11

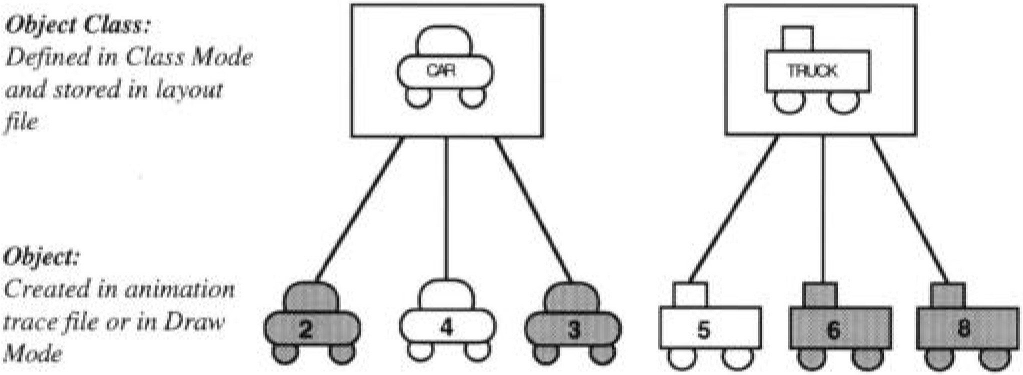
Chapter Eleven
DEFINING OBJECT CLASSES

Once you are reasonably comfortable with using Draw Mode (Chapter 10), you can easily move into defining Object Classes.

The Role of Object Classes

When you create an Object in the animation trace file you must specify its *Object Class*. This establishes the shape and other initial properties of the Object.

If there are many similar Objects in your animation, you may be able to rely on a surprisingly small number of Object Classes. An Object's properties are copied from the "parent" Object Class when the Object is created, but any Class property can be overridden for any given Object.



Object Class: *Defined in Class Mode and stored in layout file*

Object: *Created in animation trace file or in Draw Mode*

Figure 11-1. The relationship between Object Classes and Objects

Class definition is an essential step for virtually all animations. Until you have built up a collection of predefined Classes, you will define Classes with each animation you build.

(Although it may be possible to keep a "standard" set of Object Classes on hand for moving Objects in different animations, you still might need special Classes for other parts of each animation, such as resources that are subject to visible state changes.)

Defining a Class is similar to drawing a background, with the following exceptions:

- Classes tend to be smaller than the background drawing
- Classes have a "hot point," "rear guide point" and other properties
- Classes cannot contain Plots or Bars

*Class definitions are stored in the layout file. Like background elements, Class definitions are not permanently saved until you select **Save Layout** from the **File** menu.*

Using Class Mode

Class definition takes place in **Class Mode**. You can access Class Mode by selecting the **Class** option in the **Mode** menu.

Once you have entered Class Mode, you will notice two differences from Draw Mode. First, the viewing scale is initially zoomed in 10 times closer than the default grid in Draw Mode, and the point (0, 0) is at the center of the screen.

Second, the Class Mode toolbar includes a few differences from the Draw Mode toolbar:

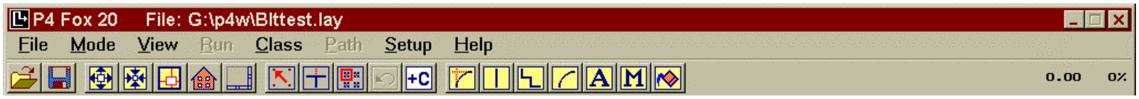


Figure 11-2. The Class Mode Toolbar

The Snap, Trim, Box Edit, and Undo tools are present on the toolbar and identical in function to those in Draw Mode. The Fillet, Line, Polyline, Arc, Text, Message, and Fill tools are likewise present and similar in function to those in Draw Mode. Since a Class definition cannot contain Bars, Plots or Objects, these tools found in Draw Mode are not available in the Class Mode toolbar. The Class Mode toolbar contains the +C New Class tool for adding new Classes.

Creating a New Class

To define a new Class, click on the +C **New Class** tool, or choose **Class**, **New**. Proof keeps track of Classes by name. Therefore, the first thing Proof will do when you request creation of a new Class will be to prompt you to enter a name for the Class. Class names are case sensitive and are of unrestricted length. "FULLAGV" is a different name from "FullAGV."

When you finish defining the shape and properties of the new Class, you can proceed directly to another action. Your Class is automatically incorporated into your layout. However, as mentioned above, new Classes are not saved permanently to disk until you save the current Layout.

How Classes are Saved

Class definitions are part of the layout (even though they are invisible from Draw Mode). Class additions, changes, or deletions are made immediately to the in-memory copy of the layout. There is no "Save Class" prompt.

However, these layout changes, like all layout changes, are saved to disk only when you save the layout. You can save the layout at any time by selecting **Save Layout** from the **File** menu.

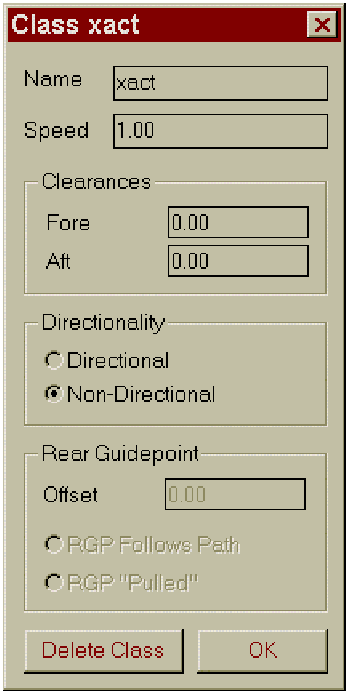
Editing an Existing Class

To select an existing Class for editing and/or examination, choose **Class**, **Select.** A list showing all existing Classes in the current Layout will appear. You can click on the Class that you would like to examine or edit, and it will become the current Class. If you have many Classes and you know the name of the Class you want to select, you may find it easier to type the name of the Class, rather than scrolling through a very long list of Class names.

When you are finished examining or editing a Class, you can proceed directly to any other action. Your changes to a Class are immediately incorporated (though not yet saved to disk). If you change to another Mode and then change back, the current Class is still active.

The Class Definition Dialog Box

When you create a new Class or select an existing Class, Class definition dialog box appears:



Class xact

Name: xact

Speed: 1.00

Clearances

Fore: 0.00

Aft: 0.00

Directionality

○ Directional

● Non-Directional

Rear Guidepoint

Offset: 0.00

○ RGP Follows Path

○ RGP "Pulled"

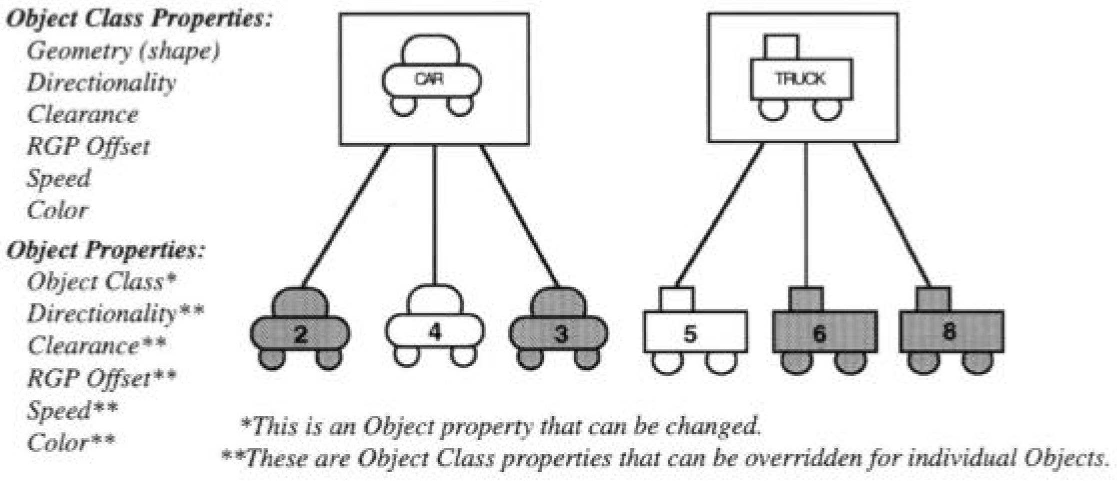
Deleting a Class

Clicking the Delete Class button removes the current Class from the layout. You are prompted for verification. There is no way to “undo” a Class deletion unless you re-open the previously saved copy of your layout without saving your most recent work.Rename a Class

The name of the active Class is displayed in the dialog box’s **Name** control. To rename a class, simply select this control and edit the current name or type in a new name. The new name will be permanent if you save your layout file.

Specifying Class Properties

The Class dialog box allows you to specify a Class’s properties at any time, i.e., before, during, or after you draw the Class’s geometry. We’ll discuss these properties in sections that follow.



*This is an Object property that can be changed.
**These are Object Class properties that can be overridden for individual Objects.

Figure 11-2. Object Class properties and Object properties

Hot Point

The hot point of an Object Class is the (0, 0) point in the Class Mode coordinate space. Proof considers Objects created from the Class to be located at the hot point. It is the point at which an Object stays attached to its Path and also the point about which the Object may rotate.

Suppose you've misdrawn an Object Class so the desired Hotpoint rests at a location other than the (0,0) point of the Class Mode coordinate space. You can correct this condition using **Box Edit** to capture and move the entire geometry of the Class so that the desired Hotpoint does rest at (0, 0).Note that the hot point cannot be set directly from the Class dialog box.

Directionality and Orientation

An Object Class may be defined as *directional* or as *non-directional*. If you define a Class to be directional, Objects from that Class will always turn to "point" in the direction they are moving. With non-directional Classes, Objects will only "slide" around on the screen in one orientation.

You can change the directionality of an individual Object from within a trace file by using set object…directional or set object…nondirectional (see Chapter 14). This is helpful if you have a moving Object that sometimes rotates and sometimes slides.

For a directional Object, Proof needs to know which direction points "forward."

The forward "orientation" of a Class is assumed to be along the positive x axis of the Class definition, i.e., pointing to the right in a non-rotated view.

In a sense, orientation is another Class Property. You can use **Box Edit** to rotate the geometry of a Class so that it "points" in the desired direction. (Remember to check and adjust the hot point, if needed, after rotating the geometry.)

Clearance

When Objects line up consecutively on an accumulating Path, Proof can keep them from bumping into one another based on defined *clearance* values. An object's initial clearance values are copied from its Class when the Object is created. Clearance values for an individual Object can be changed from within a trace stream by using a set object…clearance command (see Chapter 14).

Clearance is specified in linear units measured along the Path occupied by Objects from the Class. There are two values: *fore* clearance and *aft* clearance.

The default values of fore and aft clearances are zero. If you do not specify at least one of these values to be non-zero, Objects will overlap when they reach the end of accumulating Paths.

The hot point of a trailing Object stops *c* units (measured along the Path) behind the hot point of the Object in front of it. The value of *c* is the fore clearance of the trailing Object plus the aft clearance of the leading Object. When a Path contains Objects that have the same Clearances, it makes no difference how much of the total clearance is defined as "aft" and how much of it is defined as "fore," as long as the total clearance is comfortably longer than the Object. In Figure 11-3, for example, it won't matter whether you specify "CLEARANCE 7 0" or "CLEARANCE 3 4," as long as the total is seven.

The only occasion when you will need to be concerned about how the total clearance is allocated between fore and aft is when you are accumulating dissimilar Objects.

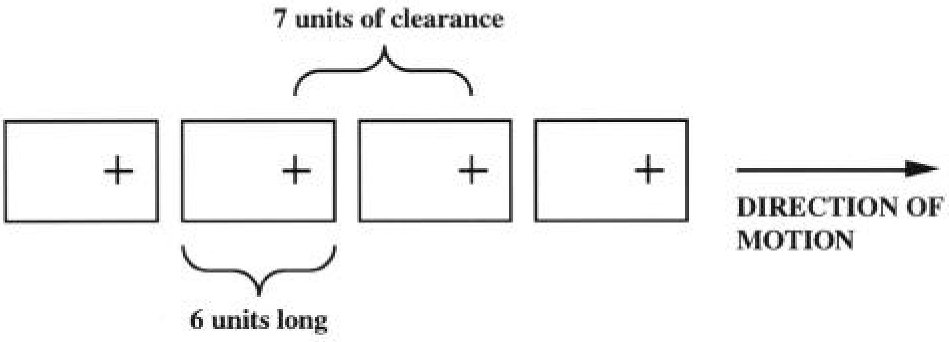


Figure 11-3. The total clearance (fore + aft) is large enough to include the length of the Object

> *When the <u>leading</u> Object reaches the end of an accumulating Path, no clearance is applied. The hot point of the Object rests on the very end of the Path.*

RGP Offset

The **RGP Offset** field in the Class dialog allows you to specify an optional *Rear Guide Point* (RGP) for the current Class. If a non-zero RGP value is specified, then the RGP acts as a secondary hot point for directional Objects moving on Paths. RGP offsets must be specified as negative numbers, considering that the RGP point is always behind the hot point.

The effect of specifying a non-zero RGP depends on the Class's RGP mode. The default RGP mode is "RGP Follows Path." In this mode, an RGP acts as a second point of attachment when directional Objects move along a Path. This mode of operation is depicted in Figure 11-5.

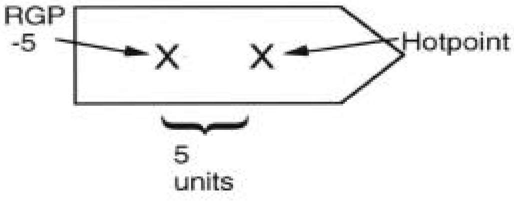


Figure 11-4. The RGP is set to -5 which is 5 units.

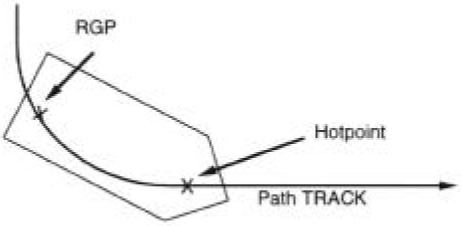


Figure 11-5. A sample of the usefulness of RGP on long Objects rounding curves.

An RGP can also operate in “RGP Pulled” mode. In this mode, a Class’s hot point is treated as the front axle of a wheeled vehicle, and the RGP is treated as the rear axle of the vehicle. When a wheeled vehicle such as an automobile turns, its rear wheels trace a different (shorter) route from that of its front wheels. When a car drives in a circle, its front and rear wheels trace out two concentric circles, with the rear wheels tracing the smaller of the two circles. The rear wheels of a car trace the same route as the front wheels only when the car is moving in a straight line and has been moving in the same direction for some time.

For Objects operating in RGP Pulled mode, Proof performs computations that approximate “correct” motion fairly well. (100% correct computations would require too much computer time.)

An Object’s RGP offset can be changed from within a trace file by using set object…rgp, and an Object’s RGP mode can be set by using set object…RGP pulled or set object…RGP follows. These commands are described in Chapter 14.

Speed

Generally, Objects moving on Paths will travel at the Path speed. However, you can also assign a Speed to a single Object or to an Object Class. Class Speed will be useful only in those circumstances in which the speeds of dissimilar Objects on a Path depend on the Classes of Objects. One example might be empty vs. full containers moving on certain types of non-powered conveyors. Another example might be vehicles moving on a highway. See Chapter 14 for more on the different ways of specifying Speed.

Color

Color is a Class property, although it does not appear explicitly in the Class properties dialog box. Classes can be multi-colored. Each Object Class can have any combination of foreground and Layout colors. You can add detail by leaving transparent “windows” in your Class shape that show the Backdrop color. For example, you can define a Class shape to consist of two concentric circles and fill the area between the circles. This would look like a doughnut, with the Backdrop color (or anything in any of the Layout colors that is behind the Object) showing through the center.

You can select or change the Color of any element of the Class at any time by selecting the element, clicking the **Color** button in the element’s dialog box, and selecting a color from the color palette that appears. To change the Color of *all* elements of a Class to a single color, use **Box Edit**. Draw a box around the entire Class and then click on the desired color in the Color Palette.

Class Properties Illustrated

Figure 11-6 is an annotated picture of a sample Class, showing the relationship between the Class geometry, the hot point, the forward orientation (relevant only for directional Classes), and the clearances. Note that the total clearance, 6.4, minus the length of the Object in the x-direction, 4.8, equals the amount of space that will be left between similar Objects (of this Class) when they accumulate on an accumulating Path.

Figures 11-7 and 11-8 reflect the differences between directional and non-directional Classes when they are moving upward and to the right at a 45° angle. Notice that the clearances are measured in the direction of motion in either case.

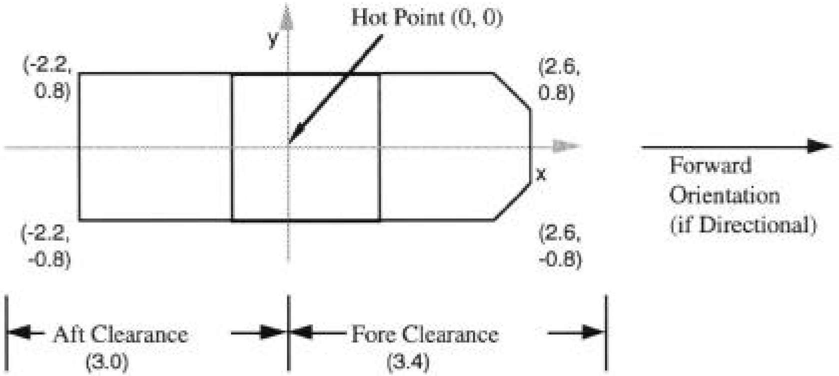


Figure 11-6. A sample Class with several properties illustrated

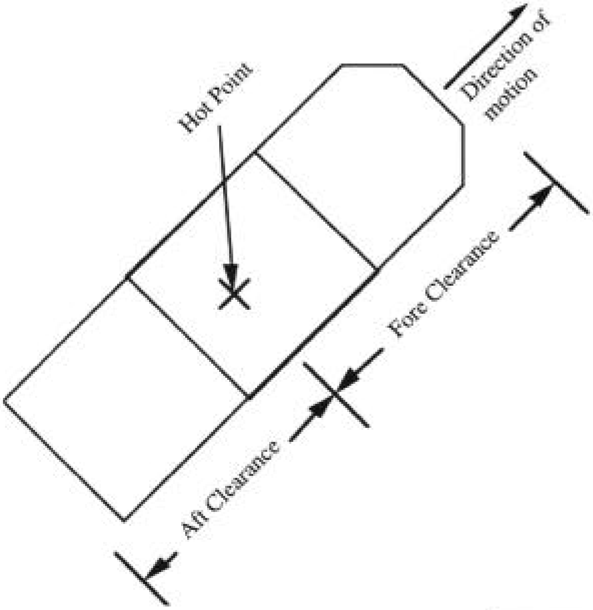


Figure 11-7. Directional Class

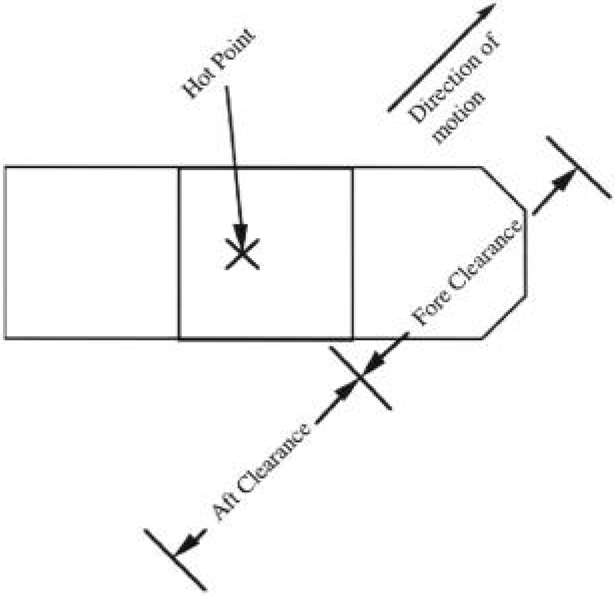


Figure 11-8. Non-directional Class

Drawing an Object Class

The tools for drawing elements of a Class are on the Class Mode toolbar. These drawing tools can be used at any time. **Line**, **Arc**, **Text**, **Polyline**, **Fillet**, **Message** and **Fill** tools are available.

When you have completed drawing a Class's geometry, you can go back and change properties shown in the Class dialog box, or you can move on to another Class or Mode. Your Class changes are always kept current in memory. *Remember that you must save the layout file in order to make Class changes permanent.*

Exercise 11-1: Defining and Using Object Classes

Let's try out Class Mode. Open apath.lay, located in the exercise folder.

Choose **Mode, Class** and define the following Classes. Remember that Class names, like all user-defined identifiers in Proof, are case-sensitive.

Arrow – The shape shown in Figure 11-9 (including the crosshair, but not the numbers), colored F6

DirArrow – The same as Arrow, with "directionality" turned on and with clearances of your choice; colored F7

Comet – The same as DirArrow, except with the hot point at the end of the Arrow and with altered clearance values; colored F3

Hint: You can use **Box Edit** to copy geometry information from one Class to another; to copy geometry information between Draw and Class modes (which is very useful, for example, when working with imported geometry); and also to drag an entire shape so that the hot point changes.

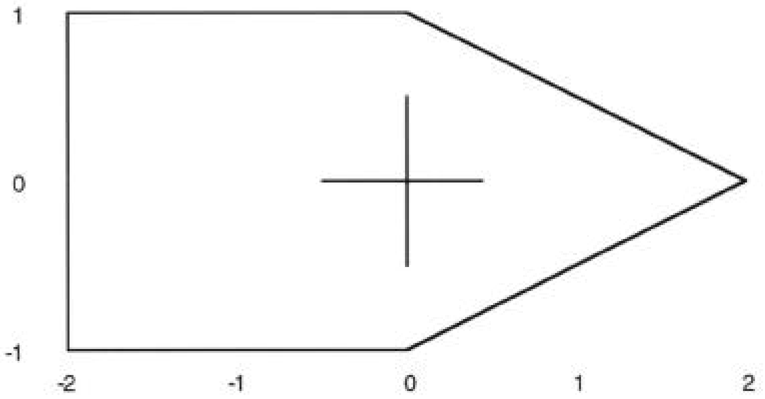


Figure 11-9. Use this illustration as a guideline to define the Classes in this Exercise. Note the x, y coordinates – the hot point will be at the center of the plus.

Save the modified layout as myclass.lay.

Using your favorite program for editing text files, open the trace file mypath.atf (from Exercise 7-2) Make your own copy by saving this animation trace file as myclass.atf. (If you have not done Exercise 7-2, you can start with a completed version of mypath.atf from the sample folder, and save that as myclass.atf in the exercise folder.)

You are going to modify myclass.atf so that it uses the Classes you just defined. Using your text editing or word processing program, insert trace commands that cause Objects of each Class to be created and placed on TheAccPath. Save the myclass.atf trace file. Return to Proof and look at the animation by running myclass.lay with myclass.atf. See how the Objects look. Be sure you observe all three Classes going around corners, and take a close look at how the Objects with clearances interact when they accumulate.

Tips on Defining Classes

Here are some tips to consider when you are defining Object Classes:

- Keep Classes small and simple if they are going to be used for moving Objects. Extra detail sometimes looks worse than no detail at typical viewing scales.
- Understand how Fills work (Chapter 10). (The **Fill** tool operates as a “flood fill” from a point, and may not work well when complex Objects appear at small scales.)
- Consider using multiple Classes to represent different states of the same Object (see set class in Chapter 14).

- Remember to define clearances and directionality for Classes that will be used by Objects traveling on accumulating Paths.
- Remember to "point" the Class in the direction of the positive x axis while drawing it.
- Consider carefully the best location for the hot point – and the optionally separate Rear Guide Point if the Class has an oblong shape. Remember the Class named Comet from Exercise 11-1.

Exercise 11-2: Thinking About Directionality and Clearances

This is a look-and-think exercise.

From Proof, open and run hanoi.lay + hanoi.atf, located in the exercise folder.

An older version of this animation used sliding (non-directional) disks that rose off their pegs, slid sideways along a curved Path, then settled onto another peg. In Class Mode the disks looked like horizontal bars.

In the version you are viewing the disks flip over like hamburgers or pancakes. In order to successfully change the disks to directional, the developer of this animation had to change the geometric description of the disk Classes. What do you think the Classes look like now? Stop the animation, go into Class Mode, and take a look at the DISK4 Class.

Look at the clearances for DISK4 also. All the other disks have the same clearances as DISK4. With the disks doing a half-rotation each time they transfer from one peg to the next, why doesn't this animation look very strange (i.e., why don't the disks show unequal spacing when stacked)?

Turn the Grid on and look at the disks on the bottom of the piles. Now look closely (slowly) at what happens as a top disk begins to leave its peg.

What could we do to cause the disks to have their "true" side up (alternating sides with each move)? What would happen to the animation if we did that? What could we then do to correct the new problem? You might want to revisit these questions after covering the set object…nondirectional command in Chapter 14.