CS4499
Project 2 - billiards
Due Sept 25, 5:00 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 WebGL
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. You must have at least two balls; one and only one of them should be white.
2. 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.
3. The balls must have walls to bounce off.
4. The balls must bounce off or interact with each other in some meaningful way.
5. 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.
6. List all additional features that you add in README.txt. If you do not list the features you will not get credit for them.
7. Run closure linter on `billiards.js` and any other javascript files you add.
8. Zip all files into project2.zip and submit your work using Moodle.

Hints

1. You will be using separate shaders for your balls and cue sticks. This means you will have two separate "programs". You may find it easier to first implement the shaders for the balls and get the balls rendering and then work on the cue stick.
2. Skeleton code for an animation technique is in `billiards.js`. This technique is more realistic than what we talked about in class. Fill in code as indicated by the TODO comments.
3. For full points, you may send to the gpu only vertices of a single ball. Each ball will be drawn after sending a transformation matrix which is applied in the vertex shader. See section 4.12.1 of the text.

4. 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.
5. You will likely make heavy use of the definition of the dot product (section 4.1.9 of the text), of its geometric interpretation (figure 4.14 of the text) and of the reflection vector we derived in class.
6. You are given almost no skeleton code. You are welcome to copy code from your implementation of project1 or any of the lecture code to get started.

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.

Notes

1. You may not use any third-party libraries other than what is bundled in projecd2.zip.