gl-billiards

Dan Lapp
Taras Mychaskiw
COSC 3P98 Project

January 8th, 2014

Abstract

Contents

1 Physics Engine

1.1 Taking a Shot

Once a ball is hit by the cue, there are two ways energy is transferred to the ball. First, and more noticeable of the two is the transfer of linear momentum. This sends the ball moving around the table in straight lines. Second is the transfer of angular momentum, which applies spin to the ball. Spin can cause the ball to stop in it's tracks after a collision, bounce off cushions in strange (but predictable) ways and to any pool player, is just as if not more important an aspect to the game as the linear momentum.

1.1.1 Linear Momentum

To simplify the shot making process, the pool cue is assuming to act like a spring. That is, the further you pull the cue away from the cue you are going to hit, the more powerful your shot will be. Spring motion can be described using Hooke's Law,

$$\vec{F} = -k\vec{x} \tag{1}$$

where, $\vec{F}$ is the force, $\vec{x}$ is how much the spring is stretched and k is the spring constant, which describes basically how powerful (springy) the spring is.
Hooke's Law is equivalent to Newton's Second Law, which states

$$\vec{F} = m\vec{a} \tag{2}$$

where, $\vec{F}$ is the force, m is the mass of the object and $\vec{a}$ is the acceleration of the object. Setting Equation 1 equal to Equation 2 gives

$$\begin{aligned} -k\vec{x} &= m\vec{a} \\ -k\vec{x} &= m\frac{\Delta\vec{v}}{\Delta t} \\ \Delta\vec{v} &= \frac{-k\Delta t}{m}\vec{x} \end{aligned}$$

where $\vec{v}$ is the velocity of the object, and t is the time. For pool, the initial velocity of the ball being struck is always zero, so we can simplify the above equation to

$$\vec{v} = \frac{-kt}{m}\vec{x} \tag{3}$$

where $\vec{x}$ is the distance between the ball and the tip of the cue, m is the mass of the ball, t is how much time the cue is in contact with the ball and k is the spring constant of the cue. The values for t and k needed to be experimentally found though trial and error, essentially trying values until what happened in the simulation closely represented real life. The values for t and k were found to be 0.1s and 10N/m.

1.1.2 Angular Momentum

Angular momentum transfer is more complex. The starting point for this section is also Newton's Second Law, but written in a form more usable to the problem at hand.

$$\vec{\tau} = I\vec{\alpha} \tag{4}$$

where $\vec{\tau}$ is the torque applied to the object around it's center of rotation, I is the moment of inertia of the object and $\vec{\alpha}$ is the angular acceleration of the object.
Torque is always given by the equation,

$$\vec{\tau} = \vec{r} \times \vec{F} \tag{5}$$

where $\vec{F}$ is the force being applied, $\vec{r}$ is the vector between where the force is being applied and the center of rotation of the object, and $\times$ is the vector cross product.
An important note is the inertia for a solid sphere is given by,

$$I_{sphere} = \frac{2}{5}mR^2 \tag{6}$$

where R is the radius and m is the mass of the sphere.
Setting Equation 4 equal to Equation 5, subbing in Equation 6 for I and Equation 1 for $\vec{F}$, and solving gives,

$$\begin{aligned} \vec{r} \times -k\vec{x} &= \frac{2}{5}mR^2\frac{\Delta\vec{\omega}}{\Delta t} \\ \Delta\vec{\omega} &= \frac{5}{2}\left(\frac{\Delta t}{mR^2}\right)(\vec{r} \times -k\vec{x}) \end{aligned}$$

where $\vec{\omega}$ is the angular velocity of the ball, and t t is how much time the cue is in contact with the ball (the same t as in the previous section). Again, for pool, when the ball is struck it is entirely at rest, so the initial angular velocity is zero. The above equation then simplifies to

$$\vec{\omega} = \frac{5}{2}\left(\frac{\Delta t}{mR^2}\right)(\vec{r} \times -k\vec{x}) \tag{7}$$

where $\vec{x}$ is the distance between the ball and the tip of the cue, m is the mass of the ball, R is the radius of the ball, t is how much time the cue is in contact with the ball, k is the spring constant of the cue and finally $\vec{r}$ is the vector between the center of the ball and where the cue strikes the ball.

1.2 Event Detection

For the purposes of this program, an event is a ball-ball collision, or a ball banking off of the cushion. The way gl-billiards works is each time the screen is redisplayed, some physics updater function is called with a parameter dt, which is how much time has passed since

the last frame. The engine simulates the pool game up until a time dt has passed. For example, if 1s was passed, then it would simulate 1s of play before returning, then the screen would be updated as if 1s had passed.
So, the engine determines which events occur within time dt. Event detection then becomes *when* do events happen instead of *if* they happened. The entire update function works as follows:
`update(`dt`)`

```
find earliest event that occurs within time dt, call it event
if no such event

  roll balls until dt

otherwise

  roll balls until event occurs
  handle event
  dt <-- dt - event.getTime()
  call update(dt) again
```

In this manner, all events that happen within dt and handled automatically before the frame updates and the balls slow down due to friction. Note that "roll balls" simply means each ball moves at it's speed for some amount of time. Pocketing detection needs to be done there, as it is not a standard event. That is, if a ball is pocketed, it cannot possible participate in any events in time dt, since it would bank off the cushion or collide with another ball first.

1.2.1 Collision Detection

1.2.2 Bank off the Cushion Detection

1.2.3 Pocketing Detection