEN = «FileName»

Opens file *FileName* (with extension *.3D, *.ACL, and *.HOT).

DATA.DIRECTORY.SCENES = «Path»

Sets *Path* to search a directory for a scene file.

DATA.DIRECTORY.IMAGES = «Path»

Sets *Path* to search a directory for an image file.

DATA.CURRENT = «Scene»

Sets the named *Scene* to be the current (active) scene.

DATA.CURRENT.CAMERA = «Camera»

Sets the named *Camera* to be the current virtual camera.

DATA.CURRENT.NODE = «Object»

Selects *Object* in tree structure for control (for example, for use by a mouse/joystick).

DATA.RESET = 1

Return all objects to their initial positions as at the opening of a scene.

DATA.PLAY = iSpeed

Presents choices track speeds and also zero speed. Parameter values *iSpeed* can be:

0 – stop;
1 – forward with normal speed;
-1 – back up with normal speed;
2 – forward with increased speed;
-2 – back up with increased speed.

1.3.2. Camera Commands

DATA.NODE.«Camera».FOV = fFOV, fTime

Sets **FOV** (Field Of View) for *Camera* in *fFOV* (in degrees) and time for the setting with the parameter *fTime* (in seconds). If the parameter *fTime* is not specified, then **FOV** is immediately set at the specified degrees.

DATA.NODE.«Camera».CLIPNEAR = fDistance

The distance to the nearer depth clipping plane for the camera *Camera*. This distance must be positive.

DATA.NODE.«Camera».CLIPFAR = fDistance

The distance to the farther depth clipping plane for the camera *Camera*. This distance must be positive.

1.3.3. Light Sources Commands

DATA NODE.«Light».MULT = fMult

Sets a multiplier for light brightness in *fMult*. This is the same parameter as is used in *3D Studio MAX*.

DATA NODE.«Light».RANGE = fNear, fFar

Sets for light source *Light* a distance range. This is the same parameter as is used in *3D Studio MAX.*

DATA NODE.«Light». ANGLE = fHotAngle, fFallAngle

Sets for a direction angle for the light source *Light*. This is the same parameter as is used in *3D Studio MAX*.

1.3.4. 3D-Object Commands

DATA.NODE.«Object».HIDE = iHide

Hides/shows object with the name *Object* in a scene; value **1** for *iHide* corresponds to hiding of object, **0** – to display of object.

DATA.NODE.«Object».POS = fX, fY, fZ

Sets a new position for object with the name *Object*, determined by new coordinates *fX, fY*, and *fZ* in the scene. Coordinates {0, 0, 0} correspond to the initial position in the scene.

DATA.NODE.«Object».ROT = fα, fβ, fγ

Rotation of object with the name *Object* from its initial position. Euler angles *fα*, *fβ*, and *fγ* (angles turning on axes *X, Y, Z*) are defined in degrees.

DATA.NODE.«Object».SCL = fa, fb, fc

Scaling of the object named *Object*, expressed as a multiple of its three dimensions by the factors *fa*, *fb*, and *fc*.

DATA.NODE.«Object».VEL = fMOVx, fMOVy, fMOVz, fROTx, fROT y, fROTz, fSCLx, fSCLy, fSCLz

Sets speed of movement, speed of rotation and speed of scaling of the object named *Object* by parameters *fMOVx, fMOVy, fMOVz, fROTx, fROTy, fROTz, fSCLx, fSCLy, fSCLz* on all axes accordingly (measured as units per millisecond).

DATA.NODE.«Object».DPOS = fMOVx, fMOVy, fMOVz, fROTx, fROT y, fROTz, fSCLx, fSCLy, fSCLz

Sets movement, rotation and scaling of the object named *Object* with parameters *fMOVx, fMOVy, fMOVz, fROTx, fROTy, fROTz, fSCLx, fSCLy, fSCLz* on all axes accordingly.

1.3.5. Materials Commands

DATA.MATERIAL.MaterialName.CACHE = iState

Switches between on (*iState* value 1) or off (*iState* value 0) the texture-caching when repeated textures are loaded from memory. The default value is on or 1.

DATA.MATERIAL.MaterialName.MAP = «GraphicsFile.ext»

Points to a file with the name *GraphicsFile.ext* to load new texture source for ***Diffuse Map*** of the material named *MaterialName*. The command works only for textural materials with the ***Diffuse Map***, such as those made in *3D Studio Max*. Supported file extensions are jpg, bmp, tiff, tga.

DATA.MATERIAL.MaterialName.SIZE = iWidth, iHeight

Changes the size of a textural map for use by the command **TEXT**. The texture is also cleared.

DATA.MATERIAL.MaterialName.TEXT = «Text», «Font», iFontSize,iX, iY, iR, iG, iB, iMix

Replaces a material texture named "*MaterialName"* with the text specified by *«Text»* with the use of the following parameters:

«Font»	font name;
iFontSize	font size (in texels);
iX	indent on **X**;

iY	indent on **Y**;
iR	font color (red component);
iG	font color (green component);
iB	font color (blue component);
iMix	mixing mode with a textural map:

1 – text is added to texture;

0 – text replaces a texture.

The indent is counted from the left top corner of texture. All parameters by default are equal to zero, except for font, which by default – **System**. *iR, iG*, and *iB* change from 0 up to 255. The textural material should have textural coordinates (made in *3D Studio Max*) for use with this command. If the material has a transparent texture and *iMix*=0 the text appears on a transparent substrate.

DATA.MATERIAL.MaterialName.TEXT.FONT = «Font»

Sets font for command **TEXT**. *«Font»* – uses the system font name.

DATA.MATERIAL.MaterialName.TEXT.SIZE = iFontSize

Sets font size for command **TEXT**. *iFontSize* – font size (in texels).

DATA.MATERIAL.MaterialName.TEXT.OFFSET = iX, iY

Sets text displacement for command **TEXT**. The offset is counted from the left top corner of a texture.

DATA.MATERIAL.MaterialName.TEXT.COLOR = iR, iG, iB

Sets text color for command **TEXT**. *iR*, *iG*, and *iB* as specified by combinations of RGB (red, green and blue) from 0 to 255.

DATA.MATERIAL.MaterialName.TEXT.SHADOW = 0/1

Switches on/off a shadow around the text in the size of 1 texel.

DATA.MATERIAL.MaterialName.TEXT.SHADOW.COLOR = iR, iG, iB

Sets a shadow color. *iR*, *iG*, and *iB* as specified by combinations of RGB (red, green and blue) from 0 to 255.

DATA.MATERIAL.MaterialName.RESET = 1

Restores an object material after replacement by text by command **DATA.MATERIAL.MaterialName.MAP**. or ***.TEXT**.

DATA.MATERIAL.Reset = 1

Globally restores all objects materials to their initial condition.

1.3.6. 3D-text Commands

☝ *Please note, that file used for font (as in next commands) should contains256 images – one for each character of the font. These images should be 16x16 and should be separated by pixel frame with alpha=0. Thus images themselves should not contain pixels with alpha=0. Images might have different width but should have equal heights.*

DATA.FONT = «File», fFontHeightRatio, fFontWidthRatio

Sets a font by default for 3D-text.

«File» – name of *.tga or *.tiff –file describing a font or system font name.

fFontHeightRatio –ratio for font characters height.

fFontWidthRatio – ratio for font characters width. If this parameter is not specified, defaults to proportionally to height.

DATA.FONT.FontName = «File», iFontHeight, iFontWidth

Create font and give it a name **FontName.**

«File» – name of *.tga or *.tiff –file describing a font or system font name.

fFontHeightRatio –ratio for font characters height.

fFontWidthRatio – ratio for font characters width. If this parameter is not specified, defaults to proportionally to height.

DATA.NODE.«Object».FONT = «FontName»

Sets font **FontName** for the object named *Object*.

DATA.NODE.«Object».TEXT = «Text», fHeight, fWidth

Replaces object with the name *Object* in the scene with 3D-text **Text**.

You may define more than one string in **Text** to display at once by separating them with "\n".

Parameters *fHeight* and *fWidth* sets the coefficients for text scaling, such that the number of shown strings is 1/*fHeight*. Other strings are clipped. These parameters can have values from 0.0 to 1.0.

In **JUSTIFY** mode (see descriprion for the next command) the auto line feed is used. That means that if the string is wider than object, it is divided into two strings, which are displayed one under another.

By default text is centered by its width and its height is equal to objects height.

It is uses the system font by default or font which is set by the command **DATA.FONT** or a font directly specified for this object by the command **DATA.NODE.«Object».FONT = «FontName»**.

DATA.NODE.«Object».ALIGN = «Mode»

Sets the mode for lining up the text with dimensions of the object named *Object*.

«CENTER» – centering across;

«RIGHT» – alignment to the right;

«LEFT» – alignment to the left;

«JUSTIFY» – completely filling in from the right to the left and using spaces as filler where necessary.

1.4 Track Commands

Note: After each track ends, the system issues the event «TRACK.TrackName». This event can be used as parameter for the command SYS.WAIT = «event» to gage when an action track is completed. For example: SYS.WAIT = Track Sport.

TRACK.«TrackName».NODE = «NodeName»

This command sets a track name *«TrackName»*. Scene objects in *3D Studio Max* can have animation (trajectory). *3D Studio Max* does not set names for trajectories. For control of trajectories in VS2000 appropriate names must be determined. Thus this command assigns the trajectory of the object named «*NodeName*» a name *«TrackName»*. It is used for further work with trajectories of this object, for example, for start.

TRACK.«TrackName».NOTIFY = «Event»

Generates the event named *Event* when track *TrackName* reaches its end; by default at end of the playing of a track event «**TRACK.TrackName**» is generated.

TRACK.«TrackName».LOOP = iNLoop

Defines whether track will be automatically replayed or not by *START* or *PLAY* commands (see further). *iNLoop* = 1 corresponds to an infinite number of playbacks while *iNLoop* = 0 means playback just one time.

TRACK.«TrackName».START = iFrom, iTo

Plays back the track named *TrackName* over the range from *iFrom* frame to the *iTo* frame. If *iFrom* > *iTo* the track is played over the range of frames in the inverse order. If the second parameter is not specified, the track is set in the initial setting in the **iFrom** frame.

TRACK.«TrackName».RANGE = iFrom, iTo, iRepeat

Set the playback of the track named *TrackName* over a range of frames set from *iFrom* to *iTo*, to be repeated *iRepeat* number of times. For playing this set of frames it is possible to use the command **TRACK.«TrackName».PLAY = iSpeed** to determine the rate of play.

TRACK.«TrackName».PLAY = iSpeed

Playback the track named *TrackName* as it was set by last command **START** or **RANGE**, in the direction of play determined by the value of the parameter *iSpeed*:

0 – stops playback;

1 – playbacks track forward;

–1 – playbacks track back up.

TRACK.«TrackName».STOP = iNFrame

Completely stops playing of the track named *TrackName* on the frame number determined by the value of *iNFrame*. Whether track is be replayed or not is not determined.

TRACK.«TrackName».GOTO = iNFrame

Playbacks the track named *TrackName* from current frame to the frame number determined by the value of *iNFrame*. The number of repetitions set by the command **RANGE** is ignored. If *iNFrame* is a number greater than the current frame number then the track is played forward. If *iNFrame* is a number less than the current frame number then the track is played backwards up to the frame *iNFrame*.

TRACK.«TrackName».QUEUE = iState

Switches between on (1)/switch off (0) for command queue for the track *TrackName*. Each track has its own queue. If *iState* =1 every command for the track named *TrackName* will start automatically after previous is finished; else if *iState* =0 every command will start immediately. By default – it is 1 for switched on.

The command sequence using command auto queue:

```
TRACK.A.QUEUE = 1
TRACK.A.START = 100, 200
TRACK.A.START = 00, 100
TRACK.A.START = 100, 200
```

Can be replaced using the system wait command:

```
TRACK.A.QUEUE = 0
TRACK.A.START = 100,200
SYS.WAIT = «TRACK.A»
TRACK.A.START = 200,100
SYS.WAIT = «TRACK.A»
TRACK.A.START = 100, 200
```

TRACK.«TrackName».DELETE = 1

Deletes the contents of the track named *Track*.

TRACK.DELETE = 1

Deletes all named tracks.

TRACK.RESET = 1

Stops all animation. Stops all tracks and clears any actions in a wait state.

1.5 Morphing

RENDER.MORPH = «SourceObjectName», «DestObjectName», fTime

Initiates morphing of the initial object with name *SourceObjectName* to the object with name *DestObjectName* (it is usually hidden in a scene) over a period time *fTime* expressed in seconds. Both objects should have the same number of vertices.

1.6 Video and ImageCommands

RENDER.VIDEO.VideoStreamName.CREATE = iParam

Creates a video texture with name *VideoStreamName*, which can be one of the following:

LIVE_1 – video texture from the first board *FD300* in the system;

LIVE_2, **LIVE_3** – video textures from second and third boards *FD300* in the system, if there are any;

FILE_1, **FILE_2** etc. for video texture from DV–files;

MSDS_1, **MSDS_2** etc. video texture can be created with video files by *DirectShow* filters.

Parametr *iParam* can have following values:

IParam = 1 for video stream creation;

IParam = 0 for video stream destruction.

☝ *Note: It is strongly recommended to use video textures as* ***FILE_**** *for video texture from DV-files. It then allows using additional commands with them. For other video files, codecs for which are installed in operational system, i.e. for all other files that can be viewed with help Windows Media Player, you should use* ***MSDS_****.

RENDER.VIDEO.VideoStreamName.FORMAT = «Format string»

This command should be executed after the creation but before start of a video stream. It sets the format of the video stream in the line *«Format string»* through which point parameters can be specified. For use with video texture from an AVI-stream and from boards *FD300*:

For video stream from *FD300* (**LIVE_1**, **LIVE_2**,…):

- **ALPHA** – switchs to use of chroma key. The effective result is to make transparent certain background colors of the input video. Setting of chroma key parameters is made in dialog *KeyConfigPro* (see special document on working with it). This value is set by default. It includes the mask set in dialog *KeyConfigPro*;
- **NOALPHA** – switchs off chroma key use for the input video. Switches off a mask set in dialog *KeyConfigPro*;
- **NOFIELDS** – switchs on to a full frame mode of a video stream, however it could reduce the perfomance of the whole system;
- **HIRES** – switchs on use of the deinterlace filter;
- **LOWRES** – switchs off use of the deinterlace filter.

For stream files from AVI (**FILE_1**, **FILE_2**, … **MSDS_1**, **MSDS_2,...**):

- **ADD** – sets mode to add specified files in the queue of the video stream, i.e. files are added by commands **RENDER.VIDEO.VideoStreamName.DATA** (for more about this command see further below). These files are played in the order in that they are added. This value is set on by default.

☝ *Decoding of frames for **MSDS_*** is concurrent with reading data from file that is why command START always starts playing the last file added with command **ADD**.*

- **REPLACE** – sets mode to replacement files in the queue of the specified video stream, i.e. the command **RENDER.VIDEO.VideoStreamName.DATA** (for more about it see further below) interrupts playing previous, clearing all files waiting before starting the new file;
- **LOOP** – switches on mode for recycled playback of a video stream;
- **NOLOOP** – switches off mode for recycled playback of a video stream. This value is set on by default.

For working with video textures made with help from *DirectShow* (**MSDS_1**, **MSDS_2** etc.) parameters used are:

- **AUDIO** – playing of the video stream together with its sound track;
- **NOAUDIO** – playing of a video stream will be made without its sound track.

Default value: **AUDIO**.

RENDER.VIDEO.VideoStreamName.START = iParam

Starts (*iParam* = 1) and stops (*iParam* = 0) preliminary created video texture with name *VideoStreamName*. Depending on parameter, the event «*VIDEO.VideoStreamName.START* » or «*VIDEO.VideoStreamName.STOP*» is generated.

Thus, for video textures from files (**FILE_*** and **MSDS_ ***) the ending of the fragment given on playing the event *VIDEO.VideoStreamName.END* is generated. Note, that if video stream playback is in the mode of recycled playback for **FILE_*** this message is not generated, and for **MSDS_*** the message is generated and signals positioning of the file at the beginning.

☝ *The actual time of the playback start for **MSDS_*** is depended on several circumstances (such as type of file, system computational load etc.). However second playback is started much faster.*

The following command is specific for working with video stream from AVI*:*

RENDER.VIDEO.VideoStreamName.DATA = «Filename», iStartFrame, iLength

This command attaches data to the video stream with name *VideoStreamName* (accepting values **FILE_i** or **MSDS_i**, where «**i**» – stream number): adds the frames from an AVI-file with a name *«Filename»* to AVI-video stream file with name *VideoStreamName*, since *iStartFrame* and *iLength* frames. If both parameters are equal to zero or are absent, the entire file is added.

The following command is specific for working with video stream from *FD300:*

RENDER.VIDEO.VideoStreamName.LINE = «Line»

Changes the channel for video texture from board *FD300* (i.e. *VideoStreamName* accepts values **LIVE_1**, **LIVE_2** etc.) since it has two channels:

Line = A corresponds to the first channel;

Line = B corresponds to the second channel.

Thus event *VIDEO.VideoStreamName.LINE* is generated.

The following commands are used for working with texture materials in the scene*:*

RENDER.MATERIAL.MaterialName.SOURCE = VideoStreamName

Fixes a material with the name *MaterialName* to a video texture with name *VideoStreamName* (i.e. **LIVE_1** or **FILE_2** etc.). The video texture will be laid on all objects in a scene with this material. If the video textures with such name is not present, last texture loaded for a material, i.e. a primary texture from 3D-scene or its replacement from a file or the text if used commands **DATA.MATERIAL.MaterialName.MAP** or ***.TEXT** is restored.

RENDER.MATERIAL.MaterialName.POS = fPosU, fPosV, fUnused

Adjusts textural coordinates, shifting them by *fPosU* and *fPosV* units on axes **U** and **V** accordingly; value **0.0** corresponds to a starting position of a texture.

RENDER.MATERIAL.MaterialName.SCL = fScaleU, fScaleV, fUnused

Adjusts textural coordinates, scaling them in *1/fScaleU* and in *1/fScaleV* time on axes **U** and **V** accordingly. Value **1.0** corresponds to initial scale of a texture.

1.7 Event Commands

SYS.WAIT = «Event»

Stops execution of the script before occurrence of the event named *Event.*

SYS.DELAY = fTime

Stops execution of the script at *Time* expressed in seconds.

SYS.RESET = 1 Resets any waiting of any events.

SYS.OUTPUT = «Message»

Outputs text *Message* to **Debug Output**

1.8 Sound Commands

SOUND.DIRECTORY = «Path»

Sets *Path* for working directory for audio files.

SOUND.Name.OPEN = «FileName»; SOUND.Name.FILE = «FileName»

Loads wav–file *FileName* and names created sound track with the name *Name.*

SOUND.Name.PLAY = iMode

Switches off (*iMode*=0) and switches on (*iMode*=1) playback of sound track with the name *Name.*

SOUND.Name.LOOP = iMode

Switches off (*iMode*=0) and switches on (*iMode*=1) recycled playback of sound track with the name *Name.*

1.9 Joystick Commands

JOYSTICK.Name = iID

Names joystick with the identifier *iID* with the name *Name*.

iID is from 1 up to number of connected joysticks. This command is used for naming connected joysticks by their number. That is why *iID* of connected joystick should be found experimentally. Nevertheless once founded it remains the same if the connection of joysticks is not changed.

if *iID* of two joysticks are the same, then the last joystick assigned the identifier *iID* is used;

JOYSTICK.Name.MOVE.RANGE = fMoveX,fMoveY,fMoveZ,fRotX,fRotY,fRotZ

Sets movement and rotation sensitivity of the joystick with the name *Name* by six numbers, in limits from 0.0 up to 1.0 (three for rotation and three for movement on each axis).

JOYSTICK.Name.BUTTON.Butt = iID

Sets to the button *Butt* of a joystick with the name *Name* the identifier *iID* for further use in commands. If *iID* of some buttons of one joystick coincide, last button used will be declared to have *iID* .

The commands below set a command line of Commands which is sent to the system whenever a button is used on a joystick.

JOYSTICK.Name.MOVE = «Commands»

Sets a commands line *Commands* which is sent when stick of the joystick *Name* is moved.

JOYSTICK.Name.BUTTON.Butt.UP = «Commands»

Sets up the commands line *Commands,* which is sent at release of the button with the identifier *iID* of the joystick with the name *Name*.

JOYSTICK.Name.BUTTON.Butt.DOWN = «Commands»

Sets up the commands line *Commands* that is sent by pressing of a button with the identifier *iID* on a joystick with the name *Name.*

JOYSTICK.Name.BUTTON.Butt.HOLD = «Commands»

Sets up a commands line *Commands,* which is sent when a button with the identifier *iID* is held down on a joystick with the name *Name*. This commands line is sent in each frame.

1.10 Robotized Camera Commands

RCAM.«CameraName».OPEN = «CameraType», «ComPortName»

This command initializes the control module and names the plugged camera for further use as *«CameraName»*.

CameraType parameter sets the type of used camera as "EVI-D30", "EVI-D31", "CANON VC-C1" or "CANON VC-C3";

Use *ComPortName* = *COM1* for connection to the first COM-port of a computer. *ComPortName* = *COM2* – controls the camera connected to the second COM-port of the computer etc.

RCAM.«CameraName».POWER = iValue

This command switches on the camera with name *«CameraName»* (*iValue* = 1) and switches it off when *iValue* = 0;

RCAM.«CameraName».FOV = iValue

Sets the FOV (Field Of View) for the camera with name *«CameraName»*:

iValue – horizontal FOV in degrees;

RCAM.«CameraName».ROT = fα, fβ, fγ, fTime

Rotates the camera with name *«CameraName»*:

fα, fβ, *fγ* –Euler angles (clockwise rotation about X,Y and Z axis);

fTime – time (in seconds) to perform given rotation.

☝ *Note: After each robotized camera command ends, the system issues the event «CameraName». This event can be used as parameter for the command SYS.WAIT = «event» to gage when an action is completed. For example: SYS.WAIT = «Camera1».*

RCAM.«CameraName».REACTION = fValue

Set the reaction time for camera *«CameraName»*:

fValue –time (in seconds) to perform any command, all the subsequent commands will be performed for a time length not less then the given time;

RCAM.«CameraName».QUEUE = iValue

Switches between on (*iValue* = 1) and off (*iValue* = 0) queue for camera with the name *«CameraName»*. If the queue is on then any subsequent command will start automatically after previous is finished; else every command will start immediately. By default – it is 1 for switched on.

RCAM.«CameraName».RESET = 1

Clears the command queue for camera with the name *«CameraName»*.

Note! When using a sound, joystick, or robotized camera, be certain that following libraries DLL are in the directory of HotActions for plugins (if the studio is installed on disk D this directory – \Program Files\Darim Vision\VS2000 Software\HotActions\Plugins):

- *Sound.dlf – for using audio files;*
- *Joystick.dlf – forusing joystick;*
- *rcam.dlf– for using robotized camera.*

For standard joysticks, use **Gaming Options** on control panel *Windows 2000*. In the same panel is displayed information about joysticks and their **IDs** (identifiers) for their buttons.

There may exist names connected to various objects for the further reference, such as *Sound* in line *SOUND.Sound.FILE = «FileName»*. Difficulties can arise if several scenes are opened at the same time and they have objects with same name. If you want to be certain that an action will be executed for an object of the particular scene, adjust the reference to this scene in the beginning of command definition, using the following command: *DATA.CURRENT = «Scene».*

This prevents access of use of this command by other scenes else the results may have unpredictable consequences.

To be used together names of actions, trajectories and sounds should be unique for all open scenes. In other words, two different trajectories cannot have the same name and you have to apply commands seperately to both. Using the the same variable simply will replace it.

1.11 Use of video texture

In the current version of *HotActions* use of video texture requires the following steps:

1. Video texture creation

RENDER.VIDEO.LIVE_1.CREATE = 1.

2. Video texture configuring

RENDER.VIDEO.LIVE_1.LINE = A //assigning of input channal (See **Line A** in the section about setting for keying).

3. Start stream for video texture

RENDER.VIDEO.LIVE_1.START = 1.

4. If switching between channels the following command is required:

RENDER.VIDEO.LIVE_1.LINE = B.

5. Replacement of material in a scene with the live video texture

RENDER.MATERIAL.VIDEO.SOURCE = LIVE_1.

1.12 Reserved names

The object **ALL** is the object chosen by default for manipulation by the joystick, the mouse or by arrows, at the scene opening. It is convenient, if it is the object which position, size or other parameters you want to be changed in relation to the mouse or joystick. If object **ALL** is absent, the object **World** become the object for manipulation by default. This object is automatically added as a root of the scene during export from *3D Studio MAX*. Note no changes will be apparent since if you change this object all virtual cameras are changed with it. To be sure that a certain object is selected for control use the following command to make it current:

:DATA.CURRENT.NODE="Main", or special variant for virtual cameras **:DATA.CURRENT.CAMERA = «Camera1».**

Current is the special name for use in script commands that always indicates an action is addressed to the current chosen object. Any manipulation or changes of **<Current>** object accordingly changes the current selected object.

2 Script examples

Examination of the available script examples and their updates is a very productive way to learn the use of script commands. It is highly recommended to take adequate time to understand examples for the available projects. *VS2000* is delivered with a few simple sample projects, which are installed in a separate folder named ***VS2000 Samples***. The folder for every sample contains project (*.vsp), scene (in file format *.3d), library (***Actions Library***) and ***Hotset*** for the example. It is possible to experiment with these files to learn to achieve the your desired results when creating virtual scenes in *VS2000*.

It is necessary to note, that the majority of the commands given in this manual, work only after the startup ***Action*** for a specified scene. Principles and examples of creation of startup ***Actions*** are considered in the appropriate chapter of the ***HotActions User's Guide***. It is also strongly recommended to familiarize yourself with the examples given the document which accompanies *3D Studio MAX*.

3 Use of *Debug Output* for debugging scripts

In the application *HotActions* it is possible to trace an error with error messages and warnings, including those associated with the use of script commands. For this purpose the window of output *Debug Output* is used.

To learn more about setting up the output window *Debug Output* turn to the appropriate section of the ***HotActions User's Guide***. We only note here that a necessary condition of script debugging is the checking of the options **Runtime (from Actions)** and **Script Trace** on panel **Debug Output** of the *Options* dialog.

Index