Focus

Virtual Studio

Using the script commands

User's Guide

Revision as of November 25, 2009

VS 1.63

Table of Contents

1 COMMANDS GUIDE 3

2 ACTION COMMANDS 5

3 DATA COMMANDS 6

3.1 General Commands 6

3.2 Commands for Working with Virtual Cameras 6

3.3 Commands for Working with the Environment Parameters 7

3.4 Commands for Working with Light Sources 7

3.5 Commands for Working with 3D Objects 8

3.6 Commands for Working with Materials 9

3.7 Commands for Working with 2D Text 11

3.8 Commands for Working with Web Pages, Flash Animation and PPT Presentations 12

3.9 Commands for Working with 3D Text 12

4 TRACK COMMANDS 15

5 MORPHING COMMANDS 17

6 COMMANDS FOR DISPLAYING INFORMATION ABOUT THE SCENE 18

7 COMMANDS FOR IMAGE RENDERING 19

7.1 Commands for Working with Video Streams 19

7.2 Using Video Streams in the Scene 23

7.3 *3D Overlay* Mode of Rendering Material Textures 23

7.4 *3D Overlay* Mode Additional Commands 24

8 SYSTEM COMMANDS 25

9 COMMANDS FOR WORKING WITH SOUND 27

10 COMMANDS FOR CONTROLLING THE MOUSE CURSOR 28

11 COMMANDS FOR WORKING WITH THE JOYSTICK 29

12 COMMANDS FOR WORKING WITH A ROBOTIC CAMERA 32

13 TRACKING VIDEO STREAMS IN THE SCENE (MONITORING MOVEMENT DATA) 34

14 USING DESKTOPCAPTURE TO CAPTURE IMAGE FROM A REMOTE COMPUTER .. 35

15 COMMANDS FOR WORKING WITH EXTERNAL DEVICES VIA GPI (GENERAL PURPOSE INTERFACE) 37

16 PREDEFINED IDENTIFIERS 39

17 USING VARIABLES IN SCRIPT COMMANDS 40

18 RESTRICTING THE SCOPE OF VARIABLES 41

19 SCRIPT EXAMPLES 42

20 USING THE *DEBUG OUTPUT* WINDOW 43

1 Commands Guide

The script language is designed to control the Studio 3D scenery in the *HotActions* application.

For example, switching input from one camera to another in a scene with several cameras is achieved with one script line: DATA.CURRENT.CAMERA = Camera1, where Camera1 is to be replaced with the object name representing the virtual camera. Or the playback of a virtual camera animation (e.g., zoom-in at the actor) is achieved with a line TRACK.Camera1.START = iStartFrame, iEndFrame, where the virtual camera is the Camera1 object; iStartFrame and iEndFrame are the start and end frame numbers of the animation to be played.

The execution of a sequence of such lines arranged in the required order results in the required transformations in the 3D scene. In this way is created a TV program scenario.

The language of the lines to be executed consists of a sequence of words separated by periods as you could see in the examples above (switching the scene virtual cameras and the virtual camera animation playback). The first words in the lines are the keywords. They determine the subject of the command line. All the keywords can be viewed in the list displayed when type a colon in the Body field of the *Properties* window in the *HotActions* application (Section 3.3.1 of the *HotActions User's Guide*). These are ACTION, CONTROL, DATA, and so on. When the command keyword is selected, the further words can be selected from the list displayed when typing a period (.).

Variables in the right parts of the command lines (after the equals sign "=") can be of four types: string, integer, floating-point number, and vector. The following denotations are used in the descriptions of commands in this document to distinguish between the types:

- if a variable name in the right part starts with the letter "*i*", the variable is an integer;
- if a variable name in the right part starts with the letter "*f*", the variable is a floating-point number;
- a vector is several floating-point numbers starting with "*f*";
- a "string" parameter is indicated by quotes around. Generally, using quotes in commands with variables of this type is not obligatory, and required only if there are spaces, «+», «–» «**.**» marks, etc. However, it is recommended that you use quotes in "string" variables always not to forget to type them in the required cases.

With some exeptions, variables in the right parts of the commands described in this document are denoted by these rules. For example, *iSpeed* implies the parameter is an integer, and *fX* is a floating-point parameter.

Commands are separated by the EOL symbol (pressing **Enter**) or, if several commands are typed in one line, by a semicolon (**;**).

To type a nonexecutable line in the Body Script field, e.g., a comment to a command or a nonexecutable command, type the symbol "//" at the beginning of the line. If required to type a sequence of commands that are not to be executed currently, type the symbol "/*" before the first command, and "*/" after the last.

A summary on the syntax covered in this Section is given below.

Commands Separators

{end of line}	If one command is typed in a line, the command separator is implied automatically.
;	A semicolon if several commands in sequence are typed in a line.

Comments

//	A comment up to the end of line.
/* */	A comment between «**/*** » and «* **/**».

Data Type

String	A text; lines of the "string" type must be put in quotes, e.g., «Camera1», «End of Track».
Float	A floating-point number, e.g., 10.0.

Int	An integer, e.g., 10.
Vector	A vector: a combination of three floating-point numbers, e.g., (1.0, 2.0, 3.0).

System Symbols

:	A colon. It is recommended that you put this symbol first in the line, before the beginning of the command: the system will automatically prompt command versions and their parameters.
@	A symbol indicating that the command is not to be processed when debugging and tracing (see Sections 8 and 20).

2 Action Commands

ACTION.START = "ActionName"

Executes ***Action*** with the name *ActionName*. This name can be extended by specifying the library name.

For exapmle: ACTION.START = "*NewSport.PlayTrack"*, where *NewSport* is the ***Action Library*** name, and *PlayTrack* is the ***Action*** name.

After the ***Action*** execution is complete, the message *SYS.EVENT* = "*ACTION.ActionName"* about it appears in the **Message (Debug Output)** window (see Section 20). The right part of the message can be used as a parameter for the command *SYS.WAIT ="Event"* for waiting for the ***Action*** completion (see Section 8). For example: *SYS.WAIT = Action.ShowSport*.

ACTION.DISABLE = iMode

Enables the mode when all ***Actions*** on ***Hotsets*** become unavailable (*iMode*=1) or available (*iMode*=0) for executing.

3 Data Commands

3.1 General Commands

DATA.OPEN = "FileName"

Opens the *FileName* file (with the *.3d, *.acl, *.hot, and *.vsp extentions).

DATA.CURRENT = "Scene"

Makes the *Scene* current (active).

DATA.CURRENT.CAMERA = "Camera"

Makes the *Camera* the current virtual camera.

DATA.CURRENT.NODE = "Object"

Selects the *Object* in the tree to control (for example, with mouse/ joystick).

DATA.RESET = 1

Resets all objects to their initial positions (states) as when opening the scene.

DATA.PLAY = fSpeed

Selects a speed of the track speed and starts the track. The *fSpeed* parameter values can be:

positive, at that 1 corresponds to the speed of 25 frames per second;

equals 0 – the track animation playback stops;

negative – the animation is played backwards unless the tracks are named by the **TRACK.Track.NODE = NodeName** command (discussed in Section 4), that is, playing tracks by the commands **TRACK.Track. START = iFrom, iTo, iRepeat,** and **TRACK.Track.GOTO = iNFrame** (Section 4) is not executed, if the right part of the command **DATA.PLAY** is negative.

The commands **DATA.RESET = 1**, **DATA.PLAY = 0**,and **DATA.PLAY = 1** are executed by default at the project initialization started by the **Init** button of the main window toolbar or when switching to the ***LiveAction*** mode (see the ***HotActions*** User's Guide, Section 3.1.9). The command **DATA.PLAY = 0** stops all tracks being played, and **DATA.PLAY = 1** starts playback of all scene tracks.

DATA.STEREO = fStereoBase, fAngle

Working with stereo effects. *fStereoBase* determines the image shift for creating the effect of three-dimensionality when viewing through the polarizing filters, *fAngle* determining the angle of shear. By default, *fStereoBase* = 100, *fAngle* = 0.

Working with this command implies a special configuration of the studio for working with stereo.

3.2 Commands for Working with Virtual Cameras

DATA.CAMERA.Camera.FOV = fFOV, fTime

Sets *FOV* (Field Of View horizontally) for the virtual camera named *Camera* in *fFOV* (in degrees) during the time *fTime* (in seconds). The *fFOV* parameter can take on a value from 0 to 180. The default value is equal to the scene value.

If the *fTime* parameter is not specified, *FOV* is set immediately.

DATA.CAMERA.Camera.CLIPNEAR = fDistance

Sets a distance to the nearest clipping plane for the virtual camera named *Camera*. At opening a scene, the value is set equal to 0, which enables the automatic calculation mode. That is, the distance between the virtual camera and the nearest object is calculated in every frame. If a negative value is specified, the parameter also equals 0, accordingly, the automatic calculation mode being enabled.

If the variation of the objects Z coordinates is large, faces can be sorted incorrectly when displaying. In this case, for the correction in the automatic mode, the value of the nearest clipping plane for the virtual camera can be set smaller than the value of the nearest object.

DATA.CAMERA.Camera.CLIPFAR = fDistance

Sets a distance to the remotest clipping plane for the virtual camera named *Camera*. When loading a scene, the value is set equal to 0, which implies the automatic calculation mode. That is, the distance between the virtual camera and the remotest object is calculated in every frame. If a negative value is specified, the parameter also equals 0, which, accordingly, implies the automatic calculation mode.

DATA.CAMERA.Camera.FOGNEAR = fDistance

Sets the nearest *fDistance* bound of the fog effect for the virtual camera named *Camera*. The commands of switching on the fog effect are described in the next Section. By default, the *fDistance* value is set as it was specified in the initial scene (the *Near Range* parameter in the *Environment Ranges* group of parameters).

DATA.CAMERA.Camera.FOGFAR = fDistance

for the *Camera* virtual camera sets the *fDistance* distanse, from which the fog effect starts dissappearing. By default, the *fDistance* value is set as it was specified in the initial scene (the *Far Range* parameter in the *Environment Ranges* group of parameters).

3.3 Commands for Working with the Environment Parameters

DATA.ENVIRONMENT.DIFFUSE = iR, iG, iB

Determines the color of the environment backgroung light, *iR being* the red component, *iG* - green component, *iB* - blue component. The *iR*, *iG*, *iB* parameters can take on values in the range from 0 to 255. By default, the color is set equal to the value specified in the scene (*Color Background* in the *Rendering Environment* Section).

DATA.ENVIRONMENT.AMBIENT = iR, iG, iB

Determines the color of the environment ambient light, *iR* being the red component, *iG* - green component, *iB* - blue component. The *iR*, *iG*, *iB* parameters can take on values in the range from 0 to 255. By default, the color is set equal to the value specified in the scene (*Ambient Global Lighting* in the *Rendering Environment* Section).

DATA.ENVIRONMENT.FOG.ENABLE = iState

Enables (*iState=1*)/disables (*iState=0*) the fog effect

DATA.ENVIRONMENT.FOG.COLOR = iR, iG, iB

Determines the fog color, *iR* being the red component, *iG* - green component, *iB* - blue component. The *iR*, *iG*, *iB* parameters can take on values in the range from 0 to 255. The default color is white: *iR=iG=iB=255*.

DATA.ENVIRONMENT.FOG.NEAR = fDensity

Determines the fog density *fDensity* in percentage terms at the closer bound of the effect demonstration visibility. The visibility range is specified for each camera individually. The commands determining the range are described in the previous Section.

DATA.ENVIRONMENT.FOG.FAR = fDensity

Determines the fog density *fDensity* in percentage terms at the farther bound of the effect visibility.

3.4 Commands for Working with Light Sources

DATA.LIGHT.Light.ENABLE = iState

Switches on (*iState=1*)/off (*iState=0*) the light source *Light* in the scene. By default, this value is set as was specified in the scene.

DATA.LIGHT.Light.COLOR = iR, iG, iB

Determines the color of the light source *Light* in the scene, *iR* being the red component, *iG* - green component, *iB* - blue component. The *iR*, *iG*, *iB* parameters can take on values from 0 to 255. The default color is white: *iR=iG=iB=255*.

DATA.LIGHT.Light.MULT = fMult

Sets the light intensity *fMult*. It is the same parameter as *Multiplier* in *3D Studio MAX*. By default, the value is set equal to the value in the scene.

DATA.LIGHT.Light.RANGE = fStart, fEnd

Sets the visibility range or the distant fading area for the light source *Light*. It is the same parameter as *Far Attenuation* in *3D Studio MAX*. By default, the value is equal to the value in the scene.

DATA LIGHT.Light.ANGLE = fHotAngle, fFallAngle

Sets the range of angles regulating the ray parameters for the directional light source *Light* (the *Target Spot*, *Target Direct*, *Free Spot*, *Free Direct* types): *fHotAngle* is the light spot of the ray, *fFallAngle* is the scope of the light spot regulating the extent of the edge blur. The *fHotAngle*, *fFallAngle* parameters can take on values from 0 to 180 degrees. They are the same parameters as in *3D Studio MAX (Spotlight (Directional) Parameters: Hotspot/Beam, Falloff/Field)*. By default, the values are equal to the values in the scene.

☝ *When trying to apply the command DATA.LIGHT.Light.ANGLE to a nondirectional light source (the Omni type), an error message is displayed in the Debug Output window.*

3.5 Commands for Working with 3D Objects

DATA.NODE."Object".HIDE = iHide

Hides/shows an object named *Object* in the scene; the value of **1** for *iHide* corresponds to hiding the object, **0** – to showing it.

DATA.NODE."Object".POS = fX, fY, fZ

Sets a new position of *Object* by determining new *fX*, *fY* and *fZ* coordinates in the scene relative to the initial position: the coordinates {0, 0, 0} correspond to the initial position.

DATA.NODE."Object".ROT = fα, fβ, fγ

Rotates the *Object* relative to its initial position: rotation is specified in degrees by the *fα*, *fβ* and *fγ* Euler angles (rotation angles around the *X*, *Y*, *Z* axes).

DATA.NODE."Object".SCL = fa, fb, fc

Sets the *Object* scaling, multiplying its three dimensions by the *fa*, *fb* and *fc* coefficients.

DATA.NODE."Object".VEL = fMOVx, fMOVy, fMOVz, fROTx, fROTy, fROTz, fSCLx, fSCLy, fSCLz

Sets the speeds of the *Object* movement, rotation and scaling with the parameters *fMOVx*, *fMOVy*, *fMOVz*, *fROTx*, *fROTy*, *fROTz*, *fSCLx*, *fSCLy*, *fSCLz* by all axes, respectively (in units [units specified in the scene] per millisecond).

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

Sets a relative shift, rotation and scale of the *Object* with the *fMOVx*, *fMOVy*, *fMOVz*, *fROTx*, *fROTy*, *fROTz*, *fSCLx*, *fSCLy*, *fSCLz* parameters by all axes, respectively.

DATA.NODE."Object".LISTENER = "Listener"

This command assigns to the **Object** node another object named **Listener**, to which all the received shift, rotation, and scaling commands are sent from the **Object**.

DATA.NODE. « Object».RESET = 1

Resets *Object* to the initial position (state), which was when loading the scene.

DATA.NODE.«Object».OnCLICK = "Script"

Assigns a command *Script*, which is to be executed when clicking at an object named *Object* in the scene (see Section 10).

DATA.NODE.«Object».OnOVER = "Script"

Assigns a command *Script*, which is to be executed when moving the mouse cursor over an object named *Object* in the scene (see Section 10).

DATA.NODE.«Object».OnOUT = "Script"

Assigns a command *Script*, which is to be executed when moving the mouse cursor away from the object named *Object* in the scene (see Section 10).

3.6 Commands for Working with Materials

DATA.MATERIAL.MaterialName.CACHE = iState

Enables (1)/disables (0) caching textures for the **MaterialName** material. When caching is enabled, all recurring loads are loaded from the video card memory. Enabled by default.

DATA.MATERIAL.MaterialName.LIMIT = iMaxWidth, iMaxHeight, iMaxSize

Sets the parameters of displaying for all the **MaterialName** material textures assigned by the commands **DATA.MATERIAL.MaterialName.MAP** (see description below), **iMaxWidth** being the maximum possible width resolution, **iMaxHeight** – the maximum possible height resolution, **iMaxSize** – the maximum possible size (in megabytes).

DATA.MATERIAL.MaterialName.MAP = "GraphicsFile.ext"

or

DATA.MATERIAL.MaterialName.MAP.DIFFUSE = "GraphicsFile.ext"

Replaces the ***Diffuse*** texture of the **MaterialName** material with a texture from the **GraphicsFile.ext** file. The **gif**, **jpg**, **bmp**, **tiff**, **tga**, **ifl**, **ppt**, **swf**, **htm**, **avi** extensions are supported. After the initial loading of the static textures (files with the **gif**, **jpg**, **bmp**, **tiff**, **tga**, **ifl** extensions), they are saved in the video card RAM, the dynamic textures (files with the **ppt**, **swf**, **html** extensions) being saved in the operating system RAM).

When the command is executed, a report message *SYS.EVENT* = "*DATA.MATERIAL.MaterialName*" is displayed in the **Message (Debug Output)** window (see Section 20). The right part of the message can be used as a parameter of the command *SYS.WAIT* = "*Event*" for a command execution wait (see Section 8).

☝ *An attempt to apply this command to a material originally not having the **Diffuse** texture is treated as an error.*

DATA.MATERIAL.MaterialName.MAP.AMBIENT = "GraphicsFile.ext"

Replaces the ***Ambient*** texture of the **MaterialName** material with a texture from the **GraphicsFile.ext** file. The **gif**, **jpg**, **bmp**, **tiff**, **tga**, **ifl**, **ppt**, **swf**, **htm**, **avi** extensions are supported. After the initial loading of the static textures (files with the **gif**, **jpg**, **bmp**, **tiff**, **tga**, **ifl** extensions), they are saved in the

video card RAM, the dynamic textures (files with the **ppt**, **swf**, **html** extensions) being saved in the operating system RAM).

☝ *An attempt to apply this command to a material originally not having the **Ambient** texture is treated as an error.*

DATA.MATERIAL.MaterialName.MAP.REFLECTION = "GraphicsFile.ext"

Replaces the ***Reflection*** texture of the **MaterialName** material with a texture from the **GraphicsFile.ext** file. The **gif**, **jpg**, **bmp**, **tiff**, **tga**, **ifl**, **ppt**, **swf**, **htm**, **avi** extensions are supported. After the initial loading of the static textures (files with the **gif**, **jpg**, **bmp**, **tiff**, **tga**, **ifl** extensions), they are saved in the video card RAM, the dynamic textures (files with the **ppt**, **swf**, **html** extensions) being saved in the operating system RAM).

☝ *An attempt to apply this command to a material originally not having **Reflection** texture is treated as an error.*

DATA.MATERIAL.MaterialName.MAP.OPACITY = FileName

Adds transparency (alpha-channel) for the ***Diffuse*** textures of the material **MaterialName** from the file **FileName**. If the alpha-channel is absent in the loaded file, the transparency values are formed from the luminosity of the loaded file. The extensions supported are: **gif**, **jpg**, **bmp**, **tiff**, **tga**, **ifl**.

☝ *An attempt to apply this command to a material originally not having the **Diffuse** texture is treated as an error.*

DATA.MATERIAL.MaterialName.MAP = 0

Restores the initial material textures of the **MaterialName** object, as when loading the scene.

DATA.MATERIAL.MaterialName.SIZE = iWidth, iHeight

Changes the texture map size for the **TEXT** command. The texture is cleared.

DATA.MATERIAL.MaterialName.DIFFUSE = iR,iG,iB

Sets the material diffuse color; *iR*, *iG*, and *iB* vary from 0 to 255.

DATA.MATERIAL.MaterialName.AMBIENT = iR,iG,iB

Sets a light-independent diffuse color of the material; *iR*, *iG*, and *iB* vary from 0 to 255.

DATA.MATERIAL.MaterialName.SPECULAR = iR,iG,iB

Sets a reflected color of the material; *iR*, *iG* and *iB* vary from 0 to 255.

DATA.MATERIAL.MaterialName.SELF = iValue

Sets self luminous excitance of the material; *iValue* is changed from 0 to 100.

DATA.MATERIAL.MaterialName.POWER = iValue

Sets a value of the material reflected flare; *iValue* varies from 0 (flare is off) to 127.

DATA.MATERIAL.MaterialName.TRANSP = iValue

Sets transparency of the material; *iValue* varies from 0 to 100.

DATA.MATERIAL.MaterialName.ALPHA = 1/0

Sets/cancels a special flag of the **MaterialName** material, which indicates that all objects containing this material are objects with transparency – for correct sorting.

☝ *By default, the rendering procedure starts from the objects with materials without transparency (without alpha-channel) and is executed in the ascending z-order of the ambient sphere centers. Then, materials with transparency (alpha-channel) are visualized now starting from the largest z-order of the object ambient sphere and following in descending order. When sorting, the presence*

of the name prefixes influencing the order of sorting objects are taken into account (Section 8 of Creating 3D Scenes User's Guide).

RENDER.MATERIAL.MaterialName.MIPMAPLODBIAS = fBias

Changes the the image resolution on the *MaterialName* material. By default, the *fBias* value is equal to 0. With its growth, the degree of the image blur increases. When decreasing the parameter value, the image becomes less blurred.

DATA.MATERIAL.MaterialName.RESET = 1

Resets the initial texture for the **MaterialName** material after replacing it with the command **DATA.MATERIAL.MaterialName.MAP** or ***.TEXT** and deletes all textures loaded earlier for the material from the video card main memory.

DATA.MATERIAL.Reset = 1

Resets all object materials to their initial states (as when loading the scene) and deletes all textures that replaced initial ones from the video card main memory.

3.7 Commands for Working with 2D Text

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

Replaces the **MaterialName** texture with the text named *Text* using:

"*Font*" – the system font name;

iFontSize – the font size (in texels – a unit analogous to a pixel for textures);

iX the indent on the **X** axis;

iY the indent on the **Y** axis;

iR the font color (red component);

iG the font color (green component);

iB the font color (blue component).

iMix mode of mixing with the texture map:

1 – the text is added to the texture;

0 – the text replaces the texture.

The indent is counted off from the top-left corner of the texture, *iR*, *iG* and *iB* vary from 0 to 255. The default values are: the font – **System**, the size – 24, the color – white: *iR=iG=iB=255*. The other parameters are equal to 0. The text parameters can also be specified by separate commands (see below), then it is not necessary to specify them in the end of the **TEXT** command itself.

To apply this command, the material must have a texture, it being necessary to specify its texture coordinates (in *3D Studio Max*). If the material has a transparent texture and *iMix*=0, the text is placed on a transparent substrate.

DATA.MATERIAL.MaterialName.TEXT.FONT = "Font"

Sets a font for the **TEXT** command. "*Font*" is the system font name.

DATA.MATERIAL.MaterialName.TEXT.SIZE = iFontSize

Sets a size of the font for the command **DATA.MATERIAL.MaterialName.TEXT.FONT**. *IFontSize* is the font size (in texels – a unit analogous to a pixel for textures).

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

Sets an offset of the text for the **TEXT** command. The shift is counted off from the top left corner of the texture.

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

Sets a color of the text for the **TEXT** command, *iR*, *iG* and *iB* varying from 0 to 255.

DATA.MATERIAL.MaterialName.TEXT.OUTLINE = iTexels

Sets an edging of the text, *iTexels* being the width of the edging in texels; it can take on values from 0 (the edging is not drawn) on up. The edging color is black by default (see also the following command).

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

Sets a color of the text edging; *iR*, *iG* and *iB* vary from 0 to 255.

DATA.MATERIAL.MaterialName.TEXT.SHADOW = iTexels

Sets shadows for a text image of the *iTexels* size in texels. The *iTexels* parameter can take on any integer values. When 0, the shadow is not displayed. When other values, the *iShadow* parameter specifies the direction of casting the shadow: right-downward (when positive values of the *iShadow* parameter) or left-upward (when negative values of the *iShadow* parameter). The shadow color is black by default (see also the following command).

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

Sets a color of the text shadow; *iR*, *iG*, and *iB* vary from 0 to 255.

3.8 Commands for Working with Web Pages, Flash Animation and PPT Presentations

Web pages (files with the *.htm, *.html extensions), *PowerPoint* presentations and *Macromedia Flash* files can be used as a ***Diffuse Map*** for a material texture.

DATA.MATERIAL.Name.MAP = "File.ext"

The supported extensions are **htm**, **html**, **ppt** and **swf**. After the initial loading, files are saved to the RAM of the operating system.

DATA.MATERIAL.Name.MAP.FRAGMENT = iFrame

Changes slides in the PPT presentation:

IFrame = +1 – a switch to the next slide;

IFrame = –1 – a switch to the previous slide.

☝ *To display PPT files in the PowerPoint 95/97 format, it is necessary to previously install the standard ActiveX Control application (by launching the axplayer.exe file: http://activex.microsoft.com/activex/controls/ppoint/ppoint.asp http://activex.microsoft.com/activex/controls/ppoint/axplayer.exe).*

To display PPT files in a format later than PowerPoint 97 (Power Point 2000-2003), it is necessary to previously install the Microsoft Office PowerPoint package (2000-2003 versions) in the operating system.

☝ *To play back Macromedia Flash files, it is necessary to previously install the Shockwave ActiveX plug-in (http://www.macromedia.com/shockwave/download/).*

3.9 Commands for Working with 3D Text

☝ *A file of the *.tga or *.tiff format describing a font (see the next commands) must consist of 256 images, i. e. all characters of the font presented in a table of 16 rows by 16 characters in each. The characters can be of an arbitrary size (when observing the standard texture restrictions); they must be of the same height, though may differ by the width. The order of the characters is determined by the current ANSI single-byte coding. The coding of a character can be determined using the **Character Map** standard utility (**Start** > **Programs** > **Accessories** > **System Tools**). Enable the **Advanced View** option when converting character cods displayed in character tips from the hexadecimal system to the decimal system.*

Characters are to be arranged in a file from the top left corner to the right and downwards. The width of the top leftmost null character is used for determining the free space around all other

characters, therefore the null character must not contain images. It is recommended that the width of the null character be no less than 40% of the font height. Each character is separated from the adjoining ones by a completely transparent color – frame of any width (with alfa=0). The character itself must not have pixels with alfa=0.

DATA.FONT = "Font", "ContentFileName", fFontHeightRatio, fFontWidthRatio, iR, iG, iB, iOutline, iShadow, fQuality

Creates a font and sets it for a 3D text by default.

«Font» – the name of a ***.tga** or ***.tiff** file describing the font, or a system font name.

«ContentFileName» – the name of the text (*.txt) file with phrases – text substitutions in the UNICODE coding. The file format is key words in angle brackets that are followed by the substitution text. For example:

<Text1>Phrase1

<Text2>Phrase2

If fragments <T1> or <T2> occur in the **«Text»** text, they are to be substituted for the corresponding text from the *«ContentFileName»* file: Phrase1 or Phrase2. For example, as a result of the command **DATA.NODE. «Object». TEXT = «This is the text from the <T1> phragment, and this is the text from the <T2> phragment »** execution, there is to be displayed the following text:

"This is the text from the Phrase1 fragment, and this is the text from the Phrase2 fragment"

fFontHeightRatio, *fFontWidthRatio* – height and width aspect ratios of the created font characters. They can take on values from 0.0 to 1.0. Default values are 1 so that when an object is replaced by a text in a 3D scene, the text occupies the maximum possible space of the replaced object.

The *iR*, *iG*, *iB* parameters set a color of the font and can take on values from 0 to 255. The default color is white, that is: *iR= iG= iB=255.*

The *iOutline* parameter value sets the width of the text black edging. Units are determined by the size of the font used (see the *fQuality* parameter description below). The default value is 1 (when *fQuality=1*).

The *iShadow* parameter sets the text black shadow size. The shadow direction is determined according to the specified *iShadow* value: right downward if the *iShadow* value is positive or left upward if negative. Units are determined by the size of the font used (see the *fQuality* paremeter description below). The default value of the *iShadow* parameter is 1 (when *fQuality=1*).

The *fQuality* parameter allows to adjust the character size. This parameter may be significant for some oriental languages (Korean, Japanese, Chinese and others), where the number of hieroglyphs in a texture map can exceed the default value – 256 characters (character size of 64 texels, the texture map size of 1024 texels). When increasing the number of characters in a texture map, the size of thr characters is redused. When the size is reduced by half (32 texels), the texture map is scaled to 2048 texels. When further increasing the number of characters, the procedure repeats with the texture map consistent increasing up to 4096 texels.

When increasing the *fQuality* parameter, the character size in the texture is enlarged, which increases the time of creating the font, the created characters having a higher definition.

The default value of the *fQuality* parameter is 1, which corresponds to a character size of 64 texels and the texture of 1024 texels for 256 characters.

DATA.FONT.FontName = "Font", "ContentFileName", fFontHeightRatio, fFontWidthRatio, iR, iG, iB, iOutline, iShadow

This command creates a font which can be later on referred to as *FontName*.

For description of the parameters, see the specification of the previous command.

DATA.NODE."Object".FONT = "FontName"

Assigns a font named **FontName** for the object named *Object*.

DATA.NODE."Object".TEXT = "Text", fTextHeightRatio, fTextWidthRatio

Replaces the object named *Object* with the 3D text named **Text** in the scene. The text can consist of several lines separated by the '\n' characters.

The text is fitted in the object so that it does not exceed its limits under any conditions. If a text string is too long, it is automatically devided into several lines. The font is scaled so that the text always fits within the object bounds. The alignment of the text lines is centered by default (the **CENTER** mode) unless another was specified (see description of the next command **ALIGN**).

The *fTextHeightRatio* and *fTextWidthRatio* parameters set the text scaling ratio from the object original size and can take on values from 0.0 to 1.0. The *fTextHeightRatio* parameter sets the maximum possible scale by the font height. If the text exceeds the object bounds, the font height is reduced automatically so that the text fits in the object. The width of the characters is proportionate to their height but can be adjusted by the *fTextWidthRatio* parameter.

The command **DATA.NODE."Object".TEXT = 0** removes the 3D text and restores the object original state.

The font used is either default set by the command **DATA.FONT**, or set by the command **DATA.NODE."Object".FONT = "FontName"** for the specific object.

It is possible to use key words in angle brackets in the text; they will be replaced with a **UNICODE** text from the **"ContentFileName"** file, whose name is specified in the **DATA.FONT** command. For example:

DATA.NODE.Example.TEXT = "<T1>"

In addition to Latin characters, it is possible to use the local system language characters in the text. For Latin and Cyrillic, all the characters are available, for oriental languages, only the basic set of hieroglyphs being available:

1. Japanese – (0x3040...0x30FF) Hiragana + Katakana
2. Korean – (0x3130...0x318F) Hangul Compatibility Jamo
3. Chinese – (0x4E00...0x4EFF) Chinese Simplified
4. Thai – (0x0E00...0x0E7F)

DATA.NODE."Object".ALIGN = "Mode"

sets the text alignment mode within the *Object* bounds.

"CENTER" – horizontal center alignment.

"RIGHT" – align right.

"LEFT" – align left.

"JUSTIFY" – justification.

4 Track commands

For any scene object having animation (trajectory), it is possible to specify one or several tracks determining particular sections of the trajectory.

When playing a track animation is complete, a message of the event completion SYS.EVENT="TRACK.Track_name" (Section 8) is displayed in the Message (Debug Output) window. The right part of the message can be used as a parameter of the command SYS.WAIT ="Event" for the track playback wait. For example: SYS.WAIT = Track.Sport.

TRACK."Track".NODE = "NodeName"

This command specifies a track named "*Track*" to control the animation of the object(s) "*NodeName*". The animation of the "*NodeName*" object(s) is excluded from the general animation and becomes controlled by the specified track. The current frame is set to 0. If the object is not specified, an object with a name identical with the track name is used in the track commands.

TRACK."Track".NOTIFY = "Event"

Sets a text of the "*Event"* message that is to be displayed in the **Message (Debug Output)** window when playing track named *Track* is complete; by default, when playing track named *Track* is complete, a message *SYS.EVENT* = "TRACK.*Track*" is displayed in the **Message** window.

TRACK."Track".LOOP = iNLoop

Enables/disables looping of the *Track* playback when executing the commands **TRACK."Track".START** or **TRACK.Track.PLAY**, which are discussed below.

iNLoop = 1 refers to a looped playback of the track, i. e. the animation is continuously played up to the moment when interrupted;

iNLoop = 0 – playing the animation is not looped, i. e., the track playback command is to be executed once.

TRACK.Track.RANGE = iFrom, iTo, iRepeat

Adds a set of frames from the *iFrom* to the *iTo* to the queue of the *Track* track *iRepeat* times. The command **TRACK.Track.PLAY = iRepeat** can be used to play back the set.

TRACK.Track.MATERIAL = MaterialName

Creates a track named *Track* to display the *MaterialName* material. The command allows you to display the files in part when a video or **IFL** file is assigned as a texture map of the material.

TRACK."Track".START = iFrom, iTo, iRepeat

Plays back the *Track* from the *iFrom* to the *iTo* frames *iRepeat* times. If *iFrom* > *iTo*, the track is played backwards. If the *iTo* and *iRepeat* parameters are not specified, the track is set to the **iFrom** frame.

TRACK.Track.PLAY = iRepeat

Plays back the track named *Track* (or a sequence of frames of *Track* that is specified by the command **TRACK.Track.RANGE = iFrom, iTo, iRepeat**) *iRepeat* times.

TRACK.Track.STOP = iNFrame

Stops the *Track* playback at the *iNFrame* regardless of the specified number of repeats.

TRACK.Track.GOTO = iNFrame

Plays back the track named *Track* from the current frame to the *iNFrame*, the number of repeats specified by the **RANGE** command being ignored. If *iNFrame* is larger than the current frame number, the track is played forward. If

iNFrame is less than the current frame number, the track is played backwards up to the frame *iNFrame.*

TRACK.Track.QUEUE = iState

Enables (1)/disables (0) queuing for the track.

Each track has its command queue: if queuing is enabled, each command is added to the queue and executed automatically when the previous one is complete. If queuing is disabled, the command is executed immediatelly. By default, queuing is enabled.

It is possible to replace queuing with wait; instead of:

TRACK.A.QUEUE = 1

TRACK.A.START = 100, 200

TRACK.A.START = 200, 100

TRACK.A.START = 100, 200

it is possible to write:

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.Track.TARGET = «Command», fVelocity, "DataFormat"

Forms a track named *Track* – execution the command *Command* at a speed determined in each frame of the track by the value “the track frame number × *fVelocity*”. The *“DataFormat”* value sets the parameter of the command *Command*, for example, the commands

TRACK.Track.TARGET = “ACTION.START”, 1, “Action%d”

TRACK.Track.START = 1

TRACK.Track.START = 2

TRACK.Track.START = 3

are equivalent to the commands

ACTION.START = Action1

ACTION.START = Action2

ACTION.START = Action3

TRACK.Track.DELETE = 1

Deletes the *Track* contents.

TRACK.DELETE = 1

Deletes all the named tracks.

TRACK.RESET = 1

Stops the animation. This command stops all the tracks and clears the queues. It is executed by default at the project initialization started by the **Init** button on the toolbar of the application main window or when switching to the ***LiveAction*** operating mode (see the ***HotActions*** User’s Guide, Section 3.1.9).

5 Morphing commands

RENDER.MORPH = "SourceObjectName", "DestObjectName", fTime

Performs the morphing of an initial object named *SourceObjectName* to a target object named *DestObjectName* (usually hidden in the scene) in *fTime* seconds. The objects must have the same number of vertices. When the morphing is complete, the message **RENDER.MORPH. "SourceObjectName"** of the event completion is displayed.

RENDER.MORPH.RESET = 1

Resets all morphed objects to their initial states (as they were when opening the scene). This command is executed by default at the project initialization started by the **Init** button on the application main window toolbar or when switching to the ***LiveAction*** operating mode (see the ***HotActions*** User's Guide, Section 3.1.9).

6 Commands for Displaying Information about the Scene

RENDER.DUMP=1

Displays information about the loaded scene in the **Message (Debug Output)** window: the video card memory amount used for loading the textures (both local memory of the graphics accelerator and a part of the main memory (non-local), whose value can be specified in the BIOS of the computer).

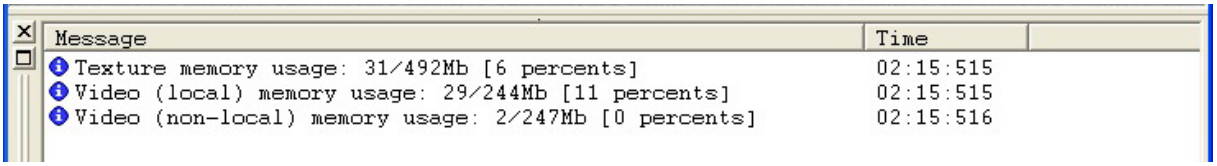

Figure 1. Information about the scene opened in *HotActions*. The information is displayed in the *Debug Output* window when executing the command RENDER.DUMP=1.

RENDER.DUMP.VIDEO = 1

Displays information about the video streams in the **Message (Debug Output)** window (see description in the next Section): signal sources for the streams of the LIVE type and the names of the files with the data for the streams of the FILE and MSDS types.

7 Commands for Image Rendering

7.1 Commands for Working with Video Streams

The following commands are common when working with the video streams of all types.

Examples of controlling a live video stream of the connected video source and a video file stream are discussed in Sections 8.3.1 and 8.3.4 respectively of *Creating 3D Scenes User's Guide*.

RENDER.VIDEO.VideoStreamName.CREATE = iParam

Creates a video stream with the name *VideoStreamName, which* can be as follows:

LIVE_1, **LIVE_2**, **LIVE_3**, etc. – video streams from the connected input boards (*FD300* or *Aja XENA*; the settings are described in *Video & Audio Settings*);

FILE_1, **FILE_2**, **MSDS_1**, **MSDS_2** etc. – video streams from video and graphics files via the *DirectShow* filters, as well as from external devices. It is allowable to play *.avi, *.mpg, *.wmv, and other files via the video streams if the codec for playing is installed in the operating system. It is also possible to display the *.jpg, *.tga, *.tiff, *.gif graphics images and others.

iParam can take on one of the following values:

iParam = 1 for creating a video stream;

iParam = 0 for deleting a created video stream.

A video stream is created in the stopped state (see below descriptions of the commands **RENDER.VIDEO.VideoStreamName.START = 1** and **RENDER.VIDEO.VideoStreamName.STOP = 1**).

When creating **FILE_*** and **MSDS_*** video streams, the data are not attached to them (see below descriptions of the commands **RENDER.VIDEO.VideoStreamName.DATA = "Filename", iStartFrame, iLength,**
RENDER.VIDEO.VideoStreamName.DATAINOUT = "Filename", iFrom, iTo).

RENDER.VIDEO.VideoStreamName.DESTROY = iParam

Deletes the created video stream named *VideoStreamName* if *iParam* = 1 or creates a video stream named *VideoStreamName* if *iParam* = 0.

RENDER.VIDEO.VideoStreamName.RESET = 1

Resets the *VideoStreamName* video stream to the default state.

The commands **RENDER.VIDEO.<LIVE>.RESET = 1**, **RENDER.VIDEO.<FILE>.RESET = 1**, **RENDER.VIDEO.<MSDS>.RESET = 1** are executed by default at the project initialization started by the **Init** button of the application main window toolbar or when switching to the ***LiveAction*** operating mode (see *HotActions. User's Guide*, Section 3.1.9).

RENDER.VIDEO.VideoStreamName.START = iParam

When *iParam* = 1, starts playing a video stream named *VideoStreamName*. The *SYS.EVENT = "VIDEO.VideoStreamName.START"* message about this event is displayed in the **Message (Debug Output)** window (see Section 8).

When *iParam* = 0, playing the video stream named *VideoStreamName* is stopped (the same as the **RENDER.VIDEO.VideoStreamName.STOP =1** command discussed below). The *SYS.EVENT = "VIDEO.VideoStreamName.STOP*" message about the event is displayed in the **Message (Debug Output)** window (see Section 8).

☝ *A second start of a file goes faster, since it is played from the system main memory.*

RENDER.VIDEO.VideoStreamName.PAUSE = 1

Stops the *VideoStreamName* video stream. When executed, the *SYS.EVENT = "VIDEO. VideoStreamName.PAUSE"* message is displayed in the **Message (Debug Output)** window (see Section 8).

RENDER.VIDEO.VideoStreamName.STOP = iParam

When *iParam* = 1, stops playing the *VideoStreamName* video stream. The *SYS.EVENT = "VIDEO.VideoStreamName.STOP*" message about the event is displayed in the **Message (Debug Output)** window (see Section 8).

When *iParam* = 0, starts playing a video stream named *VideoStreamName* (analogous to the **RENDER.VIDEO.VideoStreamName.START =1** command described above). The *SYS.EVENT* = "*VIDEO.VideoStreamName.START"* message about the event is displayed in the **Message (Debug Output)** window (see Section 8).

RENDER.VIDEO.VideoStreamName.FORMAT = "Format string"

This command is to be executed after a video stream is created, and before it is played back. It specifies the video stream format with the "*Format string*" string, where the parameters can be specified separated with a comma. For video streams from the video input boards (**LIVE_1**, **LIVE_2**,…):

- **ALPHA** enables using the chroma keying. As a result, the background colors of the input video look 'transparent'. The chroma key parameters are adjusted in the *KeyConfigPro* dialog box (see *Video and Sound Settings*). The value is set by default. The command also enables the mask specified in the *KeyConfigPro* dialog.
- **NOALPHA** disables the chroma keying for the input video signal. Disables the mask specified in the *KeyConfigPro* dialog.
- **CROP** enables using the mask for cropping frame edges. The value is set by default.
- **NOCROP** disables using the mask for cropping frame edges.
- **MIPMAP** – enables trilinear (pyramidal) filtering for the elimination of distortions when displaying reduced scale video and movements in a scene. The filtering degree can be adjusted by the command **RENDER.MATERIAL.MaterialName.MIPMAPLODBIAS = fBias** (see description in Section 3.6 *Commands for Working with Materials*).
- **NOMIPMAP** – disables the trilinear (pyramidal) filtering mode. By default, the mode is disabled.
- **SYNC** – setting the field parity synchronization adjustment mode of the input and output frames.
- **NOSYNC** – disables the ability to synchronize the input and output frame fields.

For the **FILE_*** and **MSDS_*** file video streams, the following parameters are used additionally:

- **ADD** – for file streams from AVI (**FILE_1**, **FILE_2**,...) enables the mode in which files are added to the stream queue, i.e., the files added with the **RENDER.VIDEO.VideoStreamName.DATA** command (the command is discussed below) are played in the order they were added.
- **REPLACE** enables the mode in which files in the stream queue are superseded, i. e. each file added by the **RENDER.VIDEO.VideoStreamName.DATA** command (see below) interrupts the playback of the previous file and clears the queue before it starts. This value is set by default.
- **LOOP** enables the loop mode of a video stream playback.
- **NOLOOP** disables the loop mode of a video stream playback. The value is set by default.
- **AUDIO** – a video stream is played back along with its audio track.

- **NOAUDIO** – a video stream is played back without its audio track. The value is set by default.
- **ALPHA** enables the mode of displaying transparency for AVI files with alpha channel or indefinite channel (for example, when compressing with the SoftLab-NSK Forward JPEG + Alpha (FRWT) codec).
- **NOALPHA** disables the mode of displaying transparency for video files.
- **AUTOALPHA** – the video texture is to be displayed transparent, if the video file contains an image with transparency (alpha-channel). The value is set by default.

☝ *Displaying image with transparency is supported only for AVI files. When trying to play back an MPEG or WMV file in the ALPHA format, the image is not displayed in video texture.*

- **LFF** – the Lower Field First mode of playing a video file, i.e., when playing a video file, the lower field is to be displayed first. The mode is set by default for the **FILE** type video streams.
- **UFF** – the Upper Field First mode of playing a video file, i.e., when playing a video file, the upper field is to be displayed first.
- **NOFIELDS** – image is displaed full frame, without splitting into fields.
- **AUTOFIELDS** – for the ***MPEG*** files the field order is determined by the file compression type. For the rest of the file types, the field order in this mode depends on the presence of information in the system. Absent information, a video file is played in the progressive scanning mode. The mode is set by default for the **MSDS** video streams.

☝ *Owing to the specificity of determining the field order in the AUTOFIELDS mode, we recommend that you always explicitly specify the field order – LFF, UFF, or NOFIELDS when playing files with interlacing.*

- **REMOTE** – forming a video stream for displaying image from a remote computer (see below the description of the command **RENDER.VIDEO.VideoStreamName.DATA = Remote_Computer_IP**).
- **CAPTURE** – formatting a video stream when working with external video sources: web camera, the **Vision RGB-PRO** video capture board, etc. (see below the description of the command **RENDER.DUMP.MSDS.SOURCES = 1**).
  - **LOCAL** – formatting a video stream after working with external video sources (see above description of the **CAPTURE** video stream format) to restore the correct mode of working with video files located on the computer disk. The value is set by default.

When creating a video stream (see above the description of the command **RENDER.VIDEO.VideoStreamName.CREATE = iParam**) the following formats are set by default:

for **LIVE_*** – **ALPHA, CROP, LOWRES, NOMIPMAP, SYNC**;

for **FILE_*** – **NOLOOP, NOAUDIO, LOWRES, REPLACE, AUTOALPHA, LFF**;

for **MSDS_*** – **NOLOOP, NOAUDIO, LOWRES, REPLACE, AUTOALPHA, AUTOFIELDS, LOCAL**;

☝ *The execution of the command* ***RENDER.VIDEO.VideoStreamName.FORMAT = FormatString*** *stops playing a video stream if it was previously started by the command* ***RENDER.VIDEO.VideoStreamName.START = 1***.

The following commands are specific for working with <u>video streams from files</u>:

RENDER.VIDEO.VideoStreamName.DATA = "Filename", iStartFrame, iLength

This command attaches data to the video stream named *VideoStreamName* (which takes on values **FILE_*i*** or **MSDS_*i*** where ***i*** is the stream number): adds frames from the video file *"Filename"* to the file video stream *VideoStreamName* starting from *iStartFrame*, *iLength* frames. If both parameters are equal to 0 or absent, the entire file is added.

(using the command for working with external sources is discussed below).

RENDER.VIDEO.VideoStreamName.DATAINOUT = «Filename», iFrom, iTo

This command attaches data to the video stream named *VideoStreamName* (which takes on values **FILE_*i*** or **MSDS_*i***, *where* ***i*** is the stream number)*: adds frames from the video file «Filename»* to the file video stream *VideoStreamName* starting from the *iFrom* frame to the *iTo* frame. If the *iFrom*, *iTo* parameters are not specified or *iFrom = iTo*, the whole file is added to the video stream.

*For files with the *.wmv extension, the fragment frame-by-frame mode is not supported. When trying to add data from such a file to the video stream having specified iFrom and iTo values for it, the corresponding message is displayed in the Message (Debug Output) window.*

RENDER.VIDEO.VideoStreamName.VOLUME = iVolume

Sets an *iVolume* value of the sound volume when playing video files via the *VideoStreamName* video stream (**MSDS**). *iVolume* can take values from 0 (absence of sound – 10000 dB) to 100 (- 0 dB). The value is set via the system of prompts 50 (-5000 dB). When specifying a value out of the allowable range, it is to be set as one of the closest values: 0 or 100.

RENDER.DUMP.MSDS.SOURCES = 1

This command displays a list of all video sources of the *DirectShow* filter (Figure 2) in the **Message (Debug Output)** window. The list can contain both external sources (for example, the video capture board **Vision RGB-PRO Capture** or the Web camera **Dual-Mode USB Camera**), which can be specified as a source of video signal in commands and internal sources, whose resources are used when working in the **HotActions** application (**SLTM Dshow Video Capture board 1, SLTM Dshow Video Output Capture board 1**, etc.). When further working with external video sources, it is possible to specify either the full name specified in the line, e.g., **RENDER.VIDEO.MSDS_1.DATA="Vision RGB-PRO Capture"**, or individual key words from the line, e.g., **RENDER.VIDEO.MSDS_1.DATA="USB Camera"**.

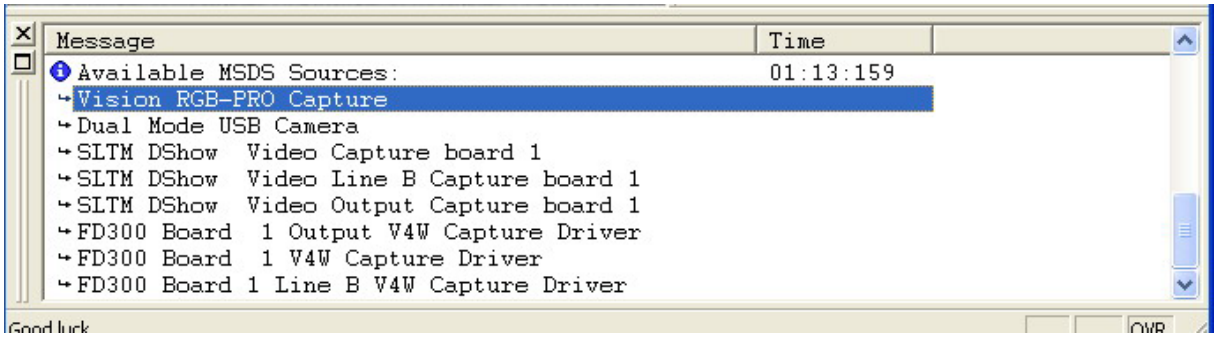

Figure 2. A list of the DirectShow filter video sources displayed in the Message (Debug Output) window when executing the command RENDER.DUMP.MSDS.SOURCES=1.

RENDER.VIDEO.VideoStreamName.DATA = Device

This command attaches data with the image from the external video source *Device* (see description of the previous command) to the *VideoStreamName* video stream (of the **MSDS_*i*** type, where ***i*** is the number of the stream).

The following command is specific for rendering image from a remote computer:

RENDER.VIDEO.VideoStreamName.DATA = Remote_Computer_IP

This command attaches the *VideoStreamName* video stream (of the **MSDS_*i*** type, where ***i*** is the number of the stream) to the image coming from a remote computer with an IP address specified by the *Remote_Computer_IP* parameter. For information about the corresponding setting of the remote computer, see Section 14.

RENDER.VIDEO.VideoStreamName.SIZE= fsizeX, fsizeY

Sets the video resolution for the video stream named *VideoStreamName* (of the **MSDS_*i*** type, where ***i*** is the number of the stream):

fsizeX – the horizontal resolution;

fsizeY – the vertical resolution.

This command sets the video resolution of those supported by the specified video source (see description of the next command), which corresponds to the specified values *fsizeX*, *fsizeY*.

7.2 Using Video Streams in the Scene

The following command is used for working with materials in the scene:

RENDER.MATERIAL.MaterialName.SOURCE = VideoStreamName

This command assigns the *VideoStreamName* video stream (**LIVE_1** or **FILE_2**. etc.) to the *MaterialName* material. The video texture is to be overlaid on all the objects with this material in the scene. Absent a video texture with such a name, the last 'usual' texture loaded for the material is restored (that is, the initial texture from the 3D scene, or its substitution from a file or a text if the commands **DATA.MATERIAL.MaterialName.MAP** or ***.TEXT** were used).

Using a video texture proposes the following steps to be done:

1. Creating a video stream

RENDER.VIDEO.LIVE_1.CREATE = 1.

2. Starting a stream for the video texture

RENDER.VIDEO.LIVE_1.START = 1.

3. Substituting a VIDEO material for the video stream in the scene

RENDER.MATERIAL.VIDEO.SOURCE = LIVE_1.

7.3 *3D Overlay* Mode of Rendering Material Textures

The interlaced scanning mode used in television may cause flicker noise when producing small details. A filter used to prevent this is based on the output image softening.

In some cases, when the texture resolution (for example, of a video stream from a camera or file) coincides with the resolution of the material on which it is overlaid, the image softening becomes visible.

The solution to the problem is the ***3D Overlay*** mode of rendering textures, in which the texture coordinates (for example, a close-up of the actor image) are re-counted projectively from the current virtual camera. The result of the procedure is that a texture texel coincides exactly with a pixel of the screen. The rest of the 3D scene in which the texture is placed is drawn as usual.

Video texture (for example, the actor image) remains in the scene on its initial object and it is still possible to put different objects ahead of it. Thus, this mode results only in refinement of the texture rendering.

☝ *It is recommended to create the size of the full screen textures for working in the 3D Overlay mode. The monitor and texture resolutions coinciding, the transfer to the mode is executed automatically, it being not required to execute the commands described below.*

RENDER.MATERIAL.MaterialName.OVERLAY = iMode, fPeriod

The **iMode** parameter enables/disables the ***3D Overlay*** mode for rendering the texture of the **MaterialName** material: 1 or 0, respectively.

The **fPeriod** parameter sets the time (in seconds) for the cross fade which is performed to smooth the switch between the ***3D Overlay*** mode and ordinary texture rendering.

This command is asynchronous, i. e., the system does not wait for the end of its execution. The **MATERIAL.MaterialName.OVERLAY** event is generated when the mode is actually enabled or disabled. It is generated immediatelly if **fPeriod**=0.0 or the command sets a switch to the mode that is already active.

7.4 *3D Overlay* Mode Additional Commands

If a frame is not drawn completely during the assigned period of time, the previous frame is displayed instead of it. That is, the frame that has not enough time to be drawn is dropped. Accordingly, from the next frame, the field parity used in the generated frame does not coincide with the output field parity in all materials displayed in the ***3D Overlay*** mode.

The following commands were introduced for the automatic correction of this situation:

RENDER.SYNC = iSync, iSkip

If *iSync*=0, the manual mode of switching parity is used. In this case, it is necessary to control whether the fields are displayed correctly and call when necessary the command **RENDER.FLIP** described below .

If *iSync* is greater than 0, the field parity is adjusted automatically, at that, the *iSync* parameter value indicates the number of asynchronous frames in succession, after which it is necessary to resynchronize. The *iSkip* parameter sets the way of synchronization:

iSkip = 1 – by dropping a half-frame of the input video;

iSkip = 0 – by changing the output field parity of the input video.

RENDER.FLIP = 1

If the automatic synchronization is disabled (see the previous command), this command changes the output field order for all the overlaid materials.

8 System Commands

SYS.WAIT = "Event" or

SYS.WAIT.EVENT = "Event"

Stops the script execution until the *Event* completion.

SYS.EVENT = "Event"

Registration of the *Event* completion. This command starts the script execution deferred by the **SYS.WAIT = "Event"** or **SYS.WAIT.EVENT = "Event"** command.

SYS.INFO="Wait"

Displays information about all unfinished events **SYS.WAIT = "Event"** or **SYS.WAIT.EVENT = "Event"** in the **Message (Debug Output)** window. After the message about an unfinished command **SYS.WAIT = "Event"**, a list of the commands that should have been executed after the end of it is displayed in the **Message** window.

SYS.DELAY = fTime

Delays the script execution for *fTime* seconds.

SYS.RESET = 1

Cancels waiting for any events. This command is executed by default at the project initialization started by the **Init** button on the toolbar of the application main window or when switching to the ***LiveAction*** mode (see the ***HotActions*** User's Guide, Section 3.1.9).

SYS.SPLASH = Text

Displays a dialog panel with the **Please, wait...** message and the **Text** text in the working area of the application. The panel can be hidden by the command **SYS.SPLASH = 0**.

SYS.OUTPUT = "Message"

Displays a text *Message* in the **Debug Output** window.

SYS.OUTPUT.FILTER = Word1,Word2,...

Allows to perform selective trace into the ***Debug Output*** window. Only the messages about the execution of commands containing words from the **Word1, Word2, ...** list, i.e., the words **Word1**, **Word2** and so on will be displayed in the window.

SYS.MACRO.Name = "Data"

Macro definition. The word **Name** is to be replaced with **"Data"** in the texts of all scripts. The command **SYS.MACRO = ""** cancels all macro definitions. It is recommended to prefix the names of macro definitions with the "**$**" character (see Section 17).

☝ *If a text is used as the* ***"Data"*** *macro definition, the name of the* ***Name*** *macro must be quoted in all commands, for example,* ***DATA.NODE.Text_node.TEXT="$text".***

SYS.ACTION = "File",fPeriod

Executes commands from the **"File"** text file. If the *fPeriod* value is set nonzero, the commands from the file are executed with the period *fPeriod* (in seconds). If the period is not specified, the commands are executed each time the file is changed (and the changes are saved). The file can be loaded not only from the disk but also from the Internet via HTTP, FTP and POP3. For that, it is sufficient to type the address in the standard URL form. For example:

"http://www.softlab-nsk.com/actions.txt"

"ftp://www.softlab-nsk.com/actions.txt"

"ftp://user:password@www.softlab-nsk.com/actions.txt"

"mailto: user:password@softlab-nsk.com"

As when launching usual ***Actions***, the corresponding message about the event completion is displayed in the **Message (Debug Output)** window after the execution of each command from the file.

☝ *If several commands* ***SYS.ACTION = "File",fPeriod*** *have been executed, valid is the one executed last. Of two executed commands SYS.ACTION = File1 and SYS.ACTION = File2, valid is only one, that is, only commands of one of the files are to be executed after the changes in text files File1, File2.*

SYS.ACTION.RECORD = "Filename"

This command initiates recording the *Debug Output* tracing (see Section 20) to the file named **"Filename"** (if **"Filename" = 0**, the recording is stopped). When the drawing of each frame is complete, the command of the **SYS.DELAY = {time of frame}** format is added to the file.

SYS.ACTION.FILTER = "command1", "command2"...

This command sets a list of commands for tracing to the file, that is, commands beginning with **"command1", "command2"** are to be recorded.

☝ *To exclude the tracing of a command specific call,it is sufficient to prefix the command with the "@" character. That may be necessary to avoid 'littering'* ***Debug Output****, for example, with a continuous stream of commands from the joystick.*

9 Commands for Working with Sound

SOUND.DIRECTORY = "Path"

Sets the path named *Path* to the working directory for audio files.

SOUND.Name.OPEN = "FileName"

or

SOUND.Name.FILE = "FileName"

Creates an audio stream named *Name* and attaches it to the audio file named *FileName.* All the audio streams are played back in *DirectShow*.

SOUND.Name.PLAY = iMode

Stops (*iMode*=0) and starts (*iMode*=1) playing the audio stream named *Name.*

SOUND.Name.NOTIFY = "Event"

Sets the text *"Event"* of the message that is to be displayed in the **Message (Debug Output)** window when the *Name* audio stream playback is complete. The *SYS.EVENT = "SOUND.Name"* message is displayed in the **Message** window by default when the stream playback is complete.

SOUND.Name.LOOP = iMode

Enables (*iMode*=1) and disables (*iMode*=0) the loop mode of playing the *Name* audio stream.

SOUND.Name.PAUSE=1

Pauses playing the *Name* audio stream.

SOUND.RESET = 1

Stops playing all the audio streams and deletes all the named audio objects. It is executed by default at the project initialization started by the **Init** button on the toolbar of the application main window or when switching to the ***LiveAction*** mode (see the ***HotActions*** User's Guide, Section 3.1.9).

10 Commands for Controlling the mouse cursor

The commands discussed in this Section are valid if the studio image is rendered via the second output of the video card (**Display Adapter** is selected from the **Display device** drop-down list in the *Render Options* dialog box; see section 8 of HotActions User's Guide). For example, when working with HD video in the Virtual Studio (see section 11 of HotActions User's Guide).

CONTROL.MOUSE.SELECT = iMode

When *iMode=1,* enables you to select objects in the scene with the mouse in the ***LiveAction*** working mode. When *iMode = 0,* the ability is disabled.

CONTROL.MOUSE.CURSOR = sMode

Determines the shape of the mouse cursor. When *sMode = HAND,* the cursor takes the shape of a hand; when *sMode = ARROW,* it takes the shape of an arrow. The default is *ARROW*.

CONTROL.MOUSE.SELECT.ALPHA = iMode

When *iMode=1,* enables you to select objects with transparency with the mouse. The ability is enabled only when the **CONTROL.MOUSE.SELECT = 1** command described above has been executed. When *iMode = 0,* the ability to select objects with transparency is disabled.

11 Commands for Working with the Joystick

JOYSTICK.Name = iID

Associates a joystick that has the *iID* identifier with the name *Name*:

iID can take on values from 1 to the number of the joysticks attached, that is, this command is used to name joysticks by their serial numbers. Therefore, the *iID* of a particular joystick can be identified experimentally, and once determined it remains the same if joysticks are not reattached to the computer;

if the *iID* values of some joysticks coincide, the last joystick assigned to the *iID* is used.

JOYSTICK.Name.MOVE = "Commands"

Sets the *Commands* command execution at moving the handle or slider of the joystick named *Name*, for example, the command **JOYSTICK.JOY.MOVE="DATA.NODE<CURRENT>.POS"** means that a movement of the joystick *JOY* handle or slider causes executing the command **DATA.NODE<CURRENT>.POS**, i.e., shifting the current selected object. The range of the object shift is determined by the next command (see below). All the values are equal to 0.5 by default.

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

Sets the sensitivity to the slider movement or handle movement and rotation for the *Name* joystick. It is set by six values in the range from 0.0 to 1.0 by each axis (by three values for the movement and rotation).

JOYSTICK.Name.BUTTON.Butt = iID

Assigns an integer identifier *iID* to the button named *Butt* of the *Name* joystick for further use in commands. If the *iID* values of some buttons of a joystick coincide, the last button assigned to the *iID* is used.

JOYSTICK.Name.BUTTON.Butt.UP = "Commands", "Argument"

Sets the left (*Commands*) and the right (*Argument*) parts of the command that is executed when the *Butt* button of the *Name* joystick is released. For example, the command

JOYSTICK.Name.BUTTON.Butt.UP = "TRACK.Track.PLAY", "0" means that the release of the *Butt* button executes the command **TRACK.Track.PLAY=0**, i.e., stops the track named *Track.*

JOYSTICK.Name.BUTTON.Butt.DOWN = "Commands", "Argument"

Sets the left (*Commands*) and the right (*Argument*) parts of the command that is executed when the *Butt* button of the *Name* joystick is pressed. For example, the command

JOYSTICK.Name.BUTTON.Butt.DOWN = "SOUND.Name.PLAY", "1"

means that pressing the *Butt* button executes the command **SOUND.Name.PLAY = 1**, i.e., starts the *Name* audio stream playback.

JOYSTICK.Name.BUTTON.Butt.HOLD = "Commands", "Argument"

Sets the left (*Commands*) and the right (*Argument*) parts of the command that is executed when keeping the *Butt* button of the *Name* joystick pressed, the commands being given in every frame.

JOYSTICK.Name.PATH = "Commands"

Sets the line of commands *Commands* with the values cofigured by the commands *AXIS, SLIDER, POV* (see below). The line of commands is sent for execution on the movement of the joystick handle or sliders and on pressing its buttons.

For example: *JOYSTICK.Name.PATH = "DATA.NODE.<CURRENT>"*.

JOYSTICK.Name.AXIS."Axis" = "Dest", fMinRange, fMaxRange

Sets a range of values and the line added to the commands set by the *JOYSTICK.Name.PATH* command (see above):

Name – the joystick name;

Axis – the joystick axis name (for example, **X**, **Y**, **Z** for the axes of movement, **RX**, **RY**, **RZ** for the axes of rotation);

Dest – a line that specifies the way the values received from the joystick handle are interpreted for the commands set by *JOYSTICK.Name.PATH*;

(for example, **POS.X**, **POS.Y**, **POS.Z** for interpreting the values as movements along the corresponding axes, **ROT.X**, **ROT.Y**, **ROT.Z** for interpreting them as rotations);

fMinRange, fMaxRange set a range of values for the current joystick axis (from 0.0 to 1.0).

For example, the command *JOYSTICK. "JOY".AXIS. "X" = "POS.Y", 0.5, 1.0*

sets that a movement of the joystick handle along the X axis, for example by 0.6, sends the following command to the system (if the command *JOYSTICK.Name.PATH = "DATA.NODE.<CURRENT>"* was executed before this):

DATA.NODE.<CURRENT>.POS = 0, 0.6, 0.

JOYSTICK.Name.SLIDER.iNum = "Dest", fMinRange, fMaxRange

Sets a range of values and the line added to the commands set by *JOYSTICK.Name.PATH* (see above) for execution when some events occur from the corresponding slider:

Name – the joystick name;

iNum – the joystick slider number (1,2,...). If a number is dropped, the slider is interpreted as number 1;

Dest – a line that specifies the way the values received from the corresponding slider are interpreted for the commands set by *JOYSTICK.Name.PATH*. For example, **POS.X, POS.Y, POS.Z** for interpreting the values as movements along the corresponding axes, and **ROT.X, ROT.Y, ROT.Z** for interpreting them as rotation;

fMinRange, fMaxRange set a range of values for the particular slider (from 0.0 to 1.0).

For example, the command *JOYSTICK."JOY".SLIDER.1 = "ROT.Z",0.5,1.0* determines that a movement of the slider along the X axis, for example by 0.6, sends the following command to the system (if the command *JOYSTICK.Name.PATH = "DATA.NODE.<CURRENT>"* was executed before this*):*

DATA.NODE.<CURRENT>.ROT = 0, 0, 0.6.

JOYSTICK.Name.POV.iNum."Axis" = "Dest", fMinRange, fMaxRange

Sets a range of values and the line added to the commands set by *JOYSTICK.Name.PATH* (see above) for execition when some events occur from the corresponding ***Point of View Hat*** joystick:

Name – the joystick name;

iNum – the ***Point of View Hat*** joystick number (1,2,...). If the number is dropped, the joystick is interpreted as ***Point of View Hat*** number 1;

Axis – the ***Point of View Hat*** axes (X or Y);

Dest – a line that specifies the way the values received from the corresponding ***Point of View Hat*** joystick are interpreted for the commands set by *JOYSTICK.Name.PATH* (**POS.X, POS.Y, POS.Z** for interpreting the values as movements along the corresponding axises, and **ROT.X, ROT.Y, ROT.Z** interpreting them as rotation);

fMinRange, fMaxRange set a range of values for the particular ***Point of View Hat*** joystick (from 0.0 to 1.0).

For example, the command *JOYSTICK.*"*JOY".POV.*"*X" =* "*ROT.Y", 0.5, 1.0* determines that the ***Point of View Hat*** movement along the X axis, for example by 0.6, sends the following command to the system (if the command *JOYSTICK.Name.PATH =* "*DATA.NODE.<CURRENT>"* was executed before this):

DATA.NODE.<CURRENT>. ROT = 0, 0.6,0.

☝ *It is not necessary to specify the 'common' command line with JOYSTICK.Name.PATH for completing configuring by the commands AXIS, SLIDER, POV. In this case the necessary commands must be specified entirely (by the* ***Dest*** *parameter of the commands AXIS, SLIDER, POV), not just as additions to the 'common' part specified by JOYSTICK.Name.PATH.*

JOYSTICK.Name.PLAY =iMode

Deactivates (iMode=0) /activates (iMode=1) all the adjusted settings of the joystick named *Name*. It is necessary to give this command with the parameter value equal to 1 to make actual all the joystick settings specified in the given commands (the sensitivity of the handle movement and rotation, button identifiers, value range of the slider, etc.).

JOYSTICK.PLAY =iMode

Deactivates (iMode=0) /activates (iMode=1) all the adjusted settings for all joysticks specified in the commands. It is necessary to give this command with the parameter value equal to 1 to make actual all the settings of all the joysticks specified in the given commands.

JOYSTICK.Name.RESET=iMode

Cancels all settings of the joystick named *Name.*

JOYSTICK.RESET=iMode

Cancels settings of all the joysticks.

☝ *Use the* ***Gaming Options*** *on the Windows 2000 Control Panel to adjust a joystick. There you can also find information about the joystick and the IDs (identifiers) of its buttons.*

☝ *Note: To use the commands for working with joystick described in this Section, it is necessary to have a DLL (plug-in) library of the appropriatre version installed. The commands described above are supported by the* ***JOYSTICK*** *plug-in version no lower than 1.0.0.497. The information about the DLL (plug-in) libraries loaded is displayed in* ***Debug Output*** *when launching the* ***HotActions*** *application (see Figure below).*

Figure 3. Information about the *JOYSTICK* library version displayed in the *Debug Output* when launching *HotActions*

12 Commands for Working with a Robotic Camera

It's possible to work with a robotic camera only when an extra module *VSPTZControl* is installed.

PTZ."CameraName".CREATE = "CameraType", "ComPortName"

First of all, it is necessary to initialize the camera control module with this command. It is also necessary to assign a name to the camera attached to a certain COM port of the computer in order to use the name in other commands.

The *CameraType* parameter specifies a camera type. The types supported are **EVI-D30, EVI-D31, CANON VC-C1, CANON VC-C3**.

Specify the *ComPortName* parameter as *COM1* to connect to the first COM port of the computer, *ComPortName = COM2* – to the second COM port, and so on.

PTZ."CameraName".CALIBRATE = 1

Calibrates the initialized "*CameraName*" camera.

PTZ."CameraName".POWER = iValue

Controls the power of the *"CameraName"* camera:

IValue = 1 – switch on the power;

IValue = 0 – switch off the power.

PTZ."CameraName".FOV = iValue

Specifies FOV of the "*CameraName*"camera:

iValue – horizontal visual angle in degrees.

PTZ."CameraName".ROT = fα, fβ, fγ

Controls the "*CameraName*" camera rotation:

fα, fβ, and *fγ* – Euler angles (angles of rotation around the X, Y, Z axes).

PTZ."CameraName".DROT = fα, fβ, fγ

Performs the "*CameraName*" camera rotation by the *fα, fβ, fγ* angles in the X, Y, Z coordinate system relative to the current camera position.

Completion of any of the robotic camera conrol commands described above is indicated with the **SYS.EVENT="CameraName"** event.

PTZ."CameraName".REACTION = fValue

Sets the time of the "*CameraName*" camera reaction:

fValue – the command execution time in seconds; all commands are executed during the time no less than the specified value.

PTZ."CameraName".QUEUE = iValue

This command enables the command queuing of the "*CameraName*" camera (in this mode all commands are added to a queue and are executed in sequence):

iValue = 1 – enable (this value is set by default);

iValue = 0 – disable.

PTZ."CameraName".RESET = 1

Clears the command queuing of the "*CameraName*" camera.

PTZ.RESET = 1

Clears the command queueing of all the attached cameras.

PTZ."CameraName".DELETE = 1

deactivates the initialized module controlling the "*CameraName*" camera control.

PTZ.DELETE = 1

deactivates all initialized modules controlling cameras.

☝ *Note: To use the commands for working with a robotic camera described in this Section, it is necessary to have a DLL (plug-in) library of the appropriate version installed. The commands*

*described above are supported by the **PTZControl** plug-in version no lower than 1.5.0.13. The information about the DLL (plug-in) libraries loaded is displayed in the **Message (Debug Output)** window when openning a project in the **HotActions** application (see Figure below) if the **Script Trace** option is enabled in the **Debug Output** tab of the application settings (see the **HotActions** User'sGuide, Section 9.3).*

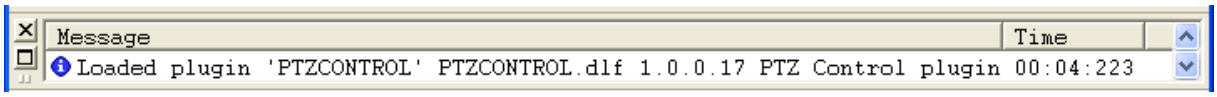

Figure 4. Information about the PTZControl library version displayed in the *Debug Output* window when launching the *HotActions* application

13 Tracking Video Streams in the Scene (Monitoring Movement Data)

It's possible to track the actor's movements in a scene only when an extra module *VSTracker* is installed.

TRACKER.Name.SOURCE = «Video»

Specifies a video stream named *Name* for tracking movements and assigns it to the *Video*.

TRACKER.Name.LISTENER = Path

Specifies a path named *Path* to which the *Name* data stream is to be transferred. The name of a robotic camera can be specified as a *Path* (see previous Section).

TRACKER.Name.START = 1

Starts the *Name* video stream for tracking movements.

TRACKER.Name.STOP = 1

Starts the *Name* video stream for tracking movements.

TRACKER.RESET = 1

Stops all tracking streams.

TRACKER.Name.SMOOTH = iDepth

Determines the number of frames *iDepth*, by which the Name stream data are to be smoothed. By default, *iDepth = 4*.

TRACKER.Name.FOV = fFOV

Determines a horizontal tracking angle *fFOV* (in degrees) for the Name stream. Can take values from 0 to 180. By default, *fFOV = 50*.

TRACKER.Name.SENS = fX, fY, fZ, fRX, fRY, fRZ

Determines scale factors when transferring the tracking data by the command *Tracker.Name.Listener = Path*: *fX* – the shift scale factor along the X axis, *fY* – along the *Y* axis, *fZ* – along the *Z* axis, *fRX* – the rotation scale factor around the X axis, *fRY* – around the *Y* axis, *fRZ* – around the *Z* axis. By default, *fX* = 0, *fY* = 0, *fZ* = 0, *fRX* = 1, *fRY* = 1, *fRZ* = 1.

TRACKER.Name.DELETE = 1

Cancels the *Name* tracking stream.

TRACKER.DELETE = 1

Cancels all tracking streams.

14 Using DesktopCapture to Capture Image from a Remote Computer

To capture image from a remote computer, an appropriate version of additional software must be installed. The settings described below are supported by the *VSDesktopCapture* version number 107 (that is, *VSDesktopCapture107.exe* must be installed on the remote computer). It is necessary to start the *DesktopCapture.exe* application to control the image captured from the remote computer. This can be performed from the **Start** menu (Figure 5) or via the **DesktopCapture** icon  on the *Windows* desktop.

Figure 5. Starting The *DesktopCapture.exe* application on the remote computer

The *DesktopCapture* application is presented by the *DesktopCapture control pane* dialog box with three tabs.

The *Capture Control Panel* tab (Figure 6) is intended to start/stop the procedure of capturing and transferring image (the *Start* button) and also to minimize the application to the system tray.

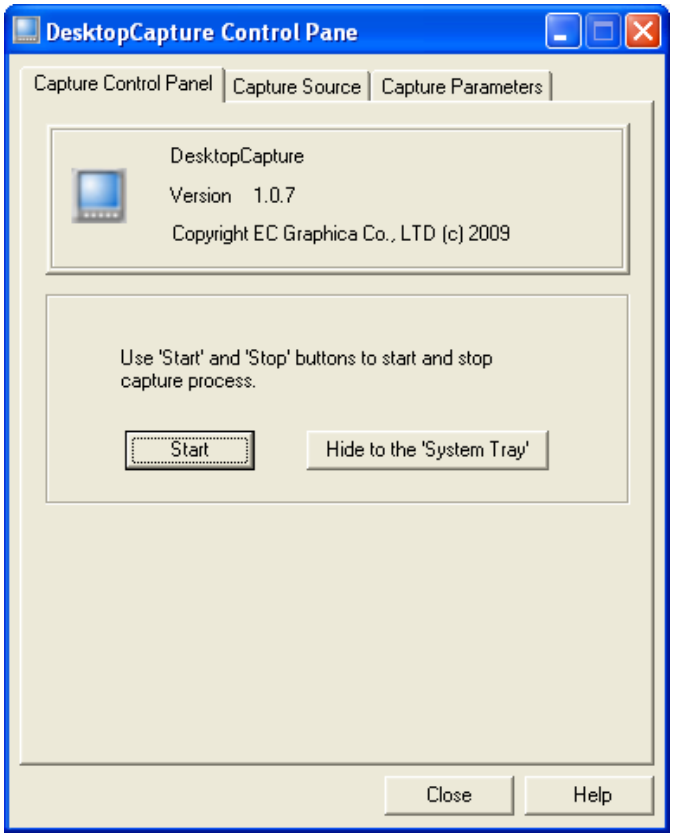

Figure 6. The *Capture Control Panel* tab of the *DesktopCapture Control Pane* dialog box

The *Capture Source* tab (Figure 7) is intended to select a window as a source of image that is to be captured (the **Window selector tool** area). If the **Capture all windows on desktop** option is enabled, the captured image is the desktop and all windows on it.

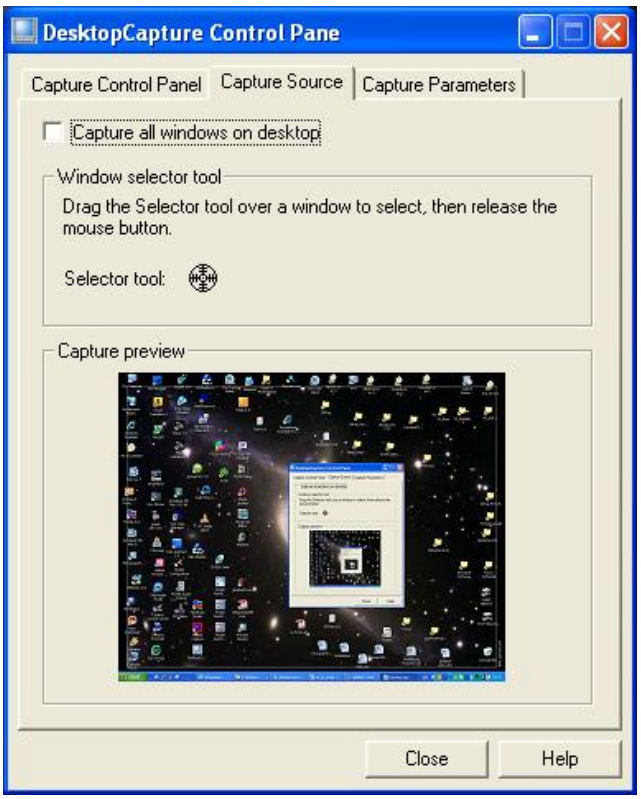

Figure 7. The *Capture Source* tab of the *DesktopCapture Control Pane* dialog box

The *Capture Parameters* tab (Figure 8) is intended to specify the image refresh rate of the captured window (the **Capture frequency** area) and to specify the frame size to which the captured window image is to be scaled (the **Frame buffer size** area).

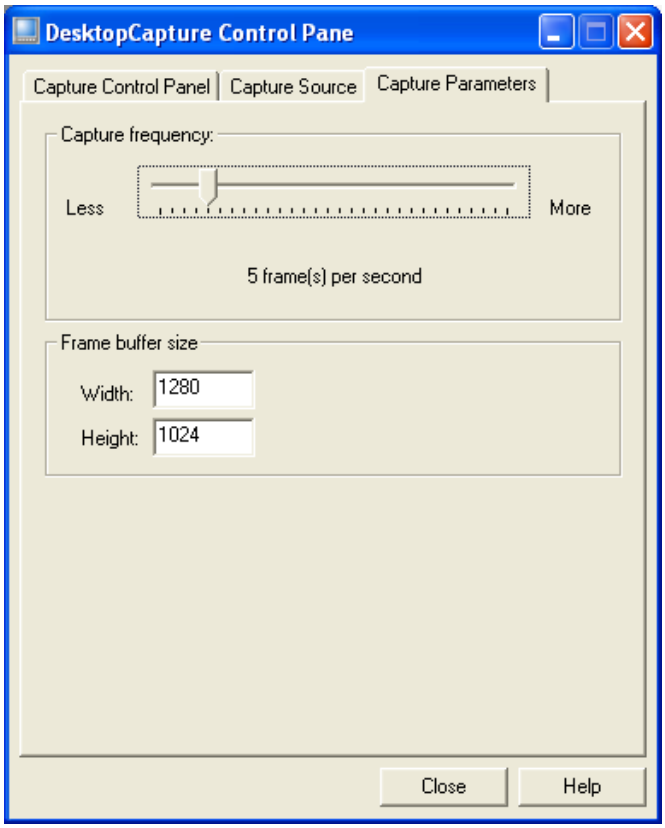

Figure 8. The *Capture Parameters* tab of the *DesktopCapture Control Pane* dialog box

15 Commands for Working with External Devices via GPI (General Purpose Interface)

The application supports external control via in and out devices connected to the computer COM ports (GPI). The features may be useful when is necessary to execute ***Actions*** with a command sent from an external device, or to mark the operating camera providing the signal for the scene (***Tally light control***).

Commands described below are valid if the application modules folder (as a rule, C:\Program Files\Focus Software\HotActions\Plugins) contains VSGPIControl.dlf (installed by VSVSGPIControl.exe).

GPI.IN."Name".CREATE = GPI_IN_DEVICE_NAME

Assigns the "*Name*" name to the input signal from the *GPI_IN_DEVICE_NAME* device connected to the *DEVICE_NAME* COM port. For example, the command **GPI.IN.IN1.CREATE = GPI_On_COM2_0_Input** assigns the name *IN1* to the input signal transmitted to input 0 of the COM port *COM2*.

GPI.IN."Name".DELETE = 1

Cancels the signal assigned to the COM port input named "*Name*".

GPI.IN.DELETE = 1

Cancels all input signals assigned to COM ports.

GPI.IN."Name".NOTIFY.1 = Event

Specifies the text of the *Event* message which is to be displayed in the **Debug Output** window when turning on the device connected to the COM port with the assigned input signal *"Name"*. By default (without specifying with the command), the message **SYS.EVENT = "GPI.IN.Name.1"** is displayed.

GPI.IN."Name".NOTIFY.0 = Event

Specifies the text of the *Event* message which is to be displayed in the **Debug Output** window when turning off the device connected to the COM port with the assigned input signal *"Name"*. By default (without specifying with the command), the message **SYS.EVENT = "GPI.IN.Name.0"** is displayed.

GPI.IN."Name".NOTIFY = Event0, Event1

Specifies the texts of the *Event* messages which are to be displayed in the **Debug Output** window when turning on (*Event0*) or off (*Event1*) the device connected to the COM port with the assigned input signal *"Name"*. By default (without specifying with the command), the messages **SYS.EVENT = "GPI.IN.Name.0"** and **SYS.EVENT = "GPI.IN.Name.1"** are displayed.

GPI.IN."Name".RESET = 1

Resets to the default texts all the messages that are displayed in the **Debug Output** window when changing states of the device connected to the COM port with the assigned input signal "*Name*".

GPI.IN.RESET = 1

Resets to the default texts all the messages that are displayed in the **Debug Output** window when changing states of devices connected to the COM ports with the assigned input signals.

GPI.IN."Name".CHECK = ActionName0, ActionName1

Performs the *Action* named **ActionName0** if the device connected to a COM port with the assigned input signal "*Name*" is off; if the device is on, the command performs the *Action* named **ActionName1**.

GPI.OUT."Name".CREATE = GPI_OUT_DEVICE_NAME

Assigns the name "*Name*" to an output signal from the *GPI_OUT_DEVICE_NAME* device connected to the *DEVICE_NAME* COM port. For example, the command **GPI.OUT.OUT1.CREATE =**

GPI_On_COM1_1_Output assigns the name *OUT1* to the output signal transmitted to output 1 of the port *COM1*.

GPI.OUT."Name".DELETE = 1

Cancels the signal assigned to the COM port output named "*Name*".

GPI.OUT.DELETE = 1

Cancels all signals assigned to the outputs of COM ports.

GPI.OUT."Name".SET = iState

Sets the state of the output signal "*Name*"; *iState = 1* sets 'on', *iState = 0* sets 'off'. By default, *iState = 0*.

GPI.OUT."Name".RESET = 1

Resets the output signal "*Name*" to the default state.

GPI.OUT.RESET = 1

Resets all output signals connected with COM ports to the default state.

GPI.OUT."Name".PULSE = fTime

Pulses the output signal "*Name*" during ***fTime*** seconds.

16 Predefined Identifiers

The object selected by default (when opening the scene) for manipulations with joystick, mouse or arrow keys is an object named <**World**>. It is added as the main node of a scene when exporting the scene. Nothing visible happens when manipulating the object <**World**> because the virtual cameras change along with the scene. There is also a special name to be used in the script commands, which always indicates the name of the current selected object – <**Current**> (Section 3.1). To select a required object for manipulation, put the command specifying it as the current one to the initializing **Action**, for example: **DATA.CURRENT.NODE = "Main"**. A special command can also be used for virtual cameras: **DATA.CURRENT.CAMERA = "Camera1"**.

When switching to another manipulated object, <**Current**> refers to another name, correspondingly.

17 Using Variables in Script Commands

Variables can be used in script commands. The most important use of variables is in ***Theme***. Theme is a version of a project presented as a set of variables. When the theme is activated, the values of the variables become available in commands. For details about themes, see the ***HotActions*** User's Guide, Section 3.8.

It is recommended to prefix the variable names with the "**$**" character to mark them out, for example: **DATA.CURRENT.NODE = $NodeName**.

After the activation of any theme in which the **$NodeName** variable is specified, the value is substituted when executing the command.

18 Restricting the Scope of Variables

There is a way of restricting the scope of such variables as *Sound* in the line *SOUND.Sound.FILE* = "*FileName*". Some difficulties can arise with duplicate names, for example, if they are referred to from ***Actions*** initializing different scenes combined into a common project. If you want to be sure that a certain ***Action*** will be executed for a certain scene, make this scene current before executing a command of this kind. For that, use *DATA.CURRENT = "Scene"*. That restricts the scope of the next commands and excludes using them for a different scene, which can lead to unpredictable consequences.

Names of tracks, ***Actions*** and audio fragments should be unique for all opened scenes so they can be used together. In any case, two different tracks (or audio fragments) cannot have the same name: specifying a duplicate name simply reassigns it.

19 Script examples

Examining examples and modifying them is one of the effective ways to learn how to use script commands and understand already created projects. The *HotActions* package includes simple projects, which are installed to a separate folder usually named ***VS_Sample*** and each of them contains: a scene in the***.3d** fornat, an ***Actions Library*** and a ***Hotset***. You can experiment with these files to learn how to achieve the required results when creating virtual scenes in *Focus*.

Note that most commands descussed in this User's Guide are available only after starting the initializing (***startup***) ***Action*** for a certain scene. Principles and examples of creating initializing ***Actions*** are discussed in Section 3.4.6 of the ***HotActions*** User's Guide. It is also recommended to examine the examples discussed in Creating 3D Scenes User's Guide.

20 Using the *Debug Output* window

The *HotActions* application has the ability to track error and warning messages including those connected with using the script commands. Such messages are displayed in the *Message (Debug Output)* window (Figures 1 – 4).

For more details about work with the *Debug Output* window and its settings, see the ***HotActions*** User's Guide, Section 6. Note that enabling the **Runtime (from Actions)** and **Script Trace** options on the **Debug Output** tab of the *Options* dialog is a requirement for the debugging of scripts in the *Debug Output* window. See also a note in Section 8 of this document about filtering of commands and tracing to a file.