**Animation & More Features**

**Animation**

Actually the effect of animation had been achieved with some programs by us so far. For example, by using delay(100) in the Pow program, each circle appears just one by one.

To erase a circle you can re-plot is using the background color, number 0.

Here is the modified Pow Program in animation.

% The modified "Pow" program in animation

% Plot a magenta circle of increasing radius centered

% in window until the edge of the window is reached

setscreen ("graphics")

const centerX := maxx div 2

const centerY := maxy div 2

for radius : 1 .. centerY

drawoval (centerX, centerY, radius, radius, magenta)

delay(100)

drawoval (centerX, centerY, radius, radius, 0)

end for

Note about animation:

* Animation is the effect of “moving pictures”
* Animation can achieved by erasing old picture and drawing new
* Erasing a picture
* Delay is often used in animation
* Remember draw🡪erase🡪 move🡪erase

🡪 draw🡪erase🡪 move🡪erase

You can check more examples here: animationEx1.t, animationEx2.t, imageAnimation.t

**Plotting a Mathematical Function**

To keep things simple we will plot a mathematical function that can be drawn with the origin of its coordinates at the lower left of the window, the bottom of the window as the x-axis, and the left side as the y-axis. We will plot the curve for the parabola

y = x \*\* 2

for values of x going from 0 to 30. We will let each unit in x-direction be represented by 20 pixels. One pixel is thus 0.05 units. This means that in the x-direction we will use from pixel 0 to pixel 600.

% The "Parabola" program

% Draws the graph of the function y = x \*\* 2 for x going from 0 to 30 in step of .05

var x:real

drawline (0, 0, maxx, 0, blue) % Draws x-axis

drawline (0, 0, 0, maxy, blue) % Draws y-axis

for pixel : 0 .. 600

x := .05 \* pixel

drawdot (pixel, round (x \*\* 2), cyan)

delay(30)

end for

**Using Text with Pixel Graphics**

Frequently we want to add text to a pixel graphics plot. In the Parabola program we could have labeled the graph in the magenta by adding these statements after the repetition (for statement).

color (magenta)

% Labelx-axis

locatexy (160, 10)

put "x-axis"

% Label y-axis

locatexy (10, 100)

put "y-axis"

% Label graph of parabola

locatexy (100, 150)

put "Graph of parabola y = x \*\* 2"

Note: The position of characters to be output is using the statement

locatexy(x, y)

where (x, y) represents the x- and y-coordinate of the graphics window. It is different with locate(x, y) where (x, y) represents the row and column number in text window.

**Background Color**

The color of the background of the Execution window can be set by using the statement

drawfillbox(0, 0, maxx, maxy, colorNumber)

Here is a program to change the window randomly.

% The "FlashWindow" program

% Randomly changes the window color

setscreen("graphics")

var c: int

loop

randint(c, 0, maxcolor)

drawfillbox(0, 0, maxx, maxy, c)

colorback(c)

locatexy(200, 300)

put "Flashing Window"

delay(500)

end loop

**Sound with Graphics**

A simple sound is in graphics display can be obtained using the statement

sound(frequency, duration)

Where the frequency is in hertz or cycles per second and the duration is in milliseconds.

The frequency should generally be between 200 and 2000. Frequencies outside that range are too low or high to be reproduced on a computer speaker.

Here a program draws graphics and plays a sound that goes up and then down.

% The "MakeSound" program

% Plays a rising then falling frequency along with colorful graphics

for i:0 .. maxy

drawfillbox (0, 0, i \* 2, maxy - i, i mod 16)

sound (i \* 2 + 200, 50)

end for

for decreasing i: maxy .. 0

drawfilloval (0, 0, i \* 2, maxy - i, i mod 16)

sound (i \* 2 + 200, 50)

end for

**Current Value of Graphics Parameters**

We can find out the current values of various parameters in pixel graphics mode. A number of functions provide such values.

whatdotcolor(x, y)

gives the color of the pixel at (*x, y*); the value of

whatcolorback

is the current background color number. Similarly the whatcolor function returns the current text color.

# Loading graphic Files

Pic.ScreenLoad is a handy function for displaying graphic files.

## Syntax

**Pic.ScreenLoad** (*fileName* : **string**, *x*, *y*, *mode* : **int**)

# Example

Pic.ScreenLoad("logo.bmp",0,0,picCopy)

The fileName must give the format of the file:

|  |  |
| --- | --- |
| JPG files | "JPG:filename" or "filename.JPG" |
| BMP files | "BMP:filename" or "filename.BMP" |

The x and y parameters set the lower left hand corner of the picture.

The mode parameter has one of the following values:

|  |  |  |
| --- | --- | --- |
| picCopy |  | This draws the picture on top of what was underneath, obscuring it completely. |
| picXOR |  | This draws the picture XORing it with the background. In DOS, you can use this function to do animation. Drawing an object on top of itself with XOR erases it and restores the background. |
| picMerge |  | This draws the picture like picCopy except that any occurrence of the background color in the picture is not drawn to the screen. This allows you to draw an irregularly-shaped object and draw it to the screen. |
| picUnderMerge |  | This draws the picture, but only where the background color was displayed underneath it. The effect of this is to make the picture appear to be displayed behind the background. |