# CSCI 1100 — Computer Science 1
# Homework 7
# CS 1 Birds

## Overview

Homework 7, worth **100 points** towards your homework grade, is due by **11:59:59 pm on Thursday, November 6, 2014.** All submission rules, academic integrity rules, and late penalties apply. Please pay careful attention to the submission instructions at the end of this document.

Computer games such as Angry Birds are based on simulating the basic rules of physics, or at least some rough approximation to these rules. These simulations involve a simple basic loop where in each iteration

1. The positions and sometimes the velocities (however, not in this homework) of moving objects are both updated by a small amount.
2. Objects are checked for collisions, and changes are made to the simulation based on these collisions.

While never 100% accurate, realistic looking results and physically useful predictions (for scientific simulations) can be obtained by making sure the changes in each loop iteration are small.

We will apply this idea to a simple version of angry birds called *CS 1 Birds*. Along the way you will get practice writing and using classes. Read this whole homework first before starting to write code.

The simulation occurs over a rectangular region whose corners are locations $(0, 0)$ and $(1000, 1000)$. The simulation will include birds and pigs and both will be represented by circles. The pigs are stationary, but the birds will move along a line (no gravity!). Each bird will move in turn, slowing down when it strikes a pig, and stopping when it becomes 1.) too slow or 2.) when it goes outside the game rectangle. When a bird strikes a pig, the pig will "pop" and disappear from the simulation. The simulation ends when either all pigs have been "popped" or when all birds have stopped, whichever occurs first.

## Input Files

The program should ask for two separate files from the user, one for birds and one for pigs.

Each line of the birds file will be in the form

```
name|x0|y0|radius|dx|dy
```

where `name` is a string giving the name of the bird, `x0` and `y0` specify the bird's initial position (center of the circle), `radius` is the bird's radius, and `dx` and `dy` specify how far the bird moves in each time step in the x and y dimensions.

Each line of the pigs file will include

```
name|xc|yc|radius
```

where `name` is a string giving the name of the pig, `xc` and `yc` specify the pig's center position, and `radius` is the pig's radius.

You may assume all input is properly formatted, all birds are entirely inside the bounding rectangle at the start, and none of the birds or pigs intersect each other at the start. You may also assume that each bird's movement $(dx, dy)$ is much smaller than the radii of the pigs, so that you do not need to worry about a bird completely passing through a pig in one time step.

After reading all birds and pigs, output the number of birds, the name of each bird and its starting center location, the number of pigs, and the name of each pig and its center location. The order should be the order the birds/pigs were read in. Follow the formatting of the output examples we have provided.

### Simulation and Output

The simulation starts at time 0 with the first bird in its initial location. Output something like:

```
Time 0: Freddie starts at (28.3,29.1)
```

(All position output should be accurate to 1 decimal place.)

In each time step:

1. Increment the time counter
2. Move the current bird by its dx and dy values.
3. Check to see if the bird's circle intersects the circle of a pig that has not yet been popped. (Intersection occurs when the distance between the two circle centers is less than or equal to the sum of the two circle radii.) If so,
   (a) Print one line giving the time, the name of the bird, the position of the bird, and the name of the pig. Something like,

   ```
   Time 25: Freddie at (15.3,19.1) pops Wilbur
   ```

   (b) "Pop" the pig. One easy thing to do is to remove the pig from the list of pigs.
   (c) Decrease the dx value (the x speed ONLY) of the bird by 50% You might want to make sure that python knows that the speed is a float... Note that this changes the direction the bird is heading. Print one more line,

   ```
   Time 25: Freddie at (15.3,19.1) has (dx,dy) = (4.5,5)
   ```

4. Check:
   (a) If the bird reaches a mininum speed of MINSPEED==6, stop moving the bird, remove it from the simulation, and move on to the next bird. (Note that the speed is $\sqrt{dx^2 + dy^2}$.) When this occurs output the time, the name, the location and the speed the bird, e.g.

   ```
   Time 281: Super Chicken at location (668.0,364.7) with speed 4.6 stops
   ```

   (b) If any part of the bird's circle has gone outside the rectangle, e.g. its x position (or y position) minus its radius is less than 0 or its x position (or y position) plus its radius is greater than 1000. When this occurs, output the time, the name and location of the bird. For example,

   ```
   Time 25: Big-Bird at location (975,234) has left the game
   ```

When a bird stops and there are more birds and more pigs left, start the next bird immediately at its initial location. Output something like

```
Time 25: Daffy starts at (123.7,88.5)
```

Just to be clear, the time Daffy starts is the same time the previous bird (Big-Bird) leaves the board.

5. If all pigs are popped then output a message saying something like

   ```
   Time 33: All pigs are popped. The birds win!
   ```

   and stop the game.

6. Otherwise, if there are no more birds, output a message with the names of the surviving pigs. For example,

   ```
   Time 33: No more birds. The pigs win!
   Remaining pigs:
   Porky
   Brick
   ```

   and stop the game.

The logic of this is a bit complicated, with a number of cases to check, so implement it carefully!

### Use of Classes and Functions

You **must** use at least two classes, one for a pig and one for a bird. The pig class in our solution is very simple and is already given to you in `Pig.py`. Do not change the pig class and instead put a lot of functionality into the `Bird` class.

To determine the functionality of the Bird class, we suggest you start by outlining the main functionality of the overall program and even start to write the main loop and the functions. Stick close to the outline we gave above. As you do this, the methods that you want the Bird class to have will become more clear, and as you think of these methods, you should make a note of what these functions should be in your `Bird.py` file. Do this initially without actually writing the methods in order to outline what your class should do and get a handle on what kind of code you will be writing. In our implementation, the Bird class does things like (a) move the bird, (b) check for a collision between the bird and a pig (A Pig object is passed as a parameter!), and (c) check to see the bird has crossed the boundary of the rectangle. There are others. Also implementing a function that allows you to print the bird info will simplify your main code quite a bit.

### Test Cases

We will post four data files to help you in your development and testing.

- `birds1.txt` and `pigs1.txt`. There are two pigs and both bigs survive (i.e. none of the birds hit a pig). This will allow you to test just your code that moves the birds and checks the boundaries. In this case, of course, the pigs will win.

- `birds2.txt` and `pigs2.txt`. In this case, each bird will pop a pig, and the birds will win.
- `birds3.txt` and `pigs3.txt`. This is a more involved example where the pigs win.
- `birds4.txt` and `pigs4.txt`. This is a more involved example where the birds win (Daffy pops 2 pigs, and the others each pop one).

Substantial partial credit will be given for correctly handling some but not all of these cases.

### A Quick Note

The simulation here is simple enough that it could be "solved" using pure algebra. This is not possible with more complicated objects and motions, so we do not consider an algebraic solution here.

### Module Organization and Submission Instructions

**Please follow these instructions carefully:** Your program should be split across three files, `Pig.py` (already given to you), `Bird.py`, and `hw7.py`. The file `Bird.py` will start with

```
import math
from Pig import *
```

and then continue with the `Bird` class implementation and any testing functionality you write. `hw7.py` should start with

```
from Pig import *
from Bird import *
```

and include the rest of your code.

`hw7.py` should read using `raw_input` the name of the file containing the birds and the name of file containing the pigs, in that order.

**In order to submit your solution:** Submit a zip file called `hw7.zip` containing exactly the two files `Bird.py` and `hw7.py`.