Final Project

AE 227 – Spring 2019

Assigned: 5 April 2019 **Due: 8 May 2019**

Learning Objectives:

For this assignment, you will apply the programming fundamentals that you have learned this semester, especially loops (for, while), conditionals (if), functions, and arrays to create an animation in MATLAB that simulates the collision of balls. Also, you will submit a readme.txt file that describes how your program works.

Description:

Using the programming concepts that you have acquired this semester, you are to implement a program in MATLAB that simulates the interaction of balls that have to interact with the walls and with each other according to the rules given in the following pages. You will be provided with several functions to help you with the animation.

Guidelines:

- **This is an INDIVIDUAL project.** Each one of you will create your own program. Your .m files for the simulation will consist of one main program called **animate.m** and other functions that you have defined.
- The program should be written modularly, meaning that different tasks should be placed in different functions defined by you.
- All of your .m files must be well commented, including descriptions of your functions that can be seen when "help functionname" is typed into the MATLAB command window.

Functions:

Below is a list of the graphic functions needed to work on this project. These functions can be found in the Project section on the class Blackboard page. All of the functions here have MATLAB help paragraphs that give a description of the function.

createWindow:

format: createWindow(w, h)

purpose: generates a window whose width is w units and height is h units.

drawBall:

format: h = drawBall(xc, yc, r)

purpose: draws a ball (filled circle) whose center is at (xc, yc) units and whose radius is r units. The function returns a handle h that identifies the ball being created. The handle can then be used by the following functions below.

xMove:
format: xMove(h, xd)
purpose: moves the object whose handle is h by xd units horizontally. xd can be either positive or negative.
yMove:
format: yMove(h, yd)
purpose: moves the object whose handle is h by yd unitss vertically. yd can be either positive or negative.

getcenter:
format: [x, y] = getCenter(h)
purpose: returns the coordinates in units of the center of the object whose handle is h.

redraw:
format: redraw
purpose: redraws all objects to reflect their new positions and flushes the buffer.

Suggested Inputs

1. The size (width and height) of the window
2. The size and number of balls to simulate and their positions (or random placement)
3. The initial direction of all of the balls
4. Alternatively, a file that will provide the inputs to the program. stack1.txt/dat, stack2.txt/dat, and stack3.txt/dat are provided as examples

Input File:
The structure of the file to be read will contain the following:

- Line 1: Width, Height
- Line 2: Size
- Line 3-end: X-coordinate, Y-coordinate

Rules for the Animation:

1. **Motion of the Balls:**
    a. A ball will only move in one of four possible directions in unit space: $\pm 45^o$ or $\pm 135^o$. The four directions correspond to NE, SE, NW, or SW.
    b. Therefore, the minimum distance of a ball's motion per iteration is ± 1 unit horizontally and ± 1 unit vertically.
2. **Interaction of the Balls with Each Other and with the Walls:**
    a. A ball interacting with a wall will behave in a manner similar to a mirror reflection, thus maintaining the rules of motion above.
    b. Two balls interacting with each other will behave as follows:
        i. If the interaction between the balls is head on, both balls will reverse direction.

ii. If the interaction is orthogonal. Each ball will perform a mirror reflection such that the balls' common direction component remains unchanged.

c. Two balls moving in the same direction will not interact.

3. **Guidelines and Assumptions:**
   a. A ball must be fully inside the window upon its creation.
   b. No two balls can initially overlap upon their creation.
   c. All balls have the same size.
   d. All balls move at the same velocity.
   e. All balls initially move in the same direction (NE).
   f. It can be assumed that three balls will never hit each other simultaneously.

The Readme file:

You will need to provide a **readme.txt** file along with your project .m files. The readme file should consist of the following sections, which should be labeled with titles:

- A brief description of what your program does (one paragraph only).
- An explanation of your algorithm for ball-to-ball interaction.
- An explanation of your algorithm for ball-to-wall interaction.
- A list of the functions that you have written and a brief description of what each function does.
- Any further comments (if any) that you feel would help the reader.

Grading:

- Points assigned to the project itself (up to 12 points):
  - 1 ball that interacts correctly with the walls: up to 4 pts
  - 2 balls that interact correctly with the walls: up to 5 pts
  - 2 balls that interact correctly with the walls and with each other: up to 6.5 pts
  - 3 balls that interact correctly with the walls and with each other: up to 7.5 pts
  - Any number of balls that interact correctly with the walls and with each other. The number of balls, location, and size must be entered using the formatted input file: up to 10 pts
  - Desirable features (user defined) up to 12 pts

    You are required to implement other features as well in your program. This is an opportunity to show your creativity and individuality. ***Do not share what additional features you are incorporating with your colleagues***. Do not add sound to the animation.

- Points assigned to the readme file: up to 3 pts

Points will be deducted for any submissions after 8:30 AM on 8 May 2019.

Submission Requirements

- All relevant .m files (animate.m and all of your functions) to be turned in on Blackboard by 8:30 AM on 8 May 2019.
- Your readme.txt to be turned in on blackboard by 8:30 AM on 8 May 2019.
- A hardcopy version of your readme.txt turned in to me at class on 8 May 2019.

Reference:

The project and the functions provided for the project come from the University of Notre Dame EG 112 – Introduction to Engineering Systems course's third project from Spring 2004.