14

Chapter Fourteen ADVANCED OBJECT MANIPULATION

This chapter describes advanced commands for manipulating Objects from a trace stream. It also covers the creation and positioning of "layout Objects" from Draw Mode.

Topics covered include the following:

- set object class
- rotate
- Layout Objects and the place in command
- Commands that vary the speeds of Objects, including set object speed, set object travel, and set path speed
- Several motion-related commands, including place on, place at, set object directional, set object nondirectional, and set object clearance
- attach and detach

There are five kinds of set commands. They are set object, set bar, set class, set path, and set view.

Some set object commands, such as set object color and set object class, are used frequently. The object keyword is optional for all set object commands. In the examples in this book we often omit the optional "object." So, for example, you may see set color instead of set [object] color.

Using "Set Object…Class" to Change an Object's Class

The set object…class trace stream command is one you might use often. This command selects a different Object Class to provide the underlying properties of an Object, *after* the

Object has been created, even if the Object is currently on-screen.

All of the Object properties are changed as a result of set object…class. In practice, the most important change is a new shape for the Object. When you issue a set object…class command that refers to a Class with a different shape, the shape of the Object will instantly change on the screen. Other properties, such as directionality, clearance, color, and class speed, will change too. The syntax is:

set [**object**] *ObjectID* **class** *classname*

For example,

```
set agv1 class emptyAGV
```

Changing the Object to the default colors of the new Class can be an unwanted effect of set object…class. Issue a set object…color command after the set object…class command if this is a problem.

You might like to take another look at the kanban animation (or refer to Figure 14-1). The guided vehicles (AGVs) change shape when they load and unload: first empty, then loaded with a full bin, then loaded with an empty bin, then empty again. This is a perfect application for set object…class.

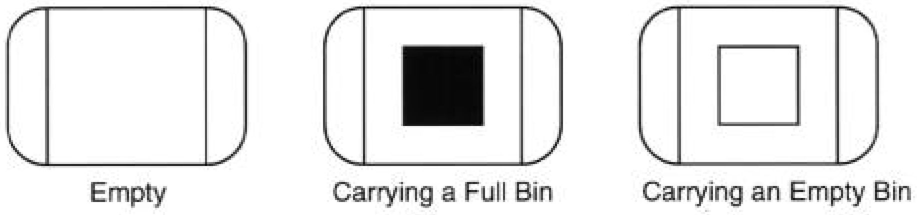


Figure 14-1. Shapes used with Set Class *in the* kanban *model*

Using "Set Class" to Change a Class Property

The set class trace stream command is used to change the properties of an Object Class definition while the animation is running. In this command, no Object ID is referenced, because the command applies to the Class definition as a whole. Existing Objects that are members of the class are unaffected by the changes. *New* Objects created after the set class commands have been executed will have the new properties.

Class properties that can be changed using the set class command are speed, directionality, clearances, and rear guide point offset.

The syntax is:

set class *classname* **speed** *s*

set class *classname* **directional | nondirectional**

set class *classname* **clearance** *fore aft*

set class *classname* **RGP** *offset*

set class *classname* **RGP pulled**

set class *classname* **RGP follows**

For example:

set class truck speed 3.4

set class Gurney directional

set class CPUbit clearance 3 2

set class AGV RGP –2

set class Automobile RGP pulled

In each of the examples, the class properties have been changed. Individual Object properties can be changed using the set [object] trace stream commands covered later in this chapter.

These changes will be in effect for objects *subsequently* created using these classes. Currently existing (previously created) Objects are not affected by set class commands.

Rotating Objects

The rotate command allows you to rotate an Object around its Hot Point. Do not confuse rotate, the trace stream command, with **Rotation**, a control found in the Active Window Properties dialog box. The rotate command rotates an Object with respect to the layout, while **Rotation** rotates the entire layout with respect to the eyes of viewers of the animation.

The syntax for rotate is:

rotate *ObjectID* [**to**] *angle* [**time** *duration*] [**step** *stepsize*]

rotate *ObjectID* [[**to**] *angle*] [**speed** *rotspeed*] [**step** *stepsize*]

The *angle* and *stepsize* are in degrees, the *duration* is in Proof Animation time units, and the *rotspeed* is in degrees per time unit.

A positive *angle* is counter-clockwise and a negative *angle* is clockwise. The sign of *rotspeed* is ignored, unless *angle* is omitted. The sign of *stepsize* is always ignored.

The Three Uses of Rotate

The *angle* is the most important parameter. Given an *angle* of rotation and a meaningful *duration* or *rotspeed*, Proof Animation will rotate the Object at a constant rate (in degrees per time unit) so that it ends up at the desired orientation.

If you specify an *angle* but no *duration* and no *rotspeed*, then the rotation happens instantaneously. In this case the Object will instantaneously appear at a different orientation.

If you do not specify an *angle*, then you must specify a *rotspeed*. In this case the Object will rotate indefinitely at that speed until it is subjected to another rotate command or destroyed. Using rotate with a *rotspeed* of 0 will stop the rotation.

Independence of Rotate From Other Object Manipulation

Rotation using rotate is independent of all other Object activity. For example, you can change the color or Class of a rotating Object, move it with move, or even place it on a Path.

Controlling the Smoothness of the Rotational Motion

Smoothness of rotational motion is controlled by the *stepsize*. By default, Proof Animation rotates in 30-degree increments. If this is too coarse, you can override it. However, be aware that Proof Animation must store a bit map for every directional orientation used. It's easy to use very large amounts of bit map memory space by specifying, say, a one degree *stepsize*.

The rotational orientations of Objects are subject to a master setting for angular resolution. You can specify angular resolution by choosing **Setup**, **Angular Resolution**, and entering a new value. Any value you set will be stored as part of the layout file.

Note that the step size is only important if the rotational speed is such that the person viewing the animation at a typical viewing speed will notice discrete rotational jumps. For example, if the animation speed were 70 time units per viewing second, then a rotate command with time 1 specified will always be completely finished in a single frame refresh (1/70 of a second will always be less than or equal to the time between frame refreshes), making the step size irrelevant.

Relative vs. Absolute Rotation

The *angle* can represent either an absolute angle or a relative angle. If "to" is specified before the *angle*, the Object will rotate from its current position to the absolute rotational angle designated.

With absolute rotation, the difference between the "to" angle and the current angle determines the direction of the rotation. For example, an initial rotate to 90 will be counter-clockwise (90 minus 0 is positive), but if the next rotate applied to the same Object is rotate to 45, the rotation will be clockwise (45 minus 90 is negative).

Figure 14-2 illustrates absolute rotation.

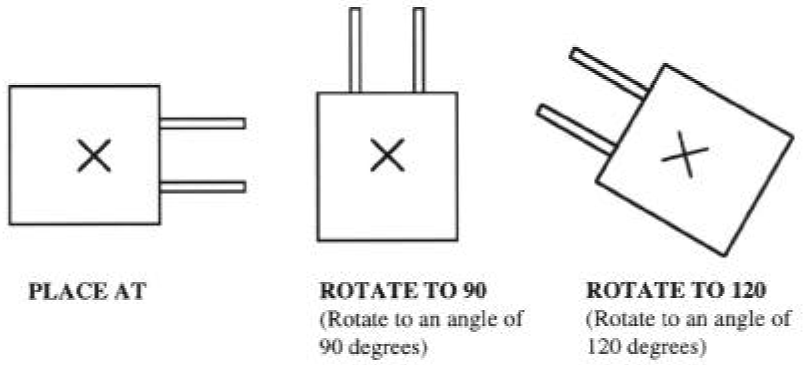

PLACE AT
ROTATE TO 90
(Rotate to an angle of 90 degrees)
ROTATE TO 120
(Rotate to an angle of 120 degrees)

Figure 14-2. Absolute rotation of a forklift

If you omit the "to" the Object will rotate by *angle* degrees relative to its current position. Figure 14-3 illustrates relative rotation as a result of two successive rotate commands.

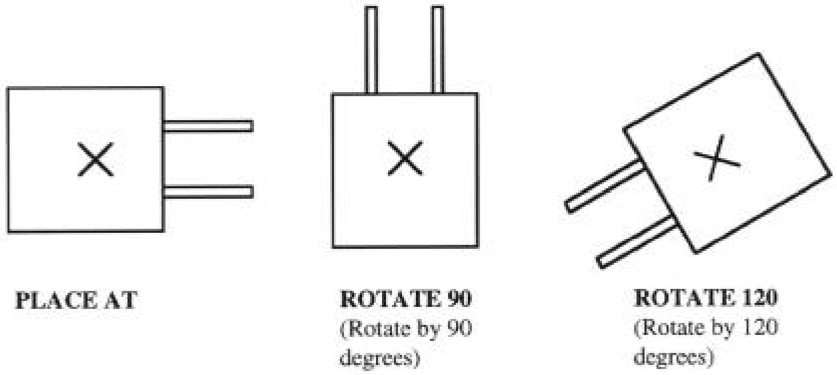


Figure 14-3. Relative rotation of a forklift

Examples of Rotate

Here are some examples of the Rotate command.

```
rotate 1 to 90
```

The above command rotates Object 1 so that its rotational angle is 90 degrees counterclockwise from “unrotated.” (See Figure 14-2.)

```
rotate 1 120
```

The above command rotates Object 1 by 120 degrees counterclockwise. (See Figure 14-3.)

```
rotate 1 -120
```

The above command rotates Object 1 by 120 degrees clockwise.

```
rotate 1 speed -90
```

The above command causes Object 1 to begin spinning continually clockwise about its Hot Point at a rate equivalent to one complete revolution (360 degrees) per four Proof Animation time units. Issuing another rotate command with an angle will stop the continuous rotation.

```
rotate 1 90 time10
```

```
rotate 1 90 speed 9
```

The two equivalent variations shown above cause Object 1 to rotate in three steps over 10 Proof Animation time units, ending up rotated 90 degrees counterclockwise from its current orientation.

```
rotate 1 90 time 10 step 5
```

The above example is similar to the previous example, except that Object 1 will make 18 steps instead of three to rotate 90 relative degrees (the default step size is 30 degrees, and we are overriding that value with five degrees).

A Note on Sources of Rotation

There are two ways an Object can change to a different angle using trace commands. One is through the rotate command — we'll call this the "rotational angle." The other is through movement of a Directional Object using move or place on — we'll call this the "directional angle."

The rotational angle is independent of the directional angle. An Object's visible orientation is determined by the *sum* of these two angles.

The rotational angle maintained by Proof Animation for an Object that has undergone directional motion can be different from the visible orientation of the Object on the screen. For example, when you rotate such an Object to an *angle*, this absolute angle is absolute with respect to "zero rotational angle" but not necessarily absolute with respect to the positive x-axis of the coordinate system.

Rotation can occur simultaneously with directional motion.

Exercise 14-1: Rotating an Object

Please run the rotate1 animation, located in the exercise folder. This animation illustrates a number of rotate commands.

Using a text editor, edit the file rotate1.atf and experiment with the rotate command. Change Messages using write commands so that you will be able to see which rotate command is currently executing. Add the following types of rotate commands (note that what is the name of a message that has been defined in the rotate1.lay Layout):

```
time 67
write what rotate 1 speed 20 step 5
rotate 1 speed 20 step 5
time 80
write what rotate 1 to -75 speed 50
rotate 1 to -75 speed 50
time 90
write what rotate 1 speed 40
rotate 1 speed 40
time 125
```

After you've made some experimental changes, rerun the rotate1 animation.

To see another interesting example of the rotate command, run the rotate5 animation, located in the exercise folder.

About Layout Objects

Previous chapters covered Objects that are created using the create command in an animation trace stream. Layout Objects are created and positioned in Draw Mode. Layout Objects are saved as part of the Layout file. A layout Object has the same properties and capabilities as an Object that is created in a trace stream.

There are at least four reasons why you might want to use layout Objects. First, some animations contain large numbers of Objects that are part of the animation from Time 0. In some cases it may be easier to define layout Objects than to cause the necessary create and place commands to appear in the trace stream.

Second, some Objects are naturally stationary. You want them to appear at a particular place, and stay there. An example would be a fixed-location resource that changes color. (If it didn't change color, you could just draw it as part of the background and not make it an Object at all.) Another good use of layout Objects is to repeat identical elements of the background based on a template, even if you don't need to manipulate them in the trace stream. (Users with CAD experience may recognize this usage as similar to a Block in AutoCAD.) It is often easier to position fixed-location Objects graphically (using layout Objects in Draw Mode) than it is to "hard-wire" the (x,y) coordinates of the Objects into the model or program that creates the trace stream.

Third, Layout Objects provide a convenient way for pre-positioning Objects that are present at fixed locations at the beginning of an animation, but will eventually move.

Finally, it is sometimes useful to define a position on the screen that can be referenced by name. Layout Objects can be used as "position markers" for use in conjunction with the place in trace stream command, covered later in this chapter. A layout Object used this way can be visible, such as a machine into which parts are placed, or invisible. An invisible layout Object can be thought of as a "named point" at which other Objects may be placed using place in.

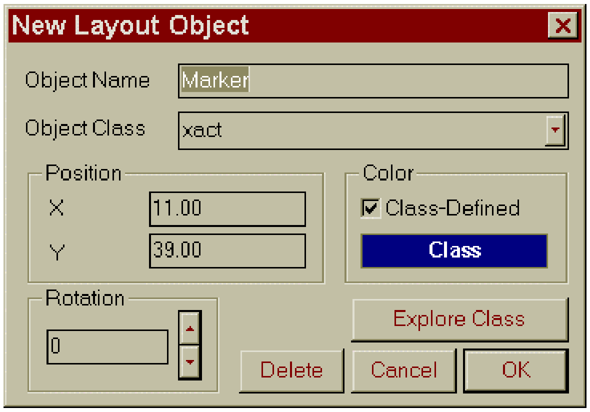
Creating and Editing a Layout Object

Like an Object created in an animation trace stream, a layout Object is defined as an instance of a previously defined Object Class. For this reason, the **Add Layout Object** button is dimmed (unusable) if no Classes have been defined.

We use the term "Creating" in the title of this section because while the animation is running, an Object that was defined in the layout file behaves just like an Object that is created from the trace stream.

To create a layout Object based on an existing Object Class, click on Draw Mode's **Add Layout Object** button. Proof will immediately prompt you to specify a name for the layout Object. The case-sensitive name must be unique and start with an alphabetic character. Names are of unrestricted length. Although a trace-created Object can have either a numeric identifier or a name, a layout Object must have a name.

After you have specified a name for the layout Object, a dialog box similar to the following will appear:



At this point, a Class name appears in the dialog box’s **Object Class** control, and an Object from that Class appears on the screen with a red “X” over its Hot Point. The Class will be the one most recently used in this session to create a layout Object or, if this is the first one created, it will be an arbitrary choice from among the defined Classes. If this is not the Class you wanted for your layout Object, click on downward arrow at the right end of the **Object Class** control, and an alphabetical list of all the Classes will appear. You can select a Class from the list or type in a Class name. When you have selected the desired Class, the Class name changes in the menu and the layout Object takes the shape and color of the newly chosen Class.

You can now position the layout Object by moving the mouse. The layout Object will follow the mouse, and the **X** and **Y** controls will be updated in the **Position** box, reflecting the layout Object’s position at all times. Once you are satisfied with the layout Object’s position, click the mouse to “nail down” the layout Object.

If you would like to initialize the Object’s rotational angle, type a value into the **Rotation** control. A positive angle will cause a counterclockwise rotation and a negative angle will cause the Object to be rotated clockwise. To quickly change the rotation in 10-degree increments, click on the **Rotation** control’s up- or down-arrowsTo change the color of the currently active layout Object, click on the **Color** button and select a color from the color palette that appears. If you would like the Object to have the color of the Class as it appears in Class Mode, click on **Class-Defined** button in the **Color** box. The Object will assume the Class’s color or colors.

If you want to rename the current layout Object, simply type a new name into the **Object Name** control. The new name takes effect from that point on. Remember to save your layout to make the change permanent.

When you are finished with the layout Object, click the **OK** button. If you’d like to immediately create another layout Object, click the mouse in an open area of your layout, away from any layout element. You will then be prompted to enter a name for the next new layout Object. If you are adding several layout Objects based on the same Class, you will not need to change the Class each time, because the Class of the most recently created layout Object will be used each time. To abort creation of a layout Object, click on the **Cancel** button, or press the <**Esc**> key.

You will notice that not all properties associated with Objects and their motion can be initialized in the layout Object dialog box. For example, there is no provision for initializing a layout Object's directionality or speed from Draw Mode. Layout Objects inherit such properties from their Classes. These properties can be changed using trace stream commands, just as you would to change the properties of any Object; however, you probably will not need to, because most layout Objects never move.

To edit any layout Object that has already been created, select the Object by clicking on or near any part of its geometry in Draw Mode. Be sure you release the mouse button to highlight the Object (you'll see the red "X") before initiating a drag operation to move it.

A layout Object *can* be manipulated with trace stream commands. For example, you can issue a move command on a layout Object and the Object will move just like Objects created in the trace stream. If you destroy a layout Object using the destroy command and then save your layout after the destroy command has been executed, the destroyed layout Object will not be saved as part of your layout file. You will either have to replace the Object using Draw Mode or use the backup (.zzz) layout file that was created when you clicked on Save Layout in the File menu.

Exercise 14-2: Using Layout Objects

Repeat Exercise 10-1 (creating multiple machines using **Box Edit**). Instead of using background geometry for the ten machines, use layout Objects. We need six machines this time.

If you still have the boxedit.lay file from exercise 10-1, **Open** it now.

If you do not have boxedit.lay, you can easily repeat the necessary part of Exercise 10-1 by drawing one or more simple machines in Draw Mode.

Use **Box Edit** to capture one of the machines, and click the **Copy** button. Switch to Class Mode and click on the +C **Add Class** button. Specify a name such as "MachineShape" for your new Class. Click on **Box Edit** and **Paste** the geometry into the Class.

Now return to **Draw Mode** and delete all the machines using Box Edit's **Cut**. (You'll be saving this layout with a different name.)

Now click on the **Add Layout Object** button to create a single layout Object named Machine1. The Class will be chosen for you. Make the color of this layout Object F2.

The layout Object will be displayed with a red "X" at its Hot Point. The "X" also marks the (x, y) coordinates of the layout Object's location. (You could change the Hot Point of the Class by moving the geometry of the Class while in Class Mode, but this should not be necessary in this case.)

Next, using **Box Edit**, **Paste** a first copy of the layout Object you have just created. Carefully position the copy to the right of its "parent." Now click **Repeat** four times. Notice that **Repeat** automatically renumbers each new Object created. You should end up with six layout Objects named: Machine1, Machine2, ..., Machine6.

Add a line of **Text** that says, "GREEN=BUSY, RED=IDLE." **Save** your layout as flash.lay in the exercise folder.

We have pre-defined a trace stream that you can use with this layout. It is called flash.atf. Try running flash.lay & flash.atf.

Choose **View**, **View Proof File as Text**, **Trace File**. Flash.atf will be the default trace file name, since it is the active trace file. (Note: the very first time you use **View Proof File as Text**, you will be prompted to specify a "launch" command for your editor of choice. The default launch command is "start WordPad.") What would the trace stream look like if you had no layout Object capability at your disposal?

The Place In Command

The place in command lets you place one Object so that its Hot Point is at the same location as the Hot Point of another Object. It is like place at, except you do not need to know any (x, y) coordinates to place the Object.

Place in has two primary uses. First, if you represent a resource with a layout Object, then you may choose to show a resource state such as "in use" by placing a unit of work "in" the resource.

Note that place in does not permanently "connect" one Object to another, so it is not usually meaningful to place an Object in a moving Object.

Second, you may want to use place in as a "place at a named point" command. This works as follows. First, you might define a simple Class and call it Point. You might draw two short intersecting lines in the BACKDROP color, with the point of intersection at (0,0). Then, from Draw Mode, you could create a number of layout Objects of the "Point" Class and position them where you want to be able to place other Objects. Because they are Backdrop-colored, they would be invisible in Run Mode. Finally, from the trace stream, you could place visible Objects at any of your "named points" by referring to the names of the layout Objects of the Point Class.

The syntax for place in is:

place [**object**] *ObjectID1* **in** *ObjectID2*

place in causes the first Object to appear at the (x, y) location currently occupied by the second Object. (The two Hot Points are in the same spot.)

Here are two examples of the syntax of place in:

```
place 26 in WaitArea

place Carrier2 in Machine12
```

If the second Object is moving (via move, place on, or place object on at), then the place in command will place the first Object wherever the second Object is at the instant the place in command is processed. Place in never causes the first Object to move.

Place In and Rotation

When you use place in, the Object being placed (*ObjectID1*) can assume the rotational orientation of the other Object (*ObjectID2*), as follows. If the first Object is *non-Directional*, it will take on *neither* the rotational angle nor the directional-motion angle of the second Object. If the first Object is *Directional*, it will take on *both* the current rotational angle and the current directional-motion angle of the second Object.

This is illustrated in Exercise 14-3. Because the layout Objects (chairs) in the top row have a 180-degree rotation, the Objects (customers) undergoing the place in must be Directional.

Exercise 14-3: Using Place In

Let's take a look at an animation named stop. It's a cross between the sixbars and flash models seen earlier in this chapter.

The underlying system is the same as in sixbars. There are six servers and a single queue.

As in flash, stop uses layout Objects to represent each of six resources. However, instead of using color changes to represent changes in the state of each resource, stop uses place in to show units of work "in" each resource.

Try running shop.lay & shop.atf.

After you've looked at the animation, try repositioning and/or rotating the chairs in the barbershop by manipulating the layout Objects from Draw Mode. Save your modified layout as myshop.lay, and run it opening myshop.lay and shop.atf (in two steps).

Imagine the extra work you'd have to do to animate the reconfigured shop if your customers were using (x, y) coordinates or even Paths to get into the chairs. With place in, it's automatic.

Varying the Speed of an Object on a Path

Overriding the Path Speed

An Object travels on a Path at a particular speed, measured in animation coordinate units per unit of animated time. Normally, the speed at which an Object moves is determined by the speed of the Path designated during Path definition.

There are other ways to designate speed.

Every Object Class has a speed property. Class speed is set from the properties dialog box in Class Mode and saved in the layout file. If you specify a non-zero value for Class speed, it will override the Path speed for Objects of that Class.

An individual Object can have its own speed. Object speed is set from the trace stream using set object…speed. If you set an Object's speed, it will override both the Class speed and the Path speed. To set the speed of an Object, insert either of the following commands into the trace stream:

set [**object**] *ObjectID* **speed** *newspeed*

set [**object**] *ObjectID* **travel** *traveltime*

For example,

```
set AGV1 speed 10
```

```
set Truck travel 30
```

The travel option is the number of animated time units from "now" (when the command is processed) until the Object should reach the end of the Path, assuming no interruption in its motion. If you use set object travel, the Object speed is calculated inside Proof Animation as "(distance remaining)/(*traveltime*)."

The Object speed assigned as a result of set object speed or set object travel operates until overridden by another set object speed or set object travel command, or by either of the following two variations of set object speed:

set [**object**] *ObjectID* **speed path**

set [**object**] *ObjectID* **speed class**

Set object speed path designates the Path speed as the dominant speed for this Object. Set object speed class designates the Class speed as the dominant speed for this Object.

When you use set object travel, which calculates and sets an Object speed that is dependent on the remaining length of the Path, you will generally need to override it with set object speed or set object travel when the Object is placed on its next Path. This is because the Object speed you want for the next Path is probably unrelated to the speed assigned as a result of set object travel.

Objects can move at different speeds along the same Path, and will appear with proper accumulation (even while moving) and enforcement of clearances.

Using SET SPEED on Objects not on a Path

If you use the set speed command on an Object that is not currently on a Path, the speed will be set anyway, and the Object will assume that speed if it is later placed on a Path.

Set speed is a Path-oriented construct, so it does not normally affect the visible motion of an Object that is undergoing a move. However, if set speed 0 is used on an Object that is undergoing a move, the Object will stop. This is a special case.

Set travel cannot be used on an Object that is not on a Path, because the travel time cannot be calculated without a "remaining distance on the Path."

Changing the Speed of a Path

The speed of all Objects on a single Path can be changed at once using the following syntax:

set path *pathname* **speed** *newspeed*

set path pathname **travel** traveltime

For example,

set path Conveyor speed 0

set path Street1 travel 45

set path Aisle speed -10

Set path...speed affects the motion on *pathname* of all Objects currently on the Path, as well as any Objects subsequently placed on the Path. This will continue until that Path speed is overridden with another set path...speed command. However, set path...speed only affects those Objects that get their speed from the Path speed. Objects that have a Class speed or an Object speed will be unaffected, because they do not get their speed from the Path speed.

For non-accumulating Paths only, speeds and travel time can be set to a negative number. This causes Objects to travel backwards or back up on the Path. This is especially useful when an Object must back in, then pull out.

set path...travel works similarly to set path...speed. However, a travel time to traverse the entire Path is set. Proof Animation calculates the speed of the Objects on the Path using the *traveltime* and the Path length.

If you run a trace stream that contains set path...speed or set path...travel commands, and save your layout after those commands have been executed, your saved layout file will reflect the newspeed or traveltime issued in the SET PATH command for that particular Path.

It is good practice never to save a layout file after running any part of a trace stream.

The set path...speed command allows you to bring all Objects moving on an entire conveyor, transfer line, etc. to a halt by setting the Path speed to zero (and back to moving again later using a positive *speed*). Remember that setting the Path speed has no effect on Objects that have their own speeds, because Object speed always overrides Path speed.

Remember that Class speed overrides Path speed, and Object speed overrides Class speed or Path speed. This means that once you have set an Object's speed directly using a set speed *value, the Object's speed will not be affected by changes to the Path speed.*

If you are relying on set path speed *(e.g. for conveyors), you should avoid using Object speeds or Class speeds. If you do use Object speeds or Class speeds, you may have to rely on* set speed path *when you want those Objects to use the Path speed.*

Place On At

You may want to place an Object on a Path at a point other than the beginning. The place…on…at command does this. Here is the syntax:

place *ObjectID* **on** *pathname* [**squeeze**] **at** *offset*

place *ObjectID* **on** *pathname* [**squeeze**] **at end**

place *ObjectID* **on** *pathname* [**squeeze**] **before** *ObjectID2*

place *ObjectID* **on** *pathname* [**squeeze**] **after** *ObjectID2*

For example,

place AGV on Path5 at 2

place 12 on lane squeeze at 20

place AGV on Path5 at end

place 11 on loop squeeze at end

place BOX1 on conveyor squeeze before box2

place 2 on lane after 1

The *offset* is in linear units beyond the beginning of the Path. The distances are computed along the Lines and Arcs that comprise the Path. place...ON...AT 0 is the same as place...ON. In the first example, the hotpoint of the Object AGV will be placed ON Path5 2 units from the beginning of Path5.

The squeeze option is meaningful for accumulating Paths only. Squeeze causes Objects already on an accumulating Path to be pushed back in order to squeeze in the new Object. Squeeze never moves Objects already on a Path forward. Squeeze may push Objects already on a Path so far back that they fall off the beginning of the Path. If this occurs and validation is enabled for the Path, a special “Ouch” Alert will occur stating “Object *ObjectID* squeezed off the end of Path *pathname*.”

If the squeeze option is used with non-accumulating Paths, the Object will be placed on the Path at the specified *offset*, but there will be no effect on Objects already on the Path. In other words, no Object will be pushed back on a non-accumulating Path. You may see the new Object placed on top of an existing Object on the Path.

Place…on…at end allows you to place one or more Objects at the very end of a Path. For accumulating Paths, the squeeze option in this case (squeeze at end) pushes back all Objects currently on the Path so that the new Object can be placed at the very end of the Path.

Place…on…before and place…on…after allow you to place Objects on Paths before or after individual Objects already on the Path. Remember that the squeeze option is useful for accumulating Paths only.

Place On and Rear Guide Points

For Objects that have a Rear Guide Point (RGP), there are a number of options for Path placement of the RGP using special versions of the place…on command. You may need the RGP to travel on a different Path than that on which the hotpoint is traveling. The Path on which the RGP is placed can be specified using these commands:

place ObjectID **on** *pathname* **RGP on** *path2*

place ObjectID **RGP on** *pathname*

Here are some examples:

place 12 on GUIDE1 RGP on GUIDE2

place crane RGP on track

In the first example, the hotpoint of Object 12 is placed at the beginning of Path GUIDE1 and the RGP of Object 12 is placed on Path GUIDE2.

In the second example, only the RGP of the Object named crane is placed on a Path “track”.

The RGP of the Object always trails the hotpoint, so the initial placement of the Object’s hotpoint will be the same regardless of whether the RGP will be placed on a different Path.

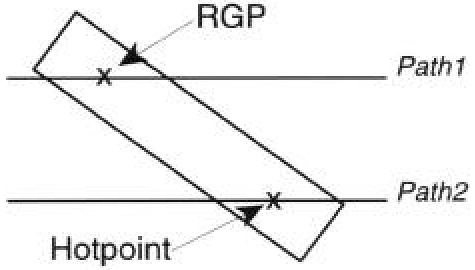


Figure 14-4 Example of an Object Whose Hot Point and RGP are on Different Paths

Exercise 14-4: Leapfrogs

Use the existing mypath.lay and mypath.atf files (either the files you created in the exercise folder in Exercise 11-1, or the already completed files in the sample folder) as the basis for this exercise.

Edit mypath.atf to destroy the second Object placed on TheAccPath before you destroy the first object. Run the animation and see what happens. Did you see an Alert? Try turning validation off for TheAccPath (do this from Path mode). Save the Layout, exit Proof Animation, and re-run the animation.

Changing an Object's Directionality

We've already seen how Objects inherit a "directional" or "non-directional" flag from the Object Class. A directional Object will point in the direction of movement during guided motion (place on) or unguided motion (move). A non-directional Object will appear to "slide" instead of turning around a corner.

You can dynamically change the directionality of an Object by using the set object… directional or set object…noindirectional trace stream commands. These commands operate on individual Objects. The syntax is:

set [**object**] *ObjectID* **directional**

set [object] *ObjectID* **nondirectional**

For example,

set CAR1 directional

set Car3 nondir

Why would you use this feature? You probably wouldn't, unless you were doing a detailed material handling system animation. An Object that moves directionally most of the time might occasionally need to slide. Examples include conveyor or other handling systems with a mixture of 90-degree transfers and curves or angled diverts, all-wheel-steering vehicles that park themselves sideways, and any system in which long narrow Objects are "bias banked" (stored at an angle in order to save space).

The steel animation, in the sample folder, has Objects that follow Paths for the most part, but slide when awaiting initial pickup by the crane. This is done with set object…directional and set object nondirectional. For an example of bias banking, see the bias animation, also in the sample folder.

When you use set object…nondirectional, the *current* orientation of the Object is "frozen" and used for subsequent sliding motion.

When you use set object directional, the original forward orientation will become the new orientation, but not until the Object embarks on another move or moves onto another Path *segment*. If you need an Object to rotate in the middle of a move or in the middle of a Path segment, you will need to use rotate.

Setting Object Clearances for Accumulating Paths

We've already seen examples of accumulating Paths in which the Objects "stack up" with gaps between them. These gaps are a property of the Objects on the Path, and can be changed.

When you define an Object Class you can set the fore clearance and the aft clearance.

The default fore and aft clearances are 0, measured from the Object's Hot Point. If you have an accumulating Path but the Objects are all coming to rest atop each other at the end of the Path, try setting the appropriate clearances in the Object Class definition.

You can change the clearance of an Object "on the fly" by using the set object…clearance trace stream command. This is useful in conjunction with set object…directional and set object…nondirectional (see the previous section), because if asymmetrical Objects are to accumulate in a new orientation, the clearances will probably need to be adjusted. The syntax is:

set [**object**] *ObjectID* **clearance** *fore aft*

For example,

```
set car1 clearance 3.1 4.8
```

If the change in clearance is temporary, remember to change it back before the Object proceeds on its way. The trace stream must specify the values of the original Fore and Aft clearances in order to change back to the original clearances using set object…clearance.

Attaching Objects to One Another

In some cases you may want to attach or "hitch" an Object to another Object and have them move on Paths as a chain. This is especially useful for animating trains or other traffic scenarios. To do this you can use the attach trace stream command.Attach lets you connect an Object to another for movement on Paths with their Hot Points separated by the applicable clearance values Only one Object can be attached to another. You cannot attach two or more Objects to the same "leader" Object. A chain of multiple attached Objects can be formed by executing a series of attach commands. For example, Object 2 can be attached to Object 1 and Object 3 can be attached to Object 2.

When a chain of attached Objects is to move along a Path, only the lead Object need be placed on the Path. The follower Objects will move onto the path, following the lead Object, as soon as there is physically enough room on the Path for that follower Object.

If the lead Object enters a Path that has a faster or slower speed than that of the previous Path, all follower Objects assume the new speed of the lead Object regardless of the speed of the Path on which they are currently moving.

The syntax of the attach command is:

attach *ObjectID1* **to** *ObjectID2*

ObjectID1 is the unique Object ID of the Object that will be attached to the Object referenced by *ObjectID2*. The spacing of the attached Objects is based on the clearances of the Objects.

Here are some examples of the attach command:

```
attach 2 to 1

attach 2 to 3

attach car to engine
```

In the first example Object 2 is attached to Object 1. In the next example, Object 3 is attached to Object 2. We have now created a chain of three Objects that are all attached. The third example attaches Objects with names for IDs instead of numbers.

Detaching Objects

To detach Objects that are currently attached to another Object, use the detach trace stream command. In order to use detach, the Objects must first be attached. If you try to detach an Object that is not attached to another, you will get an error Alert during execution.

The syntax for detach is:

detach *ObjectID1* **from** *ObjectID2*

ObjectID1 is the unique Object ID of the Object that is to be detached from the Object referenced by *ObjectID2*.

Here are some examples of the detach command:

```
detach 2 from 1

detach 3 from 2

detach car from engine
```

In the first example Object 2 is detached from Object 1. In the next example, Object 3 is detached from Object 2. The third example detaches Object car from Object engine.

Compound strings of attached Objects can be detached at any point in the chain simply by detaching the Objects at the point at which you want to break the chain.

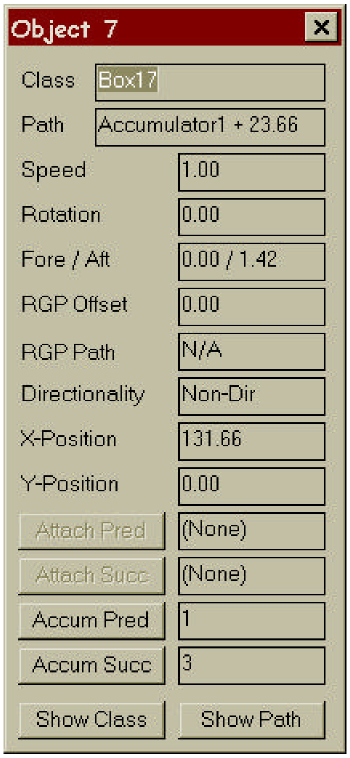
Advanced Debugging

In Chapter 5, we introduced some basic techniques for debugging an animation. In this section, we'll explore some advanced techniques. To utilize these techniques, you must be in Debug Mode.

Selecting and Displaying Individual Objects

There are two ways to select an Object for debugging purposes in Debug Mode. If you're interested in a particular Object that you can see on the screen, you can simply left click on it. On the other hand, if you know exactly which Object you'd like to examine, but you're unsure about its current location, you can click on the (Find Object) icon in the Debug Mode toolbar. Doing so will cause Proof to prompt your for the name or number of the desired object.

Once you've selected an Object (by either of the above methods), a dialog box of the following form will pop up:


Object 7

Class	Box17
Path	Accumulator1 + 23.66
Speed	1.00
Rotation	0.00
Fore / Aft	0.00 / 1.42
RGP Offset	0.00
RGP Path	N/A
Directionality	Non-Dir
X-Position	131.66
Y-Position	0.00
Attach Pred	(None)
Attach Succ	(None)
Accum Pred	1
Accum Succ	3

As you can see, this dialog box contains a wealth of information about an Object. The dialog box also contains five buttons that you can use to further explore an object:

- For Objects that are attached to other Objects, you can examine the chain of attached Objects by clicking on **Attach Pred** or **Attach Succ**.
- For Objects that are on an accumulating Path, you can examine the chain of Objects by clicking on **Accum Pred** or **Accum Succ**.
- You can explore an Object's Class by clicking on **Show Class**. If you do so, Proof will automatically jump into Class Mode, with the selected Object's Class active.
- If an Object is on a Path, you can explore the Path by clicking on **Show Path**. If you do so, Proof will automatically jump into Path Mode, with the selected Object's Path active.

Attaching debugging information to objects

One of the most effective debugging techniques at your disposal is the ability to attach descriptive information to Objects. Proof provides a pair of Trace Stream commands for doing so:

set *ObjectID* userinfo *descriptive_text_string*

set *ObjectID* usertitle *descriptive_title*

Setting an Object's userinfo attaches to an Object the descriptive text supplied. (The text begins with the second character after the userinfo keyword.)

To display the descriptive text, simply right click on the Object. When you do so, a dialog box will pop up, showing the text. You can set the title that is used in the dialog box by specifying a usertitle.

If you want format a multi-line display, use "\n" to indicate line breaks.

The following is an example of the above commands taken from a hypothetical air traffic control animation:

```
Set 387 userinfo UAL 327 From ORD to LAX\nDeparture: 09:35\nEnroute
Set 387 usertitle UAL 327
```

Note: Right-click display of userinfo *text can be done in Run Mode as well as Debug Mode.*