

Виртуальная студия
Фокус

*Использование
команд сценария*

Руководство пользователя

Версия от 19 ноября 2009 г.

VS 1.63

Copyright © SoftLab-NSK Ltd

Содержание

1	СПРАВОЧНИК ПО КОМАНДАМ	3
2	КОМАНДЫ РАБОТЫ С АСТІОН'АМИ.....	5
3	КОМАНДЫ РАБОТЫ С ДАННЫМИ	6
3.1	Общие команды	6
3.2	Команды работы с виртуальными камерами.....	6
3.3	Команды работы с параметрами среды	7
3.4	Команды работы с источниками света.....	8
3.5	Команды работы с 3D-объектами.....	8
3.6	Команды работы с материалами	9
3.7	Команды работы с 2D-текстом	11
	DATA.MATERIAL.MaterialName.TEXT = «Text», «Font», iFontSize,iX, iY, iR, iG, iB, iMix ..	11
3.8	Команды для работы с Web-страницами, Flash-анимацией и PPT-презентациями.....	13
3.9	Команды работы с 3D-текстом	13
4	КОМАНДЫ РАБОТЫ С ТРЕКАМИ	16
5	МОРФИНГ	18
6	КОМАНДЫ ВЫВОДА ИНФОРМАЦИИ О СЦЕНЕ	19
7	КОМАНДЫ РАБОТЫ С ВЫВОДОМ ИЗОБРАЖЕНИЯ	20
7.1	Команды работы с видеопотоками	20
7.2	Использование видеопотоков в сцене	24
7.3	Режим вывода текстур материалов <i>3D Оверлей</i>	25
7.4	Дополнительные команды режима <i>3D Оверлей</i>	25
8	СИСТЕМНЫЕ КОМАНДЫ	27
9	КОМАНДЫ РАБОТЫ СО ЗВУКОМ	29
10	КОМАНДЫ УПРАВЛЕНИЯ КУРСОРОМ МЫШИ	30
11	КОМАНДЫ РАБОТЫ С ДЖОЙСТИКОМ	31
12	КОМАНДЫ РАБОТЫ С РОБОТИЗИРОВАННОЙ КАМЕРОЙ	34
13	РАБОТА С ТРЕКИНГОМ ВИДЕОПОТОКОВ В СЦЕНЕ (ПЕРЕДАЧА ДАННЫХ О ДВИЖЕНИИ)	36
14	ИСПОЛЬЗОВАНИЕ DESKTOPCAPTURE ДЛЯ ВЫВОДА ИЗОБРАЖЕНИЯ С УДАЛЁННОГО КОМПЬЮТЕРА.....	37
15	КОМАНДЫ РАБОТЫ С ВНЕШНИМИ УСТРОЙСТВАМИ ЧЕРЕЗ GPI (GENERAL PURPOSE INTERFACE)	39
16	ЗАРЕЗЕРВИРОВАННЫЕ ИМЕНА	41
17	ИСПОЛЬЗОВАНИЕ ПЕРЕМЕННЫХ В КОМАНДАХ СЦЕНАРИЯ.....	42
18	УТОЧНЕНИЕ ОБЛАСТИ ДЕЙСТВИЯ ПЕРЕМЕННЫХ	43
19	ПРИМЕРЫ СЦЕНАРИЕВ	44
20	ИСПОЛЬЗОВАНИЕ <i>DEBUG OUTPUT</i> ДЛЯ ОТЛАДКИ СЦЕНАРИЕВ.....	45

1 Справочник по командам

Язык сценариев позволяет управлять 3D виртуальными декорациями студии в приложении *HotActions*.

К примеру, переключение вида сцены с одной виртуальной камеры на другую производится исполнением одной строки сценария: `DATA.CURRENT.CAMERA = Camera1`, где `Camera1` должно быть заменено именем объекта, представляющим виртуальную камеру. Или проигрывание анимации виртуальной камеры (наезд камерой на актёра), исполняется строкой `TRACK.Camera1.START = NachKadr, KonKadr`, где виртуальная камера – это объект `Camera1`, а `NachKadr` и `KonKadr` – номера начального и конечного кадров анимации, которую нужно проиграть.

Исполнение последовательности из нескольких таких строк, выстроенных в нужном порядке, приводит к требуемым преобразованиям в 3D-сцене. Таким образом создаётся сюжет телепередачи.

Язык исполняемых строк, как можно было увидеть из приведённых примеров переключения виртуальной камеры сцены и проигрывания её анимации, состоит из набора слов, разделённых точками. Первые слова в этих строках – ключевые, они определяют, из какого раздела будет набираемая строка-команда. Все эти ключевые слова можно увидеть в списке, который появляется при наборе двоеточия в поле `Body` окна *Properties* в приложении *HotActions* (раздел 3.3.1 руководства пользователя *HotActions*). Это слова `ACTION`, `CONTROL`, `DATA` и так далее. После выбора первого ключевого слова команды дальнейшие её слова тоже можно выбирать из списка, появляющегося при нажатии точки (`.`).

В строках-командах переменные в их правых частях (после знака равенства “`=`”) могут быть четырёх типов: тип строка, целое значение, число с плавающей запятой и вектор. В описании команд в этом документе для различения типов будут использоваться следующие обозначения:

- если название переменной в правой части начинается с буквы “*i*”, переменная должна быть целым числом;
- название переменной, начинающееся с буквы “*f*” означает, что переменная – действительное значение;
- вектор представлен несколькими действительными значениями, начинающимися с “*f*”;
- параметр типа «строка», обозначается при помощи кавычек вокруг параметра. Вообще использование кавычек непосредственно в командах с переменными этого типа необязательно, и требуется только, если в тексте строки есть пробелы, знаки «`+`», «`-`», «`.`» и так далее. Однако, чтобы не забывать ставить кавычки только в этих случаях, рекомендуется использовать их для переменных-строк всегда.

За некоторыми исключениями переменные в правых частях команд в описаниях этого документа обозначаются в соответствии с этими правилами. Например, *iSpeed* подразумевает, что параметр *iSpeed* имеет тип «целое», а *fX* – параметр с плавающей точкой.

Команды разделяются символом конца строки (нажатие **Enter**) или, если несколько команд напечатано в одной строке, то точкой с запятой (`;`).

Чтобы напечатать неисполняемую строку в поле `Body Script`, например, комментарий-пояснение к команде или команду, которая не должна исполняться, вначале строки печатаются символы “`//`”. Если требуется напечатать последовательность команд, которые в данный момент не должны исполняться, перед первой командой печатается символ “`/*`”, а после последней “`*/`”.

Далее представлена сводная информация по упомянутому в данной главе синтаксису.

Разделители команд

- {конец строки}** Если печатается одна команда в строке, присутствие разделителя команды будет подразумеваться автоматически.
- ;** Точка с запятой, если несколько команд напечатано подряд в одной строке.

Комментарии

// Комментарий до конца строки.
/* */ Комментарий между «/* » и «* /».

Типы данных

String Текст; строки типа «текст» должны быть включены в кавычки; примеры: «Camera1», «End of Track».
Float Число с плавающей точкой, например, 10.0.
Int Целое число, например, 10.
Vector Вектор – комбинация трёх чисел с плавающей точкой, например, (1.0, 2.0, 3.0).

Системные символы

: Двоеточие. Данный символ рекомендуется ставить первым символом строки – перед началом команды: система автоматически начнёт подсказывать варианты команд и их параметров.
@ Символ, указывающий, что данная команда не должна выводиться при отладке и трассировке (см. главы 8 и 20).

2 Команды работы с *Action*'ами

ACTION.START = «ActionName»

Исполняет *Action* с именем *ActionName*. Это имя можно уточнить, расширив его имя указанием библиотеки.

Например: ACTION.START = «*NewSport.PlayTrack*», где *NewSport* – имя библиотеки, а *PlayTrack* – имя *Action*'а.

После окончания исполнения *Action*'а в окне **Message(Debug Output)** (см. главу 20) появляется сообщение об исполнении *SYS.EVENT* = «*ACTION.Имя_Action*'а». Правая часть этого сообщения может быть использована в качестве параметра команды *SYS.WAIT* = «*Event*» для ожидания выполнения *Action*'а (см. главу 8). Например: *SYS.WAIT* = *Action.ShowSport*.

ACTION.DISABLE = iMode

Включает режим, когда все *Action*'ы на *Hotset*'ах становятся недоступными для исполнения (*iMode*=1) или доступными (*iMode*=0).

3 Команды работы с данными

3.1 Общие команды

DATA.OPEN = «FileName»

Открывает файл *FileName* (с расширением *.3d, *.acl, *.hot и *.vsp).

DATA.CURRENT = «Scene»

Устанавливает *Scene* текущей (активной) сценой.

DATA.CURRENT.CAMERA = «Camera»

Устанавливает *Camera* текущей виртуальной камерой.

DATA.CURRENT.NODE = «Object»

Выбор объекта *Object* в дереве для управления (например, используя мышь/джойстик).

DATA.RESET = 1

Возвращают все объекты к их первоначальным позициям (состояниям) как при загрузке сцены.

DATA.PLAY = fSpeed

Выбор скорости трека и его запуск. Значения параметра *fSpeed* могут быть:

- положительными, при этом 1 соответствует скорости движения 25 кадров в секунду;
- равными 0 – при этом проигрывание анимации трека останавливается;
- отрицательными – анимация проигрывается в обратном направлении, только если траекториям не назначены имена командой **TRACK.Track.NODE = NodeName** (описанной в главе 4), то есть, проигрывание треков командами **TRACK.Track. START = iFrom, iTo, iRepeat** и **TRACK.Track.GOTO = iNFrame** (глава 4) не происходит, если правая часть команды **DATA.PLAY** отрицательна.

Команды **DATA.RESET = 1**, **DATA.PLAY = 0** и **DATA.PLAY = 1** исполняются по умолчанию при инициализации проекта, запускаемого кнопкой **Init** на панели инструментов основного окна приложения или при переходе в рабочий режим *LiveAction* (см. раздел 3.1.9 Руководства пользователя *HotActions*). Команда **DATA.PLAY = 0** останавливает все проигрываемые треки, а команда **DATA.PLAY = 1** запускает проигрывание всех треков сцены.

DATA.STEREO = fStereoBase, fAngle

Работа со стерео эффектами. *fStereoBase* определяет сдвиг изображения для создания эффекта объёмности при просмотре через поляризационные фильтры, *fAngle* определяет угол сдвига. По умолчанию значение *fStereoBase = 100*, *fAngle = 0*.

Работа с этой командой подразумевает специальную конфигурацию студии для работы со стереоизображением.

3.2 Команды работы с виртуальными камерами

DATA.CAMERA.Camera.FOV = fFOV, fTime

Устанавливает *FOV* (угол зрения по горизонтали) для виртуальной камеры *Camera* в *fFOV* (в градусах) за время *fTime* (в секундах). Параметр *fFOV* может принимать значения от 0 до 180. По умолчанию устанавливается равным значению в сцене.

Если параметр *fTime* не указан, то *FOV* устанавливается сразу.

DATA.CAMERA.Camera.CLIPNEAR = fDistance

Устанавливает для виртуальной камеры *Camera* расстояние до ближней плоскости клипирования. При загрузке сцены это значение устанавливается равным 0, что означает автоматический режим определения, то есть в каждом кадре вычисляется расстояние от виртуальной камеры до ближайшего к ней объекта. При указании отрицательного значения параметр также устанавливается равным 0, что, соответственно, тоже означает автоматический режим определения.

☞ При большом разбросе Z-координат объектов сортировка граней при показе может происходить некорректно, в этом случае для коррекции в автоматическом режиме значение ближней плоскости клипирования для виртуальной камеры может устанавливаться меньше, чем до ближайшего объекта.

DATA.CAMERA.Camera.CLIPFAR = fDistance

Устанавливает для виртуальной камеры *Camera* расстояние до дальней плоскости клипирования. При загрузке сцены устанавливается равным 0, что означает автоматический режим определения, то есть в каждом кадре вычисляется расстояние от виртуальной камеры до самого отдалённого от неё объекта. При указании отрицательного значения параметр также устанавливается равным 0, что, соответственно, тоже означает автоматический режим определения.

DATA.CAMERA.Camera.FOGNEAR = fDistance

Для виртуальной камеры *Camera* устанавливает ближнюю границу *fDistance* проявления эффекта дымки (тумана). Команды включения эффекта тумана описаны в следующем разделе. По умолчанию значение *fDistance* устанавливается таким, какое определено в исходной сцене (параметр *Near Range* в группе параметров *Environment Ranges*).

DATA.CAMERA.Camera.FOGFAR = fDistance

Для виртуальной камеры *Camera* устанавливает расстояние *fDistance*, начиная с которого эффект дымки (тумана) исчезает. По умолчанию значение *fDistance* устанавливается таким, какое определено в исходной сцене (параметр *Far Range* в группе параметров *Environment Ranges*).

3.3 Команды работы с параметрами среды

DATA.ENVIRONMENT.DIFFUSE = iR, iG, iB

Определяет цвет фонового света среды, *iR* - красной составляющей, *iG* - зелёной составляющей, *iB* - синей составляющей. Параметры *iR*, *iG*, *iB* могут принимать значения в диапазоне от 0 до 255. По умолчанию цвет устанавливается равным значению в сцене (*Color Background* в разделе *Rendering Environment*).

DATA.ENVIRONMENT.AMBIENT = iR, iG, iB

Определяет цвет рассеянного света среды, *iR* - красной составляющей, *iG* - зелёной составляющей, *iB* - синей составляющей. Параметры *iR*, *iG*, *iB* могут принимать значения в диапазоне от 0 до 255. По умолчанию цвет устанавливается равным значению в сцене (*Ambient Global Lighting* в разделе *Rendering Environment*).

DATA.ENVIRONMENT.FOG.ENABLE = iState

Включает (*iState=1*)/выключает (*iState=0*) эффект дымки (тумана)

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

Определяет цвет тумана, *iR* - красной составляющей, *iG* - зелёной составляющей, *iB* - синей составляющей. Параметры *iR*, *iG*, *iB* могут

принимать значения в диапазоне от 0 до 255. По умолчанию цвет белый, то есть $iR=iG=iB=255$.

DATA.ENVIRONMENT.FOG.NEAR = fDensity

Определяет степень плотности тумана $fDensity$ в процентах на ближней границе видимости проявления эффекта. Границы видимости устанавливаются для каждой камеры индивидуально и команды, определяющие эти границы, описаны в разделе выше.

DATA.ENVIRONMENT.FOG.FAR = fDensity

Определяет степень плотности тумана $fDensity$ в процентах на границе окончания видимости проявления эффекта.

3.4 Команды работы с источниками света

DATA.LIGHT.Light.ENABLE = iState

Включает($iState=1$)/выключает($iState=0$) источник света *Light* в сцене. По умолчанию устанавливается равным значению в сцене.

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

Определяет цвет источника света *Light* в сцене, iR - красной составляющей, iG - зелёной составляющей, iB - синей составляющей. Параметры iR, iG, iB могут принимать значения в диапазоне от 0 до 255. По умолчанию цвет белый, то есть $iR=iG=iB=255$.

DATA.LIGHT.Light.MULT = fMult


Устанавливает интенсивность света $fMult$, это тот же самый параметр, что и в *3D Studio MAX (Multiplier)*. По умолчанию устанавливается равным значению в сцене.

DATA.LIGHT.Light.RANGE = fStart, fEnd

Устанавливает для источника света *Light* диапазон дальности действия или область дальнего затухания света. Тот же самый параметр, что и в *3D Studio MAX (Far Attenuation)*. По умолчанию устанавливается равным значению в сцене.

DATA LIGHT.Light.ANGLE = fHotAngle, fFallAngle

Устанавливает для направленного источника света *Light* (тип *Target Spot, Target Direct, Free Spot, Free Direct*) диапазон углов, регулирующих параметры луча: $fHotAngle$ - световое пятно луча, $fFallAngle$ - область действия пятна света, регулирующая степень размытия края. Параметры $fHotAngle, fFallAngle$ могут принимать значения в диапазоне от 0 до 180 градусов. Те же самые параметры, что и в *3D Studio MAX (Spotlight (Directional)Parameters: Hotspot/Beam, Falloff/Field)*. По умолчанию устанавливаются равными значениям в сцене.

 При попытке применить команду `DATA.LIGHT.Light.ANGLE` к ненаправленному источнику света (тип *Omni*) в окне *Debug Output* появляется сообщение об ошибке.

3.5 Команды работы с 3D-объектами

DATA.NODE.«Object».HIDE = iHide

Скрывает/показывает объект *Object* в сцене; значение **1** для $iHide$ соответствует сокрытию объекта, **0** – показу объекта.

DATA.NODE.«Object».POS = fX, fY, fZ

Устанавливает новое местоположение для объекта *Object*, определяя новые координаты fX, fY , и fZ в сцене, относительно первоначального местоположения; то есть координаты $\{0, 0, 0\}$ соответствуют первоначальному местоположению.

DATA.NODE.«Object».ROT = fa, fb, fy

Вращение объекта *Object* относительно его первоначальной позиции задаётся в градусах углами Эйлера *fa*, *fb*, и *fy* (углы поворота вокруг осей X,Y,Z).

DATA.NODE.«Object».SCL = fa, fb, fc

Масштабирование объекта *Object*, умножение его трёх измерений на коэффициенты *fa*, *fb*, и *fc*.

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

Установка скорости движения, скорости вращения и скорости масштабирования объекта *Object* параметрами *fMOVx*, *fMOVy*, *fMOVz*, *fROTx*, *fROTy*, *fROTz*, *fSCLx*, *fSCLy*, *fSCLz* по всем осям соответственно (в units(единицы измерения, установленные в сцене) в миллисекунду).

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

Установка относительного сдвига, поворота и масштаба объекта *Object* параметрами *fMOVx*, *fMOVy*, *fMOVz*, *fROTx*, *fROTy*, *fROTz*, *fSCLx*, *fSCLy*, *fSCLz* по всем осям соответственно.

DATA.NODE.«Object».LISTENER = «Listener»

Данная команда назначает узлу *Object* другой объект с именем *Listener*, которому посылаются от *Object* все приходящие ему команды сдвига, поворота и масштабирования.

DATA.NODE.«Object».RESET = 1

Возвращает объект *Object* в первоначальную позицию (состояние), которое он занимал при загрузке сцены.

DATA.NODE.«Object».OnClick = "Script"

Назначает команду *Script*, которая будет исполняться при щелчке курсором мыши по объекту с именем *Object* в сцене (см. главу 10).

DATA.NODE.«Object».OnOVER = "Script"

Назначает команду *Script*, которая будет исполняться при проведении курсором мыши над объектом с именем *Object* в сцене (см. главу 10).

DATA.NODE.«Object».OnOUT = "Script"

Назначает команду *Script*, которая будет исполняться при удалении курсора мыши с объекта с именем *Object* в сцене (см. главу 10).

3.6 Команды работы с материалами

DATA.MATERIAL.MaterialName.CACHE = iState

Включает(1)/выключает(0) кэширование текстур для материала **MaterialName**; при включенном кэшировании все повторяющиеся загрузки будут производиться из памяти видеокарты. По умолчанию включено.

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

Устанавливает параметры отображения для всех текстур материала с именем **MaterialName**, выводимых командами **DATA.MATERIAL.MaterialName.MAP** (описание приведено ниже), **iMaxWidth** – максимально возможное разрешение по ширине, **iMaxHeight** – максимально возможное разрешение по высоте, **iMaxSize** – максимальный размер (в мегабайтах).


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

или

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


Заменяет *Diffuse* текстуру материала **MaterialName** на текстуру из файла **GraphicsFile.ext**. Поддерживаемые расширения – **gif, jpg, bmp, tiff, tga, ifl, ppt, swf, htm, avi**. После первичной загрузки статичных текстур (файлы с расширением **gif, jpg, bmp, tiff, tga, ifl**) они сохраняются в оперативной памяти видеокарты, динамические текстуры (файлы с расширением **ppt, swf, html**) сохраняются в оперативной памяти операционной системы.

После исполнения команды в окне **Message (Debug Output)** (см. главу 20) появляется сообщение об исполнении **SYS.EVENT = «DATA.MATERIAL.MaterialName»**. Правая часть этого сообщения может быть использована в качестве параметра команды **SYS.WAIT = «Event»** для ожидания исполнения команды (см. главу 8).

 *Попытка применить эту команду к материалу, не имеющему изначально **Diffuse** текстуру, воспринимается как ошибка.*

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

Заменяет *Ambient* текстуру материала **MaterialName** на текстуру из файла **GraphicsFile.ext**. Поддерживаемые расширения – **gif, jpg, bmp, tiff, tga, ifl, ppt, swf, htm, avi**. После первичной загрузки статичных текстур (файлы с расширением **gif, jpg, bmp, tiff, tga, ifl**) они сохраняются в оперативной памяти видеокарты, динамические текстуры (файлы с расширением **ppt, swf, html**) сохраняются в оперативной памяти операционной системы.

 *Попытка применить эту команду к материалу, не имеющему изначально **Ambient** текстуру, воспринимается как ошибка.*


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

Заменяет *Reflection* текстуру материала **MaterialName** на текстуру из файла **GraphicsFile.ext**. Поддерживаемые расширения – **gif, jpg, bmp, tiff, tga, ifl, ppt, swf, htm, avi**. После первичной загрузки статичных текстур (файлы с расширением **gif, jpg, bmp, tiff, tga, ifl**) они сохраняются в оперативной памяти видеокарты, динамические текстуры (файлы с расширением **ppt, swf, html**) сохраняются в оперативной памяти операционной системы.

 *Попытка применить эту команду к материалу, не имеющему изначально **Reflection** текстуру, воспринимается как ошибка.*

DATA.MATERIAL.MaterialName.MAP.OPACITY = FileName

Добавляет прозрачность(альфа-канал) для *Diffuse* текстур материала **MaterialName** из файла **FileName**. Если в загружаемом файле альфа-канал отсутствует, то значения прозрачности формируются из яркостей загружаемого файла. Поддерживаемые расширения **gif, jpg, bmp, tiff, tga, ifl**.

 *Попытка применить эту команду к материалу, не имеющему изначально **Diffuse** текстуру, воспринимается как ошибка.*

DATA.MATERIAL.MaterialName.MAP = 0

Восстанавливает исходные текстуры материала *MaterialName* объекта, как при загрузке сцены.

DATA.MATERIAL.MaterialName.SIZE = iWidth, iHeight

Меняет размер текстурной карты для команды **TEXT**. Текстура очищается.

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

Задаёт диффузный цвет материала; *iR*, *iG*, и *iB* изменяются от 0 до 255.

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

Задаёт рассеянный цвет материала, не зависящий от освещения. *iR*, *iG*, и *iB* изменяются от 0 до 255.

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

Задаёт отражённый цвет материала; *iR*, *iG*, и *iB* изменяются от 0 до 255.

DATA.MATERIAL.MaterialName.SELF = iValue

Задаёт собственную светимость материала; *iValue* меняется от 0 до 100.

DATA.MATERIAL.MaterialName.POWER = iValue


Задаёт величину отражённого блика для материала; *iValue* меняется от 0 (блик выключен) до 127.

DATA.MATERIAL.MaterialName.TRANSP = iValue

Задаёт прозрачность материала; *iValue* меняется от 0 до 100.

DATA.MATERIAL.MaterialName.ALPHA = 1/0

Устанавливает/снимает специальный признак для материала **MaterialName**, по которому все объекты, в которые входит этот материал, помечаются как имеющие прозрачность - для правильной сортировки.

 По умолчанию процедура рендеринга начинается с объектов с непрозрачными материалами (без альфа-канала) и проводится в порядке возрастания Z-координат центров сфер, охватывающих объекты. Далее визуализируются объекты с материалами с прозрачностью (альфа-каналом), уже начиная с наибольшей Z-координаты охватывающей сферы объекта и следуя в порядке их убывания. При сортировке учитывается наличие префиксов в именах (глава 8 Руководства пользователя по созданию 3-х мерных сцен), влияющие на порядок сортировки объектов.

RENDER.MATERIAL.MaterialName.MIPMAPLODBIAS = fBias

Изменяет степень чёткости изображения материала *MaterialName*. По умолчанию значение *fBias* равно 0 и с его увеличением повышается степень размытости изображения. При уменьшении значения параметра изображение становится менее размытым.

DATA.MATERIAL.MaterialName.RESET = 1

Восстанавливает исходную текстуру для материала *MaterialName* после замены её командой **DATA.MATERIAL.MaterialName.MAP** или ***.TEXT** и удаляет все загруженные ранее текстуры для этого материала из оперативной памяти видеокарты.

DATA.MATERIAL.Reset = 1

Приводит все материалы объектов в первоначальное состояние (как при загрузке сцены) и из оперативной памяти видеокарты удаляет все текстуры, которые замещали первоначальные.

3.7 Команды работы с 2D-текстом

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

Заменяет текстуру материала *MaterialName* текстом *Text* с использованием:

«Font» – название системного шрифта;

iFontSize – размер шрифта (в текселах – единицах измерения – аналог пикселей для текстур);

<i>iX</i>	отступ по X ;
<i>iY</i>	отступ по Y ;
<i>iR</i>	цвет шрифта (красной составляющей);
<i>iG</i>	цвет шрифта (зелёной составляющей);
<i>iB</i>	цвет шрифта (голубой составляющей).
<i>iMix</i>	режим смешивания с текстурной картой: 1 – текст добавляется к текстуре; 0 – текст заменяет текстуру.

Отступ отсчитывается от левого верхнего угла текстуры, *iR*, *iG*, и *iB* изменяются от 0 до 255. По умолчанию: шрифт **System**, размер – 24, цвет – белый, то есть $iR=iG=iB=255$. Остальные параметры по умолчанию равны нулю. Параметры текста можно также задать отдельными командами (см. ниже), тогда в самой команде **TEXT** их можно не указывать.

Кроме того, что материал должен иметь текстуру для использования этой команды, у него необходимо задать текстурные координаты (в *3D Studio Max*). Если материал имеет прозрачную текстуру и *iMix=0*, то текст окажется на прозрачной подложке.

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

Задаёт шрифт для команды **TEXT**. «Font» – название системного шрифта.

DATA.MATERIAL.MaterialName.TEXT.SIZE = iFontSize

Задаёт размер шрифта для команды **DATA.MATERIAL.MaterialName.TEXT.FONT**. *iFontSize* – размер шрифта (в текселях – единицах измерения – аналог пикселей для текстур).

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

Задаёт смещение текста для команды **TEXT**. Отсчитывается от левого верхнего угла текстуры.

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

Задаёт цвет текста для команды **TEXT**; *iR*, *iG*, и *iB* изменяются от 0 до 255.

DATA.MATERIAL.MaterialName.TEXT.OUTLINE = iTexels

Задаёт окантовку текста; *iTexels* – ширина окантовки в текселях, может принимать значения от 0 (окантовка не рисуется) и больше. Цвет окантовки по умолчанию задан чёрным (см. также следующую команду).

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

Задаёт цвет окантовки текста; *iR*, *iG*, и *iB* изменяются от 0 до 255.

DATA.MATERIAL.MaterialName.TEXT.SHADOW = iTexels

Устанавливает изображение тени для текста размером в *iTexels* текселей. Параметр *iTexels* может принимать любые целые значения. При значении 0 тень не рисуется, при других значениях знак *iShadow* задаёт направление отбрасывания тени: вправо-вниз (при положительных значениях *iShadow*) или влево-вверх (при отрицательных значениях *iShadow*). По умолчанию тень имеет чёрный цвет (см. также следующую команду).

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

Задаёт цвет тени текста; *iR*, *iG*, и *iB* изменяются от 0 до 255.

3.8 Команды для работы с Web-страницами, Flash-анимацией и PPT-презентациями

В качестве *Diffuse Map* для текстуры материала можно использовать Web-страницы (файлы с расширением *.htm, *.html), презентации *PowerPoint* и файлы *Macromedia Flash*.

DATA.MATERIAL.Name.MAP = «File.ext»


Поддерживаемые расширения – **htm, html, ppt** и **swf**. После первичной загрузки файлы сохраняются в оперативной памяти операционной системы.

DATA.MATERIAL.Name.MAP.FRAGMENT = iFrame

Команда служит для смены слайдов PPT презентации:

iFrame = +1 – переход к следующему слайду;


iFrame = -1 – к предыдущему слайду.

 Для показа PPT файлов в формате PowerPoint 95/97 необходима предварительная установка стандартного ActiveX Control (с помощью запуска *axplayer.exe* <http://activex.microsoft.com/activex/controls/ppoint/ppoint.asp> <http://activex.microsoft.com/activex/controls/ppoint/axplayer.exe>).

Для показа PPT-файлов формата выше PowerPoint 97 (Power Point 2000-2003) необходимо, чтобы в операционной системе был установлен пакет Microsoft Office PowerPoint(версии 2000-2003).

 Для проигрывания файлов Macromedia Flash необходима предварительная установка Shockwave ActiveX plugin (<http://www.macromedia.com/shockwave/download/>).

3.9 Команды работы с 3D-текстом

 Файл *.tga или *.tiff формата, описывающий шрифт (см. следующие команды), должен состоять из 256 изображений – всех символов шрифта, расположенных в таблице по 16 символов в 16-ти рядах. Символы могут быть произвольного размера (при соблюдении ограничений на обычную текстуру), но при этом все должны иметь одинаковую высоту, хотя при этом и могут различаться по ширине. Порядок расположения символов определяется текущей однобайтовой ANSI-кодировкой. Кодировку символа можно определить, используя стандартную утилиту “Таблица символов”(Пуск > Программы > Стандартные > Служебные), включив в ней опцию “Дополнительные параметры просмотра” при пересчёте кода символов, содержащегося в подсказках к символам, из шестнадцатеричной в десятичную систему отсчёта.

Символы в файле следует располагать начиная с левого верхнего угла, слева направо и сверху вниз. Ширина самого левого верхнего, нулевого символа, служит для определения свободного пространства вокруг изображения всех остальных символов, поэтому нулевой символ не должен содержать изображений. Ширину нулевого символа рекомендуется определять не менее 40% от высоты шрифта. Каждый символ отделяется от соседнего полностью прозрачным цветом – рамкой любой ширины с альфа=0. Сам символ не должен иметь пикселей с альфа=0.

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

Создаёт шрифт и задаёт его по умолчанию для 3D-текста.

«Font» – имя *.tga или *.tiff файла, описывающего шрифт, либо имя системного шрифта.

«ContentFileName» – имя текстового (*.txt) файла с фразами – подстановками текста, в кодировке UNICODE. Формат файла – ключевые слова в угловых скобках, за которыми следует текст - замена. Пример:

<T1>Фраза1

<T2>Фраза2

если далее в используемых командах **DATA.NODE. «Object». TEXT = «Text», fTextHeightRatio, fTextWidthRatio** (описание приведено ниже) в тексте «Text» встретятся фрагменты <T1> или <T2>, они будут заменены на соответствующий текст из файла «ContentFileName»: Фраза1 или Фраза2. Например, в результате исполнения команды

DATA.NODE. «Object». TEXT = «Это текст из фрагмента <T1>, а это – из фрагмента <T2>»

будет показан текст

”Это текст из фрагмента Фраза1, а это – из фрагмента Фраза2”.

fFontHeightRatio, fFontWidthRatio – коэффициенты пропорциональности высоты и ширины символов созданного шрифта. Принимают значения от 0.0 до 1.0. По умолчанию равны 1, то есть размеры шрифта будут установлены такими, чтобы при замене объекта в 3D сцене на текст этот текст занимал бы максимально возможное пространство исходного объекта.

Параметры *iR, iG, iB* устанавливают цвет шрифта и могут принимать значения от 0 до 255. По умолчанию установлен белый цвет шрифта, то есть все значения $iR = iG = iB = 255$.

Значение параметра *iOutline* задаёт ширину окантовки текста чёрным цветом. Единицы измерения определяются размерами используемого шрифта(см. ниже описание параметра *fQuality*). По умолчанию значение равно 1(при $fQuality = 1$).

Параметр *iShadow* устанавливает для текста размер тени чёрного цвета. Направление тени определяет знак указанного значения: вправо-вниз при положительных значениях *iShadow* или влево-вверх при отрицательных значениях *iShadow*. Единицы измерения определяются размерами используемого шрифта(см. ниже описание параметра *fQuality*). По умолчанию значение *iShadow* равно 1(при $fQuality = 1$).

Параметр *fQuality* позволяет регулировать размер символа. Этот параметр может быть значим для некоторых восточных языков (Корея, Япония, Китай и так далее), для которых количество иероглифов в текстурной карте может превышать установленное по умолчанию количество – 256 символов (размер символа 64 тексела, размер текстурной карты 1024 текселя). При увеличении количества символов в текстурной карте происходит уменьшение размера символов, как только символы уменьшаются вдвое (32 тексела), текстурная карта масштабируется до 2048 текселей, при дальнейшем увеличении количества символов процедура повторяется с последовательным увеличением размеров текстурной карты до 4096 текселей.

При увеличении параметра *fQuality* увеличивается размер символа в текстуре, это увеличивает время создания шрифта, но символы при этом получают более высокого разрешения.

По умолчанию параметр *fQuality* задан 1, что соответствует размеру символа 64 тексела и текстуре 1024 текселов для 256 символов.

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

Данная команда создаёт шрифт, обращение к которому в дальнейшем возможно с помощью имени *FontName*.

Описание параметров см. в спецификации предыдущей команды.

DATA.NODE.«Object».FONT = «FontName»

Задаёт шрифт **FontName** для объекта *Object*.

DATA.NODE.«Object».TEXT = «Text», fTextHeightRatio, fTextWidthRatio

Заменяет объект *Object* в сцене 3D-текстом **Text**. Текст может состоять из нескольких строк, которые разделяются символами «\n».

Текст вписывается в объект таким образом, чтобы ни при каких условиях не выходил за его пределы. Если строка текста слишком длинная, то автоматически осуществляется её разбивка на несколько строк. При этом шрифт масштабируется таким образом, чтобы надпись всегда умещалась в пределах объекта. Выравнивание текста в строках по умолчанию производится по центру – режим **CENTER**, если предварительно не был установлен иной режим выравнивания текста (см. описание следующей команды **ALIGN**).

Параметры *fTextHeightRatio* и *fTextWidthRatio* задают коэффициенты масштабирования текста в долях от исходного размера объекта, и могут принимать значения от 0.0 до 1.0. Параметр *fTextHeightRatio* задаёт максимально возможный масштаб высоты шрифта. Если текст не умещается в размерах объекта, высота шрифта автоматически уменьшается и устанавливается такой, чтобы текст не выходил за пределы объекта. Ширина символов пропорциональна высоте, но может быть изменена параметром *fTextWidthRatio*.

По команде **DATA.NODE.«Object».TEXT = 0** 3D-текст убирается и объект восстанавливается до первоначального состояния.

Шрифт используется либо по умолчанию, заданный командой **DATA.FONT**, либо заданный для этого объекта командой **DATA.NODE.«Object».FONT = «FontName»**.

В тексте можно использовать ключевые слова в угловых скобках, они будут заменяться текстом в кодировке **UNICODE** из файла **«ContentFileName»**, имя которого указано в команде **DATA.FONT**. Например:

DATA.NODE.Example.TEXT = «<T1>»

Кроме латинских символов, в тексте можно использовать символы локального системного языка. Для латиницы и кириллицы можно использовать все символы, а для азиатских языков – только базовый набор иероглифов:

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»

Задаёт режим вывода текста в габаритах объекта *Object*.


«CENTER» – центрирование по горизонтали.


«RIGHT» – выравнивание вправо.

«LEFT» – выравнивание влево.

«JUSTIFY» – выравнивание слева и справа.

4 Команды работы с треками

 Для любого объекта сцены, имеющего анимацию (траекторию), можно задать один или несколько треков, отвечающих за отдельные участки траектории

 После окончания проигрывания анимации каждого трека в окне *Message(Debug Output)* появляется сообщение о завершении события *SYS.EVENT=«TRACK.Имя_трека»* (глава 8). Правая часть этого сообщения может использоваться в качестве параметра команды *SYS.WAIT = «Event»* для ожидания исполнения трека. Например: *SYS.WAIT = Track.Sport*.

TRACK.«Track».NODE = «NodeName»

Назначает имя трека «Track» для управления анимацией объекта(-ов) «NodeName». По этой команде анимация объекта(-ов) «NodeName» исключается из общей анимации и переходит под контроль указанного трека. Текущий кадр выставляется в 0. Если объект не назначен, то в командах трека используется объект с тем же именем, что и трек.

TRACK.«Track».NOTIFY = «Event»

Устанавливает текст «Event» сообщения, которое будет появляться в окне **Message (Debug Output)** при завершении проигрывания трека с именем *Track*; по умолчанию при завершении проигрывания этого трека в окне **Message** генерируется сообщение *SYS.EVENT = «TRACK.Track»*.

TRACK.«Track».LOOP = iNLoop

Определяет заикленность проигрывания трека *Track* при выполнении команды **TRACK.«Track».START** или **TRACK.Track.PLAY**, описания которых приведены ниже.

iNLoop = 1 соответствует заикленному проигрыванию трека, то есть анимация будет воспроизводиться непрерывно до прерывания;

iNLoop = 0 проигрывание анимации не заиклено, то есть команда воспроизведения трека будет исполнена один раз.

TRACK.Track.RANGE = iFrom, iTo, iRepeat

Добавляет в очередь трека *Track* набор кадров с *iFrom*-ого по *iTo*-ый, *iRepeat* раз. Для проигрывания этого набора можно использовать команду **TRACK.Track.PLAY = iRepeat**.

TRACK.Track.MATERIAL = MaterialName

Создаёт трек с именем *Track* для показа материала *MaterialName*. Для случаев, когда в качестве текстурной карты материала назначен видео- или **IFL**-файл, команда позволяет показывать эти файлы частично.

TRACK.«Track».START = iFrom, iTo, iRepeat

Проигрывает трек *Track* с *iFrom*-ого кадра по *iTo*-ый *iRepeat* раз. Если *iFrom* > *iTo*, то трек проигрывается в обратную сторону. Если параметры *iTo* и *iRepeat* не указаны, то трек устанавливается в кадр *iFrom*.

TRACK.Track.PLAY = iRepeat

Проигрывает трек с именем *Track* (или последовательность кадров трека *Track*, установленную командой **TRACK.Track.RANGE = iFrom, iTo, iRepeat**) *iRepeat* раз.

TRACK.Track.STOP = iNFrame

Останавливает проигрывание трека *Track* на *iNFrame*-ом кадре, независимо от установленного числа его повторений.

TRACK.Track.GOTO = iNFrame

Проигрывает *Track* с текущего кадра по *iNFrame*-ый кадр, число повторов, задаваемое командой **RANGE**, игнорируется. Если *iNFrame* больше номера

текущего кадра, то трек проигрывается вперед. Если *iNFrame* меньше номера текущего кадра, то трек проигрывается назад до кадра *iNFrame*.

TRACK.Track.QUEUE = iState

Включает(1)/выключает(0) очередность для трека.

Каждый трек имеет свою очередь команд: при включенной очередности каждая команда помещается в очередь, и исполняется автоматически, когда закончится предыдущая. При выключенной очередности команда исполняется немедленно. По умолчанию очередность включена.

Очередность может быть заменена ожиданием. Вместо:

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

можно писать:

```
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"

Формирует трек *Track* – исполнение команды *Command* со скоростью, в каждом кадре трека определяемой значением “номер кадра трека × *fVelocity*”. Значение “*DataFormat*” специфицирует параметр команды *Command*, например, команды

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

```
TRACK.Track.START = 1
```

```
TRACK.Track.START = 2
```

```
TRACK.Track.START = 3
```

эквивалентны командам

```
ACTION.START = Action1
```

```
ACTION.START = Action2
```

```
ACTION.START = Action3
```

TRACK.Track.DELETE = 1

Удаляет содержимое трека *Track*.

TRACK.DELETE = 1

Удаляет все именованные треки.

TRACK.RESET = 1

Остановка анимации. Останавливает все треки и очищает очереди. Исполняется по умолчанию при инициализации проекта, запускаемого кнопкой **Init** на панели инструментов основного окна приложения или при переходе в рабочий режим *LiveAction* (см. Руководство пользователя *HotActions*, раздел 3.1.9).

5 Морфинг

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

Осуществляет морфинг исходного объекта с именем *SourceObjectName* в объект с именем *DestObjectName* (обычно он скрыт в сцене) за время *fTime* секунд. Оба объекта при этом должны иметь одинаковое число вершин. После завершения морфинга генерируется сообщение о завершении события **RENDER.MORPH. «SourceObjectName»**.

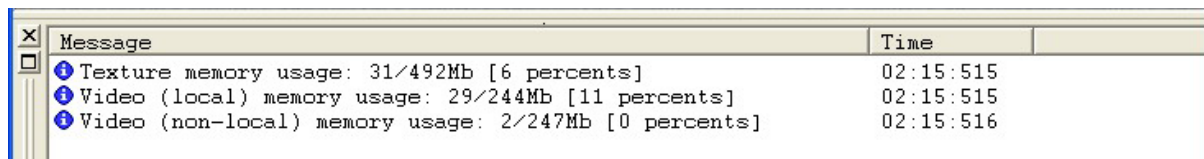
RENDER.MORPH.RESET = 1

Перевод всех морфированных объектов в исходное состояние (как при загрузке сцены). Исполняется по умолчанию при инициализации проекта, запускаемого кнопкой **Init** на панели инструментов основного окна приложения или при переходе в рабочий режим *LiveAction* (см. Руководство пользователя *HotActions*, раздел 3.1.9).

6 Команды вывода информации о сцене

RENDER.DUMP=1

Выводит в окно **Message (Debug Output)** информацию о загруженной сцене: количество памяти видеокарты, используемой для загрузки текстур, как локальной памяти графического ускорителя (local), так и части оперативной памяти(non-local), значение которой можно задавать в BIOS-е данного компьютера,



Message	Time
Texture memory usage: 31/492Mb [6 percents]	02:15:515
Video (local) memory usage: 29/244Mb [11 percents]	02:15:515
Video (non-local) memory usage: 2/247Mb [0 percents]	02:15:516

Рис. 1. Информация о сцене, загруженной в *HotActions*, выводимая в *Debug Output* при исполнении команды **RENDER.DUMP=1**

RENDER.DUMP.VIDEO = 1

Выводит в окно **Debug Output** информацию о видеопотоках (см. описание в следующей главе): об источниках сигнала для потоков типа LIVE и имена файлов с данными для потоков типа FILE и MSDS.

7 Команды работы с выводом изображения

7.1 Команды работы с видеопотоками

Следующие команды являются общими для работы с видеопотоками всех типов.

Примеры управления видеопотоком подключенного видеисточника и видеопотоком из видеофайла приведены соответственно в разделах 9.3.1 и 9.3.4 Руководства пользователя по созданию 3D-сцен.

RENDER.VIDEO.VideoStreamName.CREATE = iParam

Создаёт видеопоток с именем *VideoStreamName*, который может быть:

LIVE_1, **LIVE_2**, **LIVE_3** и так далее – видеопотоком для показа видеоизображения из видеосигнала, подключенного к плате ввода (*FD300* или *Aja XENA*), настройка описана в документе по настройке видео и звука *VS_Video.pdf*);

FILE_1, **FILE_2**, **MSDS_1**, **MSDS_2** и так далее – видеопотоком для показа видеоизображения из видео- и графических файлов посредством фильтров *DirectShow*, а также для вывода изображения со внешних устройств. Через такие видеопотоки допустимо проигрывание *.avi, *.mpg, *.wmv и других файлов, кодек для воспроизведения которых установлен в операционной системе. Можно также показывать графические изображения: *.jpg, *.tga, *.tiff, *.gif и прочие файлы.

iParam может принимать следующие значения:

iParam = 1 для создания видеопотока;

iParam = 0 для удаления созданного видеопотока.

Видеопоток создаётся в остановленном состоянии (см. ниже описания команд **RENDER.VIDEO.VideoStreamName.START = 1** и **RENDER.VIDEO.VideoStreamName.STOP = 1**).

При создании видеопотоков **FILE_*** и **MSDS_*** данные к этим видеопотокам не привязаны (см. ниже описания команд **RENDER.VIDEO.VideoStreamName.DATA = «Filename», iStartFrame, iLength**, **RENDER.VIDEO.VideoStreamName.DATAINOUT = «Filename», iFrom, iTo**).

RENDER.VIDEO.VideoStreamName.DESTROY = iParam

Удаляет созданный видеопоток с именем *VideoStreamName*, если значение *iParam* = 1 или создаёт видеопоток с именем *VideoStreamName*, если значение *iParam* = 0.

RENDER.VIDEO.VideoStreamName.RESET = 1

Сбрасывает состояние видеопотока с именем *VideoStreamName* в предусмотренное по умолчанию.


Команды **RENDER.VIDEO.<LIVE>.RESET = 1**, **RENDER.VIDEO.<FILE>.RESET = 1**, **RENDER.VIDEO.<MSDS>.RESET = 1** исполняются по умолчанию при инициализации проекта, запускаемого кнопкой **Init** на панели инструментов основного окна приложения или при переходе в рабочий режим *LiveAction* (см. Руководство пользователя *HotActions*, раздел 3.1.9).

RENDER.VIDEO.VideoStreamName.START = iParam

При значении *iParam* = 1 запускает воспроизведение видеопотока с именем *VideoStreamName*. При этом в окне **Message (Debug Output)** генерируется сообщение о событии *SYS.EVENT = «VIDEO.VideoStreamName.START»* (см. главу 8).

При значении *iParam* = 0 воспроизведение видеопотока с именем *VideoStreamName* останавливается (аналог команды **RENDER.VIDEO.VideoStreamName.STOP = 1**, описанной ниже). В окне

Message (Debug Output) генерируется сообщение о событии *SYS.EVENT* = «*VIDEO.VideoStreamName.STOP*» (глава 8).

 Повторный старт файла происходит быстрее, поскольку он воспроизводится из оперативной памяти системы.

RENDER.VIDEO.VideoStreamName.PAUSE = 1

Останавливает видеопоток *VideoStreamName*. В окне **Message (Debug Output)** после исполнения выводится сообщение *SYS.EVENT* = "*VIDEO.VideoStreamName.PAUSE*" (глава 8).

RENDER.VIDEO.VideoStreamName.STOP = iParam

Если *iParam* = 1, останавливает воспроизведение видеопотока с именем *VideoStreamName*. При этом в окне **Message (Debug Output)** генерируется сообщение о событии *SYS.EVENT* = «*VIDEO.VideoStreamName.STOP*» (глава 8).

При *iParam* = 0 запускает воспроизведение видеопотока с именем *VideoStreamName* (аналог команды **RENDER.VIDEO.VideoStreamName.START = 1**, описанной выше). В окне **Message (Debug Output)** генерируется сообщение о событии *SYS.EVENT* = «*VIDEO.VideoStreamName.START*» (глава 8).

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

Команда должна исполняться после создания и до запуска видеопотока. Она устанавливает формат видеопотока строкой «*Format string*», в которой через запятую могут быть указаны параметры. Для видеопотоков с плат ввода видеосигналов (**LIVE_1, LIVE_2,...**):


- **ALPHA** – включает использование рирпроекции (chroma key). В результате достигается эффект «прозрачности» цветов фона во входном видео. Настройка параметров рирпроекции производится в диалоге *KeyConfigPro* (см. документацию по настройке видео). Это значение установлено по умолчанию. Включает маску, задаваемую в диалоге *KeyConfigPro*.
- **NOALPHA** – выключает использование рирпроекции для входного видеосигнала. Выключает маску, задаваемую в диалоге *KeyConfigPro*.
- **CROP** – включает использование маски для «подрезания» краёв кадра. Значение установлено по умолчанию.
- **NOCROP** – выключает использование маски для «подрезания» краёв кадра.
- **MIPMAP** – включение трилинейной (пирамидальной) фильтрации для устранения искажений в изображении видео при его показе в уменьшенном масштабе и передвижениях в сцене. Степень фильтрации можно регулировать командой **RENDER.MATERIAL.MaterialName.MIPMAPLODBIAS = fBias** (описание см. в разделе команд работы с материалами 3.6).
- **NOMIPMAP** – отключение режима пирамидальной фильтрации. По умолчанию режим отключен.
- **SYNC** – установка режима подстройки синхронизации чётности полей входного и выходного кадров
- **NOSYNC** – отключает возможность синхронизовать поля входного и выходного кадров.

Для файловых видеопотоков **FILE_*** и **MSDS_*** дополнительно используются параметры:


- **ADD** – для файловых потоков из AVI (**FILE_1, FILE_2,...**) устанавливает режим, в котором происходит добавление файлов в очередь потока, то есть файлы, добавленные с помощью команд

RENDER.VIDEO.VideoStreamName.DATA (об этой команде см. далее), проигрываются в том порядке, в каком они были добавлены.

- **REPLACE** – устанавливает режим, в котором происходит замещение файлов в очереди потока, то есть каждый файл, добавленный с помощью команды **RENDER.VIDEO.VideoStreamName.DATA** (об этой команде см. далее), прерывает проигрывание предыдущего, очищая очередь перед своим запуском. Это значение установлено по умолчанию.
- **LOOP** – включение режима зацикленного проигрывания видеопотока.
- **NOLOOP** – выключение режима зацикленного проигрывания видеопотока. Это значение установлено по умолчанию.
- **AUDIO** – проигрывание видеопотока будет производиться вместе с его звуковой дорожкой.
- **NOAUDIO** – проигрывание видеопотока будет производиться без его звуковой дорожки. Значение установлено по умолчанию.
- **ALPHA** – включает режим показа прозрачности для AVI-файлов с альфа-каналом или неопределённым каналом (например, при компрессии кодеком SoftLab-NSK Forward JPEG with Alpha(FRWT)).
- **NOALPHA** – выключает режим показа прозрачности для видеофайлов.
- **AUTOALPHA** – видеотекстура будет показана прозрачной, если видеофайл содержит изображение с прозрачностью(альфа - канал). Значение установлено по умолчанию.

 Показ изображения с прозрачностью поддерживается только для файлов в формате AVI. При попытке проиграть MPEG- или WMV-файл в формате ALPHA, изображение не будет показываться в видеотекстуре.

- **LFF** – режим воспроизведения видеофайла Lower Field First, то есть при воспроизведении видеофайла первым будет показано нижнее поле кадра. Режим установлен по умолчанию для видеопотоков типа **FILE**.
- **UFF** – режим воспроизведения видеофайла Upper Field First, то есть при воспроизведении видеофайла первым будет показано верхнее поле кадра.
- **NOFIELDS** – показ изображения осуществляется полными кадрами, без разделения на поля.
- **AUTOFIELDS** – для **MPEG**-видеофайлов порядок показа полей в кадре определяется типом компрессии файла. Для остальных типов файлов очередность показа полей в этом режиме зависит от наличия информации в системе. Если информация отсутствует, видеофайл воспроизводится в режиме прогрессивной развёртки.
Установлен по умолчанию для видеопотоков типа **MSDS**.

 Из-за особенностей определения порядка показа полей при воспроизведении видеофайлов в режиме **AUTOFIELDS** рекомендуется при проигрывании файлов с чересстрочной развёрткой всегда явно указывать очередность показа полей – **LFF**, **UFF** или **NOFIELDS**.

- **REMOTE** – форматирование видеопотока для вывода изображения с удалённого компьютера (см. ниже описание команды **RENDER.VIDEO.VideoStreamName.DATA = Remote_Computer_IP**).
- **CAPTURE** – форматирование видеопотока при работе с внешними видеоисточниками: web-камерой, платой видеозахвата **Vision RGB-PRO** и так далее (см. ниже описание команды **RENDER.DUMP.MSDS.SOURCES = 1**).
- **LOCAL** - форматирование видеопотока после работы с внешними видеоисточниками (см. выше описание формата видеопотока **CAPTURE**)

для восстановления корректного режима работы с видеофайлами, находящимися на диске компьютера. Значение установлено по умолчанию.

При создании видеопотока (см. выше описание команды **RENDER.VIDEO.VideoStreamName.CREATE = iParam**) по умолчанию установлены форматы:

для **LIVE_*** - **ALPHA, CROP, LOWRES, NOMIPMAP, SYNC;**

для **FILE_*** - **NOLOOP, NOAUDIO, LOWRES, REPLACE, AUTOALPHA, LFF.**

для **MSDS_*** - **NOLOOP, NOAUDIO, LOWRES, REPLACE, AUTOALPHA, AUTOFIELDS, LOCAL**

 *Исполнение команды **RENDER.VIDEO.VideoStreamName.FORMAT = FormatString** вызывает остановку воспроизведения видеопотока, если он предварительно был запущен командой **RENDER.VIDEO.VideoStreamName.START = 1**.*

Следующие команды являются специфичными для работы с видеопотоками из файлов:

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


Команда привязывает данные видеопотоку с именем *VideoStreamName* (принимая значения **FILE_***i* или **MSDS_***i*, где *i* – номер потока): добавляет кадры из видеофайла с именем «*Filename*» к файловому видеопотоку с именем *VideoStreamName*, начиная с *iStartFrame*-ого, *iLength* кадров. Если оба параметра равны нулю или отсутствуют, то добавляется весь файл.

(описание использования команды для работы с внешними источниками видео приведено ниже).

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

Команда привязывает данные видеопотоку с именем *VideoStreamName* (принимая значения **FILE_***i* или **MSDS_***i*, где *i* – номер потока): добавляет кадры из видеофайла с именем «*Filename*» к файловому

видеопотоку с именем *VideoStreamName*, начиная с кадра *iFrom* по кадр *iTo*. Если параметры *iFrom*, *iTo* не указаны или *iFrom* = *iTo*, то к видеопотоку добавляется весь файл.

 *Для файлов формата *.wmv не поддерживается режим фрагментарного покадрового добавления из файла. При попытке добавить данные фрагмента из такого файла к видеопотоку, указав для него значения *iFrom* и *iTo* в окне *Message(Debug Output)* появляется соответствующее сообщение.*

RENDER.VIDEO.VideoStreamName.VOLUME = iVolume

Устанавливает значение громкости звука *iVolume* при воспроизведении видеофайлов через поток *VideoStreamName*. *iVolume* может принимать значения от 0 (отсутствие звука – 10000 дБ) до 100 (- 0 дБ). Значение, устанавливаемое при наборе через систему подсказок 50 (-5000 дБ). При указании значения вне допустимого диапазона оно будет установлено одним из ближайших к нему граничных значений: 0 либо 100.

RENDER.DUMP.MSDS.SOURCES = 1

Выводит в окно **Message (Debug Output)** список всех видеоисточников, работа с которыми осуществляется через фильтр *DirectShow* (Рис. 2). В списке могут содержаться как внешние источники (например, плата видеозахвата **Vision RGB-PRO Capture** или Web-камера **Dual Mode USB Camera**), которые можно указывать в командах в качестве источника видеосигнала, так и внутренние источники, ресурсы которых используются при работе приложения *HotActions* (**SLTM Dshow Video Capture board 1, SLTM Dshow Video Output Capture board 1** и так

далее). При дальнейшей работе со внешними видеоисточниками в командах можно указывать полное имя, указанное в строке, например, **RENDER.VIDEO.MSDS_1.DATA="Vision RGB-PRO Capture"**, либо отдельные ключевые слова из строки, например, **RENDER.VIDEO.MSDS_1.DATA=" USB Camera"**.

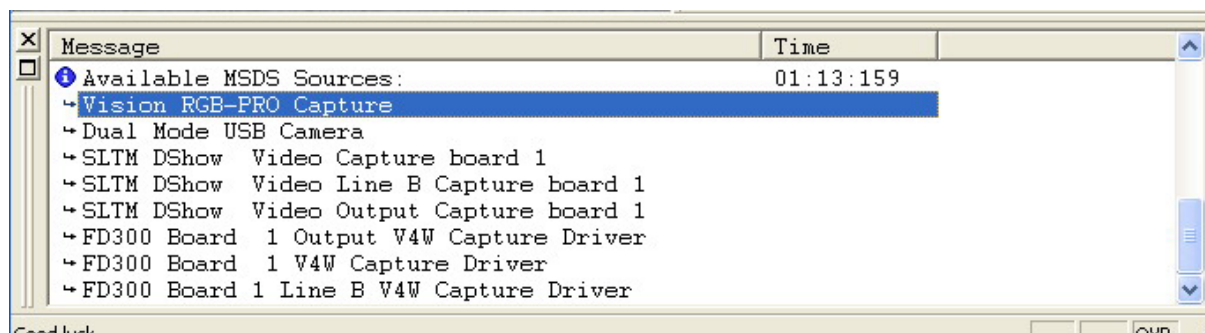


Рис. 2. Список видеоисточников для работы через фильтр DirectShow, выводимый в окно Message (Debug Output) при выполнении команды **RENDER.DUMP.MSDS.SOURCES=1**.

RENDER.VIDEO.VideoStreamName.DATA = Device

Данные с изображением, поступающим с внешнего видеоисточника *Device* (см. описание предыдущей команды) привязываются к видеопотоку с именем *VideoStreamName* (типа **MSDS_*i***, где *i* – номер потока).

Следующая команда является специфичной для вывода изображения с удалённого компьютера:

RENDER.VIDEO.VideoStreamName.DATA = Remote_Computer_IP

Команда связывает видеопоток с именем *VideoStreamName* (типа **MSDS_*i***, где *i* – номер потока) с изображением, поступающим с удалённого компьютера с ip-адресом, задаваемым параметром *Remote_Computer_IP*. О соответствующей настройке удалённого компьютера см. главу 14.

RENDER.VIDEO.VideoStreamName.SIZE= fsizeX, fsizeY

Задаёт разрешение видеоизображения для видеопотока с именем *VideoStreamName* (типа **MSDS_*i***, где *i* – номер потока):

fsizeX – разрешение по горизонтали;

fsizeY - разрешение по вертикали.

Будет установлено разрешение видеоизображения из поддерживаемых указанным видеоисточником, которое наиболее соответствует указанным значениям *fsizeX*, *fsizeY*.

7.2 Использование видеопотоков в сцене

Следующая команда служит для работы с материалами в сцене:

RENDER.MATERIAL.MaterialName.SOURCE = VideoStreamName

Назначает материалу с именем *MaterialName* видеопоток с именем *VideoStreamName* (то есть **LIVE_1** или **FILE_2** и т.п.). Видео текстура будет наложена на все объекты в сцене с этим материалом. Если видео текстуры с таким именем нет, то восстанавливается последняя загруженная для материала «обычная» текстура, то есть изначальная текстура из 3D-сцены или её замена из файла или текст, если пользовались командами **DATA.MATERIAL.MaterialName.MAP** или ***.TEXT**.

Использование видео текстуры подразумевает следующие шаги:

1. Создание видеопотока

RENDER.VIDEO.LIVE_1.CREATE = 1

2. Запуск видеопотока

RENDER.VIDEO.LIVE_1.START = 1

3. Сопоставление видеопотоку материала VIDEO в сцене, на котором будет показываться видеоизображение

RENDER.MATERIAL.VIDEO.SOURCE = LIVE_1


7.3 Режим вывода текстур материалов 3D Оверлей

В связи с тем, что в телевидении реализован чересстрочный режим развертки, при передаче мелких деталей, может возникнуть мерцание строк — так называемый фликкер-шум. Фильтр, применяемый против этого, основан на размывании выходного изображения.

В некоторых случаях, когда разрешение текстуры (например, видеопотока - с камеры или из файла) совпадает с разрешением материала, на который она накладывается, размывание изображения становится заметно.

Решением является режим вывода текстур **3D Оверлей**, при котором текстурные координаты (например, изображения актёра при крупном плане) проективно пересчитываются от текущей виртуальной камеры, результатом чего является точное попадание тексела текстуры в пиксел экрана. Оставшаяся 3D сцена, в которую помещена такая текстура, рисуется как обычно.

При этом видеотекстура (например, изображение актёра) остаётся в сцене на исходном объекте и перед ней по-прежнему можно помещать другие объекты. Таким образом, этот режим влияет только на улучшение качества вывода текстуры.

 Для работы в режиме 3D Оверлей рекомендуется создавать текстуры размером в полный экран. При совпадении разрешений монитора и текстуры переход в этот режим происходит автоматически, не требуется исполнять команды, описанные ниже.

RENDER.MATERIAL.MaterialName.OVERLAY = iMode, fPeriod

Первый параметр **iMode** включает/выключает использование режима **3D Оверлей** для вывода текстуры материала с именем **MaterialName** – 1 или 0 соответственно.

Для того чтобы сгладить переход между режимами **3D Оверлей** и обычным выводом текстур, вторым параметром команды **fPeriod** можно указать время (в секундах) за которое необходимо осуществить плавный переход.

Данная команда является асинхронной, то есть система не дожидается окончания её выполнения. Когда режим действительно включится или выключится, генерируется событие **MATERIAL.MaterialName.OVERLAY**. Оно генерируется мгновенно, если **fPeriod=0.0** или команда задаёт переход в режим, который и так установлен в данный момент.

7.4 Дополнительные команды режима 3D Оверлей

Если кадр не успеет отрисоваться за отведённое время, вместо него показывается предыдущий кадр. То есть кадр, который не успевает изображаться на экране, теряется. Соответственно, во всех материалах, выводимых в режиме **3D Оверлей**, со следующего кадра чётность поля, используемого в генерируемом кадре, не будет совпадать с чётностью выходного поля.

Следующие команды были введены для автоматического исправления такой ситуации:

RENDER.SYNC = iSync, iSkip

При **iSync**, равном нулю, используется ручной режим переключения чётности. В этом случае необходимо самостоятельно следить за правильностью отображения полей и, в случае необходимости, вызывать команду **RENDER.FLIP**, описанную далее.

Если **iSync** больше нуля, то используется автоматическая подстройка чётности поля, причем значение **iSync** указывает число несинхронных

кадров подряд, после которых необходимо пересинхронизоваться. Параметр *iSkip* определяет способ, которым происходит синхронизация:
iSkip = 1 – с помощью пропуска одного полукадра входного видео;
iSkip = 0 – с помощью смены чётности выводимого поля входного видео.

RENDER.FLIP = 1

При выключенной автоматической синхронизации (см. предыдущую команду) данная команда меняет очерёдность вывода полей кадра для всех оверлейных материалов.

8 Системные команды

SYS.WAIT = «Event» или

SYS.WAIT.EVENT = «Event»

Останавливает исполнение сценария до завершения события *Event*.

SYS.EVENT = «Event»

Регистрация завершения события *Event*. Запускает выполнение сценария, отложенного по команде **SYS.WAIT = «Event»** или **SYS.WAIT.EVENT = «Event»**.

SYS.INFO="Wait"

Выводит в окно **Message (Debug Output)** информацию о всех незавершённых событиях **SYS.WAIT = «Event»** или **SYS.WAIT.EVENT = «Event»**. После сообщения о каждой незавершённой команде **SYS.WAIT = «Event»** в окно **Message** выводится список команд, которые должны были исполниться после завершения этой команды.

SYS.DELAY = fTime

Останавливает исполнение сценария на *fTime* секунд.

SYS.RESET = 1 Сброс ожидания любых событий. Исполняется по умолчанию при инициализации проекта, запускаемого кнопкой **Init** на панели инструментов основного окна приложения или при переходе в рабочий режим *LiveAction* (см. Руководство пользователя *HotActions*, раздел 3.1.9).

SYS.SPLASH = Text

Отображает в рабочей области приложения диалоговую панель с сообщением об ожидании **Please, wait...** и текстом **Text**. Убрать панель с экрана можно командой **SYS.SPLASH = 0**.

SYS.OUTPUT = «Message»


Выводит текстовое сообщение *Message* в окно **Debug Output**.

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

Позволяет осуществлять выборочную трассировку в окно *Debug Output*. В этом окне будут появляться сообщения только об исполнении команд, содержащих слова из списка **Word1,Word2, ...,** то есть слова **Word1, Word2** и так далее.

SYS.MACRO.Name = «Data»

Макроопределение. В тексте всех сценариев слово **Name** заменится на **«Data»**. Отменить все макроопределения можно командой **SYS.MACRO = «»**. Рекомендуется имена макроопределений начинать с символа **«\$»** (см. главу 17).

 Если в качестве макроопределения **«Data»** используется текст, то имя макроса **Name** во всех командах необходимо заключать в кавычки, например **DATA.NODE.Text_node.TEXT="$\\$text$"**.

SYS.ACTION = «File»,fPeriod

Исполняет команды из текстового файла **«File»**. При ненулевом значении *fPeriod* команды из файла исполняются с этим периодом (время указывается в секундах). Если период не задан, то команды исполняются после каждого изменения файла (его сохранения). Файл может загружаться не только с диска, но и через internet по протоколам HTTP, FTP и POP3. Для этого достаточно написать имя в форме стандартного url. Например:


«<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»

Как и при обычном запуске *Action* 'ов, после исполнения каждой команды из файла в окне **Message(Debug Output)** формируется соответствующее сообщение о завершении события.


 Если исполнено несколько команд **SYS.ACTION = "File", fPeriod**, рабочей будет только та, которая исполнена последней. Из двух исполненных **SYS.ACTION = File1** и **SYS.ACTION = File2** действительной будет только одна, то есть, исполняться после изменений в текстовых файлах *File1*, *File2* будут команды только одного из этих файлов.

SYS.ACTION.RECORD = «Filename»

Команда включает запись трассировки *Debug Output* (см. главу 20) в файл «**Filename**» (если «**Filename**» = **0**, то запись прекращается). После завершения отрисовки каждого кадра в файл добавляется команда вида **SYS.DELAY = {время кадра}**.

SYS.ACTION.FILTER = «command1», «command2»...

Данная команда задаёт список команд для трассировки в файл, то есть записываются команды, начинающиеся с «**command1**», «**command2**».

 Для того, чтобы исключить трассировку конкретного вызова команды достаточно поставить перед ней символ «@». Это может быть необходимо, чтобы исключить «замусоривание» **Debug Output**, например, непрерывным потоком команд от джойстика.

9 Команды работы со звуком

SOUND.DIRECTORY = «Path»

Устанавливает путь *Path* к рабочей директории для звуковых файлов.

SOUND.Name.OPEN = «FileName»

или

SOUND.Name.FILE = «FileName»

Создаёт звуковой поток с именем *Name* и связывает его со звуковым файлом *FileName*. Воспроизведение всех звуковых потоков происходит через *DirectShow*.

SOUND.Name.PLAY = iMode

Выключает (*iMode=0*) и включает (*iMode=1*) проигрывание звукового потока с именем *Name*.

SOUND.Name.NOTIFY = «Event»

Определяет текст «*Event*» сообщения, которое будет появляться в окне **Message (Debug Output)** после завершения воспроизведения звукового потока *Name*. По умолчанию при завершении воспроизведения этого потока в окно **Message** генерируется сообщение *SYS.EVENT = «SOUND.Name»*.

SOUND.Name.LOOP = iMode

Выключает (*iMode=0*) и включает (*iMode=1*) режим циклического проигрывания звукового потока с именем *Name*.

SOUND.Name.PAUSE=1

Останавливает проигрывание звукового потока с именем *Name*.

SOUND.RESET = 1

Останавливает воспроизведение всех звуковых потоков и удаляет все именованные звуковые объекты. Исполняется по умолчанию при инициализации проекта, запускаемой кнопкой **Init** на панели инструментов основного окна приложения или при переходе в рабочий режим *LiveAction* (см. Руководство пользователя *HotActions*, раздел 3.1.9).

10 Команды управления курсором мыши

Описанные в этой главе команды действительны, если вывод изображения в студии осуществляется через второй выход видеокарты (в **Display device** диалога настроек вывода выбрано устройство *Display Adapter* - описано в главе 8 руководства пользователя *HotActions*). Например, при работе в виртуальной студии с видеосигналами HD (глава 11 Руководства пользователя *HotActions*),

CONTROL.MOUSE.SELECT = iMode

Задаёт возможность (если *iMode=1*) выбирать объекты в сцене при помощи курсора мыши в режиме работы *LiveAction*. При *iMode = 0* эта возможность отключается.

CONTROL.MOUSE.CURSOR = sMode

Определяет форму курсора мыши. При *sMode = HAND* курсор принимает форму руки, при *sMode = ARROW* устанавливается в форме стрелки. По умолчанию установлена форма *ARROW*.

CONTROL.MOUSE.SELECT.ALPHA = iMode

Задаёт возможность (*iMode=1*) выбирать объекты с прозрачностью при помощи курсора мыши. Возможность можно определить, только если исполнена команда, описанная выше – **CONTROL.MOUSE.SELECT = 1**. При *iMode = 0* возможность выбора объектов с прозрачностью отключается.

11 Команды работы с джойстиком

JOYSTICK.Name = IID

Ассоциирует джойстик, имеющий идентификатор *iID* с именем *Name*:

iID может принимать значения от 1 до числа подключенных джойстиков, то есть данная команда служит для именованного подключения джойстиков по их порядковым номерам. Поэтому *iID* конкретного джойстика должен удостоверяться опытным путём, но определённый один раз остаётся таким же, если не менять подключение джойстиков к компьютеру;

если *iID* некоторых джойстиков совпадают, то используется последний джойстик, объявленный на этот *iID*.

JOYSTICK.Name.MOVE = «Commands»

Устанавливает выполнение команды *Commands* на движение ручки или слайдера джойстика с именем *Name*, например, при выполнении команды **JOYSTICK.JOY.MOVE=«DATA.NODE<CURRENT>.POS»** будет определено, что смещение ручки или слайдера джойстика *JOY* вызывает перемещение текущего выбранного объекта – выполняется команда **DATA.NODE<CURRENT>.POS**. Диапазон смещения объекта будет определяться следующей командой (см. описание следующей команды). По умолчанию все значения установлены равными 0.5.

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

Устанавливает чувствительность на движение слайдера или движение и вращение ручки джойстика с именем *Name* шестью числами, в пределах от 0.0 до 1.0 по каждой оси (по три на вращение и движение).

JOYSTICK.Name.BUTTON.Butt = IID

Устанавливает кнопку с именем *Butt* джойстика с именем *Name* целый идентификатор *iID* для дальнейшего использования в командах. Если *iID* некоторых кнопок у одного джойстика совпадают, то используется последняя кнопка, объявленная на этот *iID*.

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

Задаёт левую *Commands* и правую *Argument* части команды, которая будет выполняться при отжатии кнопки *Butt* джойстика с именем *Name*. например, команда

JOYSTICK.Name.BUTTON.Butt.UP = «TRACK.Track.PLAY», «0»

определяет, что отжатие кнопки *Butt* будет выполнять команду **TRACK.Track.PLAY=0**, то есть останавливать трек с именем *Track*.

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

Задаёт левую *Commands* и правую *Argument* части команды, которая будет выполняться при нажатии кнопки *Butt* джойстика с именем *Name*. Например, команда

JOYSTICK.Name.BUTTON.Butt.DOWN = «SOUND.Name.PLAY », «1»

определяет, что нажатие кнопки *Butt* будет выполнять команду

SOUND.Name.PLAY = 1, то есть запускать воспроизведение звукового потока с именем *Name*.

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

Задаёт левую *Commands* и правую *Argument* части команды, которая будет выполняться при удержании кнопки *Butt* джойстика с именем *Name*, команды при этом посылаются в каждом кадре.

JOYSTICK.Name.PATH = «Commands»

Устанавливает строку команд *Commands* со значениями, доконфигурированными командами *AXIS*, *SLIDER*, *POV* (см. далее),

которая посылается системе на выполнение при движении ручки и слайдеров джойстика и нажатии его кнопок.

Например: *JOYSTICK.Name.PATH* = «*DATA.NODE.<CURRENT>*».

JOYSTICK.Name.AXIS.«Axis» = «Dest», fMinRange, fMaxRange

Задаёт диапазон значений и строку, добавляемую к командам, заданным *JOYSTICK.Name.PATH* (см. выше):

Name – имя джойстика;

Axis – имя оси джойстика (например, для осей перемещения **X, Y, Z**, для осей вращения **RX, RY, RZ**);

Dest – строка, уточняющая, как будут интерпретироваться значения, полученные от ручки джойстика для команд, заданных *JOYSTICK.Name.PATH*;

(например, для интерпретации значений как перемещений по соответствующим осям – **POS.X, POS.Y, POS.Z**, а для вращения **ROT.X, ROT.Y, ROT.Z**);

fMinRange, fMaxRange – задают диапазон значений для данной оси джойстика (в пределах от 0.0 до 1.0).

Например, команда *JOYSTICK.«JOY».AXIS.«X»* = «*POS.Y*», 0.5, 1.0

устанавливает, что движение ручки джойстика по оси X, например на 0.6, будет посылать системе следующую команду (если до этого была исполнена *JOYSTICK.Name.PATH* = «*DATA.NODE.<CURRENT>*»):

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

JOYSTICK.Name.SLIDER.iNum = «Dest», fMinRange, fMaxRange

Задаёт диапазон значений и строку, добавляемую к командам, заданным *JOYSTICK.Name.PATH* (см. выше), для исполнения системой при возникновении событий от соответствующего слайдера:

Name – имя джойстика;

iNum – номер слайдера джойстика (1,2,...). Если номер пропущен, то это интерпретируется, как слайдер с номером 1;

Dest – строка, уточняющая, как будут интерпретироваться значения, полученные от соответствующего слайдера для команд, заданных *JOYSTICK.Name.PATH*. Например, для интерпретации перемещений по соответствующим осям – **POS.X, POS.Y, POS.Z**, а для вращения – **ROT.X, ROT.Y, ROT.Z**;

fMinRange, fMaxRange задают диапазон значений для данного слайдера (в пределах от 0.0 до 1.0).

Например, команда *JOYSTICK.«JOY».SLIDER.1* = «*ROT.Z*», 0.5, 1.0 устанавливает, что движение слайдера по оси **X**, например на 0.6, будет посылать системе следующую команду, если до этого была исполнена *JOYSTICK.Name.PATH* = «*DATA.NODE.<CURRENT>*»:

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

JOYSTICK.Name.POV.iNum.«Axis» = «Dest», fMinRange, fMaxRange

Задаёт диапазон значений и строку, добавляемую к командам, заданным *JOYSTICK.Name.PATH* (см. выше), для исполнения системой при возникновении событий от соответствующего *Point of View Hat* джойстика:

Name – имя джойстика;

iNum – номер *Point of View Hat* джойстика (1,2,...). Если номер пропущен, то это интерпретируется, как *Point of View Hat* с номером 1;

Axis – оси *Point of View Hat* (X или Y);

Dest – строка, уточняющая, как будут интерпретироваться значения, полученные от соответствующего **Point of View Hat** джойстика для команд, заданных *JOYSTICK.Name.PATH* (например, для интерпретации перемещений по соответствующим осям – **POS.X, POS.Y, POS.Z**, а для вращения – **ROT.X, ROT.Y, ROT.Z**);

fMinRange, fMaxRange – задают диапазон значений для данного **Point of View Hat** джойстика (в пределах от 0.0 до 1.0).

Например, команда *JOYSTICK.«JOY».POV.«X» = «ROT.Y», 0.5, 1.0* устанавливает, что движение **Point of View Hat** по оси X, например на 0.6, будет посылать системе следующую команду (если до этого была исполнена *JOYSTICK.Name.PATH = «DATA.NODE.<CURRENT>»*):

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

☞ Вы можете не указывать «общую» строку команд с помощью *JOYSTICK.Name.PATH* для «доконфигурирования» командами *AXIS, SLIDER, POV*. В этом случае необходимые команды должны быть указаны полностью (параметром **Dest** команд *AXIS, SLIDER, POV*), а не только как дополнения к «общей» части, задаваемой *JOYSTICK.Name.PATH*.

JOYSTICK.Name.PLAY = iMode

Деактивирует (*iMode=0*)/активирует (*iMode=1*) все установленные настройки джойстика с именем *Name*. То есть, чтобы все настройки джойстика, указанные в отданных командах, - чувствительность движения и вращения ручки, идентификаторы кнопок, диапазон значений слайдера и так далее - стали действительны, необходимо отдать эту команду с параметром 1.

JOYSTICK.PLAY = iMode

Деактивирует (*iMode=0*)/активирует (*iMode=1*) все установленные настройки для всех джойстиков, определённых в командах. То есть, чтобы все настройки всех джойстиков, указанных в отданных командах, стали действительны, необходимо отдать эту команду с параметром 1.

JOYSTICK.Name.RESET=iMode

Отменяет все настройки джойстика *Name*.

JOYSTICK.RESET=iMode

Отменяет настройки всех джойстиков.

☞ Для того чтобы откалибровать джойстик, используйте **Gaming Options** панели управления Windows 2000, там же вы можете получить информацию о джойстике и *IDs* (идентификаторах) его кнопок.

☞ Обратите внимание: для использования описанных команд работы с джойстиком необходимо, чтобы библиотека *DLL (plug-in)* соответствующей версии была установлена. Команды, описанные выше, поддерживаются *plug-in*'ом **JOYSTICK** версии не ниже 1.0.0.497. Информация о том, какие библиотеки *DLL (plug-in*'ы) загружены, выводится в **Debug Output** при запуске приложения **HotActions** (см. следующий рисунок).



Рис. 3. Информация о версии библиотеки **JOYSTICK**, выводимая в **Debug Output** при запуске приложения **HotActions**.

12 Команды работы с роботизированной камерой

Осуществлять работу с роботизированной камерой возможно только при установке дополнительного модуля *VSPTZControl*.

PTZ.«CameraName».CREATE = «CameraType», «ComPortName»

Этой командой, прежде всего, необходимо инициализировать модуль управления камерой. Также необходимо назначить подключенной к определенному COM-порту компьютера камере имя, для использования в остальных командах.

Параметр *CameraType* задаёт тип камеры, поддерживаемые типы – **EVI-D30, EVI-D31, CANON VC-C1, CANON VC-C3**.

Укажите параметр *ComPortName* равным *COM1* для подключения к первому COM-порту компьютера, *ComPortName = COM2* – для подключения ко второму COM-порту компьютера и так далее.

PTZ.«CameraName».CALIBRATE = 1

Команда выполняет калибровку инициированной камеры «*CameraName*».

PTZ.«CameraName».POWER = iValue

Команда управления питанием камеры «*CameraName*»:

IValue = 1 – включить питание;

IValue = 0 – выключить питание.

PTZ.«CameraName».FOV = iValue

Устанавливает FOV камеры «*CameraName*»:

iValue – угол зрения по горизонтали в градусах.

PTZ.«CameraName».ROT = fa, fβ, fy

Команда поворота камеры «*CameraName*»:

fa, *fβ*, и *fy* – углы Эйлера (углы поворота вокруг осей X,Y,Z).

PTZ.«CameraName».DROT = fa, fβ, fy

Осуществляет поворот камеры «*CameraName*» на углы *fa*, *fβ*, *fy* в системе координат X, Y, Z относительно текущего положения камеры.

Завершение любой из вышеперечисленных команд управления роботизированной камерой сопровождается событием **SYS.EVENT=«CameraName»**.

PTZ.«CameraName».REACTION = fValue

Время реакции камеры «*CameraName*»:

fValue – время выполнения команды в секундах, все команды будут выполняться за время, не меньшее указанного.

PTZ.«CameraName».QUEUE = iValue

Данная команда включает очередь команд камеры «*CameraName*» (в этом режиме все команды помещаются в очередь и выполняются последовательно):

iValue = 1 – включить (значение по умолчанию);

iValue = 0 – выключить.

PTZ.«CameraName».RESET = 1

Команда очищает очередь команд камеры «*CameraName*».

PTZ.RESET = 1


Очищает очередь команд всех подключенных камер.

PTZ.«CameraName».DELETE = 1

Деактивирует инициализированный модуль управления камерой «*CameraName*».

PTZ.DELETE = 1

Деактивирует все инициализированные модули управления камерами.

 *Обратите внимание: для использования описанных команд управления роботизированной камерой необходимо, чтобы библиотека DLL (plug-in) соответствующей версии была установлена. Команды, описанные выше, поддерживаются plug-in'ом **PTZControl** версии не ниже 1.5.0.13. Информация о том, какие библиотеки DLL (plug-ins) загружены, выводится в окно **Message(Debug Output)** при открытии проекта в приложении **HotActions** (см. следующий рисунок), если включена опция **Script Trace** в настройках приложения на закладке **Debug Output** (см. раздел 9.3 Руководства пользователя *HotActions*).*

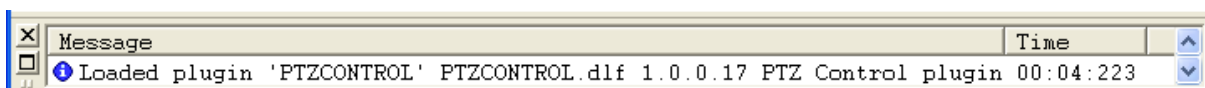


Рис. 4. Информация о версии библиотеки PTZControl, выводимая в Debug Output при запуске приложения *HotActions*.

13 Работа с трекингом видеопотоков в сцене (передача данных о движении)

Проследивать движение актёров в сцене можно только при дополнительной установке модуля *VSTracker*.

TRACKER.Name.SOURCE = «Video»

Определяет поток слежения движения с именем *Name* и привязывает его к видеопотоку *Video*.

TRACKER.Name.LISTENER = Path

Определяет путь *Path*, которому будет пересылаться поток данных *Name*. В качестве пути *Path* может быть указано, например, имя роботизированной камеры (см. предыдущую главу).

TRACKER.Name.START = 1

Запускает поток слежения с именем *Name*.

TRACKER.Name.STOP = 1

Останавливает поток слежения с именем *Name*.

TRACKER.RESET = 1

Останавливает все потоки слежения.

TRACKER.Name.SMOOTH = iDepth

Определяет количество кадров *iDepth*, по которым сглаживаются данные потока *Name*. По умолчанию *iDepth* = 4.

TRACKER.Name.FOV = fFOV

Устанавливает угол слежения *fFOV* по горизонтали (в градусах) для потока *Name*. Может принимать значение от 0 до 180. По умолчанию *fFOV* = 50.

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

Определяет масштабные множители при передаче данных слежения командой *Tracker.Name.Listener = Path*: *fX* – множитель смещения по оси *X*, *fY* – смещения по оси *Y*, *fZ* – смещения по оси *Z*, *fRX* – поворота вокруг оси *X*, *fRY* – поворота вокруг оси *Y*, *fRZ* – поворота вокруг оси *Z*. По умолчанию *fX* = 0, *fY* = 0, *fZ* = 0, *fRX* = 1, *fRY* = 1, *fRZ* = 1.

TRACKER.Name.DELETE = 1

Удаляет поток слежения с именем *Name*.

TRACKER.DELETE = 1

Удаляет все потоки слежения.

14 Использование DesktopCapture для вывода изображения с удалённого компьютера


Для использования вывода изображения с удалённого компьютера необходимо, чтобы соответствующая версия дополнительного программного обеспечения была установлена. Настройки, описанные ниже, поддерживаются версией *VSDesktopCapture* номер 107 (то есть на удалённом компьютере необходима инсталляция *VSDesktopCapture107.exe*). Для управления получением изображения с удалённого компьютера необходимо запустить приложение—*DesktopCapture.exe*. Это можно сделать с помощью меню **Start** (Рис. 5), либо с помощью иконки **DesktopCapture**  на рабочем столе *Windows*.



Рис. 5. Запуск приложения *DesktopCapture.exe* на удалённом компьютере.

Представлено приложение *DesktopCapture* одноимённым окном, содержащим три панели.

Панель *Capture Control Panel* (Рис. 6) служит для запуска/остановки процесса захвата и передачи изображения (кнопка *Start*), а также для возможности свернуть приложение в системный лоток.

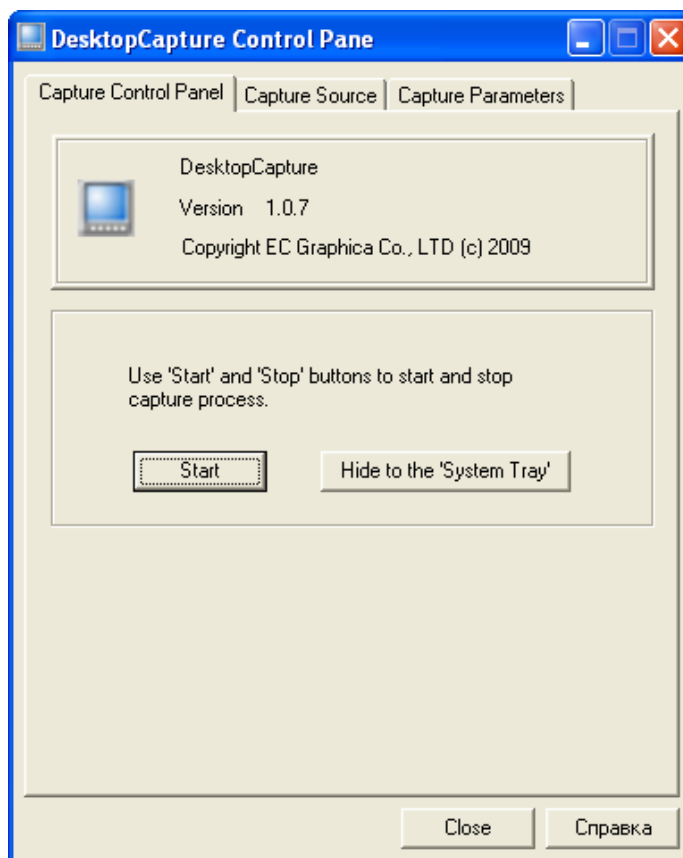


Рис. 6. Панель *Capture control panel* приложения *DesktopCapture.exe*.

Панель *Capture Source* (Рис. 7) служит для выбора окна, изображение которого будет передаваться (группа **Window selector tool**). Если установлена опция **Capture all windows on desktop**, то будет передаваться изображение рабочего стола и всех окон на нём.

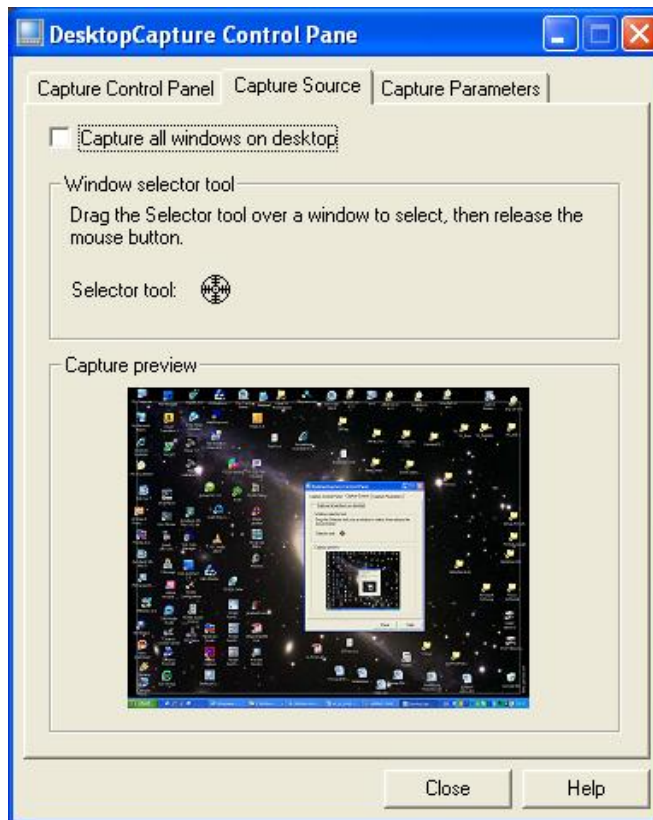


Рис. 7. Панель *Capture Source* приложения *DesktopCapture.exe*.

Панель *Capture Parameters* (Рис. 8) служит для установки частоты обновления захватываемого изображения окна в группе **Capture frequency**, а также размеров передаваемого кадра, до которых автоматически будет масштабироваться захваченное изображение окна.

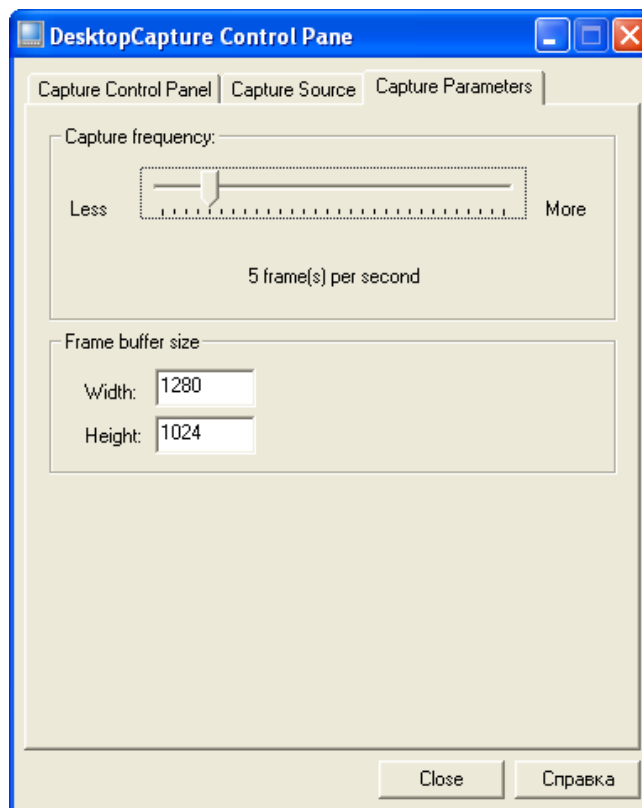


Рис. 8. Панель *Capture Parameters* приложения *DesktopCapture.exe*.

15 Команды работы с внешними устройствами через GPI (General Purpose Interface)

Приложение поддерживает внешнее управление через устройства входа(in-) и выхода(out), подключаемые к COM-портам компьютера (GPI). Эти возможности могут быть полезны для случаев, если, например, необходимо исполнять *Action*-ы командой, посылаемой с внешнего устройства, или если необходимо маркировать рабочую видеокамеру, сигнал с которой подаётся в сцену (*Tally light control*).

Описанные ниже команды действительны, если в папке модулей приложения (как правило, это C:\Program Files\Focus Software\HotActions\Plugins) находится модуль VSGPIControl.dlf (устанавливается программой VSVSGPIControl.exe).

GPI.IN.«Name».CREATE = GPI_IN_DEVICE_NAME

Присваивает имя «Name» входному сигналу от устройства *GPI_IN_DEVICE_NAME*, соединённого с com-портом *DEVICE_NAME*, например, **GPI.IN.IN1.CREATE = GPI_On_COM2_0_Input** назначает имя *INI* входному сигналу, передаваемому на вход 0 com-порта *COM2*.

GPI.IN.«Name».DELETE = 1

Удаляет сигнал, назначенный на вход com-порта, с именем «Name».

GPI.IN.DELETE = 1

Удаляет все входные сигналы, назначенные на com-порты.

GPI.IN.«Name».NOTIFY.1 = Event

Определяет текст *Event* сообщения, которое будет появляться в окне **Debug Output** при включении устройства, соединённого с com-портом с назначенным входным сигналом имени «Name». По умолчанию(без определения командой) появляется сообщение **SYS.EVENT = "GPI.IN.Name.1"**.

GPI.IN.«Name».NOTIFY.0 = Event

Определяет текст *Event* сообщения, которое будет появляться в окне **Debug Output** при выключении устройства, соединённого с com-портом с назначенным входным сигналом имени «Name». По умолчанию(без определения командой) текст сообщения **SYS.EVENT = "GPI.IN.Name.0"**.

GPI.IN.«Name».NOTIFY = Event0, Event1

Определяет тексты сообщений, которые будут появляться в окне **Debug Output** при выключении (*Event0*) и включении (*Event1*) устройства, соединённого с com-портом с назначенным входным сигналом имени «Name». По умолчанию(без определения командой) появляются сообщения **SYS.EVENT = "GPI.IN.Name.0"** и **SYS.EVENT = "GPI.IN.Name.1"**.

GPI.IN.«Name».RESET = 1

Переопределяет тексты сообщениями по умолчанию для всех сообщений, появление которых в окне **Debug Output** установлено при изменениях состояний устройства, соединённого с com-портом с назначенным входным сигналом имени «Name».

GPI.IN.RESET = 1

Переопределяет тексты сообщениями по умолчанию для всех сообщений, появление которых в окне **Debug Output** установлено при изменениях состояний устройств, соединённых с com-портами с назначенными входными сигналами.

GPI.IN.«Name».CHECK = ActionName0, ActionName1

Исполняет *Action* с именем **ActionName0**, если выключено устройство, соединённое с com-портом с назначенным входным сигналом имени

«Name»); если устройство включено, исполняет *Action* с именем **ActionName1**.

GPI.OUT.«Name».CREATE = GPI_OUT_DEVICE_NAME

Назначает выходной сигнал с именем «Name» устройству *GPI_OUT_DEVICE_NAME*, соединённому с com-портом *DEVICE_NAME*, например, **GPI.OUT.OUT1.CREATE = GPI_On_COM1_1_Output** назначает имя *OUT1* выходному сигналу, передаваемому на выход 1 com-порта *COM1*.

GPI.OUT.«Name».DELETE = 1

Удаляет сигнал, назначенный на выход com-порта, с именем «Name».

GPI.OUT.DELETE = 1

Удаляет все сигналы, назначенные на выходы com-портов.

GPI.OUT.«Name».SET = iState

Устанавливает состояние выходного сигнала с именем «Name», *iState = 1* соответствует состоянию “включено”, *iState = 0* – состоянию “выключено”. По умолчанию *iState = 0*.

GPI.OUT.«Name».RESET = 1

Устанавливает состояние выходного сигнала с именем «Name» в состояние по умолчанию.

GPI.OUT.RESET = 1

Устанавливает состояния всех выходных сигналов, соединённых с com-портами, в состояние по умолчанию.

GPI.OUT.«Name».PULSE = fTime

Включает выходной сигнал с именем «Name» на *fTime* секунд.

16 Зарезервированные имена

Объектом, выбранным по умолчанию для манипулирования джойстиком, мышью или с помощью стрелок, при открытии сцены является объект с именем **<World>**. Этот объект добавляется в качестве главного узла сцены при её экспорте. При манипулировании объектом **<World>** ничего не происходит, так как вместе со сценой соответственно изменяются и виртуальные камеры. Имеется также специальное имя для использования в командах сценария, всегда указывающее на имя текущего выбранного объекта – **<Current>** (раздел 3.1). Для выбора нужного объекта манипулирования необходимо поместить команду, устанавливающую его текущим, в инициализирующей *Action*, например **:DATA.CURRENT.NODE = «Main»**. Для виртуальных камер также можно использовать специальную команду, например **:DATA.CURRENT.CAMERA = «Camera1»**.

При смене объекта манипуляций соответственно изменяется имя, на которое указывает **<Current>**.

17 Использование переменных в командах сценария

В командах сценария можно использовать переменные. Самое важное применение переменных – это *Theme* (тема). Темы представляет собой вариацию проекта за счёт набора переменных, значения которых при активизации этой темы становятся доступными в командах. Подробнее о темах см. раздел 3.8 Руководства пользователя *HotActions*.

Для того, чтобы выделять переменные, рекомендуется начинать их имена со специального символа «\$», например: **DATA.CURRENT.NODE = \$nodeName**.

После активации любой темы, в которой задано значение переменной **\$nodeName**, при исполнении данной команды будет подставлено значение **\$nodeName**.

18 Уточнение области действия переменных

Необходимо напомнить о способе уточнения области действия переменных, таких как *Sound* в строке *SOUND.Sound.FILE = «FileName»*. Могут возникнуть трудности при пересекающихся именах, например при ссылке на них из *Action*'ов, инициализирующих разные сцены, объединённые в один проект. Если вы хотите убедиться, что действие будет выполнено для вполне определенной сцены, установите эту сцену текущей до запуска такого рода команды, используя следующую запись: *DATA.CURRENT = «Scene»*. Это ограничивает область действия последующих команд, исключая использование их для другой сцены, что могло бы привести к непредсказуемым последствиям.

Чтобы использоваться совместно, имена траекторий, *Action*'ов и звуковых фрагментов должны быть уникальны для всех открытых сцен. В любом случае, две различные траектории (или два звуковых фрагмента) не могут обозначаться одним и тем же именем: указание идентичного имени просто переназначит его.

19 Примеры сценариев

Рассмотрение уже готовых примеров и их модификация – один из продуктивных способов научиться использовать команды сценария и разобраться в примерах уже готовых проектов. Приложение *HotActions* поставляется с простыми проектами, которые устанавливаются в отдельную папку, как правило, с именем *VS_Sample* и содержат каждый: сцену в формате **.3d*, библиотеку *Actions Library* и *Hotset*. Можно экспериментировать с этими файлами, чтобы научиться добиваться нужных результатов в создании виртуальных сцен.

Необходимо отметить, что большинство команд, приведённых в данном руководстве, работают только после запуска инициализирующего (*startup*) *Action*'а для конкретной сцены. Принципы и примеры создания инициализирующих *Action*'ов рассмотрены в разделе 3.4.6 Руководства пользователя *HotActions*. Также настоятельно рекомендуется ознакомиться с примерами, приведёнными в отдельном документе, описывающем работу с *3D Studio MAX*.

20 Использование *Debug Output* для отладки сценариев

В приложении *HotActions* имеется возможность отслеживать сообщения об ошибках и предупреждения, в том числе связанные с использованием команд сценария. Для этого используется окно вывода *Message (Debug Output)* (Рис. 1-4).

О настройках окна вывода *Debug Output* и работе с ним см. главу 6 Руководства пользователя *HotActions*. Отметим только, что необходимым условием отладки сценариев с его помощью является установка опций **Runtime (from Actions)** и **Script Trace** на панели **Debug Output** диалога *Options*. О фильтрации для вывода команд, а также о трассировке в файл см. замечание в главе 8 данного документа.