Project 2

CS 320

75 points, due Apr. 6, 2009

For this project you will implement the first phase of a pool simulation, starting from `collision.c`. You have a pool table with certain attributes and pool balls with certain attributes. There is a simulation attribute or two to handle as well. Sample data is collected into the file `poolData.txt`. Your simulation will take a file name as a command line argument and use the data in the file to define the simulation. The data format is as follows. Notes: Lengths are in inches. Colors are rgb components on [0.0, 1.0]. Velocities are inches per second. Time is seconds. The coefficient of rolling friction (C_{rf}) has units of inches per second per inch. Mathematically:

$$V' = (1 - C_{\mathrm{rf}} T_{\mathrm{i}}) V$$

where V' is future velocity, T_{i} is time elapsed for current simulation step, and V is current velocity. Ball masses are ounces. Positions and velocities are given in three dimensions (x, y, z). Each data item will be separated by one **or more** whitespace characters. File format begins with next line.

```
number of simulation steps per render step
ll.x ll.y ur.x ur.y --- area of play boundary
color of area of play
width of fringe area surrounding area of play
color of fringe area
coefficient of restitution
coefficient of rolling friction
number of balls
mass radius color position velocity --- ball data, repeated for each ball
```

Your simulation should have the following characteristics:

1. I expect your program to be readable and literate. Use meaningful identifier names, adequate whitespace, and appropriate comments. Split long statements into several lines, preserving readability. Indent carefully.

2. As already mentioned, data will be read from a file. The name of the file to be read should be given on the command line. **Do not** read the file name from stdin.

   The simulation should exit if there is not a single command line argument, or if the data file can't be opened.

   Look online (phoenix) at the documentation for `fopen()`, `fscanf()`, and `fclose()`. The `fscanf()` conversion formats you will need for reading are `%lf` for reading doubles and `%ld` for reading integers. Do not forget to pass the addresses of the variables into which you are reading:

```
fscanf(data, "%lf", &ll.x);   /* FYI, data is a file handle. */
```

3. It is acceptable to use constant values to define the window size rather than compute values. I used a window of size 900 by 450. You may assume that the window will not be resized.

4. Your simulation should be physically accurate. This means you will need to record the time between simulation steps and account for it in your simulation. This also applies to the coefficient of rolling friction. These will ensure the simulation looks the same regardless of the capabilities of the machine on which it runs. The Windows function `GetTickCount()` will be useful here:

   ```
   #include <Windows.h>
   #include <Winbase.h>
   int GetTickCount(void);
   ```

   This function returns the current number of milliseconds since boot time. It overflows every 49.7 days. You should account for this overflow.

5. Using the data item given in the data file, your simulation should be capable of performing multiple simulation steps per single rendering step.

6. Your simulation should calculate average (rendered) frames per second for five second intervals and display the five second averages.

7. You may assume that this simulation will never involve more than 100 pool balls.

8. If designed and written properly, very few modifications need to be made to `collisionResponse()` for handling collisions between a ball and the boundary. It will be quite useful to have a wrapper function generate the virtual ball modeling the boundary and then pass the two balls to `collisionResponse()`. You will need to find a way to represent the infinite mass of the virtual ball and modify `collisionResponse()` to recognize this and adjust the computation. This is the only change needed to `collisionResponse()`.

Submitting Your Project

Your solution is to be e-mailed to me at kelliher[at]goucher.edu. All project files should be sent as attachments in a single e-mail. You may collect all the files into a single ZIP archive, but be sure to first change the file extension and inform me in the email that the file contains a ZIP archive. You should send all files necessary for me to build your program from source (generally, this is all .h and .c files), as well as any documentation and test files. You should send an ASCII file, named README.txt, describing the rest of the attached files. I will build your program from source and run it for myself. Your project is due at the beginning of class on the 6th. My standard late penalty (10%/three days) will apply.