For my simulations final project, I created a billiards simulation using OpenGL. Before every hit, the console asks the user to input the horizontal and vertical displacement of the cue - cue ball contact point from center as well as the velocity the cue is hitting the cue ball with. Then the user can choose the direction to hit the cue ball. After receiving the position of the contact point and initial velocity, the force of impact is calculated by $F = \frac{2mV_0}{1+\frac{m}{M}+\frac{5}{2R^2}(a^2+b^2cos^2\theta + c^2sin^2\theta - 2bc\ cos\ \theta sin\theta)}$ where c = $|\sqrt{R^2 - a^2 - b^2}|$, R is the radius of the balls, m is the ball mass, M is the cue mass, and V_0 is the initial cue velocity. Then we calculate the initial velocity using $v = (0,\ -\frac{F}{m}cos\theta,\ -\frac{F}{m}sin\theta)$ and the initial angular velocity with $w = \frac{1}{I}(-cFsin\theta + bFcos\theta,\ aFsin\theta,\ -aFcos\theta)$. By setting a to a non zero float, we can generate sidespin where if a is more than 0, then we have right spin, otherwise, we have left spin. Setting b to a non zero float allows us to generate forward spin if b is greater than 0 and backspin otherwise. Setting both to 0 generates a normal center ball shot with no angular velocity.

The cue ball goes through two stages of movements, sliding and rolling. We can calculate the sliding velocity using $v(t) = v_0 - \mu_s g t u_0$ and $w(t) = w_o - \frac{5\mu_s g}{2R}t(k\ x\ u_o)$ where μ_s is the friction constant, g is the gravity constant, and $u_0 = v(t) + Rk$ x w(t) is the relative velocity between the ball and the table surface. The duration of the sliding state is $\Gamma_s = \frac{2|u_0|}{7\mu_s g}$ and when the time step passes this point, we are in rolling state. The velocity for rolling state is $v(t) = v_0 - \mu_s g t^2 v_0$ and $w(t) = \frac{|v(t)|}{R}$. Depending on the time step, we use the appropriate velocity measurements for the 2 states and we update the position by adding the delta timestep to the velocity. Since the velocity calculations outputs the velocity in the x direction, we have to rotate the resulting vector to the cue direction vector using the standard 2d rotation matrix. After updating the positions, we test for ball-ball collisions between all the balls using conservation of energy and momentum and wall collisions simply by negating the velocity if the ball is out of bounds. When all the balls stop moving, the console once again asks the user for the information and repeats the process.