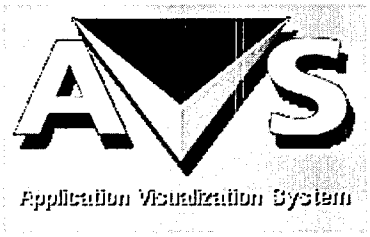
Animating AVS Data Visualizations

First Edition

CONVEX Press
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851
United States of America
(214)497-4000

Animating AVS Data Visualizations



Order No. DSW-306

First Edition
February 1992

CONVEX Press
Richardson, Texas
United States of America

Animating AVS Data Visualizations

Order No DSW-306

Printed in the United States of America

Document No. 710-019530-002

Contents

Figures

Tables

How to use this manual

Purpose and audience

This manual describes how to use the AVS Animation Application and its associated components to create computer based and video based animations of Application Visualization System (AVS) data visualizations.

This document is intended to be used in conjunction with the primary AVS documentation.

Audience

This document is intended for AVS users and programmers who wish to generate animation sequences of AVS data visualizations and record them to disk files for display on workstations and record the image sequences on film or video hardware.

This document assumes that you have a working knowledge of AVS and its subsystems.

Organization

This manual is organized into the following chapters:

- Chapter 1, "Animating AVS"—Animation and data visualization using AVS.
- Chapter 2, "Creating AVS animations"—Getting started, animation methodologies and example procedures.
- Chapter 3, "Using the AVS Animator"—Includes a description of the control panels and provides detailed functional information.
- Chapter 4, "Recording and storing images to disk"—Writing image sequences to disk.
- Chapter 5, "Reading image sequence files from disk"—Tools for reviewing a recorded image sequence.

- Chapter 6, "Recording on video systems"— Preparing image sequences for output to video hardware.
- Appendix A, "Frame sequence file formats"—Detailed description of the file format and compression techniques use in th **write frame seq** module.

Notational conventions

The following conventions are used in this guide:

- **Bold serif font** identifies module names.
- **`Bold constant-width font`** identifies user input in examples and keywords.
- *Italics*
  - Designate user-supplied variables in a command-line example
  - Identify file names and directory names
  - Introduce new and important terms
  - Identify variables in mathematical equations
  - Indicate document titles
- `Constant-width font` designates input and output, including:
  - Command names and options
  - System calls
  - Data structures and types
  - Directives, program statements, display examples, printout examples, and error messages returned
- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipsis shows that lines of code have been left out of an example.
- Words and abbreviations that indicate icon buttons that you click on with the mouse or keyboard keys you press are identified in a distinctive helvetica bold type. For example, **RETURN** refers to the carriage return key. Words separated by a hyphen indicate two keys that you must press simultaneously. For example, **CTRL-X** indicates that you must press and hold down the **CTRL** key and then press the **X** key. Buttons on the screen menu that you use a mouse to click on are alo identified in this manner.

Note

A Note highlights supplemental information.

Associated documents

Using this software may require information not specific to the tasks described in this document.

For more information about the AVS Application Visualization System (AVS), refer to the AVS documentation appropriate to your system.

Acknowledgments

I would like to thank the following people for their contributions to this document:

Technical Contributors: Bob Miller, Ham Lord, Duane Gustavus.

Review Team: Clare Bernier, Linda B. Merims, Dave Kamins.

This document would not have been possible without your help.

Henry C. Smith II
Documentation and Training Development
Convex Computer Corporation

Animating AVS

1

The AVS Animation Application provides a set of interactive tools to help you animate data visualizations generated through the Application Visualization System (AVS).

The AVS Animation Application consists of the following components:

- **AVS Animator** module—The primary interface for:
  - Creating animation scripts
  - Reading animation scripts
  - Editing scripts
  - Generating animation frames
  - Previewing the animation sequences
  - Writing animation scripts, networks and geometry scene information that may be associated with the animation
- **write frame seq** module—Writes a sequence of frames to a file. Image sequences are indexed and optionally compressed for efficient storage.
- **read frame seq** module—Reads sequence files generated through the **write frame seq** module. Frames can be accessed in sequence and played back in any direction.
- **prepare video** module—Performs low-pass filtering, interlacing and gamma correction prior to output to video recording equipment.
- **output video modules**—Write frames to video equipment.

Hardware and software requirements

The AVS Animation Application is intended to be used with Application Visualization System (AVS) software. Therefore, you must have a licensed version of AVS.

You must meet the minimum hardware and operating system requirements specified in the AVS installation procedure.

Packaging will vary from vendor to vendor. The AVS Animation Application may be a separately licensed product, or may be included in the standard AVS software installation.

The AVS Animation Application is installed as a separate module library in */usr/avs/avs_library*. The name of the module library is *Animation*. You can use the module library manipulation facilities provided by AVS to make the Animation Application part of your default environment. Some vendors include these modules as part of the *Supported* module library. Please refer to the software release notes for accurate installation and configuration information.

What is animation?

An animation is a motion picture made up of a sequence of images that contain slight variations that, when displayed, simulate movement through time. Changes in an image that might be tracked include:

- The position of an object (motion dynamics)
- The shape, color, transparency, structure, and texture of an object (update dynamics)
- Changes in lighting
- Camera position, orientation, and focus
- Changes in rendering techniques

Animation techniques bring your data to life, helping you to discover things about data that you could not see from the static data. An animated sequence also provides a convenient mechanism for presenting the results to colleagues. Animations go beyond static charts, words, and pictures to create:

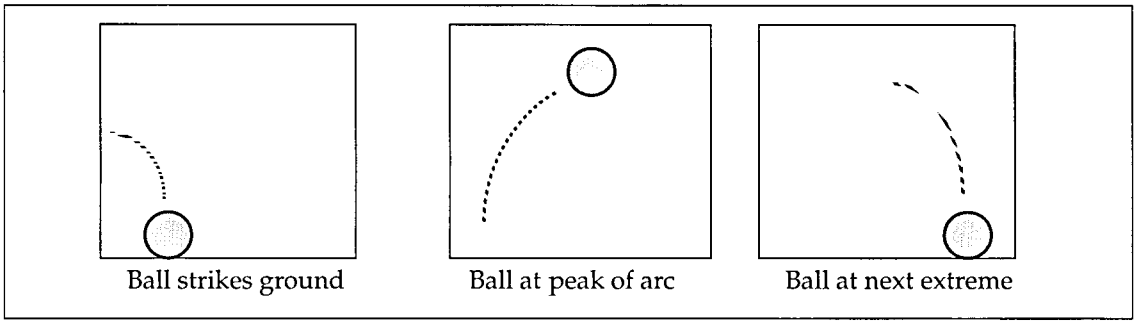
- Video tapes to back up presentations at conferences
- Demonstrations for funding proposals
- Reports to oversight agencies and principal investigators
- Informal exchange of results among colleagues

Using animation techniques to automate the visualization processes, you can create visual experiments that let you concentrate on watching the data on a continuous basis rather than on manipulating the AVS interface.

What is the AVS Animator?

In traditional hand-drawn animation, the creative team is headed by a lead animator. The lead animator develops the characters that will be animated. When actual development begins, the lead animator does not start at the beginning of a scene and work forward, drawing every cell that makes up the animation in sequence (35mm film records at 24 frames per second). Rather, the lead animator breaks down the scene into a series of dynamically or visually significant extreme poses. For example, if the lead animator were drawing a bouncing ball, the extremes would be the ball as it strikes the ground and the ball in the air at the peak of its arc where it changes direction. The lead animator draws only these extremes, as depicted in Figure 1. These extremes are then handed over to an assistant animator who goes through the laborious time-consuming process of drawing all of the in-between cells that connect the extremes to produce the finished sequence.

Figure 1
Poses at extremes



Each of these processes has an analogy in the **AVS Animator** module. You are the lead animator. The characters that you design and draw are the geometries produced by AVS modules in your data visualization network. The stage upon which they will perform is almost always an AVS Geometry Viewer scene window. You use the AVS Animator to define the extremes of an animation sequence. In computer animation technology, these extremes are called *keyframes*. The AVS Animator, acting as the assistant animator, interpolates between the extremes/keyframes that you have defined, automatically generating the many in-between frames needed to produce the smooth, finished animation.

Note

The AVS Animator differs from the AVS action flipbook facilities (found in the Geometry Viewer) in that it automatically generates the in-between frames.

You can manipulate how the observer sees data by modifying the various controls of the Geometry Viewer and by making changes to the module parameters.

Scripting the motion—stage directions

The record of changes that you create with the Animator represents an *animation script*. Much like a script used to direct a play, the animation script contains all of the settings and actions required to produce your animation image output.

Each object that you transform, module parameter value, and environmental properties that change through time are represented by *channels* in the AVS Animator.

Another analogy that might be applied to creating animations is one of laying down tracks on a recording. Each channel is created individually and then mixed together to present the finished piece.

Generating frames—keyframe animation

Keyframe animation is a technique that greatly speeds up the animation process. You only need to create frames at the beginning and end of each segment in your animation, and the system will evaluate the path and generate the in-between frames automatically by interpolating each of the keys you have specified in the keyframes.

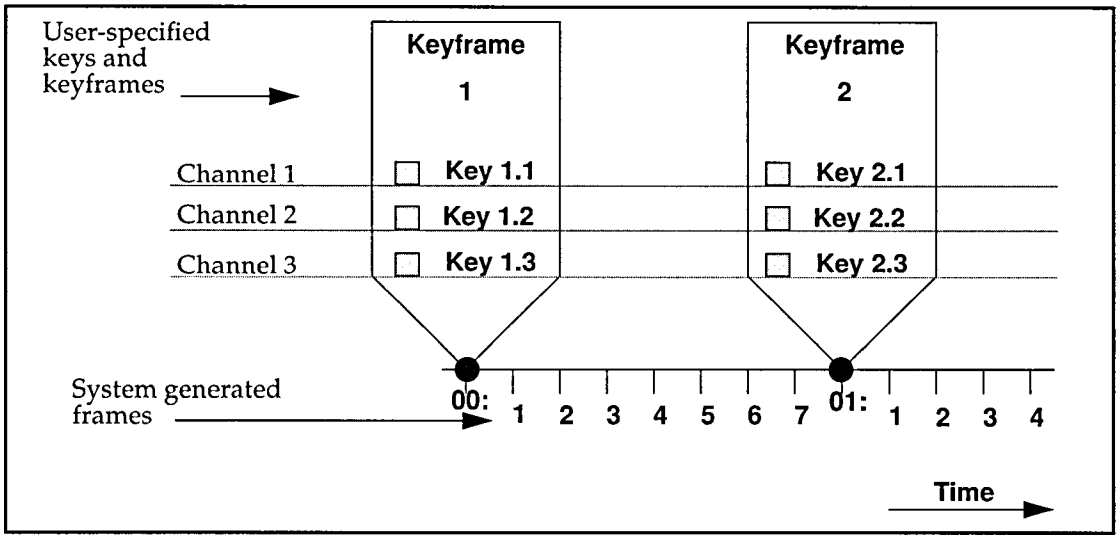
How does the AVS Animator work?

The AVS Animator stores animations as a doubly-linked list of keyframes. Each keyframe contains a time stamp of the form:

```
minutes:seconds:frame
```

The time stamp represents where the keyframe occurs in the absolute video or film running time of the animation. The keyframes are stored in the list in time-order. Each keyframe contains a list of keys, as illustrated in Figure 2.

Figure 2
Keyframes and system generated frames



A *keyframe* is composed of a set of *keys* that contains specific values and property settings that represent the channels. The values for each of these active keys are interpolated over time from one keyframe to another. The rate of change is, in part, determined by the number of frames between keyframes.

Within the AVS Animator, a keyframe provides a snapshot of an AVS network at a user-specified point in time.

Within the AVS Animator, *keys* include the following:

- Parameters for modules in your network
- Objects that make up the scene, their properties and transformations
- Geometry Viewer scene information, including:
  - Lights, settings, color, number and transformations
  - Cameras, their settings and transformations (perspective, background color, orientation, and number)

When the AVS Animator starts, it records all possible keys, the initial state of all module parameters, and all Geometry Viewer interface values. Subsequent keyframes contain only those keys that have changed from the initial state.

Playback

The AVS Animator lets you preview your animation by playing back defined keyframe sequences with the number of in-between frames that you specify.

When you play back an animation, the AVS Animator runs down its list of keyframes generating in-between frames by interpolating key values in adjacent keyframes.

The number of interpolation steps that are generated depends on the value set for frames/second in the AVS Animator control panel.

The controls are similar to viewing a video tape through a VCR. The sequence can be played forward or backward, continuously or in single steps, once through, or in a repetitive cycle.

Editing

Use the **AVS Animator** module to perform interactive editing. You can delete, move, and add complete keyframes or edit the specific values of individual keys to achieve precise effects.

Two menu interfaces are provided, a Compact Animator and Full Animator menu. The Compact Animator menu provides quick access to the controls needed for setting keyframes. The Full Animator provides more detailed controls for editing and finishing existing scripts. Refer to "Control panels," on page 30, for more information.

Storage formats

Animations can be stored as scripts, or as a sequence of finished frames.

An animation script generated by the AVS Animator is stored as a set of files that may include:

- Keyframe settings file (`.anim`)—ASCII file that contains a description of each keyframe and its associated keys.
- Geometry Viewer settings file (`.scene`)—ASCII file that contains a description of the Geometry Viewer settings.
- Network description file (`.net`)—ASCII file that contains a description of the modules that make up the network used in your data visualization.

Script files

Script files are created with the Animator's **Save Script** button. You can read these animation scripts back into the Animator and regenerate an animation sequence. Script files are used when you are creating and editing an animation, before you commit the animation to final form. Script files are also a good transport and archive medium because they represent the instructions for producing the images and not the images themselves.

Note

Animation scripts are much smaller than CLI scripts. Use the following command line to read the CLI script:

```
cli -play filename
```

Frame image sequences

Animations can also be stored as frames in *.seq* sequence files with the **write frame seq** module, described in Chapter 4. Sequence files contain the actual frames that make up an animation. The frames are numbered sequentially, but are randomly retrievable with the **read frame seq** module, described in Chapter 5. To save space, the frames in a sequence file are usually compressed. You create sequence files of *animation segments* prior to final processing.

AVS Command Language Interpreter scripts

You can also save the current state of an animation development session as a CLI script. This option might be used when you need to make a minor correction to the settings in an animation sequence without re-assembling the entire segment from scratch. Refer to "Using CLI scripts," on page 47 and the CLI chapter in your AVS documentation for more information on using them.

Note

The AVS Animator can generate CLI scripts, but cannot read them.

Rendering techniques

Many approaches can be taken to rendering a scene and storing the resulting image. These approaches use different rendering techniques that produce varying levels of image quality.

Rendering features and methods may depend on the hardware platform that you are running AVS on and the AVS Animation Application tools. Refer to the AVS documentation and your software installation and release notes for the methods used within your system.

Output to video

There are several considerations for outputting color graphic images to video. First of all, computer graphics have a much higher resolution than video.

Broadcast video standards are low resolution when compared to most workstation color graphics displays. They use interlaced video, and have insufficient bandwidth to permit abrupt horizontal color changes. A *low pass filter* helps to resolve both problems. A low pass filter smears images horizontally, to eliminate abrupt transitions, and it smears images vertically so that features don't disappear between the scan lines when the picture is interlaced. The **prepare video** module automatically performs these actions, as described in "Preparing computer graphics for output to video," on page 62.

The AVS Animation Application includes an example output video module written for the Diaquest ImageNode. This module that can be used to send images to video hardware using this controller. Source code for this module is included, so that you can write your own custom output video module for other types of controllers. The output video module lets you control the insertion point on the tape and the edit length of each frame. Refer to "Video output modules," on page 64, for more information.

Creating AVS Animations 2

In this chapter, we will lead you through the process of creating your own animation using the components of the AVS Animation Application.

There are several steps in the production of an animation. The number and order of these steps may vary depending on the complexity of your animation.

- You must first generate your data visualization.
- **Plan the action**—Decide what you want to show and how best to present the data visualization.
- **Create a test script**—Run through the presentation as a storyboard, then create the rough sequences.
- **Revise and save your script**—Add, modify, and save the changes for final generation.
- **Preview the action**—Use the AVS Animator to display the action to your workstation.
- **Frame rendering**—Choose the method most appropriate to your needs.
- **Frame recording**—Store images on disk, film, and recording on video.

Planning the action

Before creating an animation, you should consider some of the aspects of designing animations.

There are three basic actions:

- Transforming objects
- Changing module parameter values
- Changing environmental properties, such as lighting, camera position and colors

Think of ways to stage the action. Choose a view that conveys the most information about the events taking place in the animation, and try to isolate events so that only one thing at a time occupies the viewer's attention.

In the case where the events being simulated are happening simultaneously, it may still be possible to view the scene from a position in which the different events occupy different portions of the image, and can be watched independently without visual clutter from the others.

Hint

Set up one variable at a time for change, such as object movement. Once you have mastered the change and are satisfied, go on to the next variable and then add it to the previous one.

Creating animation is as much an art as it is a science. It is an evolutionary process that starts with a rough idea that is executed and then refined through as a series of iterative steps. It is up to the user/artist to decide when the piece is finished.

Note

You may wish to refer to Chapter 3 for detailed information about the controls used in the AVS Animator module as you work through the next sections.

Create a storyboard

An animation feature movie actually consists of a series of short recorded sequences. You should not attempt to lay out the entire movie as a single animation sequence.

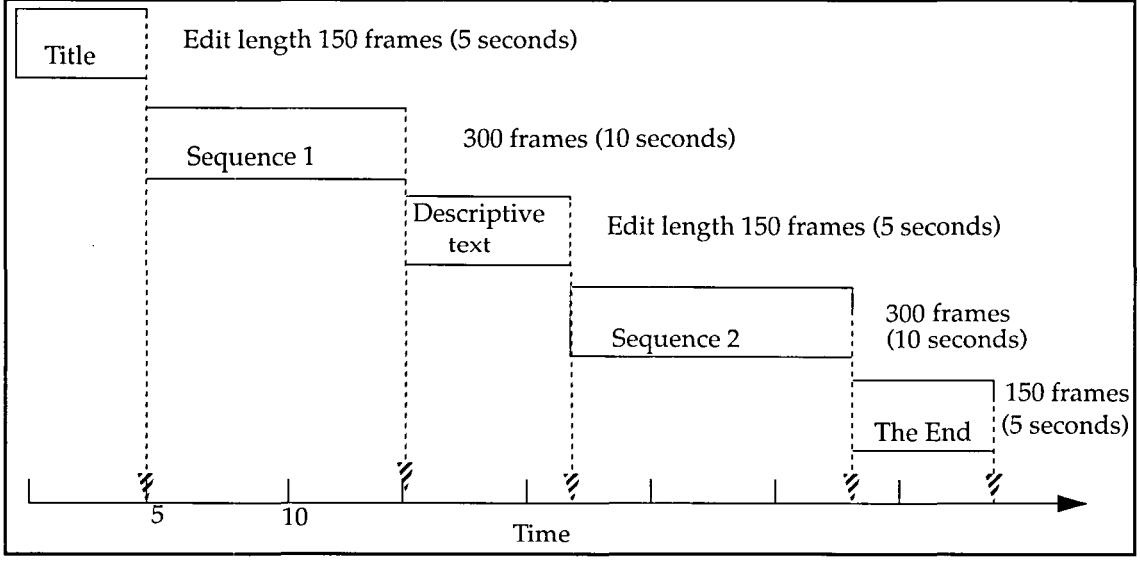
When creating a storyboard for the animation feature, plan out short, manageable segments that can be edited together to make up the final full length feature. The final editing may not occur until you are at the recording stage.

In most cases, you probably already have some idea about how you want to present your data visualization. But it is still helpful to sit down first and work out a rough draft of the actions and the order of events, as illustrated in Figure 3.

- Decide which actions (channels) you actually want to track.
- Plot changes in each channel through time, on graph paper.
- Visualize mentally what a person would be seeing, and time it with a stop watch. You might also try acting out object and camera motions and time your movements with a watch (professional animators do this all the time).
- If the production will have sound, or if you will be narrating it live, a different pacing that allows time for the narration is needed.
- An instructional and illustrative animation must be slow enough for the viewing audience to grasp what they are seeing. Sequences might be repeated with slight variations, moving faster as the audience gains understanding.
- Use anticipation. Focus the audience attention on what they are about to see by using probe pointers or labels that appear just prior to moving the object being animated.
- Copy animation techniques other people have used that you like.
- The human mind is capable of following about six changing aspects at any one time. Keep the animation as simple as possible to convey the desired effect.

For many users, single segments representing one view of the data visualization are all that is needed. For some projects, more involved animations may be required.

Figure 3
Lay out the story



Title frames before the sequence may be desired as well as the insertion of graphs and other illustrative and textual material to round out and complete the animation. You can generate text and graphics frames using your favorite text/graphics editor and then capture the results into a bitmap (using a capture utility like the X system *xwd(1)* screen capture utility) that can be converted into AVS image format. You can also use the labeling features of the Geometry Viewer to add text information to your images. These frames can then be read and sent directly to an image sequence file or to an output module for recording.

With the AVS Animation Application, it is easy to create *test scripts* for your animation before committing them to tape or film. You can choreograph and preview the motion of objects, lights, and cameras and the varying of module parameters.

Once you are happy with how a particular scene plays back, the image can be sent to a renderer and output to an image sequence file, film recorder, or video recording system.

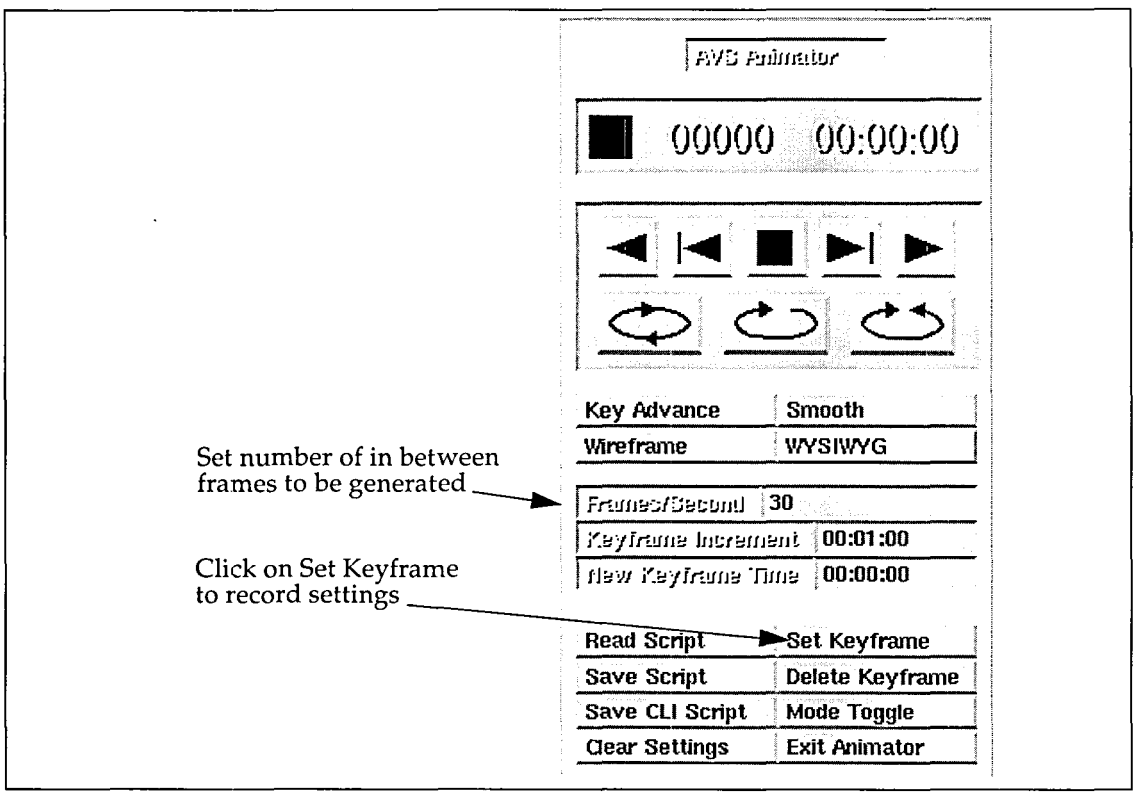
The next section describes methods for creating these test scripts.

Creating a test script

The following steps provide a rough outline of the actions needed to create an animation script by using the AVS Animator module. Following this outline is an example procedure:

1. **Scene setup**—Create the visualization network. If needed, perform any initial setup required within your network. At this time, you might want to lay out the order and number of individual sequences that you want to capture.
2. **Initialize** the AVS Animator module *after performing your setup,* by dragging it into your workspace from within the Network Editor. The Compact Animator controls appear as shown in Figure 4.

Figure 4
Compact Animator control panel



Chapter 3, "Using the AVS Animator," on page 29, describes the buttons, type-ins, and indicators for this panel in more detail.

3. **Timing**—Set the number of frames per second to an appropriate value. The default setting is 30 frames per second. This setting depends on whether you plan to output this sequence to video, film, or to a disk file for viewing on a workstation.

   - If you are creating animations intended for NTSC output devices, set the frame rate to 30 frames per second with interlacing off. Set the frames per second to 60 frames/second if you intend to output interlaced frames.
   - If you are outputting to the PAL European standard, set the frame rate to 25 frames per second or 50 frames per second, interlaced.
   - If you are sending the output to a film recorder, you should set the frame rate to 24 frames per second.
   - If you are using a workstation, lower frame rates can be used to preview the script more quickly.

When creating a script, frame rate is an arbitrary setting, because this is not real-time output. Refer to "Preparing computer graphics for output to video," on page 62, for more information about preparing output for video recording.

Note

The frames-per-second setting only affects playback, and can be changed at any time.

4. Capture the first keyframe settings by clicking on the **Set Keyframe** button.
5. Set up the next object positions and parameters for the next keyframe in the sequence.
6. Click on the **Set Keyframe** button to capture the next keyframe in the sequence.
7. Repeat steps 5 and 6 until you have completed a rough sequence.
8. If you wish, save the resulting script by clicking on the **Save Script** button.

Once you have created your script, you can then proceed to the next steps of playing back, editing and saving the scripts.

The next section contains a step-by-step example procedure that takes you through the process of creating an animation.

Animating objects—an example

Let's start out with a simple example that might help illustrate the process. In this example, we assume that AVS is already running on your workstation. We will use the Geometry Viewer subsystem to create a display of a geometric object. We will then position the object, and record the transformation information into a script using the **AVS Animator** module and finally, run the script, allowing the system to generate the in-between positions.

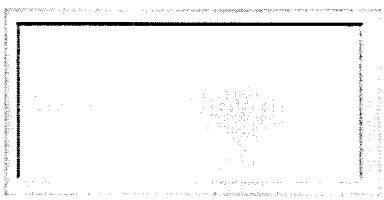
Note

All discussions and examples in this book assume that you are familiar with AVS and its subsystems. There may be more than one method for arriving at a particular step. Refer to your AVS documentation for more information.

Following this section are more detailed descriptions of the processes associated with the basic steps.

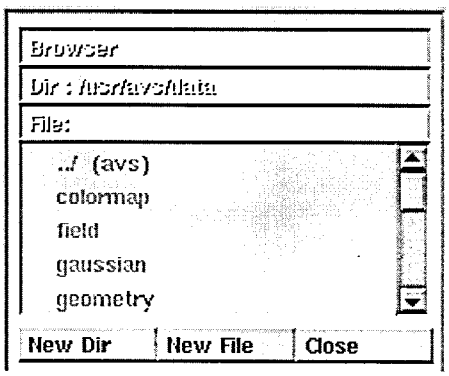
Scene setup

1. Click on **Geometry Viewer** from the Main menu.



The Geometry Viewer control panel appears.

2. Click on the **Read Object** button. The file browser is displayed.



For this example, we are using the default directories, */usr/avs/data*, and data files supplied with AVS.

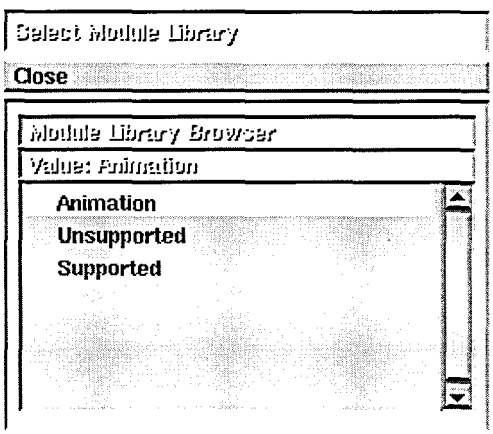
3. Click on the *geometry* directory, and then click on *teapot.geom* to display the teapot geometric object. Click on the **Close** button of the browser after making your selection.

4. Use the mouse to position and scale the teapot so that it resides within one quadrant of the display window. This represents the initial setting for your geometric object. Details on how to perform these actions are provided in the AVS user's guide provided with your version of AVS in the "Geometry Viewer" chapter.

Initialize the Animator

5. You are now ready to initialize the AVS Animator. Close the Geometry Viewer control panel and click on the **Network Editor** button. The Network Editor palette, workspace, and control panel appear.

6. Locate the **AVS Animator** module. This module may be part of your standard module palette, or may need to be accessed through one of the application module libraries. If it is not part of your standard module library, click on the **Module Tools** button in the Network Editor control panel and then click on the **Select Module Library** button to bring up the Select Module Library browser.



7. Click on the **Animation** library to read the AVS Animation Application modules into the palette. The **AVS Animator** module is located in the Data Input column of the module palette.

Note

If you do not have the Animation module library, you may need to read the library into your system through the Read Module Library option.

8. Click on and drag the AVS Animator module into the workspace. The Compact Animator control panel is displayed, as shown in Figure 4.
9. Click on the **Close** button to close the Network Editor Display.
10. Click on and hold the **Data Viewers** button at the top of the control panel and then drag the mouse until the Geometry Viewer selection is highlighted, then release the mouse button. The Geometry Viewer control panel appears.

Establish timing

11. Move your mouse cursor to the **Frames/Second** type-in in the Compact Animator menu, and change the value to 5. You must backspace over the default value. This setting can be changed at any time. For now, we are creating a rough sample run, so a low frame generation rate is sufficient.

Record the settings

12. Click on the **Set Keyframe** button located in the second column of the lower button group in the AVS Animator control panel. This action sets the initial keyframe (00:00:00). Notice that the New Keyframe time changes to 00:01:00, the *next* available keyframe.
13. Move the teapot to the opposite quadrant of the display window.
14. Click on the **Set Keyframe** button again. The New Keyframe Time is now at 00:02:00, and the frames counter shows that there are now 5 frames set for this sequence. You have created your first animation sequence, extending from time 00:00:00 to 00:01:00.
15. To preview the sequence, click on the **Play Once** button.

You should see the teapot jump back to the original position and then proceed along a course in five steps to the second keyframe position, then stop.

16. Set the **Frames/Second** value to a larger number, then click on one of the cycle buttons again. Notice that the number of in-between frames changes according to this value.
17. Move the teapot to another position. This time, try scaling and rotating the teapot in the new position.

18. Select a new keyframe position by entering **00 : 02 : 00** in the **New Keyframe Time** type-in. A dialog box appears to confirm your new position.

Note

If you attempt to set a new keyframe in a position already occupied by one, a warning message appears to confirm that you wish to overwrite the existing one.

19. Click on the **Set Keyframe** button to create a third keyframe.

20. Click on the **Play Once** button and watch what happens.

You have now generated the first pass of the animation script. The next step is to edit and save it.

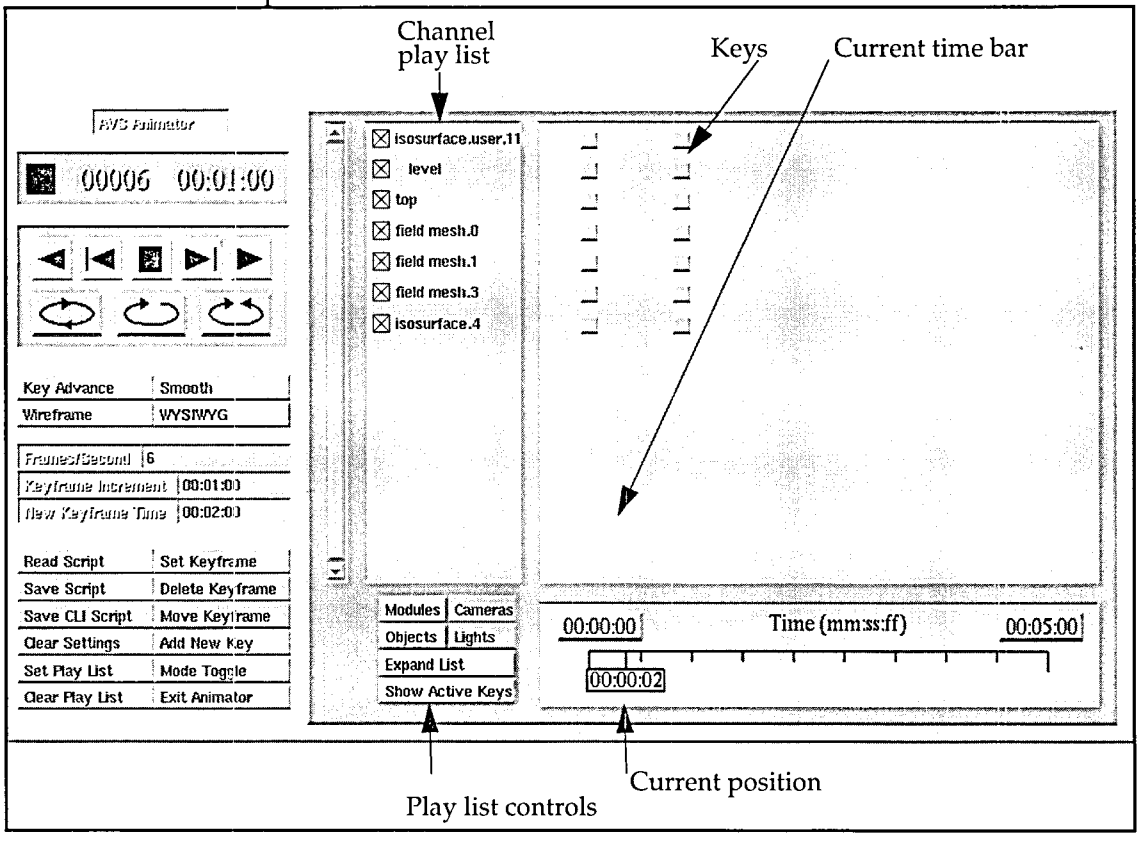
Note

You should save your work often, at each intermediate step, so that valuable work is not lost due to system failure.

Revising the script

Once you have created an initial sequence of keyframes, you may find that you need to modify the action or order of events. This is done by adding new keyframes, moving, or deleting keyframes to vary the behavior of the animation script. To gain access to the full editing capabilities of the AVS Animator, click on the **Mode Toggle** button to bring up the Full Animator control panel, as shown in Figure 5.

Figure 5
Full Animator control panel



Changing your current position

You could use the **New Keyframe Time** type-in to specify a new keyframe position to edit. This same action can also be performed from the Full Animator control panel by clicking on and dragging the current position box across the time range slider, described in "Time slider control and editor," on page 41.

The time setting specifies a position in the script, identified in minutes and seconds. The frame count specifies an in-between frame after the second specified by the time setting. Refer to "Working with time," on page 43.

For example, specifying 00:04:01 sets a new keyframe position at the first frame after the 4th second in the script sequence.

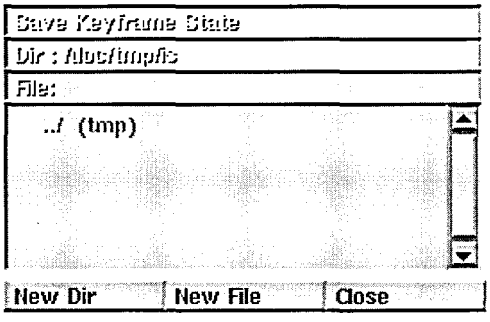
Saving animations as scripts

Animations can be stored as scripts or as a sequence of frames.

The animation script contains all of the settings and actions required to produce your animation image output. You can save this information by using the **Save Script** button on the AVS Animator.

For example, suppose that you have generated an animation segment and want to save the settings:

1. Click on the **Save Script** button on the AVS Animator. The Save Keyframe State browser appears.
2. Click on an existing animation script to overwrite or click on the **New File** button to enter a new file name. The *.anim* extension is added by the system.



In a similar fashion, you can read in a previously saved animation script with the **Read Script** function.

The system reads in the setting information and also reads in any *.scene* files, networks (*.net* files) and geometry (*.geom* files) that are associated with the animation script.

Saving the segments of your animation in script format is very efficient, since only the instructions needed to generate the animation are saved. You can also save the finished images as frame sequences in a file using the **write frame seq** module, Refer to Chapter 4, "Recording and storing images to disk," for more information about this method.

Previewing the animation

You can preview the animation, once the objects and networks are selected, the keyframes edited, and the interpolation method is chosen.

Setting up a simple animation where only a few variables change is a fairly straightforward task. However, attempting to control many variables and many keyframes can be a very confusing and complex task. A previewing system becomes an essential tool. There are two types of previewing capabilities provided in the AVS Animation Application:

- **Preview the animation interactively**—Used in building the animation.
- **Preview the rendered frames**—Render and store each image on disk using the **write frame seq** module and then play back the recorded sequence using the **read frame seq** module. This method is used at the end of the animation process as a final verification step.

The second process of saving to disk is discussed in more detail in later chapters. In this section, we will discuss the interactive previewing capabilities of the AVS Animator module.

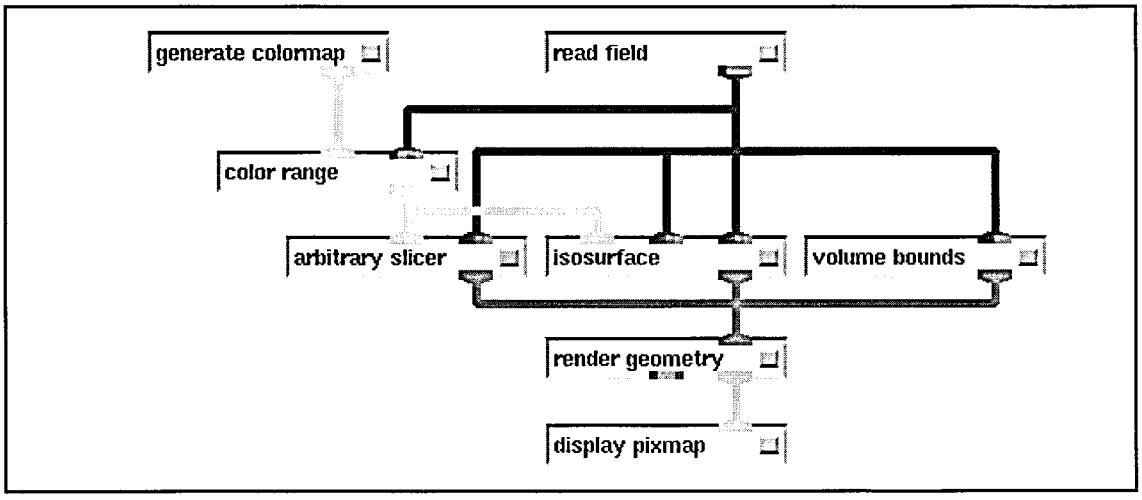
Within the AVS Animator, you control what parts of the animation are being played back. If only the motion of one object through a scene is being choreographed, you can turn off the updating of all other parameters through the Full Animator. Other options allow you to view the animation at any point in time. You can also view the action in either the forward or reverse direction.

A more advanced example

This example uses the AVS Animator to produce an eight second scene showing the point where the pressure peaks in the airflow over a blunt fin. The scene will first rotate and then zoom in to the peak pressure point.

1. Start AVS and create the network shown in Figure 6.

Figure 6
Bluntfin data visualization network



An example network is provided in */usr/avs/networks/examples/bluntfin1.net*. It contains an **isosurface, arbitrary slicer,** and a **volume bounds** module.

2. Bring the **read field** module's control panel up and read in the file */usr/avs/data/field/curv.fld*.
3. Instantiate the **AVS Animator** module. The Compact Animator control panel appears, shown in Figure 4.
4. Set the Animator's **Keyframe Increment** to **00 : 04 : 00**.
5. Click on the **Mode Toggle** button to display the Full Animator control panel, shown in Figure 5.

Note

If you are reading in an existing network, steps 6 and 7 are not needed.

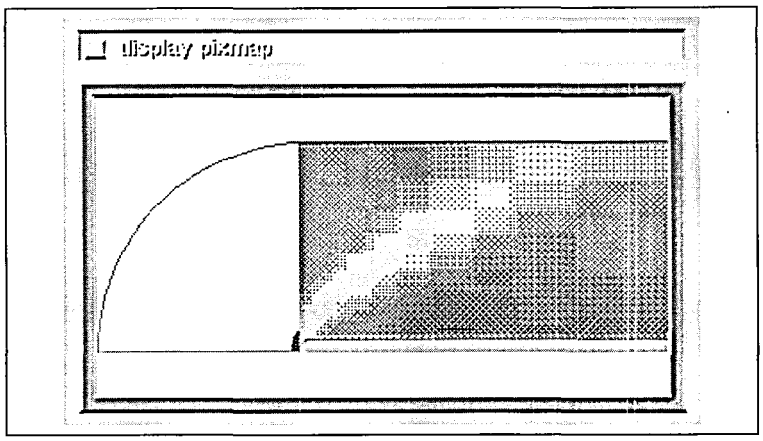
6. Bring up the **isosurface** module's control panel. Set the isosurface level to 3.

Note

For the rest of this example, it may be useful to use the Dial editor for the isosurface level control widget, moving it to a convenient portion of the screen.

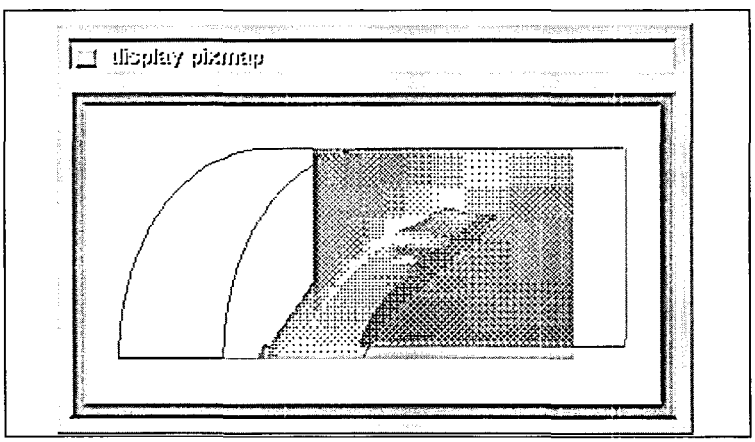
7. Bring the Geometry Viewer's panel up (use the **Data Viewers** pop-up). Verify that the **top** object is selected. Click on the **Reset** button, then the **Normalize** button. The resulting display should appear similar to the display in Figure 7.

Figure 7
Bluntfin, normalized view



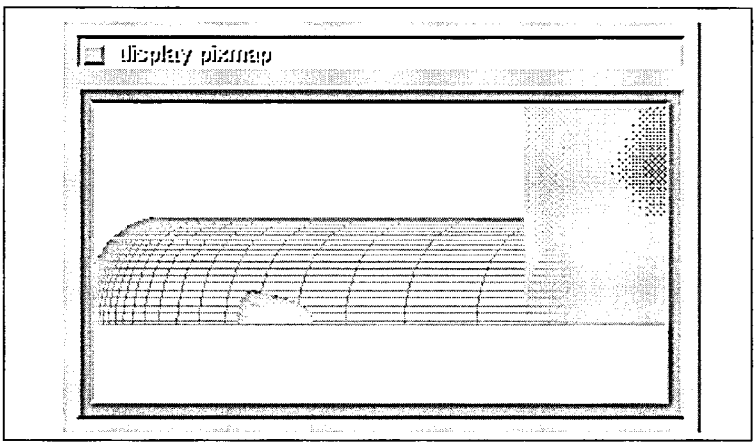
8. Click on the **Set Keyframe** button on the AVS Animator's control panel. This records the first keyframe of the animation.
9. Close the Geometry Viewer window. Adjust the isosurface level to 1.62. The isosurface should now reach nearly to the top of the volume.
10. Rotate the top object 45 degrees to the right. You can use the arrow keys to perform this action.
11. Enable the **volume bounds** module's **Min J** toggle. The display should now look like Figure 8.

Figure 8
Bluntfin, rotated 45 degrees at isosurface level 1.62



12. Use the **Set Keyframe** button to set the second keyframe. Notice that the new keyframe is at time `00:04:00`.
13. Adjust the isosurface level to 4.7. The isosurface will become a tiny orange spot at the leading edge of the fin.
14. Zoom in on the animation until the tiny spot is clearly visible. You'll see something similar to Figure 9.

Figure 9
Bluntfin, zoomed to isosurface 4.7



15. Use the **Set Keyframe** button to set the third keyframe.

The first pass of the animation script is ready to be previewed. The next step is optional.

16. Click on the **Save Script** button to save it to files. The Save Keyframe State file browser will pop up. We'll save the script in */tmp/example.anim* and */tmp/example.net*.
    - Click on the browser's **New Dir** button. A type-in will pop up.
    - Enter `/tmp` in the type-in and click on **OK**.
    - Click on the browser's **New File** button. Another type-in will pop up.
    - Enter `example` and click on **OK**. The browser disappears. ConvexAVS automatically adds the *.net* and *.anim* extensions to the files.

Preview the results

Now test-play the animation:

1. Press the **Play Once** button on the Animator's control panel. The animation begins playing. You can see that it takes quite a while for the scene to finish.
2. Press the **Stop** button when you are ready to continue.

Speed up the play back

You can speed up the playback by reducing the frame rate.

1. Set the **Frames/Second** type-in on the AVS Animator control panel to 2.
2. Press **Play Once** again. The animation should play more quickly by only producing two frames for every second. Click on the **Stop** button to halt the preview.
3. You can speed it up further by looking only at keyframes.
4. Click on the **Key Advance** toggle to enable keyframe previewing
5. Press **Play Once**. Now the animator only produces the three keyframes that you set.

However, viewing three disjointed frames does not give a good feel for the movement. Because AVS is spending much of its time recalculating isosurfaces, we need to temporarily stop AVS from performing this calculation by turning off the channel that tracks it.

The word isosurface appears in two places in the Animator's channel play list. The first one, **isosurface***N*, refers to the module, and the second, **isosurface.***N* (notice the dot), refers to the geometric object that module produces. The number will vary depending on the order in which the objects are read in

6. Before continuing, disable the **Key Advance** toggle to disable keyframe preview mode.
7. Click the check box to the left of the module name (the upper one).
8. Press **Play Once** again. The isosurface level does not change, and frames are produced more quickly.

Now draw the isosurface and the slice plane in wireframe to speed the animation up.

9. Click the check box for the isosurface channel again to enable isosurface tracking.
10. Enable the **Wireframe** toggle. You may need to enable the **lines** button in the Geometry Viewer control panel to view an isosurface as a wireframe.
11. Press the **Play Once** icon. All objects in the scene are drawn in wireframe.

On some systems, wireframe drawings are faster than polygons. On others, they are about the same speed. You may or may not see the animation speed up.

Another method for speeding up the process might be to use only the bounding box to establish the motion pattern. It is generally a good practice to get the motion of the cameras and objects established first and then add in the visualization techniques.

You have now completed the animation sequence. From this point, you can convert the output to images that can be written to a disk file and output to a recording device.

Using the AVS Animator

3

The **AVS Animator** module provides the keyframe editor and controls needed to create animation scripts, edit them, and generate animation frames. This module, along with the other AVS Animation Application modules is located in the *Animation* module library. Refer to your installation and software release notes for the exact configuration and location of the modules. You can access this library by selecting the **Read Module Library** button from the AVS Network Editor control panel. This module, shown in Figure 10, is initialized when it is dragged from the module palette in the Network Editor into the workspace.

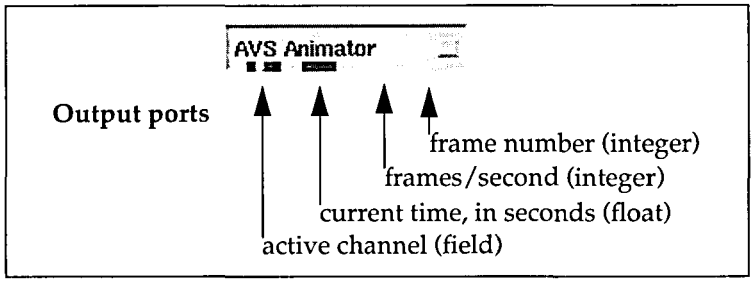


Figure 10
AVS Animator module icon

The **AVS Animator** module has the following output ports:

- **Frame number** (integer)—Outputs the frame number. Use this port as a synchronizing input for coroutine modules.
- **Frames/second** (integer)—Outputs the frames per second.
- **Current time** (real)—Outputs the current time position.
- **Active Channel** (field 2-D scalar real uniform)—Outputs the values for the currently active channel. A valid channel must be selected before a value is output. You must click on the name of the desired channel in the play list to highlight it. If no channel is selected, then there is no output. The channel selected cannot be a module name. Valid channels output integer, floating point (real) or character values.

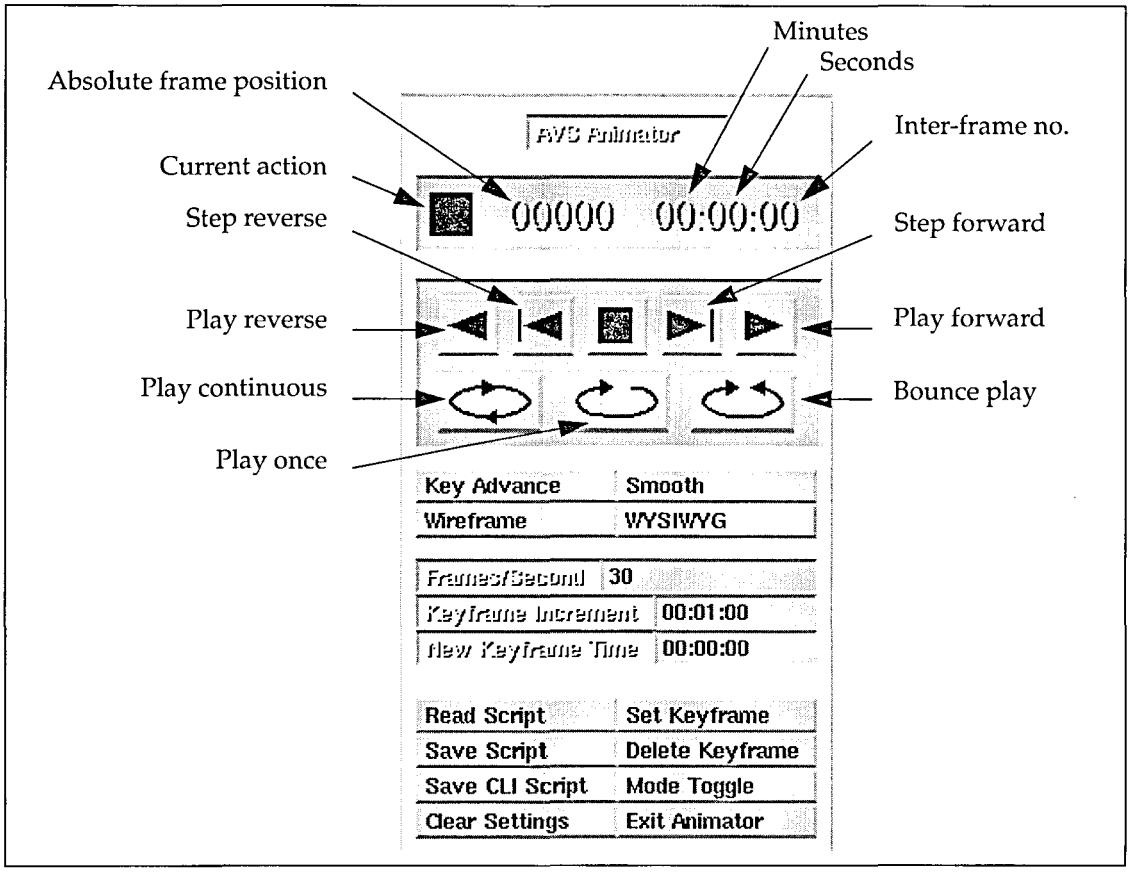
This output can be connected to the **graph viewer** module's Linear input port to view a single channel's value as a function of time.

For most operations, no explicit connections to a network are needed.

Control panels

The AVS Animator interface includes two control panels: a *Compact Animator* and a *Full Animator.* The Compact Animator, shown in Figure 11, provides quick access to the controls needed for setting keyframes. The Full Animator, shown in Figure 13, provides more detailed controls for editing and finishing existing scripts. Both control panels provide playback buttons similar to a video cassette recorder (VCR). These buttons allow you to control the playback of the animation scripts.

Figure 11
Compact Animator control panel



The Full Animator control panel, shown in Figure 13, appears when you click on the **Mode Toggle** button.

Status indicators

For both the Compact Animator and the Full Animator, the top of the control panel displays the current action icon, absolute frame position, and the current time position in minutes, seconds, and inter-frame number, shown in Figure 12.

The *frame rate* (frames per second), in conjunction with the *keyframe increment*, defines the number of frames that will be generated between each pair of keyframes.

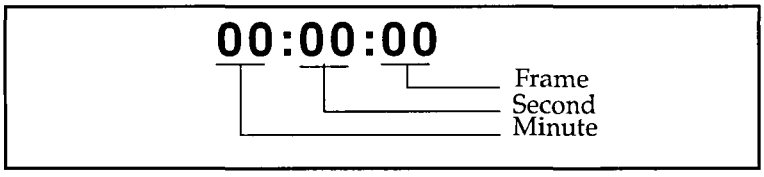
The *time setting* used in the AVS Animator identifies a specific position in the script, called out in minutes and seconds.

The *inter-frame number* specifies an in-between frame after the second specified by the time setting. Figure 12 shows the time setting.

Frame zero (00:00:00) is always a keyframe. If you set a frames per second increment of 30, then frame 29 (00:00:29) specifies the frame just prior to the next second (00:01:00).

If you specify a new keyframe time of 00:04:01, you will set a new position at the first frame after the 4th second in the script sequence.

Figure 12
Time setting



This time setting specification is normally used by video tape recording devices.

The following sections describe the control buttons on the AVS Animator menus.

Playback controls

The following buttons control the script playback:

Play reverse

Generate frames in descending order starting with the frame prior to the current frame. The sequence stops when the beginning of the current script is reached (frame zero). If the current frame is zero, the sequence starts with the last frame.

Step reverse

Generate the frame previous to the current frame. If the current frame is zero, the action is ignored.

Stop

Stops any action that is in progress.

Note

There may be some delay before the playback stops depending on the state of the network or geometry being animated.

Step forward

This button generates the frame after the current frame. If the current frame is the last frame, frame zero is generated.

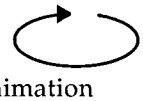
Play forward

Generate frames, one frame at a time, starting at the next frame after the current frame and stopping when the last active keyframe is reached. If the current frame is the last frame, the sequence starts at frame zero.

Play continuously (loop)

Run in a continuous loop, starting at the current frame, until the **Stop** button is pressed.

Play once



Cycle once from the 1st frame to last frame in the animation script.

Bounce play

Cycle back and forth continuously through the animation script until the **Stop** button is pressed.

Interpolation controls

The following buttons control the way keyframes behave during playback:

Key Advance

This toggle enables and disables keyframe interpolation. If it is set, only keyframe positions are displayed when you select one of the playback controls. The script jumps from keyframe to keyframe.

Smooth

This toggle enables and disables sinusoidal interpolation when new keyframes are added. When smooth is enabled, frame acceleration is interpolated along a sine curve as the in-between frames are generated. Object rotation is not affected by this button. Object rotation is always quaternion interpolated. Refer to "Controlling the way keyframes are recorded," on page 46, for more information.

Wireframe

The **Wireframe** button displays the geometric objects in your animation as wireframes, overriding rendering modes that may be set through the Geometry Viewer for the current playback. This function is the same as the **Lines** button in the Geometry Viewer. There must be at least one active geometric object in the play list to use this option.

In the early stages of your design, you might wish to render the geometric objects as wireframes to reduce the computational time. You might also use this option to preview the script action of an animation.

Notes

Whether or not an object has a lines (wireframes) representation depends on the behavior of the standard AVS module or how the developer coded the routine that generates the object. Some objects may become invisible or otherwise unusable.

You may need to select "Lines" from the Geometry Viewer control panel to generate the wireframe for an isosurface.

Frames/Second

Determines the number of in-between frames to generate between each second. There is a range of 1 to 1000 frames. The default is 30 frames. When you are first setting up the animation, a low setting (for example, 5) helps you preview the sequence.

Refer to Chapter 6, the section, "Preparing computer graphics for output to video," on page 62, for details about the requirements for interlacing frames for video output. Table 1 provides format guidelines.

Table 1
Format and frames/second settings

Format	frames/second non-interlaced	frames/second interlaced
NTSC	30	60
PAL	25	50
Film	24	n/a*

* Film does not have scan lines.

Recording Controls

The following controls affect the way new keyframes are generated.

Clear Settings

Note

Use the Clear Settings button with caution.

The **Clear Settings** button removes all key states from the current script. You are asked to confirm that you wish to clear all Animator settings.

The **Clear settings** button wipes out all of the saved transformation settings in the animation (sets them to their default values), including the keyframes.

Delete Keyframe

It may be desirable to unmark certain keyframes so that they are no longer thought of as keyframes for interpolation. Use the **Delete Keyframe** button to unmark the current frame. An error message is displayed if you attempt to delete the first keyframe or if you specify a time that does not contain a keyframe.

Keyframe Increment

The **Keyframe Increment** setting determines the time increment by which the animator advances the keyframes. For example, if you set an increment of 00:02:00, each new keyframe you set will be two seconds after the last one. When the script is run, twice the number of frames are created between each keyframe.

If your frames/second is set at 6, then 12 frames are created between each keyframe pair.

New Keyframe Time

Use this type-in to specify a new keyframe position.

Enter the new time to move the keyframe to. Use the format mm:ss:ff (minutes, seconds, frame). If you select a time beyond the last frame in the script, the last frame is used.

The time setting specifies a position in the script, identified as minutes and seconds. The frame count specifies an in-between frame after the second specified by the time setting. For example, if frame zero (00:00) identifies the keyframe, and the frames per second increment is set to 30, then frame 29 (00:00:29) specifies the in-between frame just prior to the next second (00:01:00). If you specify 00:04:01, you will set a new keyframe position at the first frame after the 4th second in the script sequence.

The type-in is automatically updated to show the next available keyframe time.

Set Keyframe

Use this button to set a new keyframe at the current time position. If a keyframe already exists at the current time position, you are asked to verify that you wish to overwrite it.

When you select this button, the system:

- Checks the state of all objects, lights, and cameras
- Checks the state of all module parameters to see which ones have been updated
- Makes changes to the keyframe and adds it to the list

Select this button after each change to the scene to set the keyframe. All active keys are included in the new keyframe. This action overrides any existing keyframe settings.

Note

The last keyframe represents the last frame of the animation.

WYSIWYG

This toggle button controls the way in which new keys are added when a new keyframe is set. If the **WYSIWYG** button is enabled, a key containing the last known value for the channel is added to all previous keyframes for each channel that has changed. If the **WYSIWYG** button is off, then only the first keyframe (at time 0) and the new keyframe contains the new key. By default, this button is on. Refer to "Controlling the way keyframes are recorded," on page 46, for more information.

Utility controls

The following controls perform actions needed to save and complete your animation.

Read Script

Read in a previously saved animation script and the associated network. A clear network operation is performed prior to adding the network. If any data files were referenced in the network, they are read in. All modules are initialized.

Save Script

Use this button to save the animation script containing the current keyframe settings and to save any associated networks or Geometry Viewer scenes.

The AVS Animator saves a copy of the current network into a network file (*.net*) as well as creating a file containing the setting information (*.anim*). The same name is used as the base for both files. Be careful that this name does not conflict with any original network file names as it will be overwritten A *.scene* file and a *.geom* file are created if geometries are present, without a network.

Note

Both Read Script and Save Script actions must be completed and the browsers must be closed before any other action can take place within AVS.

Save CLI Script

Save the current CLI command structure to a file. This file can then be used as the base for a CLI script that can be used to automate some of the process. An extension of *.anim* is used. Refer to "Using CLI scripts," on page 47, for more detail on how to use CLI scripts. Also, refer to the CLI chapter in your AVS documentation.

Mode Toggle

Use this button to change between the Compact and Full Animator display control panels.

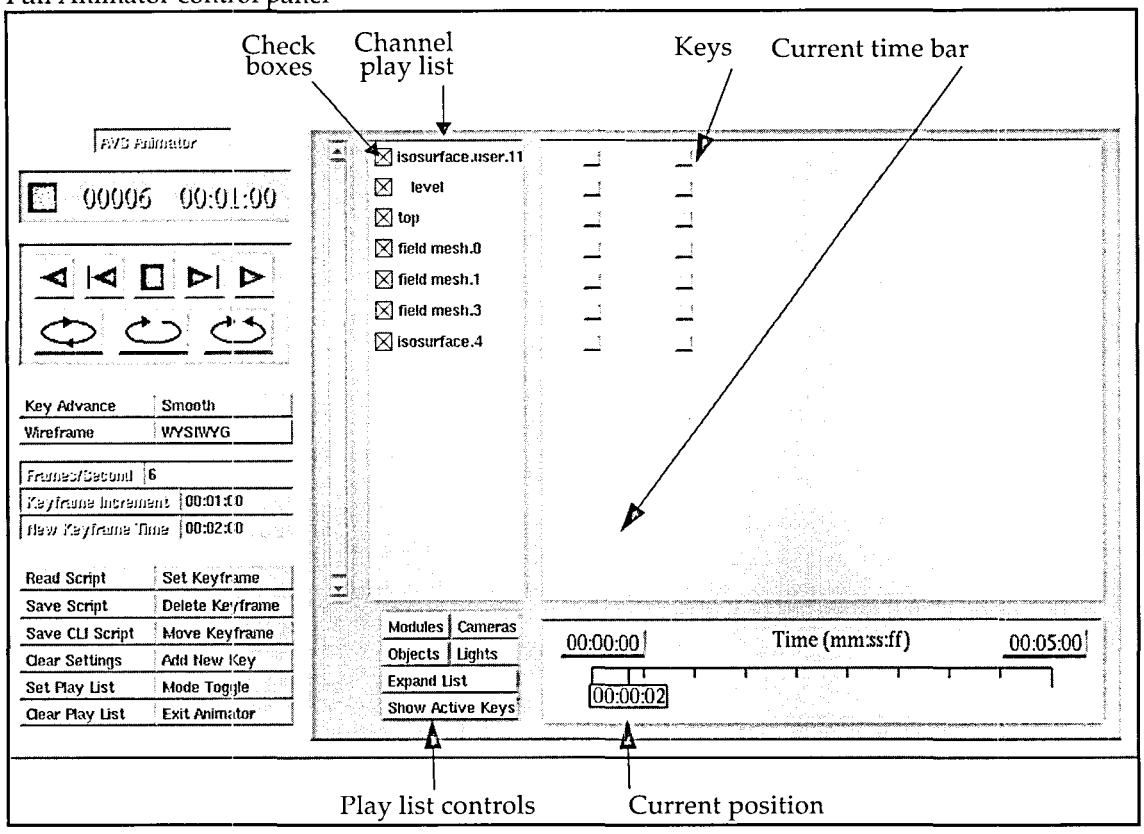
Exit Animator

Use this button to close the AVS Animator. This action frees memory and cleans up the environment. The module is no longer usable. You must remove the module by dragging it to the Hammer icon. Drag down another **AVS Animator** module from the Network Editor palette to initialize a new AVS Animator.

Full Animator control panel

The following section describes the buttons and type-ins associated with the Full Animator control panel. This panel is accessed by selecting the **Mode Toggle** button on the Compact Animator control panel.

Figure 13
Full Animator control panel



The Full Animator control panel, shown in Figure 13, provides all of the facilities of the Compact Animator, with additional editing and previewing options.

The Full Animator lets you:

- Temporarily remove or add individual channels of keys from the play list. You can, for example, have the Animator play back just an object's rotation, holding all other keys constant.
- Edit the value of individual keys for fine-grained control. Perhaps the object translation in your original keyframe goes a little too far along the X axis. You can use the Key Editor to select the X object position at a particular keyframe and type in a new value that gives a better visual effect.

A *channel* is composed of all the keys over the entire animation sequence representing a single entry that can be animated. For example, a parameter (isosurface level), a camera position, and so on.

The Full Animator controls include a display showing each channel of the animation as a function of time. Properties of objects and parameters of modules are represented as channels.

The *channel play list* can be modified according to type by a set of toggle buttons (object, module, cameras, light).

Some modules and certain parameters are exempted from the play list. You can customize your script by listing modules in a configuration file (*.animrc*). This file is described in detail at the end of this chapter.

The upper-right portion of the display shows a key button for each module, module parameter, object, light, and camera that is being tracked between each keyframe. These buttons are added automatically to the channel for each property or parameter that changes for each keyframe. These keys are then continuously tracked through all subsequent frames.

At the bottom of the control panel is a time scale. You can drag the current-position slider, and you can type in beginning and ending times for the visible range.

A vertical dotted line marks the current time.

Full Animator additional controls

Figure 14 shows the additional controls added to the panel.

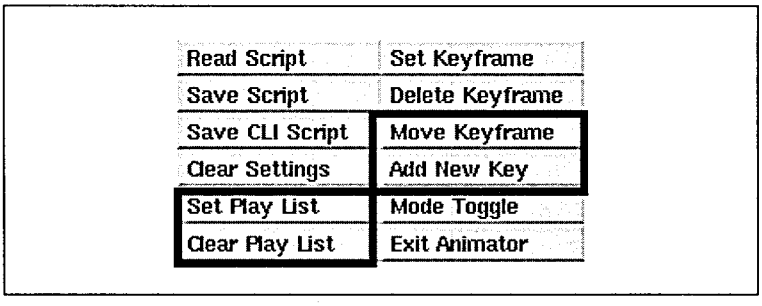


Figure 14
Extended mode buttons

Add New Key

Add keyframes between existing keyframes with the **Add New Key** button. Any new keyframes added after this key will automatically include this channel.

Use the following steps to add a new key:

1. Position the time slider to the frame that you want to add an active key to. You can also enter the frame number in the **New Keyframe Time** type-in.
2. If needed, select the **Expand List** toggle and deselect the **Show Active Keys** list to display all events in the channel list.
3. Select an event from the channel list that you wish to begin tracking (add to the play list). You must select one item from the channel list before adding a new key.
4. Click on the **Add New Key** button. A key button is added to the display in the channel for the selected channel.

Any new keyframes added after this key will automatically include this channel.

You can change an individual parameter within an existing keyframe by first changing the parameter setting and then adding a new key.

You can also edit the existing parameter value by clicking on the key and typing in a new value from the Parameter Editor window. This action overrides the existing key value.

Move Keyframe

Move an existing keyframe to a different time position. You are asked to verify that you want to move the keyframe:

```
Do you really want to move the keyframe at time (00:02:00)?
```

Enter the time position to move the keyframe to. Use time format mm:ss:ff (00:02:05) for minutes, seconds, and frame number.

The current time position must be at a keyframe.

Note

The last keyframe represents the last frame of the animation.

Set Play List and Clear Play List

Use the **Set Play List** and **Clear Play List** buttons to enable or disable all keys listed in the channel list.

Channel play list and check boxes

Each channel has a check box which, when checked, enables Animator playback for that channel. This box is checked by default.

If you click on this box or the associated name, the control panel (if one exists) for the module associated with the selected parameter appears.

The parameter field output shows that channel's value as a function of time, if the channel value is numeric.

All check boxes for the animation can be enabled or disabled by clicking on the **Set Play List** or **Clear Play List** buttons.

Play list controls

The Play list control buttons, shown in Figure 15, determine the type of event parameters that are displayed in the Key display window.

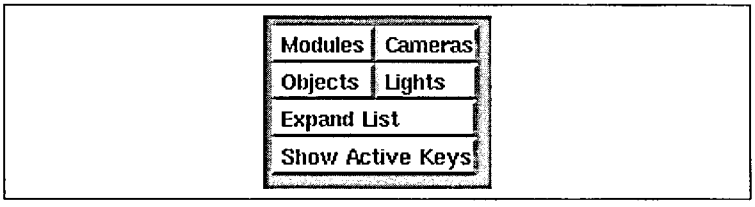


Figure 15
Play list controls

- **Modules**—Display parameters associated with modules in the network.
- **Cameras**—Display channels from the camera views in the Geometry Viewer.

Note

Only one camera (camera 1) can be used in an animation sequence.

- **Objects**—Display transformation and property channels that are applied to the objects in the scene.
- **Lights**—Display transformation and property channels that are applied to the lights in a scene.
- **Expand List**—Display all possible parameters and properties that can be tracked for each type of channel (modules, cameras, objects, and lights). When not set, only those channels that contain active keys are displayed.

- **Show Active Keys**—Set this toggle to display only those events and parameters that are actually changed in the sequence.

A scrollbar on the left side of the panel scrolls through the channel list.

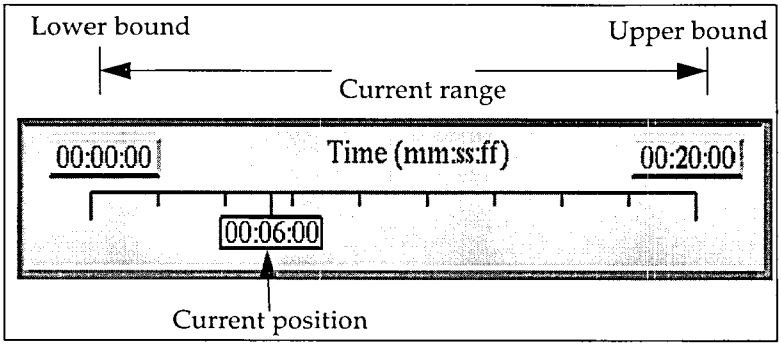
Time slider control and editor

Figure 16 shows the Time slider control. A time/frame counter at each end indicates the visible time range of your animation.

If there are more frames beyond either end of the range, then an arrow appears at the left or right end of the box.

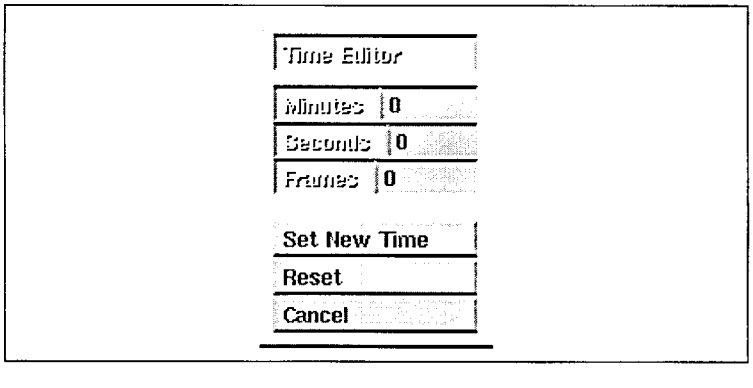
The current position counter indicates the current time position within the script. You can change the current time by holding down any mouse button over the box and dragging it to the left or right.

Figure 16
Time slider control



You can change the visible range by clicking on the beginning and ending range counters with the right mouse button to bring up the Time Editor, shown in Figure 17.

Figure 17
Time Editor popup



If you set a range that is beyond the last keyframe position in the animation and then move the Current Position box beyond the last keyframe position, the box turns yellow.

Using the Key Editor

Click on one of the key buttons to bring up the Key Editor, shown in Figure 18.

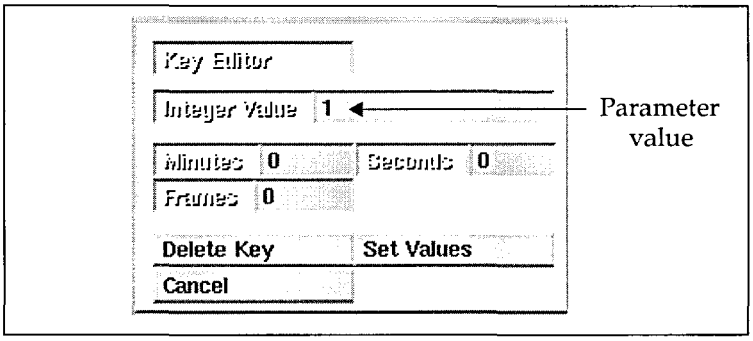


Figure 18
Key Editor popup

Use the Key Editor to perform the following tasks:

Changing parameter values—In the example shown in Figure 18, the parameter is an integer value. Parameters that have floating point, integer, or character values can be modified through the first type-in. If the key represents an object or module name instead of a parameter or property, then there are no accessible parameter values and the first field does not appear.

Moving the key—Set the Minutes, Seconds, and Frames type-ins according to the time you want this key to have. Select the **Set Values** button to change the value of the channel, if applicable, or to move the key to a new position in the script.

Removing the key—Use the **Delete Key** button to remove this key.

Changing your mind—Click on the **Cancel** button to close the Key Editor. Changes not set are ignored.

You cannot edit channel keys for module names (these act as placeholders within the channel list).

Working with time

In keyframe systems, it is useful to be able to control the kinetic (velocity and acceleration) characteristics independently of the positional information for keyframes. Kinetic control implies varying the number of in-between frames, which causes an object to move faster or slower without changing the start and final positions. There is no direct way in the AVS Animator to do this, but you can indirectly accomplish this by varying the number of seconds between keyframes and number of frames per second.

Although these settings should be established before editing individual keyframes, they can be changed at any time. The total number of frames and duration are automatically adjusted by the AVS Animator as you add new keyframes. You can, however, anticipate the limits by setting a range and setting a final keyframe.

Note

Playback time within the AVS Animator is usually proportionate to, but does not exactly correspond to absolute video or film running times. The workstation generates the frame renderings as you interact with the system. Most systems cannot do this at the 30, 25, or 24 frames per second rate.

Keyframe Interpolation methods

The *interpolation method* controls the way each active parameter, property and object changes from frame to frame as the script is played back through the AVS Animator.

Two interpolation methods are used with this version of the Animator:

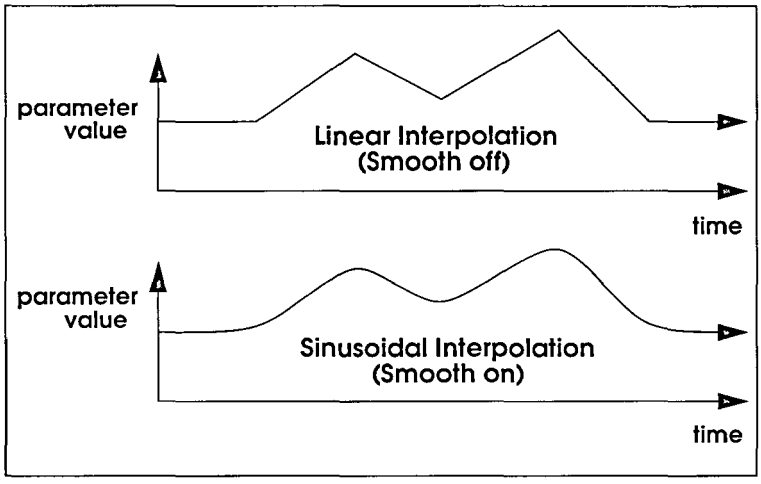
- **Linear interpolation**—Draws a straight line between key values as they change from one keyframe to the next. The parameter values for individual frames are computed such that they fall in this line. This is the default method.

  If you are animating a scene in which the action involves a steady progression through your data field, fluidity is not as important as a consistent movement through the field. Therefore, the default linear interpolation method could be used.

- **Sinusoidal interpolation**—Draws a section of a sine curve (of length π) between key values as they change from one keyframe to the next. In this form of interpolation, all derivatives are continuous.

Figure 19 shows two graphs representing the results of different interpolation methods.

Figure 19
Interpolation graphs



If you are using geometric objects to represent your data, you may want to use the **Smooth** toggle to enable sinusoidal interpolation when the objects are moving across the screen. This method reduces the jerkiness of sudden changes in position.

Quaternion interpolation for rotations

Rotation of an object is a difficult problem in animations. There are two types of rotation which can be used to interpolate the position of objects: regular rotation and quaternion rotation.

Regular rotation uses the traditional X, Y, and Z transformation dials which are accumulated before combining them in a transformation matrix. This has two unfortunate side effects. One, using absolute rotation values can create a problem sometimes known as gimbal lock, the loss of a degree of rotational freedom when trying to rotate an object to certain positions. The other unfortunate side effect is that interpolation will not choose the shortest rotation path between two positions, but will rather linearly interpolate the X rotations, the Y rotations, and the Z rotations separately, then concatenate the three matrices to create the final rotation transformation.

Quaternion rotation solves both of these problems. One, objects can be rotated intuitively (an X rotation always rotates around the horizontal, a Y rotation around the vertical, and a Z rotation perpendicular to the screen). Also, quaternion interpolation chooses the shortest path between two positions. The only drawback to using quaternion rotation is that the object must be rotated less than 180 degrees or the shortest path will then be in the opposite direction. To avoid this problem, save intermediate keyframes so that the rotations between keyframes are always less than 180 degrees.

The AVS Animator always uses quaternion rotation.

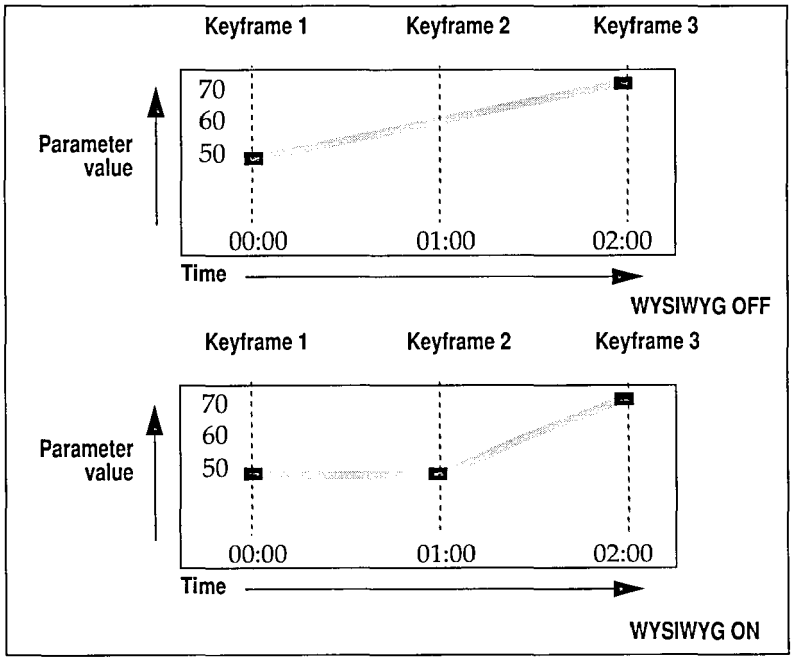
Controlling the way keyframes are recorded

The Animator propagates keys when adding new keyframes differently depending on how the **WYSIWYG** toggle is set. If it is on, a key containing the last known value for the channel is added to all previous keyframes for each channel that has changed. If the **WYSIWYG** button is off, then only the first keyframe (at time 0) and the new keyframe contains the new key.

For example, a module containing parameter par1 has this parameter set to 50 at time 0, is not changed (a value of 50) at time 1 (keyframe set), and is changed to a value of 70 at time 2 (keyframe set).

The diagram in Figure 20 shows the results if the **WYSIWYG** button is enabled or disabled.

Figure 20
How **WYSIWYG** affects adding new keys



With the **WYSIWYG** button enabled, the value for keyframe 1 is duplicated as a key in keyframe 2. This does not occur if it is off.

Note

The WYSIWYG button only affects the way in which new keyframes are added. It has no effect on existing keyframes.

Using CLI scripts

The AVS Animator lets you save the current settings as a Command Language Interpreter (CLI) script that you can then edit for special circumstances. It is a good precaution to save your session as a CLI script prior to rendering the final sequence.

CLI scripts can be used in the following ways:

- CLI scripts provide a fast way to replay an animation.

  If you execute the following command from the CLI command line, the animation will play back faster than if you select one of the Animator's play buttons.

  `script -play` *filename*

- If you end up with a bad frame or two on the video tape, you can duplicate that frame by looking in the CLI script file for the bad frame and setting the appropriate parameter and geometry values.
- If you are trying to produce an animation but there is some thing that the animator won't let you do (for example. animate color maps), write out a CLI script and then edit the script and add your changes.

Note

Modifying CLI scripts requires an intimate knowledge of the AVS system and commands. Refer to the AVS documentation for more details.

AVS Animator limitations

The Animator records, but does not interpolate changes in:

- Object rendering modes (such as a wireframe dissolving into gouraud)
- Field and UCD data values or coordinates (such as field data recorded as time-series)
- Coroutine modules, such as simulations, that act asynchronously with an AVS network
- Image and Graph Viewer control panel manipulations (however, the images and graphs appearing in these viewers are animated).
- Textures are not tracked. You cannot coordinate changes or turn textures on and off.

The AVS Animator does not record or interpolate changes made to colormaps with the **generate colormap** module's Colormap Editor widget.

You can instruct the AVS Animator to ignore changes made to specific modules by putting their names into a *.animrc* configuration file that is referenced by the **AVS Animator** module when it is initialized.

It is possible to animate data values and coordinates that change over time. To do this, you need to generate the interpolated in-between values yourself. There are two cases:

- Write an interpolate field or interpolate UCD module.
- Rewrite a simulation coroutine to run synchronously with an AVS network.

The **AVS Animator** module includes an integer frame number port that can act as a synchronizing port for coroutine modules.

User written modules that have *oneshot* parameters cannot be animated. The AVS Animator has no way to reset a oneshot, so when a scene is played in reverse, the system will behave unpredictably.

The **AVS Animator** module is not saved into a network file (**Write Network**).

AVS Animator configuration file

A configuration file (.animrc) is read by the **AVS Animator** module each time it is initialized. The animator looks for the file in these places in this order:

```
.animrc (current directory)
~/.animrc (home/default directory)
$path/runtime/animrc
```

The file contains a list of modules, listed one module name on each line, that will be ignored by the Animator. An example default file, *animrc*, is located in *the /usr/avs/runtime* directory. Here is an example list:

```
AVS Animator
display image
display pixmap
display tracker
graph viewer
prepare video
read frame seq
render geometry
write frame seq
```

As a general rule, all asynchronous coroutine modules should be included in this list of modules that will not be tracked by the AVS Animator.

Recording and storing images to disk

4

One of the final tasks in the animation process is to output the individual frames to a disk and to record the images using some recording device.You could save each image as a separate file, then read each file, one at a time. However, this method is hard to manage and can be confusing. The AVS Animation Application includes a module that gives you much more control over this process.

The **write frame seq** module writes a sequence of images to disk, compresses the output, and creates a single file containing the frame sequences. These files can be subsequently read by the **read frame seq** module. The **write frame seq** module can optionally compress the images, and it has several controls for editing sequences. The module icon is shown in Figure 21.

Figure 21
write frame seq module icon

This module can be used with the AVS Animator, but can also be used independently from the AVS Animator to store frames from any source that outputs images.

Input format requirements

The input image must be a uniform 2-D field, with a 4-vector of byte data values at each location in the field. The image can be any size. Any outputs that can connect to a **display image** module can be connected here.

Using the controls in Figure 22, you can create and modify sequence files of your animation.

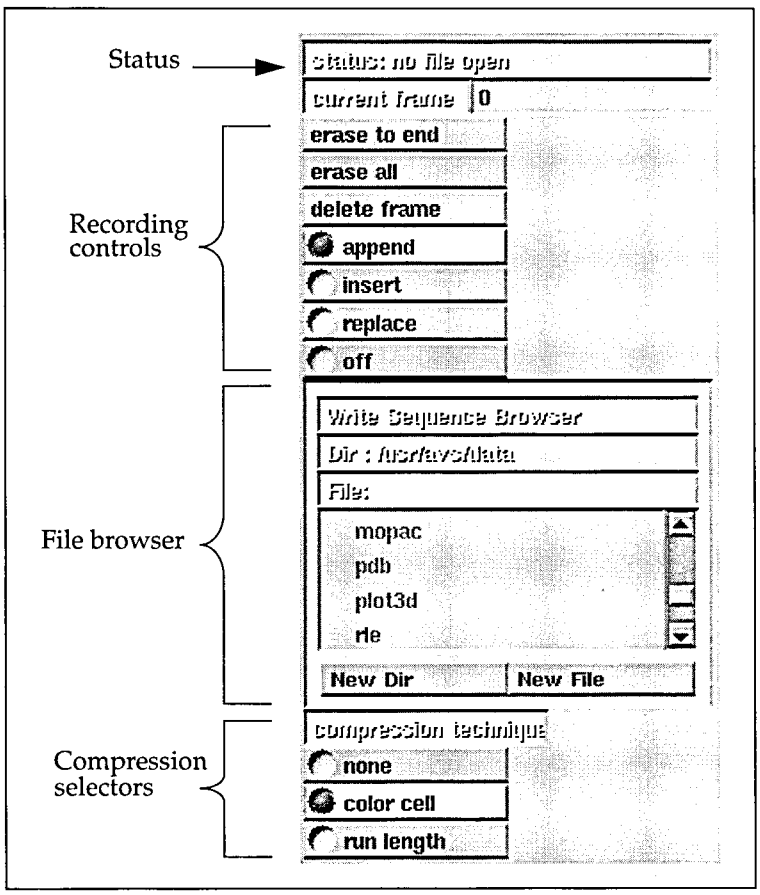


Figure 22
Write frame sequence controls

Module parameter controls

The following sections describe the parameter controls for the **write frame seq** module,

Status indicators

The control panel for this module contains two indcators: status and current frame.

status

A text widget displays the current sequence file and the number of frames in the file. If there is no current file, it displays a `no files open` message.

current frame

The integer type-in may be used to specify an absolute frame position. The **erase to end** and **delete frame** buttons and the **insert** and **replace** mode buttons refer to this value.

Recording controls

The following controls can be used to manage the image sequence file that is generated:

erase to end

A one-shot button that, when pressed, erases all frames from the current frame through the end of the sequence. There is no way to undo this action.

erase all

A one-shot button that, when pressed, erases all frames in the file. When **erase all** is pressed, a confirmation dialog box appears.

delete frame

A one-shot button that, when pressed, deletes the current frame from the file. This does not create a gap in the frame sequence; instead, frames following the deleted position are shifted down.

append

Enable this button to append each new frame to the end of the image sequences in the file.

insert

When this button is enabled, each new frame is inserted before the current frame, then the current frame number is incremented so the next frame will follow the new frame in the sequence.

replace

Enable this button to replace the current frame with the new frame, then increment the current frame number.

off

Use the **off** button to stop recording. New frames are not written out to disk.

Write Sequence Browser

Use the file browser to select an existing image sequence file or to create a new one. An extension of *.seq* is added to the file name.

compression technique

The **compression technique** radio button set lets you choose the compression technique. The default compression technique is **none**. If no technique is selected, the frames are stored uncompressed.

color cell lets you store frames using color cell compression (CCC). Color cell compression is a compression technique that requires 2 bits per pixel, regardless of image complexity. This technique results in some loss of information, but the resulting compressed files are much smaller.

run length stores frames using an enhanced run length encoding. Run length encoding (RLE) is a loss-less compression technique.

Note

A better compression rate is possible (for RLE) if the frames have zeros in their alpha channel. Frames typically compress to 8-10 bits per pixel if the alpha channel is zero, depending on image complexity. The replace alpha module **may be used upstream of the** write frame seq **module to clear the alpha channel.**

Although images are compressed, a long sequence can easily add up to many megabytes of storage.

If you enable CCC compression, the images require about 1/15th the space (color cell compression), but it takes longer to save each frame and it takes longer to read the file.

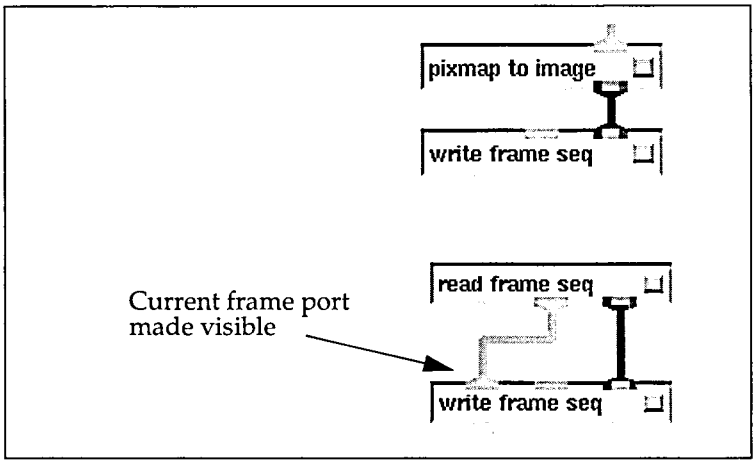
Refer to Appendix A, "Frame sequence file formats," on page 69, for details about file format and compression techniques.

Example networks

The networks in Figure 23 use the **write frame seq** module to generate an image sequence file.

You can revise the image sequence by writing portions of an image file to a new file, using the write controls to selectively stop and start the recording.

Figure 23
Write frame seq example networks



Reading image sequence files from disk

5

This section describes the method for reading animation image sequences using the **read frame seq** module and associated interface.

The **read frame seq** module reads an image sequence file that has been created by the **write frame seq** module.

When this module is initialized, the read sequence control panel is displayed, as shown in Figure 24.

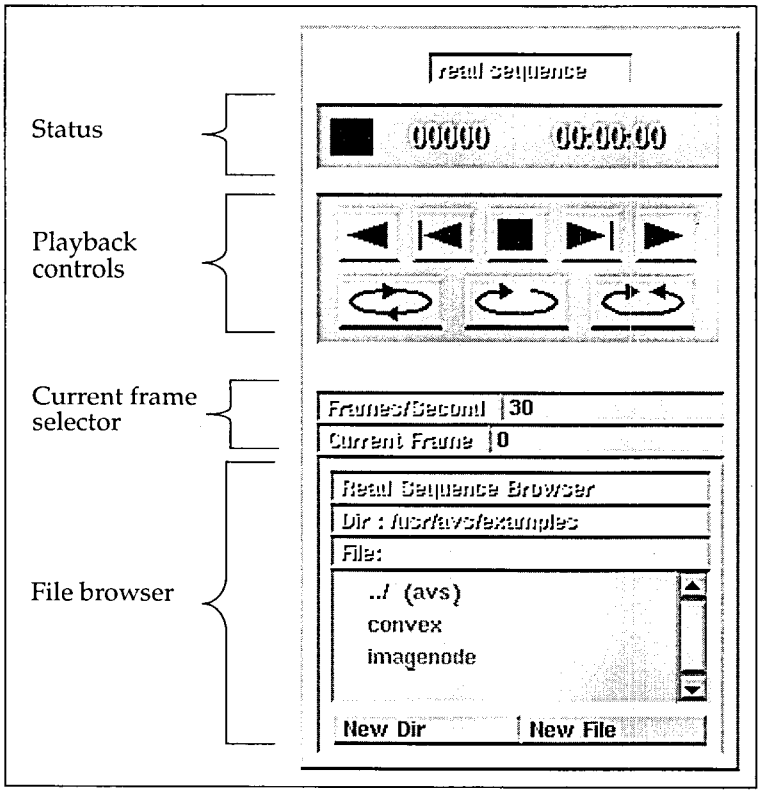


Figure 24
read sequence control panel

Module parameter controls

The play back controls act in the same way as the **AVS Animator** module. The top of the control panel displays the current action icon, absolute frame position, and the current time position in minutes, seconds, and inter-frame number.

The *current action* icon changes according to the playback controls currently active, as shown in the next section.

The *time setting* identifies a specific position in the sequence of images recorded in the file, called out in minutes and seconds. The ratio of frames to seconds varies according to the frames per second setting.

Playback controls

The following buttons control the playback of the images in the sequence file:

Play reverse

This button reads frames in descending order starting with the frame prior to the current frame. The sequence stops when the beginning of the current file is reached (frame zero). If the current frame is zero, the sequence starts with the last frame.

Step reverse

This button plays the frame previous to the current frame. If the current frame is zero, the last frame is displayed.

Stop

Stop any playback that is in progress.

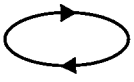
Step forward

This button plays the frame after the current frame. If the current frame is the last frame, frame zero is played.

Play forward

Use this button to play forward, towards the end of the script, one frame at a time, starting at the next frame after the current frame and stopping when the last frame in the file is reached. If the current frame is the last frame, the sequence starts at frame zero.

Play continuously (loop)



Select this button to play in a continuous loop, starting at the current frame, until the **Stop** button is pressed.

Play once

Cycle once from the first frame to last frame in the image sequence file.

Bounce play

Cycle back and forth continuously through the image sequence file until the **Stop** button is pressed.

Parameters and outputs

The following parameters and output ports are used by this module:

Frames/Second

This is an integer type-in is used to produce the *mm:ss:ff* status display. This value is initially set to 30. The setting is updated when a file is read.

Current Frame

Use this integer type-in to directly access any frame in the sequence. This value is also automatically updated when the sequence is displayed to display the current frame number.

Read Sequence Browser

A file browser allows you to select the sequence file that you wish to read. The file is automatically decompressed if previously compressed using the write sequence.

Image out (outputs)

The **read frame seq** module outputs an AVS image (field 2-D 4-vector byte).

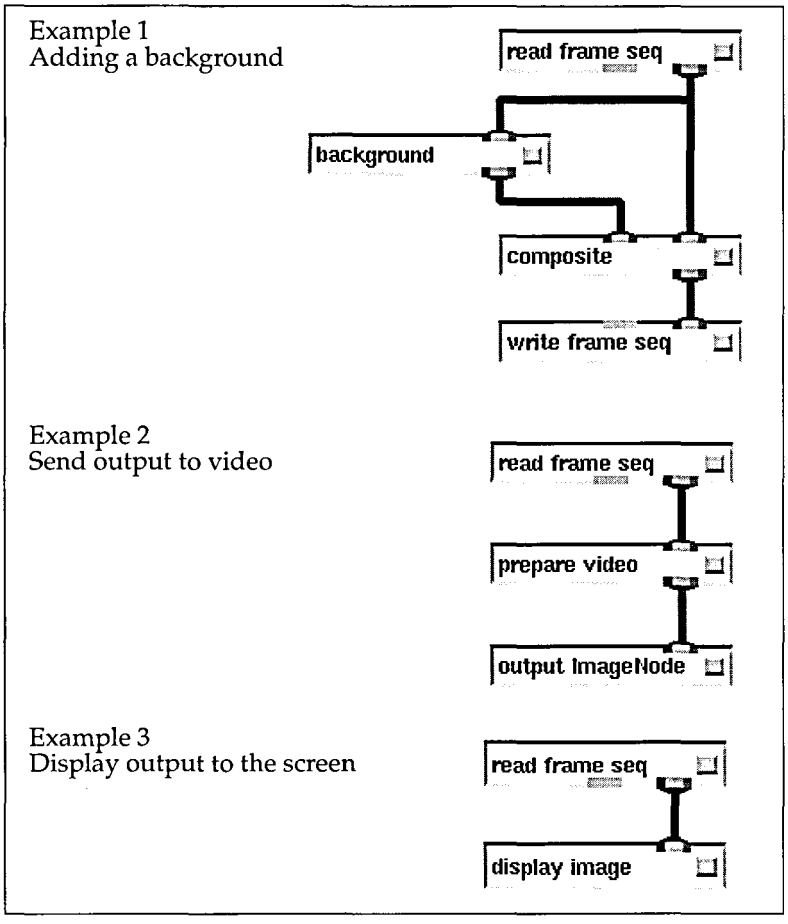
Example networks

The networks in Figure 25 represent some of the ways that you might use the **read frame seq** module. The first network reads an existing image sequence file to which the background is combined with the original images and then written back into a new sequence file with the **write frame seq** module.

In example 2, the output of the **read frame seq** module is sent to a video device after being filtered by the **prepare video** module.

In example 3, the output is displayed to the workstation screen.

Figure 25
Example networks



Recording on video systems 6

This chapter describes the modules that process the images in the sequence for output to video.

There are several considerations for outputting color graphic images to video. First of all, computer graphics have a much higher resolution than video.

NTSC (National Television Standards Committee) video is the signal format used in North American commercial television. PAL video is a signal format common in the European and South American commercial television.

NTSC and PAL, broadcast video standards, are low resolution when compared to most workstation color graphics displays. Both use interlaced video, and both have insufficient bandwidth to permit abrupt horizontal color changes. A *low pass filter* helps to resolve both problems. A low pass filter blends images horizontally, to eliminate abrupt transitions, and it also blends images vertically so that features do not disappear between the scan lines when the picture is interlaced.

Table 2
Format comparison table

Format	Width	Height	Frames/ second	Interlace
NTSC	640	480	30	yes
PAL	768	575	25	yes
Film	NA	NA	24	no
Computer graphics typical values	1280	1024	60+	no

Preparing computer graphics for output to video

Included in the Animation application is a module that prepares an image for output to a video device. This module, **prepare video**, performs low pass filtering, field interlacing, and gamma correction.

The module parameter controls are shown in Figure 26.

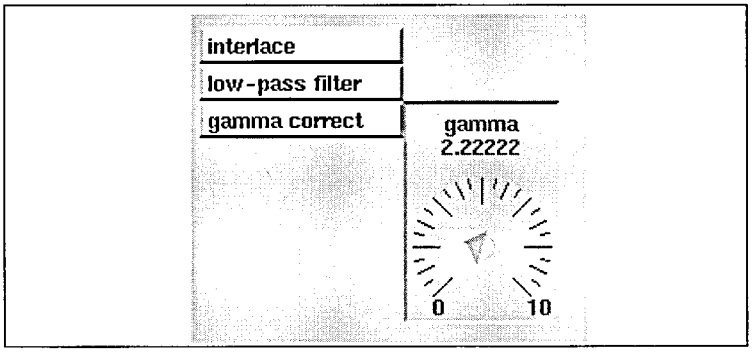


Figure 26
prepare video controls

Low-pass filtering

If the **low-pass filter** toggle is set, each input image is filtered using a filter optimized to reduce flicker in interlaced video. The filter uses an aperture four pixels high by two pixels wide.

Display refresh—interlacing

There are two types of refresh: interlaced and non-interlaced. The former is used in broadcast television and in raster displays designed to drive televisions. The refresh field is broken into two fields, each lasting 1/60th of a second so that the full refresh is 1/30th of a second. All odd-numbered scan lines are displayed in the first field, and all even-numbered ones are displayed in the second. The interlace function places new information into the screen at a 60-Hz rate, since a 30-Hz rate results in screen flicker.

When the **prepare video interlace** toggle is set, two consecutive fields input to the **prepare video** module are merged together by interlacing the lines. The even scan lines of the output come from the first image, and the odd scan lines come from the second. This doubles the apparent frame rate from 30 to 60 when outputting animation to an NTSC video device, or from 25 to 50 for PAL video.

When interlacing, you must take care to get the correct pairs of images interlaced together. One way to do that is to toggle the **interlace** button off then on again. This ensures that the **prepare video** module outputs the current image. Then, the next two images that the **prepare video** module receives are interlaced.

If two successive input images are different sizes, they are not interlaced. Instead, the first image is discarded, and the second is saved to interlace with the next image.

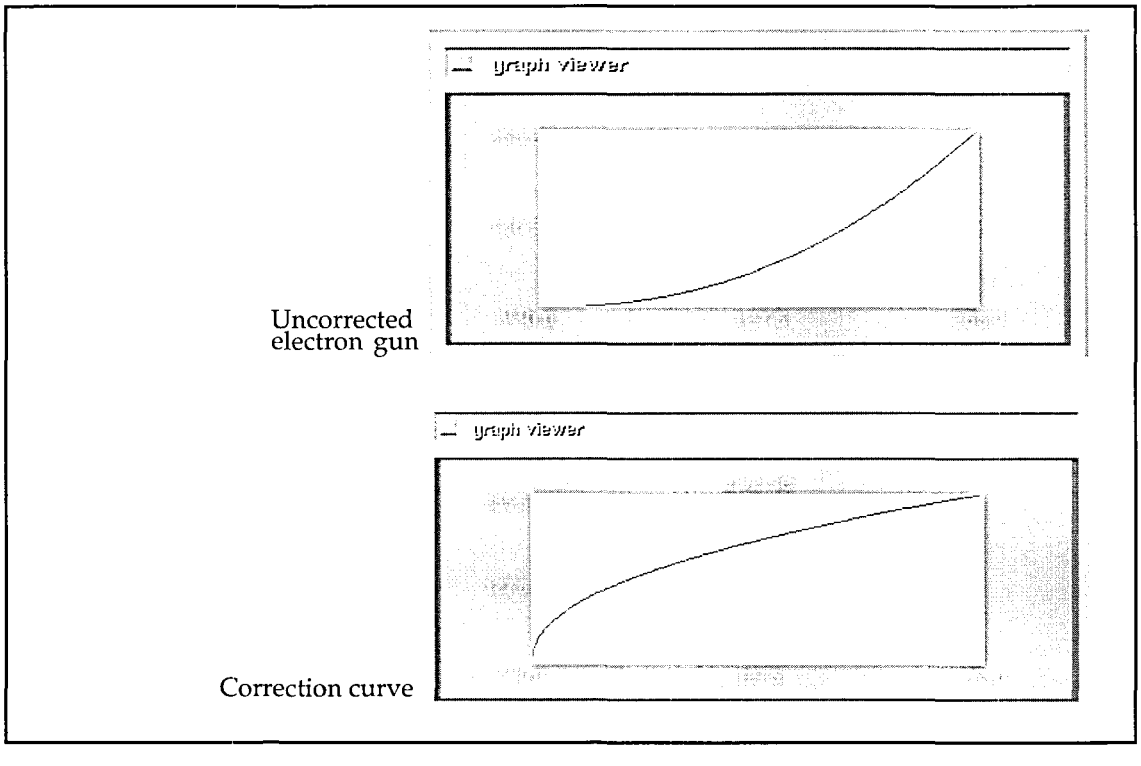
Gamma correction

The electron guns inside cathode ray tubes (CRTs) emit electrons nonlinearly with respect to their driving voltage. To compensate, the **prepare video** module multiplies each color component by a gamma correction curve.

If your video system has gamma correction elsewhere in the system, you do not need to use AVS gamma correction.

The graphs in Figure 27 show an uncorrected electron gun and the NTSC gamma correction curve.

Figure 27
Gamma correction curves



Video output modules

The following section describes video output modules.

Different hardware-specific video output modules are supported, depending on your AVS vendor. The base AVS Animation application includes an example output video module, **output ImageNode**. Source code for this module is provided as an example from which you can create your own custom output video modules.

The source code for **output ImageNode** is distributed in */usr/avs/examples/imagenode*.

Some output devices may be run through a remote module. Refer to your AVS documentation for more information about setting up and running remote modules.

Recording video through a Diaquest ImageNode

The **output ImageNode** module is used to record animation on video tape. The module displays each image it receives on the frame buffer of a Diaquest ImageNode.

You can set the **Record** toggle to record the images on an attached VCR using single-frame recording.

If the input image is smaller than the ImageNode's frame buffer, it is centered, and the remaining parts of the frame buffer are filled with black. If the input image is larger than the frame buffer, the image is cropped, and the center part is displayed. When the ImageNode uses NTSC format, the frame buffer is 648 by 486 pixels. In PAL format, the frame buffer is 768 by 576.

Configuration

The **output ImageNode** module uses three parameters to configure itself: ImageNode_hostname, ImageNode_format, and ImageNode_VCR_type. The module reads these parameters from AVS as Command Language Interpreter (CLI) variables.

`ImageNode_hostname`

This is the *Internet* host name that **output ImageNode** uses to connect to the ImageNode. It may be either a name listed in */etc/hosts* or a numeric Internet address of the form *nnn.nnn.nnn.nnn*. The default host name is `imagenode`.

`ImageNode_format`

This can have one of two values: `NTSC` or `PAL`. `NTSC` is the default.

`ImageNode_VCR_type`

When the **output ImageNode** module is instantiated, it looks for this variable. If it exists, it is passed to the ImageNode hardware followed by an `init` command. This can be used to tell the ImageNode what type of VCR is connected to it. The allowed values for this variable are shown in Table 3.

Table 3
`Image_Node_VCR_type` values

Value	VCR
`a60`	Abekas A60
`a62`	Abekas A62
`a64`	Abekas A64
`ampex`	Ampex VPR series
`lv-s100`	Hitachi LV-S100
`br-s810`	JVC BR-S810
`br-s811`	JVC BR-S811
`cr-850`	JVC CR-850
`kr-m800`	JVC KR-M800
`pr-900`	JVC PR-900
`tq-3031`	Panasonic TQ-3031
`simul`	Software VCR simulation
`lvr1`	Sony LVR-5000/6000 at 1200 baud
`lvr9`	Sony LVR-5000/6000 at 9600 baud
`sony`	Other Sony
`teac9`	TEAC LV-210A at 9600 baud

If this variable is undefined, no initialization is done, and the ImageNode uses the default VCR type set at the factory.

You may set these variables by creating an AVS CLI initialization file and referencing it in your *~/.avsrc* file. For example, if you wanted to use PAL as the default video format, you could put the following line into a file in your home directory called *~/.avs_cli_init*.

```
var_set ImageNode_format "PAL"
```

Then you would add this line to your ~/*.avsrc* file.

```
CLIinit    /your home directory/.avs_cli_init
```

The input image must be a uniform 2-D field, with a 4-vector of byte data values at each location in the field. The image can be any size. It will be cropped and matted to fit.

Recording control parameters

The parameters for this module are shown in Figure 28.

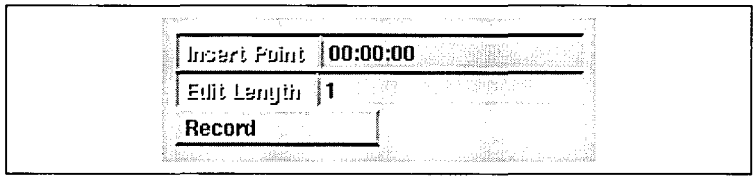


Figure 28
Module parameter widgets for **output ImageNode** module

All output modules have similar parameters.

Insert Point

Use this text type-in to specify the location on the video tape where the next image will be recorded. This may be specified as *mm:ss:ff* where *mm* is minutes, *ss* is seconds, and *ff* is the frame number. The insert point is incremented by the edit length after every entry record.

No recording occurs if the value is set to 00:00:00.

Edit Length

Use the **Edit Length** integer type-in to specify for how many frames the next image will be recorded.

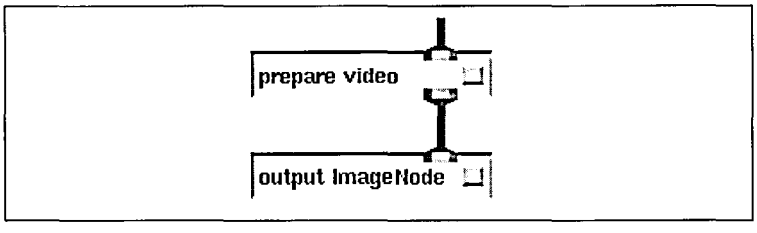
Record

Set this button to enable frame recording.

Example network

The **output ImageNode** module can be attached to any network that outputs a uniform 2-D, 4-vector field (AVS image). In the example shown in Figure 29, the **prepare video** module performs required filtering.

Figure 29
Example network for output ImageNode module



Related modules

The following module can provide the input field:

- **prepare video**
- Any module that outputs an image

The following module can replace **output ImageNode**:

- Another output video module
- **Display image** module

Note

Only the ImageNode Professional model is supported.

Frame sequence file formats

This chapter describes the file format and compression techniques used by the **write frame seq** module and the **read frame seq** modules to create and read image sequence files.

An image sequence file contains a sequence header for each frame in the stored sequence. Each header describes one frame, and contains the 64-bit offset (from the beginning of the file) for its frame.

- All headers are stored in big-endian order, with 4 bytes per INT and 8 bytes per INT 64.
- All frames are arrays of bytes, and higher level software should use consistent word order on their contents.
- All frames and frame headers are aligned to 8-byte boundaries.

The frames may be in any location in the file. No two frames may occupy the same location.

The **write frame seq** module includes two compression methods for writing images to a file: RLE (run-length encoding) and CCC (color-cell compression).

RLE provides accurate storage of 24-bit graphics. However, it is efficient only if the image has long spans of identical pixels. This is not a feature of scanned in and anti-aliased images.

CCC, although it is a lossy compression technique, results in significant reduction in storage requirements and should be considered for many projects that output to video.

Run length encoding (RLE)

The **write frame seq** module uses a byte-oriented run length encoding (RLE) method. RLE provides accurate storage of 24-bit graphics. That means that if you compress and decompress, the resulting image is identical to the original (lossless).

Run length encoding encodes sequences (runs) of adjacent, similar pixels in a small amount of space. The more similar the pixels are, the less space is required.

The encoded data consists of runs of 4-byte pixels, 3-byte pixels, 2-byte pixels, and 0-byte pixels, and of individual 1-byte pixels.

Most pixels have their values encoded as a difference from the previous pixel. Pixels are ordered left to right, top to bottom.

Each of those encodings is described below.

0-byte run

If several pixels are identical to their predecessors, they can be encoded as a run of 0-byte pixels. A run of 0-byte pixels is encoded as simply an opcode and count, as described below.

1-byte pixel

If a pixel's red, green and blue values differ from its predecessor's by +/- 2 or less, and it has the same alpha Lvalue, then it can be encoded as an individual 1-byte pixel. A 1-byte pixel is encoded as:

$$opcode = (\Delta r+2)*25+(\Delta g+2)*5+(\Delta g+2)$$

where Δr, Δg and Δb are defined as:

Δr = current_pixel.red - previous_pixel.red
Δg = current_pixel.green - previous_pixel.green
Δb = current_pixel.blue - previous_pixel.blue

Since Δr, Δg, and Δb are between -2 and +2, the op code is between 0 and 125.

2-byte run

If several pixels' red, green and blue values differ from their predecessors' by +19/-20 or less, and they have the same alpha values, they can be encoded as a run of 2-byte pixels. A run of 2-byte pixels is encoded as an opcode and count (described below) followed by a 2-byte word for each pixel. The 2-byte word is encoded as:

$$number = (\Delta r + 20)*1600+(\Delta g+20)*40+(\Delta b+20)$$

Because Δr, Δg, and Δb are between -20 and +19, the opcode is between 0 and 63999.

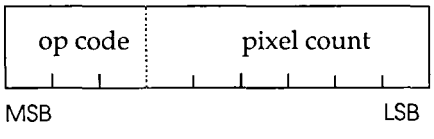
3-byte run

If several pixels have the same alpha value as their predecessors, but don't have RGB values close enough to be encoded as a 2-byte run, then they can be encoded as a 3-byte run. A 3-byte run is encoded as an op code and count (described below) followed by 3-bytes per pixel. The 3 bytes are the pixel's red, green, and blue values.

4-byte run

If several pixels have differing alpha values, they must be encoded as a 4-byte run. A 4-byte run has an opcode and count (described below) followed by four bytes per pixel. The 4 bytes are the pixel's alpha, red, green, and blue values.

Each run starts with an opcode and count. The first byte has this format.



The three opcode bits have the following meanings:

100	run of 0-byte pixels
101	run of 3-byte pixels
110	run of 4-byte pixels
111	run of 2-byte pixels

Because the high bit of the op codes for each run is set, and the high bit of all individual 1-byte pixels is clear, the two cases can be distinguished unambiguously.

The low 5 bits of the first byte encode the number of pixels in the run. A pixel count between 1 and 29 is encoded directly in the 5 bits of the count field. The value 0, 30, or 31 in the count field indicates that the following byte(s) contain the actual pixel count, according to this table. The multibyte counts are in big-endian order.

Count field	Actual count
0	in next byte
30	in next 4 bytes
31	in next 2 bytes

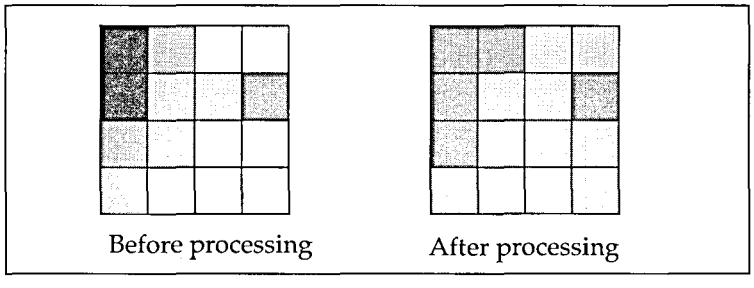
If the run has any per-pixel data, those data follow the actual count.

Color cell compression

Color cell compression (CCC) is a lossy compression technique: an image that has been CCC compressed and decompressed will not be identical to the image before compression. The losses are least visible on images containing soft edges and gradually changing colors. These are the images on which RLE compression is least effective.

CCC divides the image into 4-by-4 pixel cells. Each cell is colored using just two colors, as demonstrated in Figure 30.

Figure 30
Color cell compression



The color cell compression (CCC) algorithm is based on the observation that in normal images change in chrominance almost always indicates a change in luminance as well.

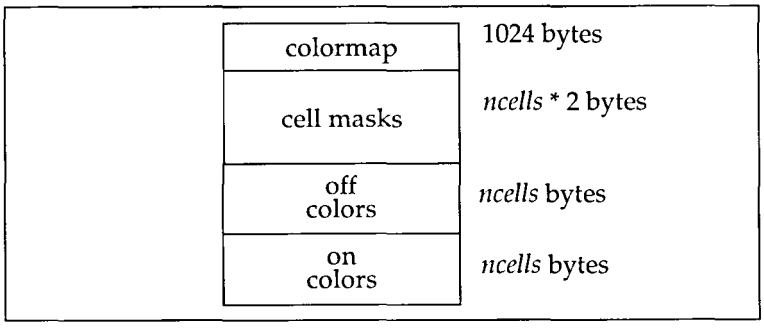
The color space of camera-digitized or synthetic images is often as high as 24 bits/pixel, or 8 bits for each of red, green, and blue. For natural images digitized at spatial resolution of 640 by 480 pixels, the number of distinct colors produced is typically 30,000 or less, which is about 10 pixels per color for this image. Many synthetic images will produce a distinct color for every pixel of the digital image. This level of color resolution, however, is seldom discernible by the human eye.

Reducing the number of RGB values from 8 bits to 5 bits reduces the number of required distinct colors without significantly affecting the resulting graphic output. The potential saving of up to 90% of the storage or transmission costs should be considered.

CCC encoding provides color images equivalent to the quality of techniques that use 6 bits per pixel but requires only 2 bits per pixel plus a fixed-size look-up table of 6144 bits.

Figure 31 shows the file format for CCC.

Figure 31
CCC file format



ncells equals the number of cells in the image, as defined in the following equation:

$$ncells = \left\lceil \frac{width}{4} \right\rceil \times \left\lceil \frac{height}{4} \right\rceil$$

The cell masks describe which bits in each cell use which colors. The pixels in the cell are numbered as shown in Figure 32.

Figure 32
Cell mask bit numbering

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Bit 0 is the least significant bit (LSB) and bit 15 is the most significant bit (MSB) of the mask.

Those pixels whose mask bits are 1 are colored with the cell's *on* color, and those whose mask bits are 0 use the cell's *off* color.

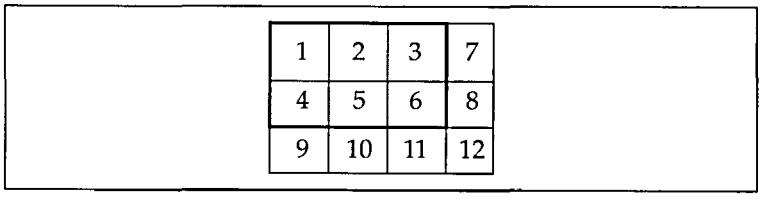
If the image width is a multiple of 4, there are no partial cells along the right edge. If the image height is a multiple of 4, there are no partial cells along the bottom.

The image is divided into cells starting at the upper-left. If the image width is not a multiple of 4, there is a column of partial cells along the right edge of the image. If the image height is not a multiple of 4, there is a row of partial cells along the bottom edge.

The full cells are ordered left to right, top to bottom. The partial cells along the right edge, if any, come next, ordered top to bottom. The partial cells along the bottom edge, if any, come next, ordered left to right. The partial cell in the bottom right corner, if any, is last.

Cell order will vary if the image width and height is not a multiple of four. Some cells are full 4 by 4. Other cells along the right and bottom edges are less than 4 by 4, as shown in Figure 33.

Figure 33
Cell order for 13-by-10 pixels



Bibliography

This bibliography contains a list of related documents that provide general information to readers interested in more information on topics, but not information that is essential to using the material covered by the current document.

Altmann, Simon L. *Rotations, Quaternions, and Double Groups*, Oxford University Press, New York City, New York, 1986.

Amanatides and Mitchell, "Antialiasing of Interlaced Video Animation," *Computer Graphics*, Volume 24, Number 4, pp. 77-85, August 1990.

Berthold K.P. Horn, "Closed-Form Solution of Absolute Orientation Using Unit Quaternions," *Journal of the Optical Society of America*, Volume 4, Number 4, April 1987.

Campbell, Graham, et al, "Two Bit/Pixel Full Color Encoding," *Computer Graphics*, Volume 20, Number 4, pp. 215-219, August 1986.

Foley, James D. et al., *Computer Graphics: principles and practice*, Addison-Wesley Publishing Company, 1990.

Fillmore, Jay P. "A Note on Rotation Matrices," *IEEE Computer Graphics and Applications*, Pages 30-33, February 1984.

Ickes, B.P. "A New Method for Performing Digital Control System Attitude Computations Using Quaternions," *AIAA Journal*, Volume 8, Number 1, January 1970

Kane, Thomas R. , Likins, Peter W.; and Levinson, David A. *Spacecraft Dynamics*, McGraw-Hill Book Company, Chapter 1, Pages 1-43.

Pletinckx, Daniel. "Quaternion Calculus as a Basic Tool in Computer Graphics," *The Visual Computer*, Pages 2-13, January 1989.

Shoemake, Ken. "Animating Rotation with Quaternion Curves," *Computer Graphics*, Volume 19 Number 3, Pages 245-254, July 1985.

Shoemake, Ken. "Quaternion Calculus and Fast Animation," *Computer Animation: 3-D Motion Specification and Control* (SIGGRAPH 1987 Tutorial), Pages 101-121, 1987.

Glossary

A

aliasing
A reduction in image quality caused by representing an image as an array of discrete pixel values. Details that are too small to be resolved can cause large, inappropriate fluctuations in pixel values unless anti-aliasing steps are taken.

alpha
A common name for opacity information calculated and stored for each pixel in a frame.

animation
A sequence of images that, when shown in rapid succession, produces the illusion of a moving image.

animation segment
A specific animation sequence that represents one discrete set of actions and movements.

anti-aliasing
A mathematical process that makes jagged or stair-stepped edges appear smoother by averaging the surrounding colors.

C

channel
A *channel* is composed of all the keys over the entire animation sequence representing a single entry that can be animated. For example, a parameter (isosurface level), a camera position, and so on, that can be tracked by the AVS Animator.

chrominance
The difference between a color and a chosen reference color of the same luminous intensity in color television.

F

flipbook
A method of viewing animation by first saving the images and then viewing them sequentially in real time.

frame
A single image that is part of a sequence.

G

Gamma correction
RGB color correction to allow for the nonlinear characteristics of CRTs.

geometric transformation
A change in the geometric configuration of a scene such as rotation or translation (change in location). A transformation can be applied to individual objects or globally to the entire scene.

Gouraud Shading
A shading technique in which the surface shade is calculated at surface vertices, and interpolated for points in the interior.

I

interpolation
The calculation of the in-between values based on two other values, usually the first and last frames of a segment.

K

keyframe
A particular frame in an animation that is used as a guide for subsequent action.

L

low pass filter
Video image processing that smears images horizontally, to eliminate abrupt transitions, and smears images vertically so that features do not disappear between scan lines when the picture is interlaced.

luminance
The given intensity of a surface in a given area per unit of projected area.

N

NTSC
The standard signal for television broadcasting. Also known as composite video and RS-170A, the acronym stands for the National Television Standards Commission in the United States.

P

PAL
European television broadcasting standard.

POV
Point of view. The perspective from which a 3-D scene is viewed (usually from the camera).

Q

quantization
The process of reducing a sample pixel value to one of a number of fixed values.

R

raster
The arrangement of pixels in a display monitor as a two-dimensional array or grid of pixels.

real time
The viewing of an event as it happens.

render
The process of taking 3-D model information and its associated values (camera position, lights, and so on) and creating an image on the computer screen.

resolution
The degree of granularity of a display monitor specified by the number of rows and columns of pixels in the display.

rotation
A transformation that turns an object around an axis.

S

scaling
A transformation changing the size of an object in the x, y, or z directions.

scene description
The process of specifying a scene to be rendered in terms of objects, light sources, and viewing devices.

storyboard
A sequence of screens used to define visual and temporal aspects of an animation design.

T

temporal aliasing
The undesirable strobing effect in an animated sequence caused by abrupt changes in a scene between frames. Temporal aliasing can be eased by motion blur to help the eye make a smooth transition between frames of a moving object.

transformation
A function applied to the points in a coordinate system to redefine their coordinates. This is usually used to convert between coordinate systems or for rotation, translation, scaling, and so on.

translation
A transformation that changes the location of an object.

transparent
The property of a surface that allows light to pass through it.

Index

Symbols

A

B

C

D

E

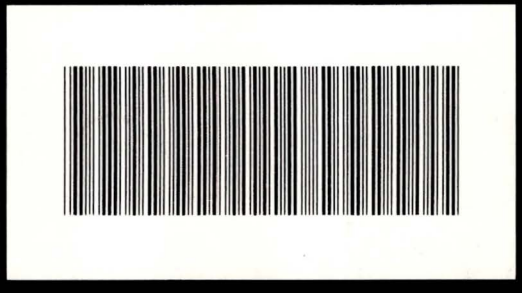
F

T

U

W

X



Order Number
DSW-306

Document Number
710-019530-002