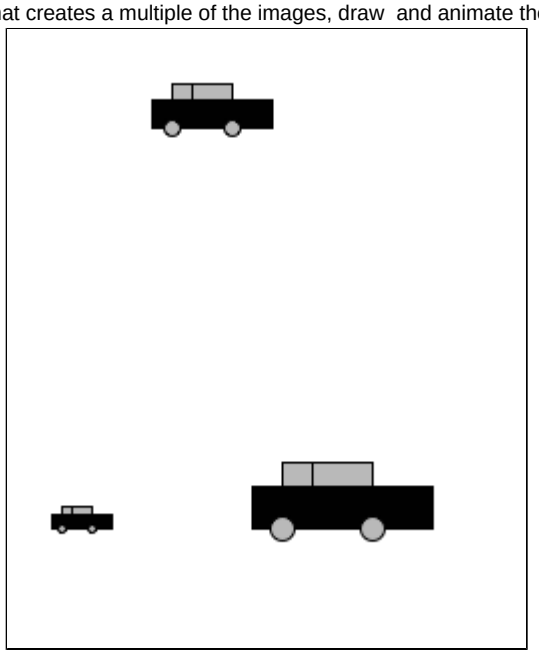
Project 6-Inuri

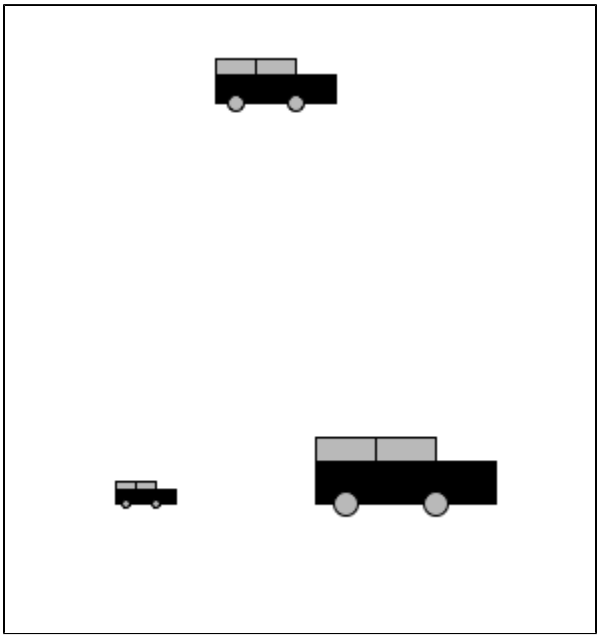
In this project we are asked to make a city scene using Zelle graphics and complex objects. First,to create two complex objects, create init and animate function,and test codes for each object and to use

Task 1 : Created three different complex objects , using Circle and Rectangle methods.Then for these objects, car ,jeep and road, an init and animate function were created. The functions of these three complex objects are in files, car.py , jeep.py and road.py respectively. Both the jeep and car drive forward. Also,I used 'setFill' method to add color to the objects.

I used 'aggregate.move' in animate function, with the variables (dx) , (dy), (objectlist) for both the car and jeep. Then I added a test function in each of the object functions, that creates a multiple of the images, draw and animate the objects.The images , img1 and img 2 are from car.py and jeep jeep.py respectively.



img1



img2

Task 2:

I created scene.py which brings all three complex objects together in a scene of a road. The key task is the use of lists that include the objectlists and name of objects in the form of duples, that are then looped through animate functions that decides which animation should be applied for each 'objectlist'.

Task 3:

This task allowed a concept from Lab3 , the use of scalars, and locations to move replicates of the scene1 in different sizes to different locations. I used this concept to create parallel roads to create a city. The scales used in init functions for car, jeep, and road allowed to use this concept effectively. Scene2.py is the product of this task.

Eg:

```
r = graphics.Rectangle(
    graphics.Point(x+10*scale,y-7*scale),
    graphics.Point(x+20*scale, y-11*scale))
```

I did an extension that further develops the above concepts, that enables to pass scales through the command line which decide the sizes of the replicates.

```
def main (args):
    if len(args) <7:
        #print "Usage:python scene2.py <scale1> <scale2> <scale3>"
        #create window
        win = graphics.GraphWin( 'accident', 800, 800 )
        #create  scenelists

        sl1 = (100,100, int(args[1]))
        sl2 = (100,300, int(args[2]))
        sl3 = (100,500, int(args[3]))
        sl4 = (100,700, int(args[4]))
        #put all the items in a list
        sls = [sl1, sl2, sl3, sl4 ]
```

Task 4 :

This task allows us to be creative with our codes. So added different colors,and a picture passed through the command line for the background.

Extensions:

1. used an elif function that used a string to decide which animate function should be used for the objectlist.

```
        if duple[0] == 'road':
            road.animate( duple[1],t,win , scale)

        elif duple[0] == 'jeep':
            jeep.animate(duple[1],t,win , scale)

        elif duple[0] == 'car':
            car.animate( duple[1],t,win , scale)
```

2. Created a program that reads scales sent through the command line and decides the size of the scenes. The concept 'lists in lists', the sys package and 'append' method, were used.

3. Uploaded an image passed through the command line as a pixmap for the background. Used scaling to make sure the picture fits between the the roads.Also used loops for (x) and (y) coordinates to create multiples copies of this image to cover the window.

```
    #background image

    filename=args[5]
    # loads an image and creates a blank image in its original size

    print 'b'

    img=[]
    for y in range (0,1000,200):
        for x in range (0,1000,200):
            a=graphics.Image(graphics.Point(x,y),filename)
            img.append(a)


    for a in img:
        a.draw(win)
```

Things Learned:

Learned to use 'while ' loop (in scene1.py)

Learned to use multiple frames in a sequence to create animations. (scene1.py and scene2.py)