Introduction to Animation Programming

Dr. Susan Rodger
PipeLINK Activity Day

Introduction

First we'll tell you how to copy files. Type:

~rodger/pipe/setup_anim

Once you've done this, to work on your animation files, you'll need to move into the anim directory. Type the three commands on three lines (after each command, wait for the prompt to come back before typing the next command):

cd
cd anim
startup

You'll need to type these three commands whenever you want to work on your animations.

There is already a file there called `myanim.in` that you will modify (type `ls` to see the file). To modify this file, type `emacs myanim.in &`

Creating the animation window

First let's define the window size to be a plane with x having the values from 0.0 to 100.0 and y having the values from 0.0 to 100.0. We use the coords command.

```
coords 0.0 0.0 100.0 100.0
```

The numbers from left to right represent the leftmost x value, the bottom (or lowest) y value, the rightmost x value, and the topmost y value.

Displaying objects

An object is a rectangle, a circle, a triangle, a text phrase, a line, etc. Each object is created by specifying information about it such as its location, its color, its height and width, etc. Each object has a unique number associated with it. This number is used to move the object around.

Creating a text phrase

We'll create a text phrase, and specify the color and the position of the text in our plane.

```
text 100 65.0 20.0 1 blue Your Name
```

The first word text indicates we are creating a text string. The 100 is the unique identifier associated with the text item. The text is placed centered at position (65.0,20.0). The 1 indicates the text is centered. The color of the text is blue, and "Your Name" is the text that is written.

Delaying so the animation doesn't move too fast

```
delay 4
```

Use the delay command followed by an integer throughout your program to slow the animation down so you can see what is happening.

Rectangle objects

```
rectangle 200 16.0 8.0 40.0 10.0 green outline
rectangle 101 16.0 45.0 36.0 8.0 red outline
```

These two lines draw two rectangles. Each has a unique number associated with it (200 is the first rectangle and 101 is the second rectangle). The next 4 real numbers on each line specify the x-y position of the lower left corner of the rectangle (rectangle 200's lower left corner is (16.0, 8.0)), followed by the width and height of the rectangle (rectangle 200 has 40.0 width and 10.0 height). Following this is the color of the rectangle (200 is green) and the fillvalue of the rectangle which could be either an outline, half shaded, or filled in completely (Both rectangles are outlines).

Exercise 1: Running an animation

First replace "Your Name" at the end of the second line with your name. To run the animation, you'll use the anim command (you must be in your anim folder to run an animation, you should already be there). If the animation file is called `myanim.in` then you would type:

```
anim < myanim.in
```

A window pops up. The scroll bar on the right is always set on the fastest speed which is too fast, so **move the scroll bar down to the bottom.** Then click on **run animation**.

When the animation is done, click on **quit**.

Exercise 2: Add to the end of the file:

- First use the **switchpos** command to instantly switch the two rectangles.
- Follow this by a **delay** command.
- Use the **exchangepos** command to smoothly move the two rectangles.

- Follow this by a **delay** command
- Save your file, and rerun the animation.

Exercise 3: Add to the end of the file:

- Move the green rectangle to position (35.0, 85.0) using the **move** command.
- Move the red rectangle to position (36.0,86.0) using the move command.
- Move your name to position (50.0, 90.0) using the move command.
- Save your file, and rerun the animation.

Exercise 4: Create a traffic light

For this you are going to create new objects (rectangle and circles). Make sure each has a unique identification number (you can't use 100, 101, or 200 because we've already created objects with those numbers). Add to the end of the file:

- Create a new rectangle at position (45.0,45.0) with width 10.0 and height 25.0. The rectangle should be yellow, with the color only half filled in. (see the **rectangle** command).
- Create a solid red circle at position (50.0,64.0) with radius 3.0.
- Create a solid yellow circle at position (50.0,58.0) with radius 3.0.
- Create a solid green circle at position (50.0,52.0) with radius 3.0.
- Save your file, and rerun the animation.

Exercise 5: Add to the end of the file:

- Let's zoom in on the traffic light by making the coordinate system smaller. Type the command:

  coords 10.0 10.0 90.0 90.0

- Only one light should be on so turn the yellow and green lights off using the **vis** command.
- Flash the red light by turning it off and on several times using the **vis** command, with it finally left on.
- Cycle through the lights. Turn the red light off and the green light on, then the green light off and the yellow light on, then the yellow light off and the red light on. Stop with the red light on.
- Let's pan back out by increasing the coordinates back to the original coordinates. Type the command:

coords 0.0 0.0 100.0 100.0

- Save your file, and rerun the animation.

Exercise 6:

Now let's add a road and car to the picture. Add to the end of the file:

- Using the **line** command, create a thin black line with one endpoint (0.0,3.0), the xsize set to 100.0 and the ysize set to 0.0.
- Create another thin black line with one endpoint (0.0,15.0) and the same xsize and ysize.
- Create a solid blue car (rectangle with lower left position at (10.0,5.0), width 5.0 and height 3.0) and use the moverelative command to move the car 25.0 units to the right.
- Save your file, and rerun the animation.

Exercise 7: Add to the end of the file:

Add two more cars to the picture.

- Create a solid red car on the right side of the screen (rectangle with lower left position at (85.0,9.0), width 5.0 and height 3.0).
- Using the moverelative command, move the red car to the left -25.0 units (to move to the left, use a negative number, so move -25.0 units).
- Create a solid orange car on the left side of the screen (rectangle with lower left position at (10.0,5.0), width 5.0 and height 3.0).
- Using the moverelative command, move the orange car to the right 19.0 units.
- Save your file, and rerun the animation.

Exercise 8: Add to the end of the file:

- Change the traffic light from red to green and use the move command to move the cars through the light and off the window. You can either move the cars one at a time or move each car a little bit and then repeat this until all the cars are off the window.
- Save your file, and rerun the animation.

Exercise 9:

Add to the end of the file one or more of the following:

- Move the cars to the side of the road, let an ambulance or red firetruck pass by, then move the cars back on the road.
- Turn the road into a hill.
- Add another road to create an intersection.
- Have cars crash into each other.
- Add a story by creating lines of text.

Exercise 10:

Now have some fun and create your own animation. Start the animation in a new file called *myanim2.in* by typing `emacs myanim2.in &` and make sure the first line in this file is:

coords 0.0 0.0 100.0 100.0

Try out all the animator commands.

When you are done

Send us any animations you created. Type the following:

```
mail rodger < myanim.in
```

If you created another animation, send it also by replacing myanim.in with the name of its file.

Moving Rectangles

- Run the animation build.in by typing: `anim < build.in`
- The rectangles are numbered from 1 to 10. Sort the rectangles by height from smallest to tallest by swapping pairs that are out of order. Use the **exchangepos** command to do this. Follow this with a **delay 10** command.
- Now that the rectangles are in sorted order, the numbers correspond to this order (e.g. the smallest rectangle has the number 1, the largest rectangle has the number 10). Fill in the rectangles in order by filling in rectangle 1, then rectangle 2, etc. Use the **fill** command. Follow this with a `delay 10` command.
- Make the rectangles disappear and then reappear using the **vis** command. Make rectangles 1-5 disappear in this order, then make 1 reappear, 6 disappear, 2 reappear, 7 disappear, 3 reappear, 8 disappear, 4 reappear, 9 disappear, 5 reappear, 10 disappear, then 6-10 reappear. Follow this with a `delay 10` command.
- Swap rectangles instantly with the **switchpos** command. Swap rectangles 1 and 10, then 2 and 9, then 3 and 8, then 4 and 7, then 5 and 6, then 6 and 5, then 7 and 4, then 8 and 3, then 9 and 2, then 10 and 1. Follow this with a `delay 10` command.
- Create three circles using the **circle** command.
  - One circle is number 20 at location (25.0, 20.0) with radius 15.0, and a solid yellow.
  - One circle is number 21 at location (75.0, 20.0) with radius 20.0, and a solid red.
  - One circle is number 22 at location (55.0, 60.0) with radius 25.0, and a solid green.
- The circles covered some of the rectangles, bring the rectangles forward by using the **raise** command for rectangles 1 - 10. Follow this with a `delay 10` command.
- Move all three circles to location (55.0, 85.0) using the **move** command.
- One by one, move the circles out of the picture in different directions.
- Use your imagination to continue the animation.