CS 354
Project 2 - billiards
Due Sept 28, 11:59 pm

Overview

For this assignment you will be implementing a game of billiards. You must have one white ball and at least one other ball. The mouse gesture for hitting the white ball is to click and drag the mouse. The direction you drag determines the direction the ball will travel and the distance you drag determines the initial velocity of the ball. The balls bounce off the walls of the "table" and off of each other. You're 80% base score will be judged on quality of the animation and realism of the physics. For the 10% creativity score you should implement extra features.

The purposes of this assignment are fourfold:

1. Continue familiarity with OpenGL and C++
2. Get a solid handle of vector algebra
3. Start building up your own vector and geometry library
4. Become familiar with physics in graphics and game-play

Specifications

1. The graphics window must be no larger than 800x600.
2. You must have at least two balls; one and only one of them should be white.
3. The white ball is the only one that is struck by the cue stick. The mouse gesture is as follows: the user clicks and drags the mouse anywhere on the canvas. Let the mouse down point be B and the mouse up point be A. The initial direction the white ball will travel is $B-A$. The speed, or magnitude of the velocity will be $\alpha|B-A|$ where α is a constant that you determine. Note that the mouse gesture defines a vector without a position. It doesn't matter whether point B is on or even near the white ball or not. Try out the included binary `example-billiards` for an example.
4. The balls must have walls to bounce off.
5. The balls must bounce off or interact with each other in some meaningful way.
6. For the creativity points you will add new and interesting features to your game. Some ideas (but you are definitely not restricted to these) might be to have holes, obstacles, explosions, rogue balls, etc. The only requirement is that you have exactly one white ball and meet other specifications listed above.
7. List all additional features that you add in README.txt. If you do not list the features you will not get credit for them.
8. This is a 2D game. You may not use any glVertex3* commands.
9. Submit your work by typing `./submit` at the command-line. A report will be written that you can consult to make sure it submitted correctly.

Helpful hints

1. Recall the definitions of position, velocity and acceleration. Velocity is the derivative of position, and acceleration is the derivative of velocity. Thus the following equations hold (assuming acceleration is constant):

$$\mathbf{v}(t) = \mathbf{a}t + \mathbf{v}(0) \tag{1}$$

$$\mathbf{x}(t) = \frac{1}{2}\mathbf{a}t^2 + \mathbf{v}(0)t + \mathbf{x}(0) \tag{2}$$

where $\mathbf{v}(t)$ is the velocity at time t, $\mathbf{a}$ is the acceleration, and $\mathbf{x}(t)$ is the position at time t.

2. You will likely make heavy use of the definition of the dot product (section 3.1.9 of the text) and of its geometric interpretation (figure 3.14 of the text).
3. It will be helpful to start creating a library. Some suggestions are to write point and vector classes as well as dot product and other legal arithmetic operations between combinations of points and vectors.
4. You are given almost no skeleton code. You are welcome to copy code from your implementation of project1 to get started.
5. If you add a cpp file, consult the contents in the last section of the file Makefile on how to add it to your build.
6. Review sections 2.11.4 and 2.11.5 of the text to know how to animate the balls.
7. Run `./check-code *.cpp *.h` frequently to make sure you are conformant with the coding guidelines.

Scoring

1. 10% - Render at least two balls, one of them white.
2. 20% - Use the correct mouse gesture to strike the white ball.
3. 20% - Realistic animation of the balls.
4. 30% - Realistic ball-wall and ball-ball interaction.
5. 10% - Additional features, listed in README.txt.
6. 10% - Coding quality and style given by report from check-code and visual inspection.

Notes

1. You may use the standard C++ library (e.g. vector). You *may not* use any other third-party libraries.
2. Run `./check-code *.cpp *.h` frequently as you program. Fixing all errors at the end is a real drag.
3. You can submit your work as frequently as you like – only the most recent submission will be retained. Suggestion: submit first thing to get familiar with how it works and submit occasionally during development. This way there won't be any surprises when you're up against the deadline.