# CSCI 1100 — Computer Science 1 Homework 8
# Classes

## Homework Overview

This homework is worth **100 points** toward your overall homework grade and is due **Thursday November 19, 2015 at 11:59:59 pm**. 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 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 speed (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 simulation that we will call *CS1 Rickverse* based on the TV Show Rick and Morty. Along the way you will get practice writing and using classes. Read this whole homework first before starting to write code.

Note that this is a long and complex homework. We will give lots of partial credit even if you do not get all the answers right. We will provide test cases of increasing difficulty. Make sure you develop slowly and test throughly. We will release new test cases throughout the week as we come up with them.

In our sample output, we will print the location of Rick's at each 10 time step to help you debug your code and print lines of 75 hyphens to make the output readable. While you do not have to match this exactly to get a perfect grade, it would help TAs reading the results to match as closely as possible.

## Homework Universe: Data

### Boards for Alternate Dimensions

In this homework, you have alternate dimensions that we will call boards. Time goes in the same speed and in parallel in each board. Each board will be given by a rectangular region whose corners are locations $(0, 0)$ (top left) and $(1000, 1000)$ (bottom right).

We will present the boards to the program in a text file which will contain lines of the form:

```
name|gravity|portalx|portaly|rightboard|upboard|leftboard|downboard|x1|y1|x2|y2|x3|y3|...
```

Let us explain these values one by one. We will give each board a different `name` to distinguish them from each other. In addition, each board will have different `gravity`, meaning that the objects move in different speeds based on the gravity of the board.

Each board has a portal, given by its `portalx,portaly` coordinates. When a new Rick enters the board, he will enter at this portal location.

Boards are adjacent to each other. For example, `leftboard` is the name of board that is to the left of the current board. So, if a Rick exited the board anywhere in the left edge (between $(0, 0)$ and $(0, 1000)$), then he will end up in the board named for `left` at its `portalx, portaly` location. An adjacent board is given for all edges (left, up,right, down) of the current board. Some edges may lead back to the same board.

Finally, each board may have some obstacles, rocks, which will be given with their `(x,y)` locations on the board. These are listed after the names of the boards along each edge of the board. The number of obstacles in each board may vary. For example, given:

```
phone|2|100|100|phone|furniture|furniture|pizza|200|200|500|20
```

We know that `phone` board has gravity 2, has a portal at `100,100`, the right edge points back to itself, up and left edges point to the furniture board, and down edge will take you to the pizza board. There are two obstacles at points `200,200` and `500,20`.

Assume the universe you read from a file is complete and correct.

You must implement a `Board` class that will include this information and relevant methods if any. In my implementation, the boards did not have any significant methods other than the constructor (`__init__`). For now, you should consider storing the above information for each board as a first step.

### Ricks

Your simulation will have different versions of the same person, Rick, from different boards. Clearly, there can only be at most one Rick that is native to a board but Rick's can travel between boards (and more than one Rick can be on the same board as a result). We will represent a Rick as a square for simplicity and keep track of the location of his upper left corner and his length.

Rick's will be given to you with an input file called `Ricks.txt` and you can assume that the Rick-universe does not violate the rules of the universe when we start the program (and no two Rick's are in overlapping locations originally).

Each line of the Rick file will be in the form

```
boardname|x0|y0|length|dx|dy
```

where `boardname` is the board that Rick belongs to. We assume that Rick is in his own board when the simulation starts. The other information gives the initial position of Rick's top left corner (`x0,y0`), Rick's size (`length`) and `dx` and `dy` specify how far the Rick moves in each time step in the x and y dimensions when gravity is 1 unit.

You must implement a `Rick` class to store all these relevant attributes as well as other information and methods that will become useful to implement the simulation. Of course, you can assume Morty is next to Rick if that makes you happier! Rick class will likely have lots of methods and the more useful your methods, the simpler the main simulation will be.

## Starting your program

Your program must read the name of two files to get started, the board and the Rick file.

```
Board file name => board1.txt
```

```
board1.txt
Rick file name => rick1.txt
rick1.txt
```

Once you have read the files and created the relevant board and rick objects, the simulation will start. Use integers for x,y locations, length of Rick, and gravity. Use floats for speed (`dx,dy`). After updating x,y coordinates, truncate them to integers.

## Homework Simulation

The simulation in CS1 Rickverse will proceed as follows.

At time 0, before the simulation starts, your program reads the boards and Ricks, and prints the information for the boards in the format:

```
Board pizza: Portal location: (50,50), Gravity: 1
    Obstacles at: [(463, 146), (32, 48)]
    Portals to: right: chair, up: zygirionsim, left: phone, down: normal
```

and the location of each Rick with a line like:

```
Time 0: Rick of pizza is in pizza board at (100,100) with speed (10.0,10.0)
```

When the simulation starts, remember to process Ricks in the order they appear in the input Rick file.

Simulation starts at Time 1. At each step of the simulation, the following happens:

1. Each Rick moves by its current speed (`dx,dy`) in their current universe adjusted by the current gravity (the gravity simply means the speed impacts the movement by a fraction).

   For example, if gravity is 3, then Rick's position will change by: `dx/3, dy/3`. If gravity is 1, you will simply add `dx,dy` respectively. After moving the specific Rick, check the following:

   (a) If this Rick hits an obstacle in its own board:

      i. reverse his direction completely, (`-dx,-dy`), and
      ii. decrease the dx value (the x speed ONLY) 50%. Note that this changes direction this Rick is heading.

      Print a line that describes the crash:

      ```
      Time 5: Rick of Chair crashed into object at (2,3) in Pizza board
         New speed is (-1.6,2.5)
      ```

   (b) If this Rick reaches the edge of his current board (if his square touches or intersects with an edge at all or Rick is outside the board), move him to the board that is adjacent at that edge. If he is intersecting with two edges simultaneously, always use the vertical edge (left or right). Rick will now be in this new board, emerging at the `portalx,portaly` coordinates of that board.

      In this case, adjust Rick's position with the name of the new board and the new x,y dimensions. The speed (`dx,dy`) will remain the same.

      Print lines that describe the change in boards.

```
Time 33: Rick of normal moved from normal board to pizza board
    Past location: Rick of normal is in normal board at (995,406) with speed (30.0,12
    Current location: Rick of normal is in pizza board at (50,50) with speed (30.0,12
```

2. This is the hardest part (add this last). After moving all Rick's once and then adjusting for board changes, check if any two Ricks clash in a board (for a clash, two Ricks must be on the same board and there must be an overlap of the squares representing the two Ricks).

   If so, then both of the Ricks will go back to their home board (emerging at the portal coordinates), and their speed will reduce by half along both dimensions (`dx/2,dy/2`) but the sign will remain the same.

   Print a line for all the relevant changes.

   ```
   Time 10: Rick of Furniture and Rick of Chair have collided in Alderaan board.
      Rick of Furniture is in Furniture board at (100,100) with speed (2.0,3.0)
      Rick of Chair is in Chair board at (200,200) with speed (3.0,-4.0)
   ```

   Process Rick pairs in the order they appear in the input list. While we will not have this situation in our test cases, if more than two Ricks collide, process the pair you find first. After these Rick's collide and move back to their board, continue checking for the collisions in the remaining order. In a real simulation, you would add some randomness here but that would make it hard for us to match inputs.

3. If a Rick reaches a mininum speed magnitude that is less than twice the length of that Rick (i.e. `MAGNITUDE <= 2*length`, enough to get away from an obstacle it just hit), stop moving him and remove him from the simulation (Note that the speed magnitude is $\sqrt{dx^2 + dy^2}$.). Basically, Rick's who stop are caught by the *The Transdimensional Council of Ricks* and put in the intergalactic jail for their crimes. When this occurs output the time, the name, the location and the speed magnitude of the Rick.

   ```
   Time 10: Rick of Chair in Pizza board location (200,200) with speed magnitude 4.6 stops.
   ```

4. Finally, increment time by 1 and continue simulation. We will simulate the universe for exactly 100 steps or until there is no moving Rick left in the simulation (whichever comes first).

Once the simulation ends, print the locations of the remaining Rick's.

```
Time 100: Simulation ended

The following Ricks are still alive:
   Rick of Pizza is in Furniture universe at location (100,100)
   Rick of Phone is in Phone universe at location (200,200)

The following Rick's were caught by Transdimensional Council of Ricks:
   Rick of Chair
```

**Note.** You must implement two classes: Board and Rick. Think about what data you must keep in your program. You need to keep track of all the Ricks and Boards, and their states. Where will you store the information of which Rick is in which Board? It depends on how you write your program. Think about which methods to implement, such as checking for overlaps and for speed, etc. Create print methods that help you simplify the program.

It is important for this program that you make good use of classes. We will seek input from your TAs and award some well-written programs with prizes!

## Module Organization and Submission Instructions

**Please follow these instructions carefully:** Your program should be split across three files, `Board.py`, `Rick.py`, and `hw8.py`. The file `hw8.py` will import both classes:

```
import math
from Board import *
from Rick import *
```

and include the rest of your code.

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

**In order to submit your solution:** Submit a zip file called `hw8.zip` containing exactly these three files `Board.py`, `Rick.py` and `hw8.py`.