CS 4550/5550 Computer Graphics (Fall 2017)
Homework1 (6%) - Programming

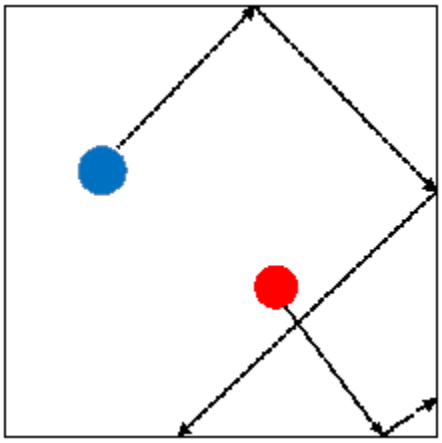
- 90 points of HW1 total score.
- No due date extension and no late submission.
- Penalty will be given when you do not follow any given instructions.
- Due: CSNS submission by Friday 9/22, 2017

OpenGL Programming; Bouncing Ball Program

☞ Submit your ***source code*** only (.cpp and .h). Insert your name and CIN# into all source files. Your program must be compiled and run using Microsoft VC++. Test your program before your submission if you programmed on a different platform or using a different compiler.

☞ Also, submit a ***readme.txt*** file that explains your submission including how to compile/run your program.

Write an OpenGL program that allows the user to render colored balls and change its moving direction and shape according to the user inputs.



Initial state: Start with two moving balls drawn as colored circle. Draw each ball using the *parametric* circle equation.

Ball generation: Any moving ball should be generated with the following properties.

- Radius R, mass M, and velocity *v* (direction and speed/magnitude).
- Assume that M is proportional to R. (e.g $M=R^3$)
- *v* should not be 0.

Ball animation and collision: In this homework, assume that the balls to be rigid, frictionless, perfect spheres. Balls are moving at constant velocity.

- Each ball's next position L' is computed by $L' = L + v * \Delta t$ where L is the current position and *v* is the velocity of the ball and $\Delta t=(T'-T)$ is a small time increment.
- When the ball hits one of four wall boundaries, calculate the new velocity and update its position (i.e. change direction and continue to move along the new direction.) Use ***reflection*** to calculate the new velocity. Do not change its magnitude.
- When two balls collide with each other, calculate each ball's new velocity based on Collision Physics (***Perfect Elastic Collision***) shown in the supplementary document.

Keyboard events: Process the following keyboard events.

- 'a' keystroke: add another moving ball to the scene. Allow the user to add up to ***three*** more balls to the scene. So the program should render up to **five** balls.

- ‘r’ keystroke: remove the most recently added ball from the scene. The user cannot remove all balls from the scene. At least one ball must be present in the scene.
- ‘1’ ~ ‘5’ keystroke: This is a required feature for graduate students only. Allow the user to choose a ball using the number. ‘1’ for the first ball through ‘5’ for the fifth ball. If any ball to be associated with a number is not on the screen, ignore the keystroke. Default is ‘1’.
- Manipulate the first ball (manipulate the selected ball if you are a graduate student) using the following keys:
  - ‘p’ keystroke: toggle between a filled ball and a non-filled ball
  - ‘Page UP’/‘Page DOWN’ keystrokes: increment/decrement radius R. (GLUT constants: `GLUT KEY PAGE UP` and `GLUT KEY PAGE DOWN`). When radius R increases, its mass M must be increased using the relation you used for the ball generation. Velocity *v* must be decreased using the equation: *v*’ = M*v*/M’ where *v*’ is the decreased *v* and M’ is the increased M.
  - ‘UP’ keystroke: increase velocity
  - ‘DOWN’ keystroke: decrease velocity
- ‘n’ keystroke: reset the program to the initial state.
- ‘q’ keystroke: quit the program.

Mouse events: Allow the user to pick a ball to change its velocity *v* by using mouse (left-click, drag and drop).