CMPSCI 119
LAB #3 – Fish Tank
Professor William T. Verts

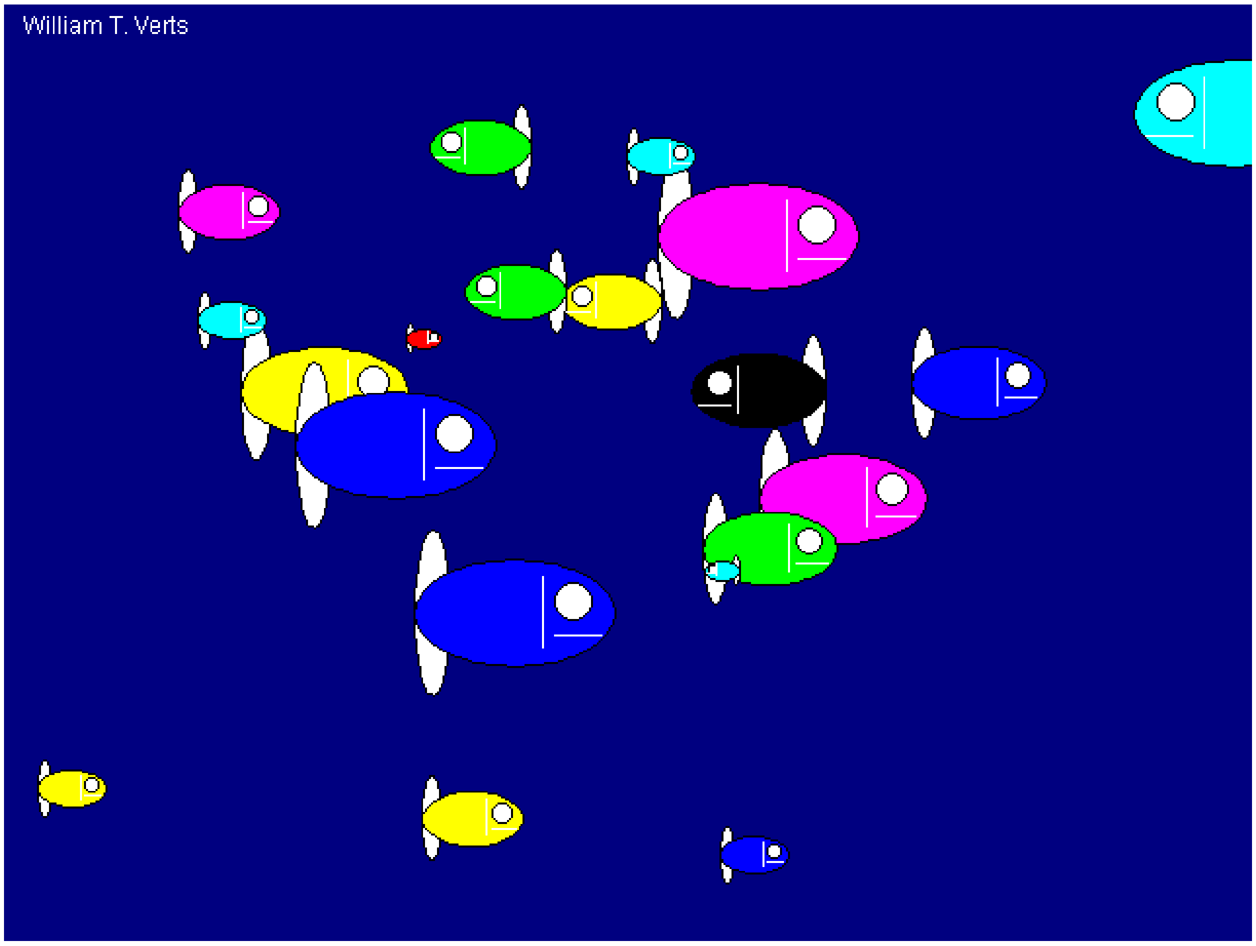
The goal of this Python programming assignment is again to write your own code inside a provided program framework, with some new graphical and mathematical considerations. Download the following code from the class site and save the program as `Lab3.py` in your Python folder. Where you see my name in the comment code, replace it with *your own name*. You will change the contents of the `FishTank` function, but ***you are not to change any other code in the program!***

```
# William T. Verts - Lab #3 - Fish Tank

import random,time

def INT(N):
    return int(round(N))

def Circles(Canvas,Xc,Yc,R,FillColor=white,EdgeColor=black):
    Diameter = INT(2*R)
    addOvalFilled(Canvas,INT(Xc-R),INT(Yc-R),Diameter,Diameter,FillColor)
    addOval      (Canvas,INT(Xc-R),INT(Yc-R),Diameter,Diameter,EdgeColor)
    return

def Ellipses(Canvas,Xc,Yc,Xr,Yr,FillColor=white,EdgeColor=black):
    addOvalFilled(Canvas,INT(Xc-Xr),INT(Yc-Yr),INT(2*Xr),INT(2*Yr),FillColor)
    addOval      (Canvas,INT(Xc-Xr),INT(Yc-Yr),INT(2*Xr),INT(2*Yr),EdgeColor)
    return

def Line (Canvas,X1,Y1,X2,Y2,NewColor=black):
    addLine(Canvas, INT(X1), INT(Y1), INT(X2), INT(Y2), NewColor)
    return

def Fish (Canvas,Xc,Yc,NewColor,Scale=1.0):

    def F(N):
        return N*Scale

    def S(N):
        return N*abs(Scale)

    Ellipses(Canvas,Xc-F(14),Yc,     S(3),  S(14), white, black)   # Tail
    Ellipses(Canvas,Xc,      Yc,     S(17), S(9),NewColor,black)   # Body
    Circles (Canvas,Xc+F(10),Yc-S(2),S(3),         white, black)   # Eye
    Line    (Canvas,Xc+F(15),Yc+S(4),Xc+F(7),Yc+S(4),white)        # Mouth
    Line    (Canvas,Xc+F(5), Yc+S(6),Xc+F(5),Yc-S(6),white)        # Gill
    return

def FishTank(Canvas, TotalFish=20):
    # All of your code goes here
    return

def Main():
    TotalFish = requestIntegerInRange("Enter Number of Fish", 1, 1000)
    FishTank(makeEmptyPicture(640,480), TotalFish)
    return
```

In JES click the Load Program button. At the >>> prompt, type `Main()` with the parentheses and press Enter↵. The program should run, and ask you for the number of desired fish, but should not do anything visible except make an empty canvas. Fix any syntax errors or other mistakes. You will not change `Main` while developing the program, but will fill in code for the `FishTank` function.

The Goal

The goal of this project is to create an animated fish tank with an arbitrary number of fish swimming around. A typical screen is shown as follows, with 20 fish, after they have been swimming around for a while:



We have provided functions for circles, ellipses, and lines. The circle and ellipse functions can paint the interior of the shape with one color and the outline with another color. All three functions are capable of handling either int or float coordinates. We have also provided a `Fish` function. If `Scale` is greater than zero the fish will face to the right, and if `Scale` is less than zero the fish will face to the left.

The `FishTank` function is currently a stub, and this function is what you must finish. Notice that the `Main` function combines the `makeEmptyPicture` into the parameter list of the call to `FishTank`, and that there is no initial color for the canvas. Both are by design, and are not errors.

Initialization

Inside the `FishTank` function you will need to create ***six separate lists***, one each for the X coordinates, the Y coordinates, the X directions, the Y directions, the colors, and the scale factors of the individual fish. The number of entries in each list will be determined by the `TotalFish` variable, which is allowed to be any value from 1 to 1000. You'll need to build these lists using either `for`-loops or list comprehensions controlled by `TotalFish`.

List #1 (Variable X): The initial values for the X coordinates must be the middle of the canvas, plus a random number between -50 and +50.

List #2 (Variable Y): The initial values for the Y coordinates must also be the middle of the canvas, plus a random number between -50 and +50.

List #3 (Variable Xdirection): The initial values for the X directions must be a random number between -1 and +1 (including 0).

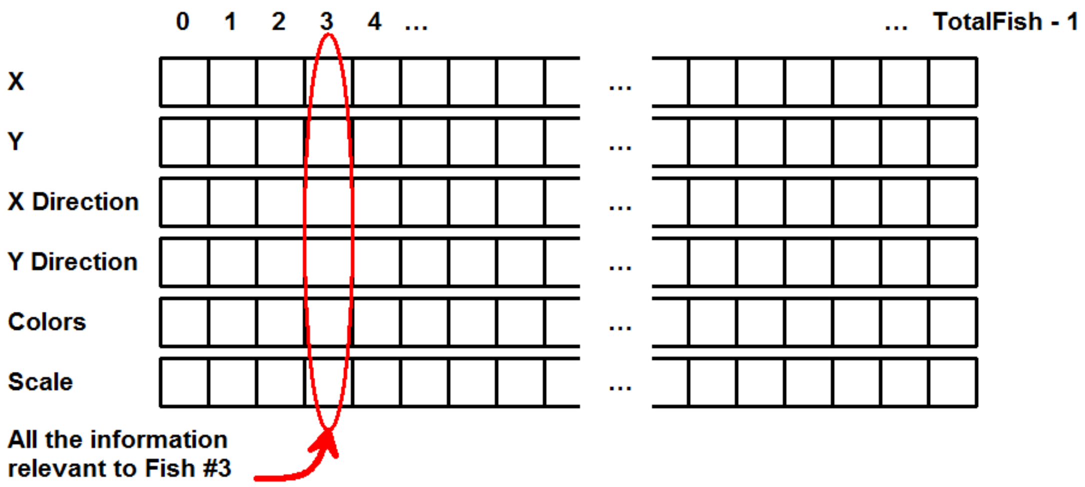
List #4 (Variable Ydirection): The initial values for the Y directions must be a random number between -1 and +1 (including 0).

List #5 (Variable Colors): The initial color for each fish will be chosen at random from the list `[black, blue, green, cyan, red, magenta, yellow]` (by intent the list does not include white, as the eye, tail, gill slit, and mouth of each fish are white).

List #6 (Variable Scale): The scale factor that determines the fish's relative size. These values must be chosen at random from 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 (all floats).

To get a random number between -N and +N, for any value of N, use the function `random.randrange(-N,N+1)`. For example, to get a random number between -1 and +1 you would use `random.randrange(-1,+2)`. To pick a random item from a list of items, you are allowed to use the `random.choice(`*`list`*`)` function.

When you are done building the lists, each fish will be defined by its position in those lists:



<u>**Testing your Fish Generation**</u>

Include the following code <u>after</u> the creation of the lists, then run the program.

```
    print "X=",X
    print "Y=",Y
    print "Xdirection=",Xdirection
    print "Ydirection=",Ydirection
    print "Colors=",Colors
    print "Scale=",Scale
```

When the program asks for the number of fish, type a small number such as 4 as a test. You <u>won't</u> see any fish on screen because you haven't written that part of the code yet, but you <u>will</u> see the results of the list generation. A sample output for four fish might look like this (but probably won't be the same values):

```
X= [291, 327, 316, 284]
Y= [247, 271, 229, 228]
Xdirection= [0, 0, -1, 1]
Ydirection= [0, -1, 0, 1]
Colors= [Color(0,0,0), Color(0,255,0), Color(255,0,0), Color(0,0,255)]
Scale= [0.5, 1.5, 3.0, 1.5]
```

From this example we can determine that:

Fish #0 is at <291,247>, is not moving at all, is black, and is tiny.
Fish #1 is at <327,271>, is floating upwards, is green, and is medium sized.
Fish #2 is at <316,229>, is swimming to the left, is red, and is huge.
Fish #3 is at <284,228>, is swimming downwards and to the right, is blue, and is medium sized.

**Do not proceed
until you can successfully generate these six lists
for any number of fish!**

**If you can't get this part working,
there is no reason to write any of the following code!**

<u>The Big Loop</u>

Once the six lists have been initialized, the program should go into an infinite loop (really!), controlled by a while-loop where the test condition is always True. The loop performs the following actions in the order shown:

1. Clear the screen to dark blue,
2. Paint all the fish at their current locations (using lists X, Y, Xdirection, Colors, Scale),
3. Update the positions and directions of each fish (using lists X, Y, Xdirection, Ydirection),
4. Repaint the canvas, then
5. Sleep for 0.05 seconds.

Dark Blue

To create the dark blue background color, use the JES functions `setAllPixelsToAColor` and `makeColor(R,G,B)`, where R=G=0, and B=128. I recommend creating a variable for the color above the loop so that it is computed only once. Using white text, plot your name in the upper-left corner of the screen – your name should always be present on-screen as the program runs.

Painting Fish

Paint each defined fish at the location indicated by the corresponding values in the X and Y coordinate lists. Use the corresponding value from the X direction list to determine direction: if the X direction is -1 the fish is swimming to the left, and if the X direction is 0 or +1 the fish is swimming to the right. Use the corresponding value from the color list to determine the color of each fish. Use the corresponding value from the scale list to determine the size of the fish. You will need to negate the value obtained from the scale list for fish facing to the left, but do not change the value actually stored in the scale list (if the scale value passed to the `Fish` function is positive the fish faces right, but if the value is negative the fish faces left). You will not use the Y direction list in painting the fish on the screen.

Updating Positions

For each fish, update its position by adding its <u>X direction</u> to its <u>X coordinate</u>, and then adding its <u>Y direction</u> to its <u>Y coordinate</u>. If the fish goes off the left side of the canvas make its entry in the X direction list *positive*, and if the fish goes off the right side of the canvas make its entry in the X direction list *negative*. If the fish goes off the top side of the canvas make its entry in the Y direction list *positive*, and if the fish goes off the bottom side of the canvas make its entry in the Y direction list *negative*. This will keep the fish always partially visible on screen (they are allowed to bounce off the walls based on their X and Y coordinates, which are defined as the center of each fish). **FOR EXTRA CREDIT: Make sure that fish do not go off screen at all, but bounce when *any portion* touches one of the walls.**

For added realism, we next use random numbers to give each fish the ability to change its direction in the middle of the canvas. For each fish, pick a random number between 0 and 9; if that value is 0 reinitialize the X direction for that fish to a random number between -1 and +1; one time out of ten the fish will suddenly “decide” to go left, go right, or stand still, regardless of what it was doing before.

Similarly, for each fish pick a random number between 0 and 19; if that value is 0 reinitialize the Y direction for that fish to a random number between -1 and +1. This means that one time out of twenty the fish will suddenly “decide” to float up, float down, or hover.

Repaint and Sleep

When the fish are painted and their positions updated, repaint the canvas and then sleep for 0.05 seconds. This will show the finished fish on the screen for a while before the next update of their positions, and will do so without annoying screen flicker.

Finishing Up

That is enough information for you to figure out how to fill out the `FishTank` function and run the program. You will need to hit the Stop button in the JES environment to break out of the infinite loop.

When you are finished and everything runs correctly, submit your final code through the on-line form as Lab #3.