Virtual Set 2000

Environment Builder
for Virtual Studio Sets Control

3D-Scene Creation Guide

Revision 1.07

DARIM

Darim Vision Co., Ltd.

Limited Warranty

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

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

Table of Contents

1 Introduction

Virtual Set 2000 (*VS2000*) uses 3D objects and scenes created in third party 3D scene generators and allows them to be rendered in real-time with live video added. The user can move the about the scenes, dynamically change virtual camera angles and lighting. Currently **3D Studio Max** (*3DStudio*) is supported by *VS2000*. *VS2000* provides a plug-in for use by *3DStudio*. Support for other 3D scene generators is under development and updates will be available through the *VS2000* web site.

From within *3DStudio*, scenes can be exported to *VS2000* in the *VS2000* scene file format. To prepare for use by *VS2000*, the *3DStudio* file is simplified but with minimal loss in quality. Never seen partsof objects to the camera are eliminated, needless texture detail is reduced and other techniques are implemented to assist real-time rendering of scenes in *VS2000*.

This manual assumes the reader is familiar with how to use the program *3D Studio Max* (versions 4.2 unless noted) and is skilled at creating objects for use in a 3D-scene. Specifically the reader should have knowledge of virtual cameras and their animation, etc. When working in *3D Studio Max* turn to chapter 2 which outlines some recommendations and restrictions for enhancing the compatibility between *3DStudio* and *VS2000*.

For use in a *VS2000* project scene prepared in *3D Studio Max*, should be exported in a file with extension ***.3d**. Use of the tool *VS2000 3D Exporter* (see section 4.3) allows for a preliminary look at a scene in a 3D-format directly from *3D Studio MAX*. It is strongly recommended to use this preview tool since rendering in *VS2000* can yield results that can be distinguished from rendering results in *3D Studio MAX*. Distinctions can amplify appreciably if at the restrictions on scenes for export listed in section 2.1 are not observed.

2 3D-scene creation for *VS2000* in *3D Studio MAX*

2.1 Restrictions on exported parameters

Export and subsequent use by *VS2000* of scene files created in *3D Studio Max* assumes adherence to the minimum requirements of a 3D-scene for use in *VS2000.* Similar to most computer games, VS2000 takes full advantage of the features of graphic 3D-accelerator hardware. Certain restrictions/requirements help VS2000 fully utilize 3D-accelerator hardware. These requirements will aide the successful functioning in real-time mode, a technique not available within 3DStudio.

The recommended settings in **3D Studio Max** are given below. By following these recommendations, it is possible to create really impressive results in *VS2000*.

2.1.1 Audio export

The sound placement in a scene with the help *3D Studio Max*, is exported in full. The lone restriction is that it is inaccessible during preview mode (section 4.3.2).

2.1.2 Geometry

Parameters exported are:

- **Standard primitives**
- **Extended primitives**
- **Compound objects**

2.1.3 Cameras

From cameras only the following parameters are exported:

- **Target camera**. Please note that cameras of this type are usually controlled through ***Dummy*** objects. It is important for ability of correct rotation of the camera about z-axis. Also track (if any) is created not for **Target camera** itself but for its ***Dummy*** object.
- **Free camera**

2.1.4 Light sources

It is possible to use up to 8 light sources. Use of the following types of light sources is available:

- **Omni** – use of sources of this type is preferable.
- **Target Spot** – the surface illuminated by such source, should be tessellated, the tessellation degree is selected experimentally.
- **Free Spot**
- **Free Direct**
- **Target Direct**

Only the following parameters of light sources are supported.

- **Attenuation** is supported, if a surface was tessellated.
- **Multiplier** is supported.

2.1.5 Hierarchy

Hierarchies supported are:

- **Pivot**
- **Linking**
- **Unlink**

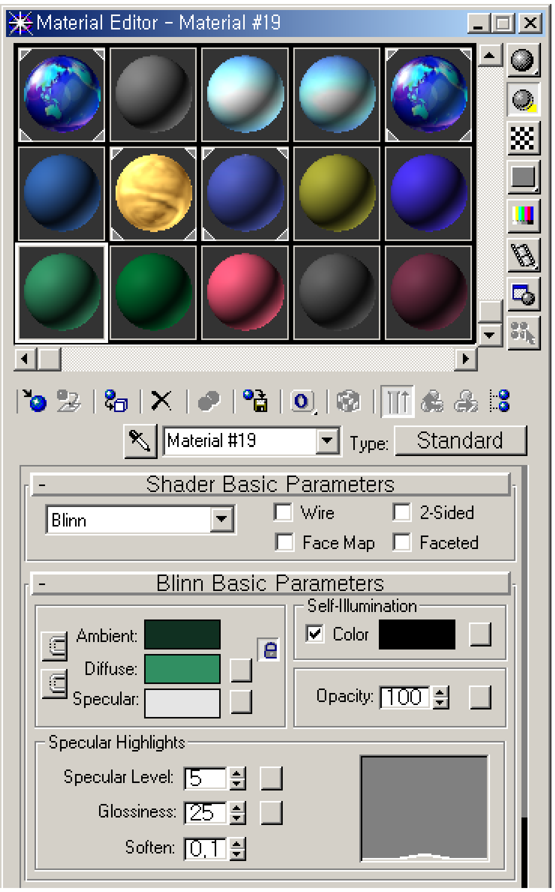
2.1.6 Environment

- Only the environment **Ambient** is supported, and no more than 8 sources can be used.

2.1.7 Materials

Types of material supported are:

- **Standard**
- **Multi/sub-object**
- For **Standard** materials only certain **Blinn** parameters from **Shader Basic Parameters** are exported (Figure 1) the others are ignored. Of the **Blinn Basic Parameters,** the only parameters exported are: **Ambient**, **Diffuse**, **Self illumination**, **Opacity**, **Specular**, **Specular Level**, **Glossiness**. Other shading-parameters are ignored.



Material Editor - Material #19

Material #19 Type: Standard

Shader Basic Parameters

Blinn

□ Wire □ 2-Sided

□ Face Map □ Faceted

Blinn Basic Parameters

Self-Illumination: ☑ Color

Ambient:

Diffuse:

Specular:

Opacity: 100

Specular Highlights

Specular Level: 5

Glossiness: 25

Soften: 0.1

Figure 1. Dialog of a choice of a material

2.1.8 Textures

When mapping textures both static images, and AVI-files can only use **Bitmap** files. In addition, for structure maps, only the following parameters are supported:

- **Diffuse color**;
- **Ambient color**;

☝ *Note: this color is used as an additional **Diffuse color**, instead of as an **Ambient color**.*

☝ *The possible total size of AVI-files used for textures should be found approximately for every scene. The reason is that they are exported as a set of low-compressed bitmaps and thus might dramatically enlarge* *size of exported 3D-scene file. This consequently increases loading time and may leads to performance problems.*

- **Opacity***;*
- **Reflection**.

Note: To learn more about mapping textures, turn to section 2.1.9.

2.1.9 Coordinates

For textures – coordinates exported are:

- **UV offset**;
- **UV tiling**;
- **W Angles**;
- from **Texture/environ** – only texture;
- from **Mapping** – only **Explicit Map Channel**.

The maximum number of textures for a material is three. The following combinations are allowable:

- **Ambient color + Diffuse color + Opacity:**

  if **Alpha source** in **Bitmap parameters** are marked as **No (Opaque)**, maps will be simply mixed proportionally to exposed values (**Amount**). The sizes of textures (in pixels) for **Diffuse color** and **Opacity** should be identical. Values **U** and **V** for **Diffuse color + Opacity** should be identical;

  if **Alpha source** in **Bitmap parameters** are marked as **RGB Intensity**, maps will be mixed in a special manner (**multiply**). In this case **Amount** - values of parameters of a structure are ignored.
- **Diffuse color + reflection:**

  modes of mixing are the same as in **Ambient color + Diffuse color**.
- **Diffuse color + Opacity:**

  the sizes of textures (in pixels) for **Diffuse color** and **Opacity** should be identical. Values **U** and **V** for **Diffuse color + Opacity** should be identical.

*Note: The size of a map should not be bigger, than is necessary, and should not exceed 1024 (2048 for GeForce 3, 4096 for GeForce 4 and so on) pixels. It is desirable to manage the size of textures for the optimal perfomance to be 256 for both height and width. The issue is that at export to a 3D-file, the sizes of textures are automatically rescaled so that they equal 2 pixels in degree N, where N –is an integer. Thus, in pixels it should be: 2, 4, 8, 16, 32, 64, 128, 256, 512, and so on. What size texture map should the designer make? Lets imagine that designer has made a texture of the size 257 by 513. At export the closest size will be automatically established such that a minimum quality of the texture is not lost. The closest size uses the guidelines from above. Based on the above criteria, the allowable size for the texture becomes a texture of 512 by 1024. This will lead to needlessly using the resources of the computer. Therefore it is better to create a texture (for example, in Photoshop) that is in natural proportions. Then the scaling of a texture to the nearest allowable values will not reduce quality of a picture. Otherwise it is possible scaling will make the image flattened or stretched. In this case it has no value, thus you see in the 3D-scene the proportion depends exclusively on how textural coordinates (**UVW Mapping**) are given. On the one hand, making textures with the suggested sizes is not a rigid requirement, but on the other hand - it is extremely desirable to use the suggested sizes especially if there are a lot different textures in the scene.*

2.1.10 Auxiliary objects

The only auxiliary objects exported are **Dummy** objects.

2.1.11 Animation

Animation is only supported for cameras, light sources and actually objects in a scene. **Bones** are not supported as well as dynamic changes of a material such as changes in transparency are not supported. Animation of the **Visibility Track** is supported.

2.2 Morphing in Virtual Studio

Morphing is executed by use of the script command **MORPH,** which is a subcommand of the RENDERcategory of commands. (see the document *Script Commands Guide*). The RENDER.MORPH command requires an original object, a target morphed object and a specification of the length of time allocated to the morphing process. The length of time is specified in seconds.

To subject objects to morphing, make the following steps:

1. create the basic original object;
2. create the target mrphed object making certain it has the same number of verticesas the original object;
3. specify the time for morphing in number of seconds.

Note: The original object and the target morphed object must have the same number of vertices). Also the basic original object must not be invisible, but target morphed object could be.

2.3 Generation of light effects and surface roughness

Shadows and light flashes are important elements for the visual perception of a scene. Limited use of flashes can make the scene look plain and lifeless. Note that export of shadows sources is not supported. The solution to this problem is the generation of the images of shadows and light flashes for use as textures. These can be made various ways. Below is one recommended method?

- Create the type and quantity of light sources necessary for an impressive illumination of the scene and creation of shadows. Ignore all restrictions described above.
- Beforehand transform to BMP-files the primary light sources of a scene on the objects. In most cases this is lighting used for the floor, walls, ceiling etc. Use a perpendicular projection of the light to a surface of object.
- Update the textures used for scene materials with the help of a program such as *Adobe Photoshop*.
- Remove all minor light sources, leaving only those required and no more than 8 sources.

2.4 Problems with use of reflection with materials

When using reflection materials, it is possible there will be a break of the image on joints of the big faces. Also distortions can appear, when the object has big anglesbetween sides. All these distortions are caused by lack of perspective transformation correction.

If distortions are too distracting, possible solutions to this problem are:

- splitting of large faces (tessellation);
- concealment of borders between sides, i.e. accommodation of objects in a scene so that borders of their large sides are not reflected;
- in case of the big angles between faces, it is possible to smooth corners by adding small superfluous faces.

2.5 Reduction of amount of polygons in a 3D-scene

Note: The 3D-scene should not contain an excessive number of objects with many faces and should not be overloaded with a great many different textures.

Objects used for scenes can be optimized. This can be done a variety of ways, below are a few suggestions for optimization of objects:

- It is desirable to know possible positions of the camerain a scene. The further an object is from the camera, the less detail required of the object.
- Remove or simplify all objects, which are not important to the perception of a scene and the scenario.
- Remove all objects, which can be modeled by a texture, for example, a sheet of a paper on a table etc.
- Optimize the flat and poorly bent surfaces of objects, editing parameters of objects and/or using modifier **Optimize**.
- Remove all surfaces of objects, which are never observed by the camera.

2.6 Live video

The live video image used in a scene could replace any material in the scene. The material used for the video image should have a **diffuse map**. After initialization of a video material in a scene, this map will be replaced with the video image. Details of this process are described in 3.3.1 and other user manuals that come with *VS2000*. During this process, all other maps and parameters of the material will be ignored.

The material name used for the video has no value. In the example projects, the materials are usually referred to as **VIDEO1** and **VIDEO2** if two streams of video are simultaneously used. If in the scene only one video stream is used, the material is frequently named **VIDEO_m** (rather than merely **VIDEO**) to avoid the confusion when "VIDEO" means category of commands. For example, **VIDEO** in RENDER.VIDEO.LIVE_1. CREATE = 1 –is part (category) of a command. And **VIDEO_m** is the material name in RENDER.MATERIAL.VIDEO_m.SOURCE = LIVE_1.

We recommend applying video material to rectangulars with aspect ratio 4:3 (both for PAL and NTSC).

2.6.1 Texture creation for a video material

The texture used for a video material, has no value. It will be replaced with a live video after initialization of a scene in *VS2000*. For a better scene composition and designer's convenience, it is commonly recommend to use the static image of the actor as the texture for material intended for live video.

For this purpose use two textures – **Diffuse** and **Opacity**.

For **Diffuse Map** use a photo (figure) of the actor.

For **Opacity Map** – a mask is drawn in *Adobe Photoshop* on the basis of the same photo or figure. Use a white color fill in of the silhouette of the actor. This will be the opaque part of a material. In the other field of a mask fill color black will be the transparent part.

All operations for **UVW Mapping** will be kept during replacement by live video. We recommend applying a video material to a rectangular with the side proportions of 4:3 both for PAL and NTSC.

Nevertheless, it is allowable to use as a basis for video any material of the scene containing the setting **diffuse map**. Thus, the video image can on any object of a scene – a wall, a screen and even a teapot.

☝ *Note: When working with VS2000 there may be black strips on the edge of the video image in a scene. The black strips on the edges can be eliminated by cutting off of edges of the frame, which makes these areas transparent. The cutting off adjustment is executed from the* **Keying adjustment** *dialog found by pressing the (button) in the KeyConfigPro dialog for HotActions. In some cases the black edges will still be present, then it is necessary to use the following additional script commands with appropriate parameters:*

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

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

☝ *The parameters such as fPosU or fPosV should be obtained experimentally.*

☝ *Note: To avoid the angle between actor plane and the camera to be small, always put before the name of an actor plane the symbol "**!**". This will cause the object to behave similarly to a wind vane, making it always facing the camera (see also section 2.7).*

2.7 Objects naming in 3D-scene and reserved prefixes for names

☝ *Note: For accuracy it is desirable to give names to materials and objects a name appropriate to their purpose, thus simplifying work with them in HotActions.*

Special names are used for an object to be manipulated by the mouse or joystick, thus enabling the user to change the object's position, size or other parameters:

- The object with name ALL is set up for manipulation by a mouse or joystick by default when scene is just opened. If object ALL is absent, then the first camera is chosen to be manipulated by a mouse or joystick.
- It is impossible to name object in a scene the name **<Current>**. This special name is used in command of scripts as the designation for the current chosen object. As the script changes the selection of an object for manipulation, the new object is automatically temporarily assigned the name **<Current>**.
- The following symbols at the beginning of the name of object have special value:

Symbols influencing position of object relative to the camera	
#	Always force object be the person facing the camera. All angles of rotation of object are equal to zero in the system of coordinates of the observer.
!	The same, as **#**, with unique distinction - the object behaves similarly to a wind vane, turning to the camera around of the vertical axis (Z).
:	The same, as **#**, with distinction, that the object decreases with distance more slowly. It is used for light sources, which from a distance tend to seem bigger, than their actual size.
Symbols varying rendering priority	
The system uses the algorithm of the Z-buffer, which sorts surfaces for rendering in a priority and eliminates surfaces invisible to the spectator. There can be cases when it it is is necessary to change rendering order of the objects, for example, in a scene with translucent objects.	
< **>**	Changes a priority of object relative to other objects with the same ancestor on the scene tree. Objects names, which are closer to the spectator, should begin with the prefix '**<**', while those that are further should begin with '**>**'.
<< **>>**	The same, as '**<**', and '**>**', but is used to change the priority concerning all objects of a scene.

3 Creation and use of an elementary scene in *VS2000*

To use a scene as virtual scenery, the minimum contents of a scene required are:

- object with any textural material for overlay of live video;
- camera;
- light source.

In the following sections we shall discuss an elementary scene created with the help of the program *3D Studio MAX* and included as an example with the program. This is an example of the minimal scene required by via *VS2000* scene.

3.1 Making a minimal scene for a virtual studio

Startup *3D Studio MAX.* With the help of command **Open** on menu **File,** open the file **Simple.max**. This scene is a completely ready example. We shall examine its components most frequently used in studios.

To required objects are:

- camera (**Target**);
- light source (**Omni**);
- the flat rectangular with the actor image. The application will project the assigned video stream from a camera onto the material of this object.
- The unessential but useful objects for any scene are:
  - a floor and a background for the actor;
  - a table or any object, used to hide the bottom portion of an actor if provided, then the actor does not need to be in the camera at full height. For example, shooting the actor above the waist, requires the actor be behind a table;
  - «monitor» which utilizes material for a video texture from a DV-file will be used. This object initially is not seen and appears only on 125-th frame of a scene.

Also in the scene is a second camera with the name **Camera02** controlled by the dummy object **CAMERA2**. This is used for demonstration of instant switching from one camera on another with the application *HotActions*. The first camera named **Camera01** (controlled by **CAMERA1**) also demonstrates animated movement of the camera in the sample virtual studio.

Important features for creation of a scene include:

- For the actor a flat rectangular is created according to recommendations of section 2.6. The rectangular textural coordinates should create a flat object with sides in the 4:3 (both for PAL and NTSC standard). Material to be replaced by live video should be mapped **Diffuse** with a photo of the actor, and in the map **Opacity** black-and-white masks of this photo should be used. To avoid turning of the camera around a small angle in the actor's plane, place the symbol «**!**» before the name of the object. Then the object behaves similarly to a wind vane and will turn in the chamber around of its vertical axis (Z).

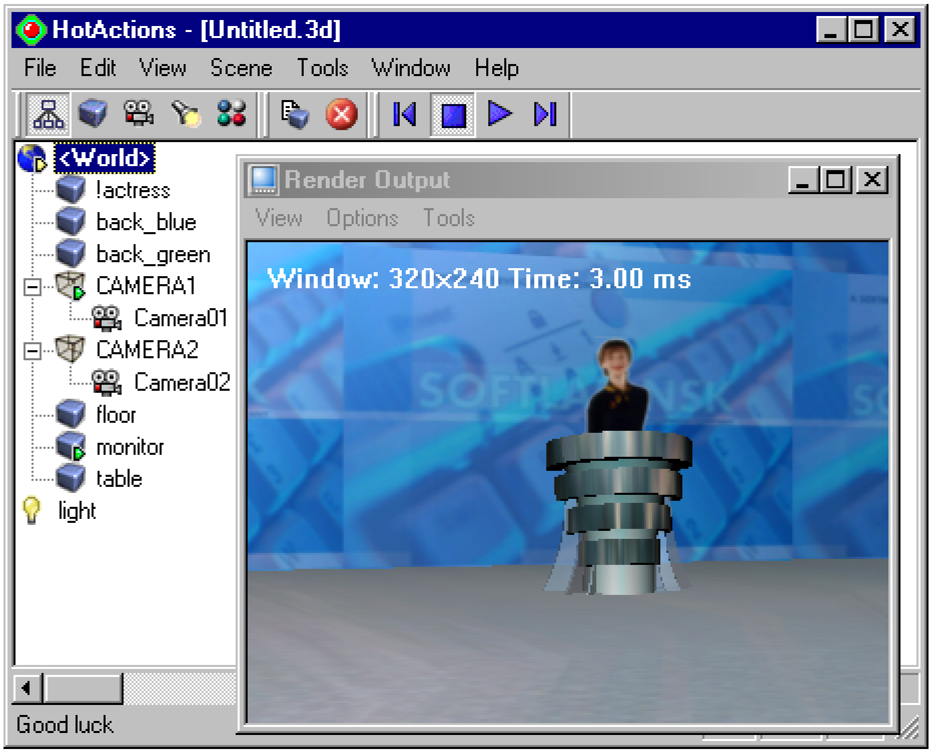
  > ☝ *Note: To make the object behave like a («wind vane») local coordinates of object, which do not coincide with global coordinates, are used. Therefore, in order to prevent problems it is recommended within 3D Studio MAX to apply the symbol «**!**» to the name of the object. Use «**Reset Xform**» to bring local coordinates of object into accord with global scene.*

- It is possible to use various scenery as a background, using the animation created in the application *3D Studio MAX*. In the example, the background is an elementary flat object. As the actor is substituted on a dark blue or green background for the more natural virtual scenery it is desirable to create a background close to green or dark blue tones depending on what background is planned for the substitution of the actor. It is possible, as in the given example, to create at once two backgrounds and as needed, hide one or the other with the help of script in *HotActions*.
- The table for the actor. This object is the closest object to the camera, therefore it requires the greatest attention to detail to make the scene more realistic. It is useful to use a map that reflects on the surface of

such object. In the given example, at the approximation of the camera angle the actor is well visible. With the help of map **Reflection,** patches of light on the material are simulated. Since the presence of a corner between big sides can break the image, a surface of cylinders has only one face in height (section 2.6).

- In the given example «monitor» – flat object without thickness with a material having one map such as **Diffuse** (similar to a plane for the actor). It is possible replace at once in *3D Studio MAX* the map of the DV-file (section 3.3.4). Later we shall understand how to use with the help of the object «monitor» an AVI-file or a slide show directly in *HotActions*.

Look in Figure 2 to see how the scene in *VS2000* will look. It is possible to look with help *VS2000 3D Exporter* (section 4.3.2).



Window: 320x240 Time: 3.00 ms

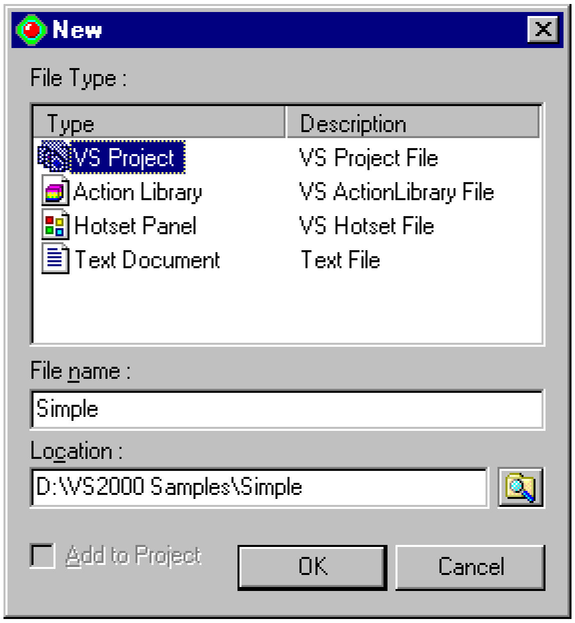
Figure 2. Example of a scene opened with the 3DStudio *VS2000 3D Exporter* dialgoue

3.2 Importing scenes from 3DStuido and new project creation in *VS2000*

When a scene created in *3D Studio Max* is export, it is given the file extension *.3d to the existing name of the scene file. This file can be opened with the **Open** command on the **File** menu of the main window of *HotActions*. It can also be saved to a different location or under a different name with the **Save As** command on the same **File** menu. (see Figure). More detail about exporting scenes from *3D Studio Max* is described in section 4.3.

Accompanying the *VS2000* application are two sample files – the first is a scene file directly created in 3DStudio, this is **Simple.max**. In the same directory is the version of this file after an export, it is in the same directory with the extenstion *.3D in other words it is named **Simple.3d**.

Scene files can be used in existing *HotActions* projects or to create a new project. For creation of a new project after starting the application *HotActions*, choose command **New Project** in menu **File** of the main *HotActions* window for dialog **New** to appear.



New

File Type :

Type	Description
VS Project	VS Project File
Action Library	VS ActionLibrary File
Hotset Panel	VS Hotset File
Text Document	Text File

File name : Simple

Location : D:\VS2000 Samples\Simple

Figure 3. Dialog *New* for Project

Enter a project name, for example «Simple», and specify a location on your hard drive. As soon as you press the **OK** button in the **New** dialog, a project window is automatically created. See Figure 4 for the new project Simple.vsp. You will see three empty group folders **Scenes**, **Librarie**s and **Hotsets**.

To add an existing file to any group, click on the group's folder and then press the right mouse button to present the contextual menu shown below in Figure 4. Now with the help of command **Add File to Group** of the contextual menu add the scene file **Simple.3d** to the **Scenes** group of the **Simple** project.

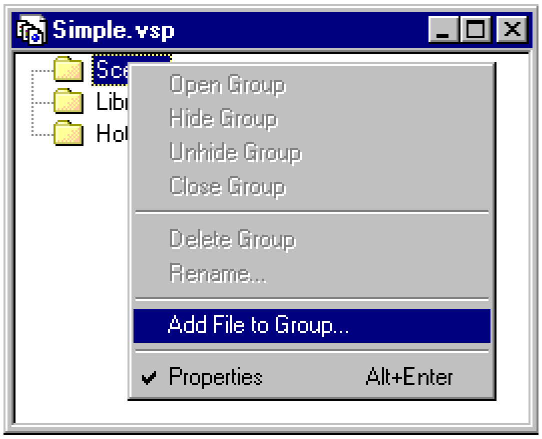
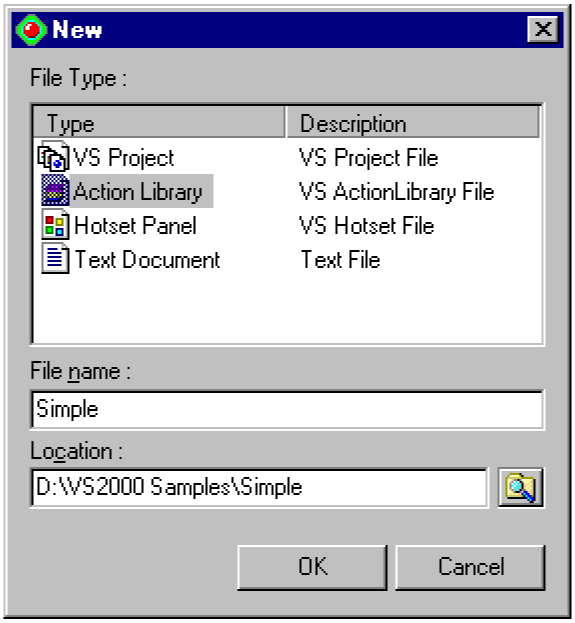


Figure 4. Addition of files in the project

In the new project it is necessary to create an ***Actions Library*** for execution of basic actions with a scene. Also a minimum of one ***Hotset*** is required for control of the start ***Actions*** from library.

Actions Library is created by choosing **Actions Library** from *New* dialog (see Figure 5) which opens by the command **New** in menu **File** of the main window of the *HotActions* application.



New

File Type :

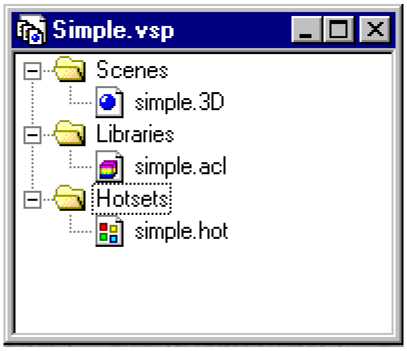
Type	Description
VS Project	VS Project File
Action Library	VS ActionLibrary File
Hotset Panel	VS Hotset File
Text Document	Text File

File name : Simple

Location : D:\VS2000 Samples\Simple

Figure 5. Dialog of a choice such as a new file for creation

Hotsets are created in the same manner by selecting **Hotset Panel** in dialog *New* (Figure 5). Since the newly created project sample is open, the creation of the new ***Action Library*** and new ***Hotset*** are automatically added to the empty groups for sample. It is shown in the expanded file tree for the **Simple** project shown in Figure 6 below.



Simple.vsp

- Scenes
  - simple.3D
- Libraries
  - simple.acl
- Hotsets
  - simple.hot

Figure 6. Updated Folders

Management of files and directories is done on the **Directories** panel of the dialog *Properties* on the contextual menu of the *Project* window.

3.3 Working with Actions in a scene

Actions are interactions within a scene such as – its display, start of separate tracks of objects etc. The following sequence is standard for creation of an ***Action*** in an open ***Action Library***.

- Click the mouse right button in an empty area of the ***Actions Library*** window. This will open up the contextual menu for the ***Actions Library*** window. (See Figure 7)
- In the opened contextual menu (Figure 7) choose command **New Action**. Immediately a new action is created with the name *New Action*. See Figure 8 below.

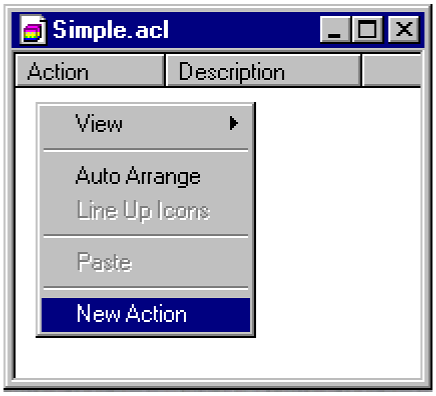


Figure 7. New Action creation in library

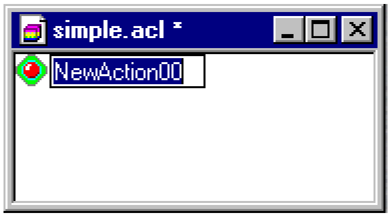


Figure 8. New Action

- It is possible to change its name by either double clicking on the **New Action** icon in the ***Actions Library*** window, by clicking on it and pressing **F2**, or by opening the contextual menu by a single click on the **New Action** and then performing a right mouse click. The contextual menu is presented below (See Figure 9). From here the name of the **New Action** can be changed by selecting the command **Rename**.

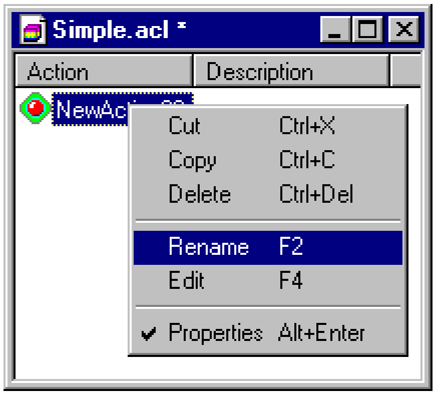
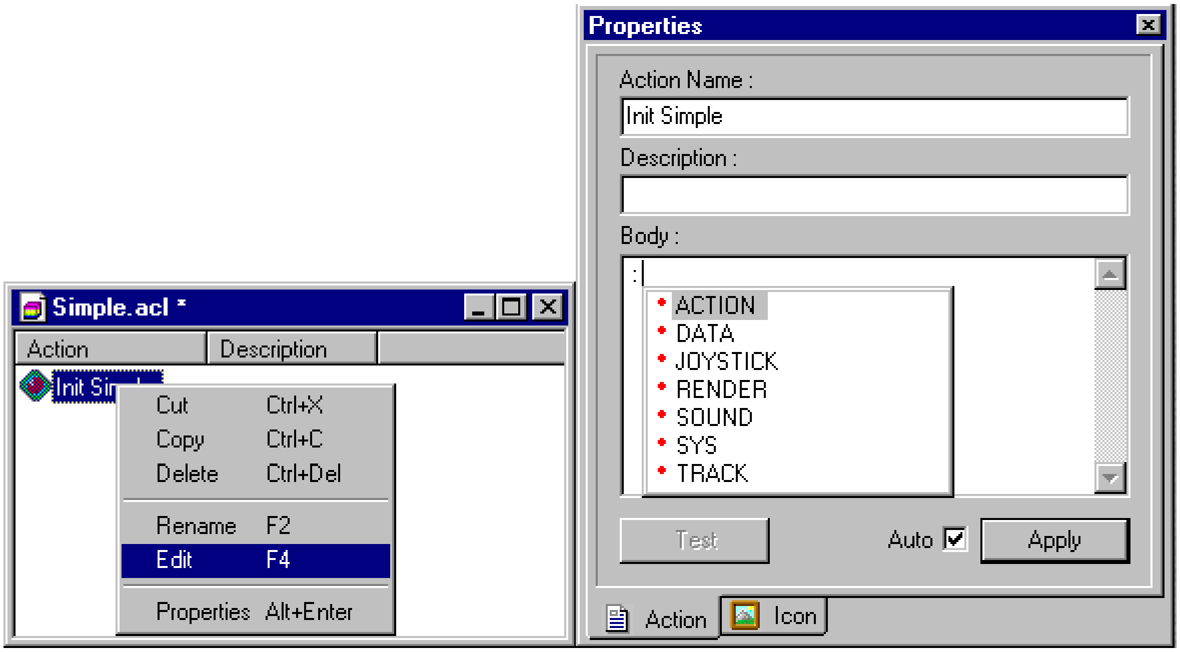


Figure 9. Action contextual menu

After creation of new ***Action*** it is possible to control the ***Action*** by writing or changing its script commands. For this purpose, use the **Edit** command on the Action contextual menu that is opened with a click of the right mouse button (See Figure 10 left). This will open the *Properties* dialog (See Figure 10 right).



Action Name : Init Simple

Description :

Body : :

Figure 10. The local menu for *Action* in library (at the left) and panel *Action* of dialog *Properties* for it

Text field **Body** in the panel ***Action*** of this dialog is used for input and editing of script commands. Entering the colon symbol («**:**») at the line beginning of a line presents the local menu (Figure 10, on the right) from which it is possible to choose accessible commands. This menu lists the categories for commands. Using the period symbol («**.**») brings up a corresponding menu which presents the available continuations for the command. This can be repeated if there are several continuations such as with the command – **DATA.CURRENT.CAMERA**. This menu system reduces the chances of incorrectly typing in commands. Commands can of course be typed in manually into the **Body** window of the ***Action*** panel.

After making any changes the **Body** window of the ***Action*** panel, it is possible to test the command by pressing the **Test** button in the ***Action*** panel of the ***Properties*** dialog. Testing can also be done by double clicking on the Action name from the ***Action Library*** window. Button **Apply** simply saves changes without initiating the actual ***Action***.

A special class of ***Actions*** are ***Startup*** actions. The following section discusses the creation of ***Startup*** actions.

3.3.1 Creation of a startup *Action*

The task of ***Startup*** commands is to initiate the working condition such as loading in a scene and putting virtual cameras in the correct position. Also there is naming convention, so in the *Sport* sample project ***Startup*** Action is called **Init Sport** while in the *Business* project it is called **Init Business**. And similarly, in the *Simple* project it is called **Init Simple**.

The ***Startup*** commands in any project could be run by the **Init** button designated by the icon 💿 when they are put to the special created folder (always named **$STARTUP$**) of any ***Actions Library***.

The **Body** window for the **Init Sample** contain the following commands which are required to carry out the initial actions of the project.

- First requirement is to load a scene file: DATA.OPEN = «Sample.3d.»
- The green background is made visible while removing (hiding) a dark blue background. To accomplish this, the command for working with the is ignored during the execution of the **Init Sample** script by placing two forward slash symbols (“**/**”) in front of the command. This is called commenting out a command line in the script., //:DATA.NODE.Back-B.HIDE = 1.

- For trajectories control in *VS2000* it is necessary to give names to them. In the scene there are present two objects having animation. The first – the virtual camera, whose control is carried out by the empty (***Dummy***) object **CAMERA1**. The second aniamted object is named **monitor**. The following commands name the tracks belonging to these objects:

  :TRACK.TV.NODE = «monitor»

  :TRACK.CAMERA.NODE = CAMERA1

  The trajectory for the object «monitor» is named TV while the trajectory for **CAMERA1** is called CAMERA.
- After naming all trajectories it is necessary to place the following command: DATA.PLAY = 1.
- During execution of the script, video streams are initiated for scene and connected to the available materials with the following commands:

  //creates a video stream connected to the first line (**LINE A**) of the first channel

  //(**LIVE_1**) of *FD300* board*:*

  :RENDER.VIDEO.LIVE_1.CREATE = 1

  :RENDER.VIDEO.LIVE_1.LINE = A

  //the format of a video stream is set at alpha - mixing is included:

  RENDER.VIDEO.LIVE_1.FORMAT = ALPHA

  //the stream is started:

  RENDER.VIDEO.LIVE_1.START = 1

  //assigns to a material with name **VIDEO_m** a video texture from video stream **LIVE_1**,

  //which will be imposed on all objects in a scene with this material:

  RENDER.MATERIAL.VIDEO_m.SOURCE = LIVE_1

The unique object in our scene named **VIDEO_m**, is plane **VIDEO** which displays the actor.

With this set of commands, we have allowed *live video* to be used in the scene.

The detail for each command is found in the document on commands. We shall note only, that any startup ***Actions***, after loading a scene and naming tracks can contain commands to operate like other ***Actions.***

Now that the ***Startup*** action is created, it is necessary to put it to the folder ***$STARTUP$*** folder of the «main» ***HotSet*** of the project. *HotActions* automatically executes contents of a folder ***$STARTUP$*** by pressing the button 💿 on the toolbar of the ***HotActions*** main window or when switching to ***LiveAction*** mode.

The folder ***$STARTUP$*** is always created at ***Hotset*** creation. By default this folder is hidden, but it is possible to display its contents by using the document view of ***Hotset*** (see Figure 11, right) activated by the **Configure** command on the Hotset contextual menu (see Figure 11 left).

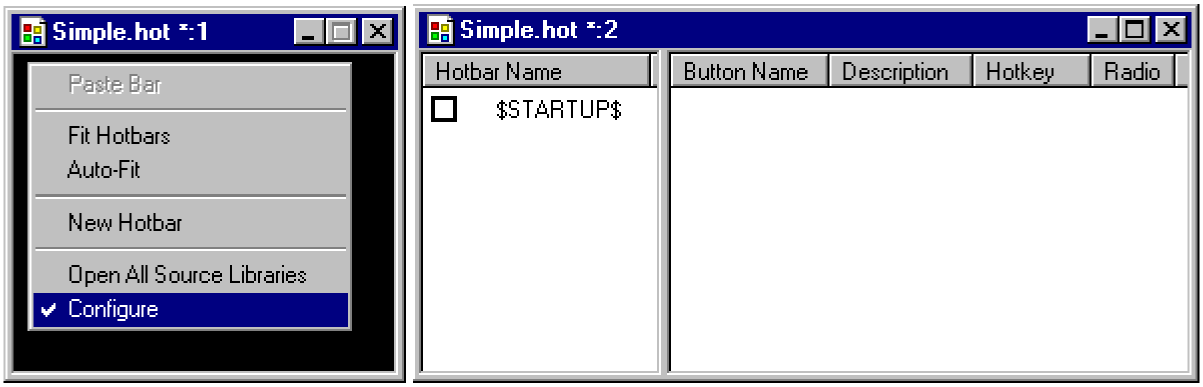


Figure 11. Contextual menu of Hotset (at the left) and its alternative representation (on the right)

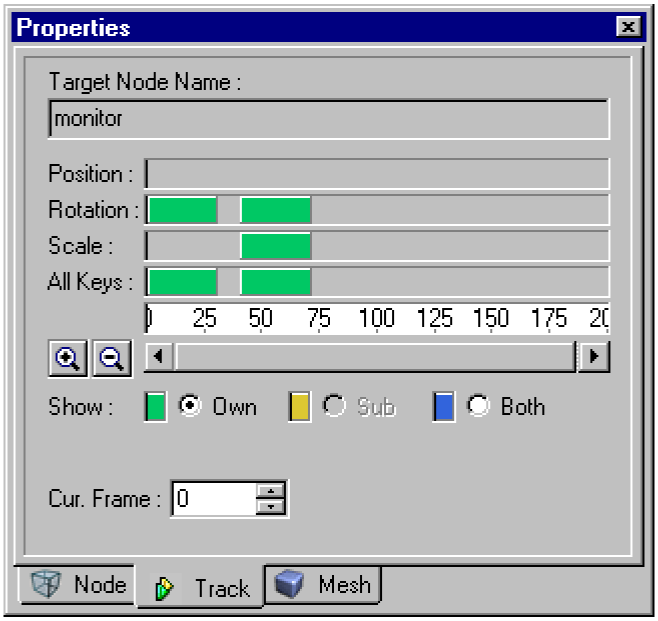
In this mode ***Hotset*** is presented by as tree structure with the groups/folders for the **Hotset** in the left window and the contents of the folders on the right. The check boxes next to the folders allow for the display or hiding of the document folders (Figure 11, on the right). In Figure 11 since the ***Hotset*** is still empty, only the unique folder ***$STARTUP$*** is displayed in its default hidden state.

Drag and drop the ***Action*** with the name **Init Sample** from the ***Actions Library*** to the folder ***$STARTUP$*** by dragging the **Init Sample** icon and then close the window of alternative representation of ***Hotset***. As the folder ***$STARTUP$*** is hidden by default (square box to the left of a name it is unchecked) any buttons for **Init Sample**, in ***Hotset*** will not appear.

3.3.2 Use of named tracks

Now tracks of objects **CAMERA1** and **monitor** are named (section 3.3.1.).

In the **Track** panel of the *Properties* dialog information for any track object contains is displayed separately by its components (Figure 12).



Properties

Target Node Name : monitor

Show : ● Own ○ Sub ○ Both

Cur. Frame : 0

Figure 12. Panel *Track* of the *Properties* dialog

On the *Properties* dialog for each object, there are panels for different types of information. On the **Track** panel (See Figure 12), displayed there are lines for changes of the object's position, changes in rotation and changes in the scale (size) of the object. On each line there are rectangles of different colors. The colors of are coded in the lower portion of the panel in the **Show** radio-buttons.

The **Show** radio-buttons indicate the following information for the object's Track:

- **Own** – shows the object's own track
- **Sub** – shows the track for the object's descendants
- **Both** – shows the tracks for the object and its descendants.

This information can be used by script commands. For example, we shall create ***Action*** for the start of the «monitor» objects track. We shall name the action **Show TV** and we shall place the following command in the body for the action:

:DATA.NODE.monitor.HIDE = 0

:TRACK.TV.GOTO = 30.

This command starts track **TV** from 100-th to the 150-th frames and providing the object TV is present by un-hiding it. An opposite result can be achieved by creating a similar command which we name **Hide TV**. The effect of this command is to smoothly remove the object from the scene over the frames 150 to 200

:TRACK.TV.START = 40,72

:DATA.NODE.monitor.HIDE = 1.

In the same way we shall create ***Actions*** using the track **CAMERA** for control of the virtual camera:

Go to Close Up, serves to have the camera to become current and approach the actor. The script command in the body the action would be

:DATA.CURRENT.CAMERA = «Camera01»

:TRACK.CAMTRACK.GOTO = 90

The action **Go to Long shot** moves the camera from the its current position to the position when it is away from the actor (frame 50) by using the following script command:

:DATA.CURRENT.CAMERA = «Camera01»

:TRACK.CAMTRACK.GOTO= 50

To focus the attention of the camera on the object «monitor», we can create an action **Go to TV** with the following script command:

:DATA.CURRENT.CAMERA = "Camera01"

:TRACK.CAMERA.GOTO = 120

:DATA.NODE.monitor.HIDE = 0

:TRACK.TV.GOTO = 30

The use of fixed frame numbers is based on knowledge of what occurs to object at in these frames. The same **Track** panel gives sufficient information to understand with the tracks of objects even if you are unfamiliar with the contents of the scene.

After naming the track of any object with the command TRACK.*NameofTrack*.NODE = *NameofObject,* in startup ***Action*** we can start the track by setting the initial frame to zero, and approximately set the last frame with the help of line **All Keys** of **Track** panel. For example: TRACK.TV.START = 0, 75.

After that, narrowing frame ranges is possible to allocate separate ranges of frames to an *Action*.

3.3.3 Working with virtual cameras

In the scene two cameras are present – **Camera01** and **Camera02**, control with which is carried out by empty ***Dummy*** objects **CAMERA1** and **CAMERA2**.

Camera01 is animated. The animation is used for actions such as zooming the camera on the *Actor* or «*monitor*» as described in the previous section. Virtual camera **Camera02** has no animation and instead is directly focused on the «*monitor*».

To switch the current camera in a scene, it is necessary to execute the following commands. These will be separate ***Actions*** with the names «Camera01» and «Camera02»):

Sets as the current camera **Camera01**   :DATA.CURRENT.CAMERA=«Camera01»

Sets as the current camera **Camera02**   :DATA.CURRENT.CAMERA=«Camera02»

3.3.4 Working with AVI–files

In the sample script are placing the video image from an AVI file on to the virtual monitor. The commands to play the video in the window of the virtual monitor are placed in the action ***Init Simple***:

We create a file video stream **FILE_1**:

:RENDER.VIDEO.FILE_1.CREATE = 1

We set a data source for video stream **FILE_1**:

:RENDER.VIDEO.FILE_1.DATA = Blue2.avi

We set video stream **FILE_1** to TV material:

:RENDER.MATERIAL.monitor_m.SOURCE = FILE_1

You can copy this actions to the special action for instance named ***Play DV*** to run them whenever you want.

Also usually we have the system wait until the file finishes playing (reaches the end of file):

:SYS.WAIT = «VIDEO.FILE_1.END»

The material of object has returned to its initial condition after working with the video file, to do so it is necessary to assign as the video source a nonexistent stream. If the stream with the given name is not found, the initial map is used. In the example, we shall return the object TV to its original texture:

:RENDER.MATERIAL.monitor_m.SOURCE = 0

Generally, after the video stream (file or «live» video) is created, it can be assigned as a material to any object in a scene, for example, video source **LIVE_1** to place on object monitor:

:RENDER.MATERIAL.monitor_m.SOURCE = LIVE_1

An example of a more professional work with a video file, using the above described ***Actions***, is given in section 3.3.6.

3.3.5 Working with graphic files

HotActions can utilize the use of graphic files in script commands.

As an example we will create an ***Action*** which we will name **Show Slide**, which will place a JPG file image on our virtual monitor.

Below is the command to use the picture file Slide.jpg on the virtual TV:

:DATA.MATERIAL.monitor_m.MAP = «Slide.jpg»

Then the material of the object is returned to its initial condition. This is done by the following command:

:DATA.MATERIAL. monitor_m.RESET = 1

It has no sense to show slide on the monitor object if it does not appear in the scene. It is of course possible to manually execute the ***Action*** with the name «**Show TV**» and then the ***Action*** with name «**Show Slide**». There is a more convenient method to connect actions so that their use will launch other ***Actions***. The following section describes working with a sequence of scenes.

3.3.6 Launching *Actions* from other *Actions*

In the previous section was described the creation of the ***Action*** «Show Slide» which places the image from a JPG–file onto the virutal monitor. It is possible to create a more complex action using existing ***Actions***. For example, we shall create the ***Action*** with the name «Monitor Show Slide», in wich «monitor» appears, the JPG slide image file is displayed for 5 seconds and then it flies away.

We will at appropriate moments also focus the camera on the proceedings. Since all described actions have already been created and stored in the library for Actions, we will now use them in a more complex action.

Since all actions need to occur consecutively one after one, we will use the fact that after each ***Action*** ends its performance it issues the event «ACTION.Name_Action». These events can be used as the parameter of command SYS.WAIT = «event» for the expectation of the fulfillment of the named ***Action***:

We focus camera at location on place, where the object monitor will appear:

:ACTION.START = «Go to TV»

This command sets the system in a wait state while ***Action*** completes its focus with the camera:

:SYS.WAIT = ACTION.« Go to TV»

This command launches the ***Action***, which places the TV object:

:ACTION.START = «Show TV»

This command sets the system in a wait state while the ***Action*** placing the object TV is completed:

:SYS.WAIT = ACTION. «Show TV»

We place the JPG image file created by the ***Action*** described in the previous section:

:ACTION.START = «Show Slide»

This command executes a 5 second pause:

:SYS.DELAY = 5.0

We now launch the ***Action*** for removing the TV object from the scene:

:ACTION.START = «Hide TV»

This command sets the system in a wait state while the Action to remove the TV from the scene is completed:

:SYS.WAIT = ACTION.« Hide TV»

We again focus the camera on the actor and set the system into a wait state while the action is finished:

:ACTION.START = «Go to Close Up»

:SYS.WAIT = ACTION.« Go to Close Up»

We return the material of the object monitor to its original setting:

:DATA.MATERIAL.monitor_m.RESET = 1

In the following ***Action*** instead of using a slide on the virtual monitor we use the contents of a certain video file. This can be accomplished by creating the ***Action*** with the name **TV Show DV**. This method differs slightly from the method described above to use a video file:

We focus the camera on the place where the monitor object will appear:

:ACTION.START = «Go to TV»

This command sets the system in a wait state until the of the ***Action*** of focusing the camera is completed:

:SYS.WAIT = ACTION.«Go to TV»

The following command starts ***Action*** to introduce the TV object into the scene:

:ACTION.START = «Show TV»

Now the system is put in a wait during the ***Action*** to introduce the TV object to the scene:

:SYS.WAIT = ACTION.«Show TV»

The following action plays the video on the virtual TV using ***Action*** created in the section 3.3.4

:ACTION.START = «Play DV»

The system now waits for the end of the video:

:SYS.WAIT = ACTION.«Play DV»

We initiate the ***Action*** to remove the TV object from the scene:

:ACTION.START = «Hide TV»

The system is in a wait state as the virtual TV exits the scene:

:SYS.WAIT = ACTION.«Hide TV»

The camera is refocused on the actor and while this is occurring the system is in a wait state:

:ACTION.START = «Go to Close Up»

:SYS.WAIT = ACTION.«Go to Close Up»

3.3.7 Hot Button creation for execution of *Actions*

With a ready supply of ***Actions*** in the *Action Library*, buttons can be assigned to various actions in order to execute the action with a single mouse click. This is the primary method for controlling actions in ***Hotset*** while in *Live Action Mode*.

Actions in the Action Library can be managed as files in a folder. They can be copied, deleted etc. They can be selected using the **Shift** or **Ctrl** or control keys such as **Ctrl-A**.

When groups of ***Actions*** are dropped to the empty space in ***Hotset***, a new ***Hotbar*** is automatically created. On the ***Hotbar*** there is a button for each action in the group. By default their names coincide with the action names. On any ***Hotbar*** it is possible to add buttons by dragging directly onto it any ***Action*** from the ***Actions library***.

Thus in essence, the ***Hotset*** is a set of ***Hotbars*** with ***Hot Buttons***. At the users convenience they can be hidden, displayed or moved simultaneously.

In the *Properties* dialog for a new ***Hotbar*** we can change its name and representations of any buttons on it. While in the *Properties* dialog for any button of a ***Hotbar*** it is possible to set or change its icon. Each

Properties dialog is accessed from the appropriate contextual menu. More detail information on working with ***Hot Buttons***, ***Hotbars*** and ***Hotsets*** are described in the appropriate chapters of user guide. The result of the working described in this section are shown in Figure 13.

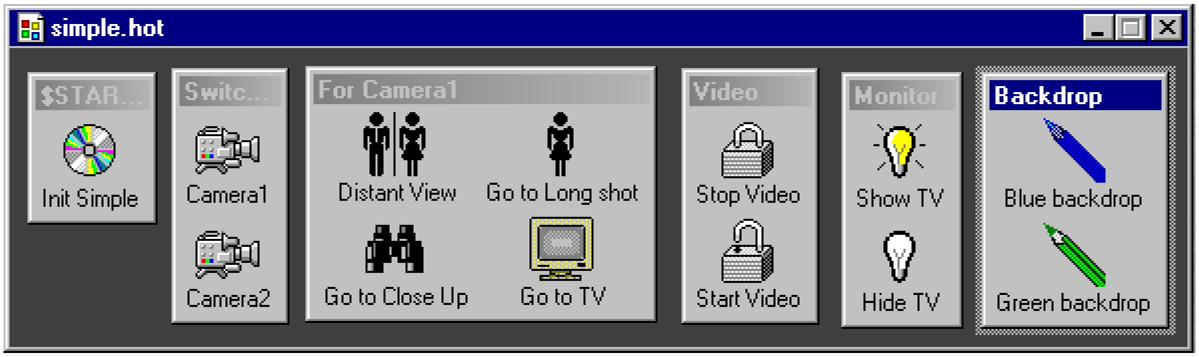


Figure 13. Ready *Hotset* and its *Hotbars* – the basic tools for working in *HotActions*

4 Scene import from *3D Studio MAX*

After a scene is created in *3D Studio MAX* it is in the file format compatible with *3DStudio* and has the extension *.MAX. To work with VS2000, it must be altered to a file format that can work with VS2000. Such files have the file extension *.3D. Scenes used in *HotActions* must have this file extension.

4.1 Installation of Plug-ins for *3D Studio MAX*

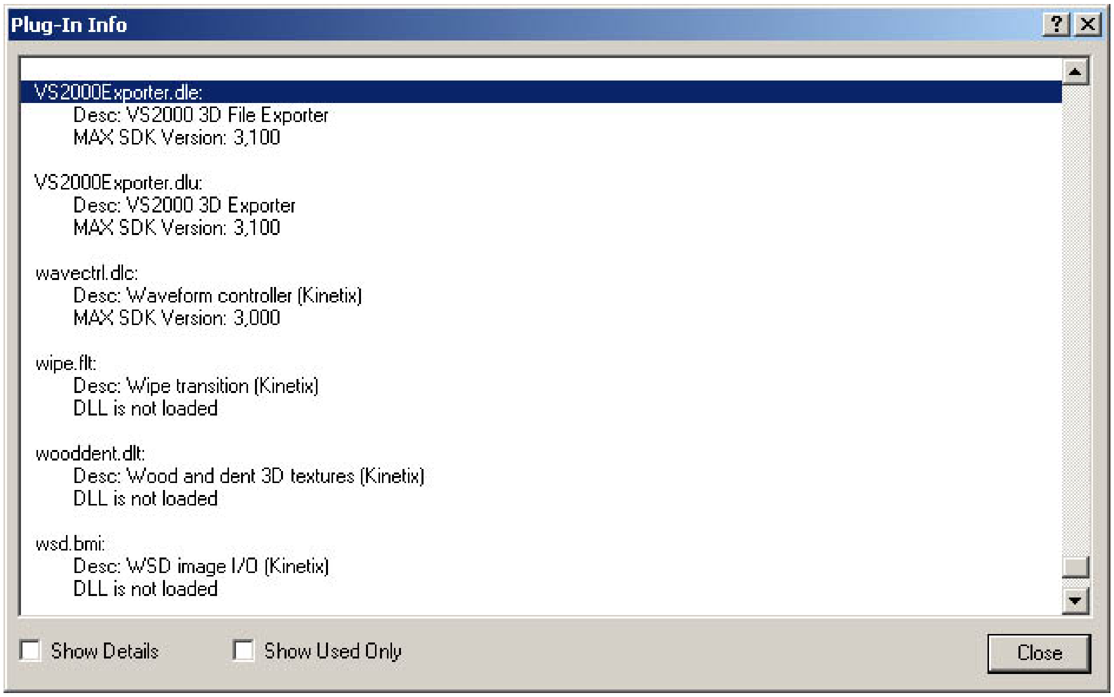
For conversion from *3D Studio MAX* format to the format compatible with *VS2000*, special modules (plug-ins) for *3D Studio MAX*, must be installed. These plug-ins are included with the VS2000 software.

If the application *3D Studio MAX* is present, then the installation of the plug-in set occurs automatically as VS2000 software is installed on your computer. If *3D Studio MAX* it is not present, then a special folder **Exporter** is created in the same directory where the program *HotActions* is installed. As a rule this will be **C:\Program Files\Darim Vision\VS2000 Software\HotActions** if VS2000 is installed on the **C:** drive of your computer.

If *3D Studio MAX* is installed on a different computer, to install the set of plug-ins on a different computer:

- copy the files **VS2000Exporter.dlu** and **VS2000Exporter.dle** from folder **Exporter** of *HotActions*
- paste these files in the folder for plug-ins for *3D Studio MAX* on the computer where *3D Studio MAX* is installed.

For check if the installation is successful, open the dialog *Plug-In Info* (Figure 14) with the command **Summary Info ...** in menu **File** *of 3D Max Studio*.



```
VS2000Exporter.dle:
      Desc: VS2000 3D File Exporter
      MAX SDK Version: 3,100

VS2000Exporter.dlu:
      Desc: VS2000 3D Exporter
      MAX SDK Version: 3,100

wavectrl.dlc:
      Desc: Waveform controller (Kinetix)
      MAX SDK Version: 3,000

wipe.flt:
      Desc: Wipe transition (Kinetix)
      DLL is not loaded

wooddent.dlt:
      Desc: Wood and dent 3D textures (Kinetix)
      DLL is not loaded

wsd.bmi:
      Desc: WSD image I/O (Kinetix)
      DLL is not loaded
```

Figure 14. ***Plug–In Info*** **Dialog**

If files **VS2000Exporter.dlu** and **VS2000Exporter.dle** are not present in the plug-ins list,, check if the full path to the is correctly specified. To perform this check use the menu **Customize** (Figure 15) and choose the command **Configure Paths ...**, this will present the dialog *Configure Path* then select the *Plug-Ins* tab (Figure 16). If the VS2000 conversion plug-ins do not appear, to manually add these plug-ins click on the **Add ...** button

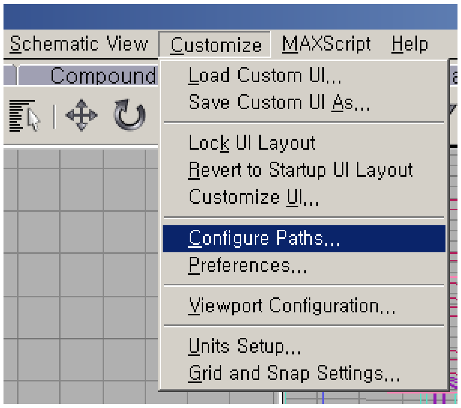
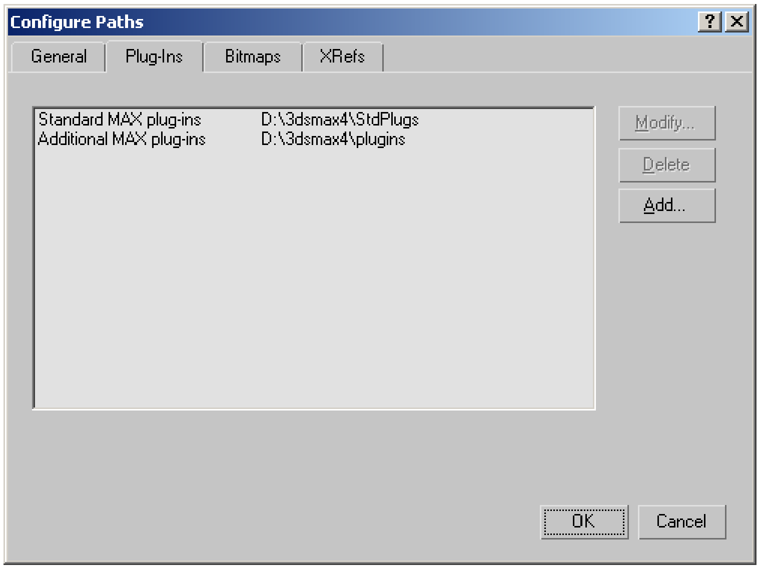


Figure 15. Menu *Customize*



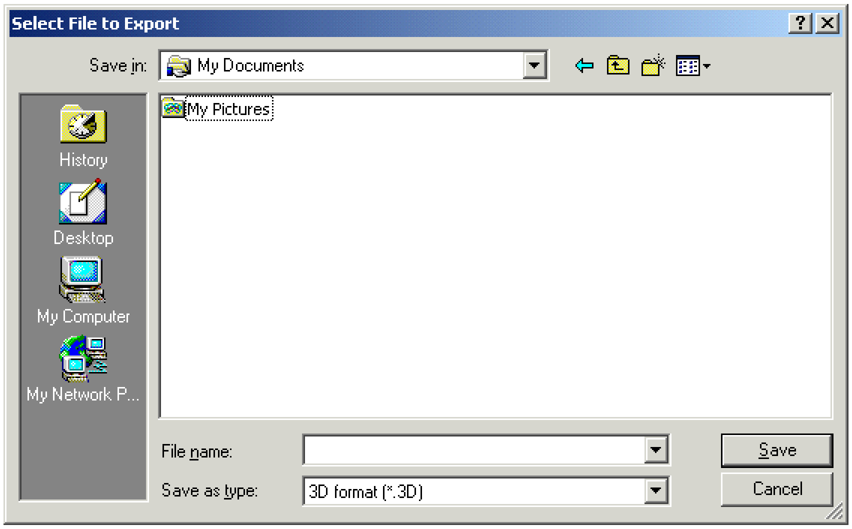
Standard MAX plug-ins	D:\3dsmax4\StdPlugs
Additional MAX plug-ins	D:\3dsmax4\plugins

Figure 16. Panel *Plug Ins* in the *Configure Paths* dialog

4.2 Using the export utility in 3DStudio

There are two methods to export files in *3DStudio* to *VS2000*. One is to use the Export Utility of *3DStudio* and the other is to use the *VS2000* plug-in.

To export a scene created in *3DStudio* with its own Export Utility, choose command **Export ...** from menu **File**. The dialog in Figure 17 will open to select a file for export. At the top portion of the dialog you can specify which location on your hard drive to save the exported file. In the center window of the dialog, choose a folder and enter the file name into field **File name**. Be certain to use file format *3D format *.3D* in the **Save File As** drop-down menu selection. Press **Save** to finish export.



Select File to Export

Save in: My Documents

File name:

Save as type: 3D format (*.3D)

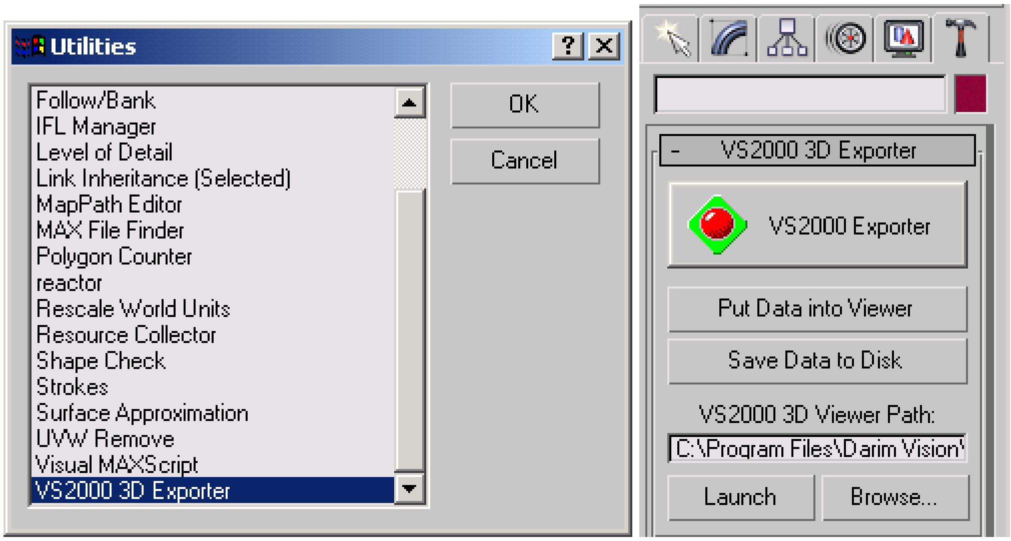
Figure 17. Dialog of files export

4.3 Using the *VS2000 3D Exporter*

By using the *VS2000 3D Exporter* (section 4.3.2) you can have a preliminary look at how the scene will look after it is exported without actually saving it in the *.3D format. This can be done without having to exit the *3D Studio Max* application.

4.3.1 Loading *VS2000 3D Exporter utility*

Open the **Utilities** dialog, scroll through the list until you find and click on *VS2000 3D Exporter* (Figure 18, on the left), then press **OK**. The **Utilities** dialog will open up the *VS2000 3D Exporter* Figure 18, on the right).



Utilities

VS2000 3D Viewer Path: C:\Program Files\Darim Vision\

Figure 18. Dialog for loading additional utilities and *VS2000 3D Exporter*

If the utility is not started after pressing button **Launch**, check up, that the path for *VS2000 3D Exporter* is correct. Press button **Browse ...** and set a correct path to ***HotActions.exe*** in *VS2000 3D Viewer Path*.

4.3.2 Scene viewing by *VS2000 3D Exporter*

To see direct results of last scene changes, press button **Put Data into Viewer**. It will start *VS2000 3D Viewer* with loaded data of the current scene. There are some restrictions that can be displayed in this means of viewing, – the scene should not be empty (Figure 19).

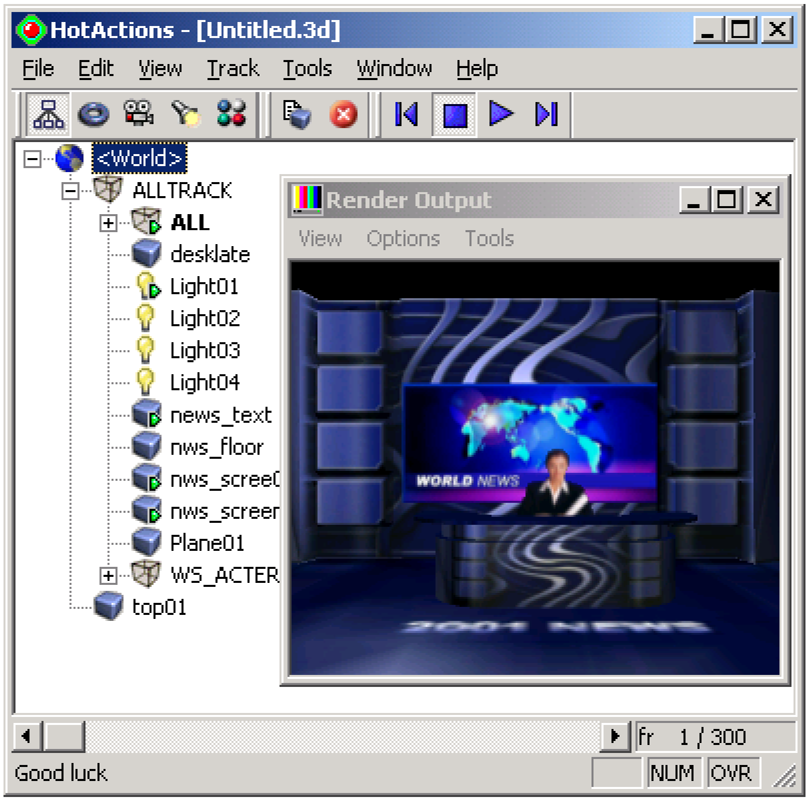


Figure 19. Window of the program after start of *VRSet 3D Viewer*

In *VS2000 3D Viewer* scene components are allocated on types between the appropriate panels. To be switched between these panels it is possible by tools panel .

Buttons ⏮ ■ ▶ ⏭ serve for playback and a stop of the common track of scene. More in detail work with a scene is described in the appropriate chapter of the user's guide.

4.3.3 Scene saving in format *VS2000* by *VS2000 3D Exporter*

For scene saving choose command **Save as...** in menu **File** on the tool panel of application *HotActions* (Figure 19). The field of the dialog will open, allowing determining a directory for saving and a name of a file. If want to save a scene in a file with extension *.3d without preliminary viewing can make it, having pressed on button **Save Data to Disk** on the panel of commands **Utilities** (Figure 18).

5 Appendixes

Index