19941025 (this version 19961029)
Bo Nordin (bosse@cb.uu.se)
Centre for Image Analysis
018-183470

Exercise 1: Visualization with AVS

Introduction to AVS

The **Application Visualization System** (*AVS*) is a powerful system for visualization of 2D and 3D data, originally developed by Stardent Computer, but now available on a wide range of computer platforms. *AVS* users can construct their own visualization applications, by combining software components, called *modules,* into executable *flow networks*. It is also possible for users to write their own modules and add them to the system (you won't be asked to do that, though). There are five major subsystems, each corresponding to one choice in *AVS*'s main menu:

Image Viewer: a tool for manipulating and viewing images.

Graph Viewer: a tool for creating 2D linear and contour graphs of data

Geometry Viewer: allows you to view and interact with geometrically-defined objects. Here you can change the objects themselves and the viewing parameters and also control the way in which the graphical images are rendered.

Network Editor: a visual programming interface for connecting computational modules together into networks.

Applications: produces another menu of choices. *AVS* comes with two prepared applications. Users can add their own applications to this menu.

In addition to the graphical user interfaces, most *AVS* modules can also be driven from a command language interpreter, the *AVS* **Command Language Interpreter** (**CLI**).

Overview of the Geometry Viewer

In this exercise you will use the **Geometry Viewer**, which is an *AVS* subsystem accessible from the main menu or from any other *AVS* subsystem through the **Data Viewers** pulldown menu. The purpose of the exercise is to familiarise you with some important concepts in computer graphics, notably scene transformations, 3D rendering methods, and colour manipulations.

You start *AVS* with the script `/opt/avs/bin/start_avs`. To simplify, you can copy it to your home directory. The script sets up required environment variables and starts *AVS*.

When you enter the **Geometry Viewer**, two X windows appear: the **Control Panel** with lots of buttons, etc. and a **view window**. The control panel can be further subdivided into header (with **Help**, **Data Viewers**, and **Exit** buttons), **Transform Selection** area, **Current Object Indicator**, and the **Menu Selection Area**. You may create several view windows.

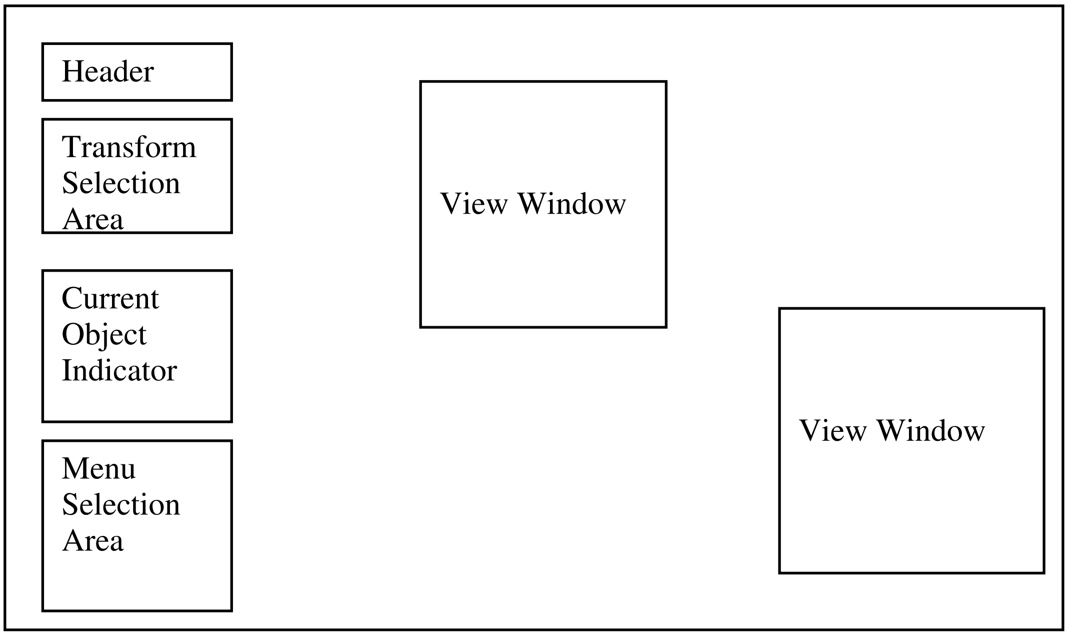

Figure 1: Overview of the geometry viewer screen

With the **Geometry Viewer** you can project an object from a 3D world, build complex objects from predefined objects and interactively make 3D scenes which are rendered by the viewer. Available tools for manipulation of the objects include are:

- adding and removing objects
- translation of objects
- rotation of objects
- scaling of objects
- changing the position of the clipping planes
- changing the position of the "camera"
- changing the illumination of the scene

The **Geometry Viewer** creates a *scene* within which you can view objects. You can create multiple scenes with the **Create Scene** function under the **Cameras** submenu. Each scene is made up of a **world space** measured in world co-ordinates, a collection of 3D **objects**, and a collection of **lights**. You view the scene in one or more **view windows** (aka **cameras**).

It is important to realise that there is a fundamental difference between the *scene* and the *image* (or display) of the scene. You change the scene through transformations of the objects, or you change the image through transformations of the viewing conditions.

The **Geometry Viewer** uses a model of a 3D world scene. Objects often consist of triangles since these are very easy to handle (as we shall see later in the course). Images shown by the viewer are 2D projections of a 3D world.

World co-ordinates in the **Geometry Viewer** are a set of right-handed X (thumb), Y (index finger), and Z (middle finger) axes. When starting the viewer, the origin (0,0,0) is positioned in the centre of the view window with the XY plane parallel to the screen face. Positive Y is straight up, positive X is to the right, and positive Z is coming straight out of the screen toward you.

In the **Transform Selection Area** you control which *transformable* (see below) is controlled by the mouse. In the menu selection area you invoke most of *AVS*'s functions. At the top of this area you choose which of five submenus is displayed in the area: **Objects**, **Lights**, **Cameras**, **Labels**, or **Action** (we won't be using Labels or Action in this exercise).

Reading in an Object

With the **Geometry Viewer**, you can manipulate and view one or more 3D objects of the fundamental *AVS* type *geometry*. A geometry is a collection of points in 3D space with some additional information. Geometries are stored in *.geom* files, created either by *AVS* itself during an earlier **Geometry Viewer** session, or by a program making calls to a special library. This may sound prohibitive, but *AVS* can read some external formats and transparently convert them into geometries.

The geometries can be pure geometries (*.geom* files), or included as part of higher abstractions called objects (*.obj* files), or as components of an entire 3D scene (*.scene* files). Sample geometries are found in the directory */opt/avs/data/geometry*. This path is applicable at the time of writing. Should the setup change, the file browser in *AVS* will change accordingly: it always starts in *AVS_PATH/data* (and AVS_PATH is set by the start script).

When the **Objects** menu selection is activated you read geometries into memory by clicking on **Read Object** which brings up the **File Browse**r. One peculiar feature of this browser (and also if the **Current Object Browser** which will be described later) is that if there are more files than there is room for in the display list, you move up in the list with the left mouse button but down with the middle button.

Manipulating an Object

You can move the object around (*translat*e, *rotate*, *scale*) with the mouse.

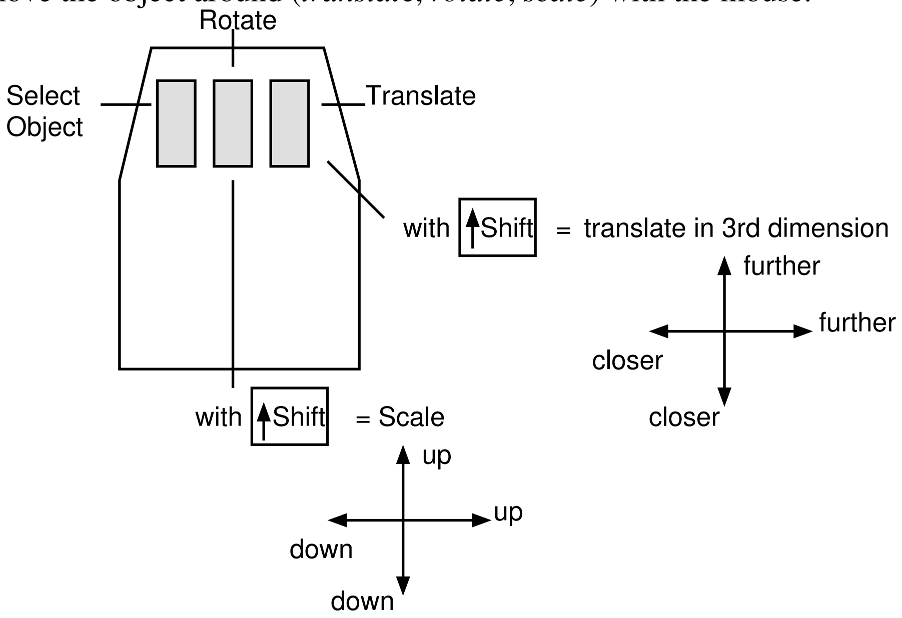

Figure 2: Functions of the Mouse Buttons

Selecting the **Bounding Box** option changes the way that object manipulation works. If this option is turned on, only a wireframe box moves as you move the mouse. It is useful when *AVS* runs on a less powerful workstation or when many objects are viewed. Note that, in computer graphics, Bounding Box in general also plays slightly different roles: that of limiting the area that must be processed and that of providing a simple initial test in hidden surface algorithms, etc.

Pressing the dimple to the right of **Transform Selection** brings up a **Transformation Options** window. In this window you can edit the values of various parameters. An oddity in this window is that you must press **Absolute** to update the values if you used the mouse or the **Reset** button to change the viewing parameters. And if **Absolute** is already active (white) you must press **Relative** followed by **Absolute**!

Note that the mouse and the **Transformation Options** are "attached" to the currently selected option in the Transform Selection part of the control panel. The mouse can be "in control of" of an object, a light, a camera, or a texture map.

It is probably best for all but seasoned *AVS* users to keep the current selection in **Transform Selection** and the **Menu Selection Area** synchronised (selecting **Transform Object** in **Transform Selection** when **Object** options are displayed in the **Menu Selection Area**, for instance).

Object hierarchy

AVS can handle multiple objects. Objects in *AVS* are organised in a hierarchy. At the top of the hierarchy is a root object named **Top**. All objects that you read into the **Geometry Viewer** are children of **Top** and normally peers of each other, but may also be children of other objects.

Each object has
- its own co-ordinate system
- extents in X, Y, and Z, a **Bounding Box**
- a centre about which the object will rotate
- a name
- optionally, surface properties as colour, transparency, reflectivity, etc.

You use the left mouse button to select object in the view window. Whenever you do anything with an object it is the *current object* you are manipulating. Actually, even when there is only one object, there are two slightly different objects: **Top** and the object itself. Because explicit settings (colour, etc.) at lower levels override setting at higher levels, you must make sure that your object (and not **Top**) is the current object if you want to change its colour. To make sure that this is the case, click on the object repeatedly with the left mouse button and watch the label above the **Current Object Indicator** (which shows the name of the current object). Each time you click you move to a different level in the scene's object hierarchy. Alternatively, you may click on the dimple to the right of the label and bring up the **Current Object Browser** in which you can select a current object simply by clicking on its name.

Transformables

A *transformable* in *AVS* parlance is anything which affects the way a scene is rendered; in other words it is things like objects, lights, and cameras - entities which may undergo what in computer graphics is called transformations. There are four transformables in the Geometry Viewer: objects, lights, cameras, and 2D texture maps. Each of these corresponds to one of the **Transform Selection** menu buttons. The buttons select which class of transformation is going to be affected by a mouse button or **Transformation Option**s panel selection: it is a transformation mode switch.

The exercise

Because this exercise is designed as if it were your first contact with an advanced (well, fairly advanced) visualization system, the instructions may be too detailed for the taste of some of you. With these cautious words as an introduction, I will now describe what is expected of you. Your answers to the questions after the instructions should be handed in before the written exam takes place. The order of the questions roughly follows the order of the instructions, but I think it's a good idea to browse through the instructions and the questions before actually starting *AVS*. If you have problems understanding how *AVS* works, try the **Help** button. Be patient, some operations may take some time; try not to queue up a lot of mouse clicks: mouse-ahead is an art that is considerably more difficult to master than type-ahead.

First start the **Geometry Viewer** and read a *geometry* (the teapot, for instance) into the work area. Familiarise yourself with the functions of the mouse buttons. Translate and rotate the object in all three dimensions. You can use the **Reset** button under the **Current Object Indicator** to reset all transformations to their defaults. If you think transformations are slow and hard to understand try the **Bounding Box** option. Also bring up the **Transformation Options** menu so you can make precise transformations. Select **Transform Camera** so you can manipulate the camera. Pressing the dimple to the right of **Transform Camera** brings up the **Camera Options** Menu.

Select **Cameras** in the **Menu Selection Area**. This brings up the **Cameras** submenu (for more about the meaning *From*, *At*, and *Up* vectors, see Foley *et al*, chapter 6.3). Select the option **Perspective** and play around with the various **Camera Options**: change the **Field of View** and turn **Perspective** and **Depth Cue** (see also Foley *et al*, 12.3.3) on and off, for instance. Note that you may have to move the cursor to the **Refresh** button in order to update the screen if you change the camera parameters by entering values in the various text fields. Hint: the effect of depth cueing is more obvious if several objects are viewed simultaneously.

The **Back Clip** and **Front Clip** are the distances from the two clipping planes from the camera position in the direction of (*From-At*). Change the various parameters that affect the clipping plane and make sure you understand how they interact. (see also Foley *et al*, Figure 6.16).

Select **Objects** in the Menu Selection Area. Close to the bottom you can select how the object is rendered using various shading models (see also Foley *et al*, chapter 14.2). Test the various rendering methods: **Lines**, **Points**, **Flat**, **Gouraud**, and possibly **Phong**.

Texture is very useful for creating virtual realism (see also Foley *et al*, pages 429 and 432, and colour plate 35). Select **Edit Texture** and try the textures in the directory */opt/avs/data/image* (the warning about taking this too literally still applies, as does the reassuring comments about the, admittedly artificial but nevertheless viable, intelligence of the file browser).

Select **Edit Property**. This functions allows you to change the reflectance properties of the current object. Intuitively speaking, you can change the material from which the object is constructed. In order to get a clearer understanding of what you are doing, it is perhaps better if you delete any texture map you may have attached to the object. Remember that you are changing the properties of the object, not the viewing conditions. Play around with the colour sliders (see also Foley *et al*, chapter 11.3).
Amb(ient Light Reflectance) is the proportion of ambient light that the object reflects.
Diff(use Light Reflectance) is the proportion of non-ambient light that the object reflects equally in all directions.
Spec(ular Highlight Intensity), Gloss, and Metal are created when the direction of incoming light is sufficiently close to the viewing direction. The intensity (Spec) determines the brightness of such highlights, **Gloss** determines what sufficiently close is, and **Meta**l specifies the colour of the specular highlight.
Trans(parency) controls the degree to which you can see through the object.
Try all these options.

Now, without closing the **Edit Property** window, select the **Lights** submenu (for more about illumination models, see Foley *et al*, chapter 14.1). The grid of numbers is a lighting panel. You can define up to 15 lights (or 7, perhaps, depending on your system). In addition, you can specify the ambient light (**AM**). Originally two lights are on: ambient light (which is non-directional) and directional light number 1. The direction of this light is parallel to your line of sight. You select a current light by pressing the appropriate number (or AM). The currently selected light may then be turned off or on and manipulated in other ways. You can change the direction of a light source (not AM, of course) by selecting **Transform Light** and using the mouse or the **Transformation Options** as usual. If you select **Show Lights** the directional lights are shown as vectors (this also happens automatically while you are rotating lights with the mouse). Play around with **Edit Property** and the light sources and make sure you understand the relation between the properties of the object and how different lights are reflected in it. If you have the time you may also try bi-directional lights, spot lights, and point lights (if they are available).

Read in one or more additional objects. Move the objects around both one at a time and all objects simultaneously.

Last, remember that although you are required to answer the questions below, the most important goal of the exercise is to get a feeling about what non-trivial computer graphics is about and what it can do for you. You are encouraged to try other *AVS* subsystems than the **Geometry Viewer** if you like (and have the time).

Questions to be answered

Q1:
When, without changing default options, you translate the object in Z, why doesn't it change size as naively expected? Which transformable controls this?

Q2:
Describe (preferably with a drawing) what the world coordinate system looks like initially, i.e. when the first object you read is first displayed. If you rotate the object 90 degrees around the Y-axis, how must you change the position in order to obtain the original view of the object?

Q3:
Explain what *From*, *At*, and *Up* in the **Camera Option** panel mean and how they relate to each other. A figure probably explains it better than even carefully chosen words. What should the relation between the (*At-From*) and *Up* vectors be in order to reflect a real-life situation?

Q4:
Why does the object change size when you change the **Field of View**? How is this related to the **Perspective** option.

Q5:
When you change the clipping planes so that the distance from either clipping plane to the camera position is close to zero, what happens? Why?

Q6:
What is depth cueing? Explain what the depth-cueing parameters mean.

Q7:
What is the difference between **Flat** and **Gouraud** shading? Why is there a distinct pattern when **Flat** shading is used? What does this tell you about how the object is modelled?

Q8:
In your own words, what does H (hue), Saturation (S) and Value (V) mean?

Q9:
What happens to the image of the object(s) if you turn off all lights except the ambient light? Why? What kind of light is ambient light? What happens if you turn off the ambient light? Why?

Q10:
In computer graphics, what is the difference between a directional light source and a point source? Can you give examples of real light sources that are modelled by the two?

Q11:
What happens if you set the **Amb**ient Light Reflectance of an object to zero and use only Ambient light in the scene? Why?

Q12:
What happens if you set both the **Amb**ient and **Dif**fuse Light Reflectances of an object to zero but the Specular reflection to a value above zero? Why?

Q13:
What happens if you move (with the right mouse button) a directional light source? Explain!

Q14:
When you read several objects to the same scene where does the Geometry Viewer place them and how does this affect the scene in terms of realistic rendering?

Q15:
This question concerns *AVS*'s user interface. What different basic user interaction tasks (see chapter 8.2 in Foley *et al* for a definition of basic user interaction) are there in *AVS* (or rather in the subsystems you have used)? Give examples. Do you think the feedback from the system is adequate? If not, point out a specific problem and sketch a solution.

Q16:
Please give any additional comments (on *AVS* or on the exercise) you may have. Constructive criticism is greatly appreciated; *AVS* is only one of several viaualization systems that could be used for this introductory exercise in the course.

Availability

The Department of Scientific Computing owns one (1!) license for *AVS*. This means that only one user can run *AVS* simultaneously. However, during the course of the course, a demo license will be installed for about a month. During that time an unlimited number of users will be permitted to run *AVS*. You will be informed about exactly when the demo license will be in effect at the lectures, by e-mail, and on the homepage of the course.