AN EVENT-BASED POOL PHYSICS SIMULATOR

Will Leckie[1] *and Michael Greenspan*[2]

Department of Electrical and Computer Engineering, Queen's University, Canada

ABSTRACT

A method to simulate the physics of the game of pool is presented. The method is based upon a parametrization of ball motion which allows the time of occurrence of events, such as collisions and transitions between motion states, to be solved analytically. It is shown that the occurrences of all possible events are determined as the roots of polynomials up to fourth order, for which closed-form solutions exist. The method is both accurate, returning continuous space solutions for both time and space parameters, and efficient, requiring no iterative numerical methods. It is suitable for use within a game tree search, which requires a great many potential shots to be modeled efficiently, and a robotic pool system, which requires high accuracy in predicting shot outcomes.

1. INTRODUCTION

Pool and its variations and close relatives billiards, carom, and snooker, are collectively classified as cue sports. While the precise origins of pool are unknown, it is an ancient game believed to have evolved from a common ancestor of golf and croquet. Shakespeare made reference to the game, and Marie Antoinette was known to be an enthusiast. There has recently been a resurgence of interest in pool world-wide: pool was recognized as a demonstration sport by the International Olympic Committee at the 1998 Nagano Olympics, and it is estimated that in 2003 over 40.7 million people picked up a cue in the U.S. alone.

The first effort to develop a robotic system to play a cue sport was the Snooker Machine of Khodabandehloo et al. [Shu (1994)], which was developed in the early 1990s. There are currently a number of such systems under development, including Deep Green [Long *et al.* (2004)], the Pool Sharc [Alian *et al.* (2004)], and others such as [Chua *et al.* (2002); Chua *et al.* (2003); Lin *et al.* (2004); Cheng *et al.* (2004)]. To play pool robotically is a challenging objective, and greater still to achieve a level of play strong enough to compete against a proficient human. In addition to the purely robotic aspects of the problem such as computer vision and mechatronics, which involve accurate sensing and actuation, there is a significant amount of strategy involved in playing pool. Placement of the cue ball following a shot is considered one of the key elements to successful play, and even moderately accomplished players tend to plan at least 3 shots ahead. In this way pool bears a similarity to other games of strategy, such as chess and checkers. One significant difference is that chess and checkers are played on a board with discrete positions so that the number of possible board states is finite, although huge. In contrast, a pool table is a continuous domain, with a truly infinite number of possible table states.

To accurately predict the outcome of a shot (*i.e.*, the final rest locations of all balls) requires a knowledge of physics. Game strategy and physics are therefore intertwined, which is one of the intriguing aspects of the game. Most players have an intuitive understanding of the physics involved, a result of heuristics and many hours of observation. Computational pool, however, requires an explicit physical model of the balls, cue, and table, and their interactions. An early investigation of pool physics was explored by Coriolis, and the most thorough treatment to date is that of Marlow [Marlow (1995)]. Recent additions include a monograph by Shepard [Shepard (1997)], who extended aspects of Marlow's work and also explored the statistical basis of certain strategies, as well as others [Petit (1997), Koehler (1989)].

[1]email:4wl1@qlink.queensu.ca
[2]email:greensm@post.queensu.ca

This paper describes the development of a pool physics simulator that is both accurate and efficient. This simulator is being developed for use within the gaming component of the Deep Green system, and it will also be distributed as the physics model behind the Computational Pool Tournament of the 10^{th} Computer Olympiad. There currently exist a number of commercial and online pool simulators, each of which have attractive graphics, automatic strategies, and an underlying physics model. While many of these games are based upon seemingly realistic physics models, pool players have varied opinions about the level of realism and usefulness of these simulators, and there are some situations where anomolies occur. All of these simulators are closed systems, with the noteable exception of foobilliard [Berger (2000)].

Our objective was to develop a pool physics simulator that was both highly accurate and efficient, which can be used by both the gaming and robotics communities to further the development of computational and robotic pool. For robotic pool it is necessary to have a highly accurate physics simulator due to the continuous nature of the game, so that the outcome of each shot can be predicted accurately. For computational pool, it is important that the physics simulator be efficient, as it may be called repeatedly to evaluate potential shots within a game tree. To satisfy both of these criteria, we have foregone the standard integration method of simualtion, which discretizes time, in favour of a event-based method that solves each shot analytically over a continuous time domain. In addition to being time efficient and accurate, a further benefit is that order of events, which may in certain cases be obscured by the integration method, will be preserved.

This paper continues in Section 2 with a description of the physics model. Section 3 describes the various motion states and events that can occur, and describes a method to analytically predict the time of each event. The method is discussed in Section 4 and the paper concludes in Section 5 with a description of future work.

2. POOL PHYSICS

In this section we model the physics of the impact of the cue and the ball, the resulting ball motion, and the collision of the ball with the rails or other balls.

2.1 Cue - Ball Impact

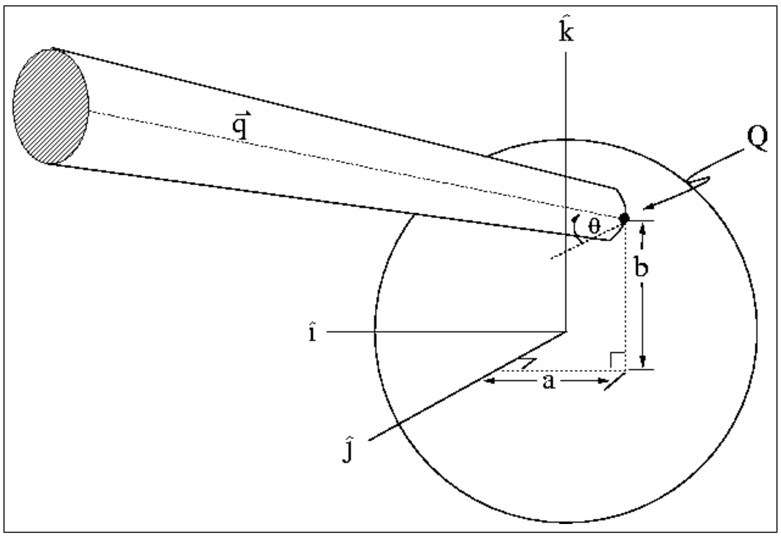

Figure 1: Cue Impact

Each shot commences with the cue striking the cue ball. Let $(\hat{\imath}, \hat{\jmath}, \hat{k})$ be the ball-centric coordinate reference frame illustrated in Fig.1. The central axis of the cue is denoted by a vector $\vec{q}$, which by definition is parallel to the vertical $\hat{\jmath}$-$\hat{k}$ plane. Vector $\vec{q}$ is oriented at an angle θ to the horizontal $\hat{\imath}$-$\hat{\jmath}$ plane, and we make the simplifying assumption that the cue contacts the ball at a single point $Q \in \vec{q}$, thereby ignoring the geometry of the cue tip. If R is the radius of the ball, a and b are the respective horizontal and vertical displacements of Q from the ball's center, and $c = |\sqrt{R^2 - a^2 - b^2}|$, then $Q = (a, c, b)$. For Q to lie on the ball's surface we must have $a^2 + b^2 \leq R^2$, so the $\hat{j}$ component of Q must always be real and non-negative.

The magnitude F of the force that the impact imparts to the ball is given by:

$$F = \frac{2mV_0}{1 + \frac{m}{M} + \frac{5}{2R^2}\left(a^2 + b^2\cos^2\theta + c^2\sin^2\theta - 2bc\cos\theta\sin\theta\right)} \tag{1}$$

where m is the ball mass, M is the cue mass, and V_0 is the initial velocity of the cue at impact. Assuming that the time duration of the collision is small[3], integrating Newton's Second Law gives $\vec{F} = m\vec{v}$, where $\vec{v}$ is the resulting post-impact instantaneous linear velocity of the ball. This velocity is expressed in the ball-centric coordinate reference frame as:

$$\vec{v} = \left(0,\ \frac{-F}{m}\cos\theta,\ \frac{-F}{m}\sin\theta\right) \tag{2}$$

The non-zero $\hat{k}$ component of $\vec{v}$ indicates that the ball has a component of initial linear velocity in the vertical direction when the cue is not horizontal, which is the basis of the *jump* shot. The event-based solution method we develop later in this paper could be easily extended to deal with three-dimensional ball motion, but in our model we ignore this $\hat{k}$-component of the cue ball's initial velocity and assume that all balls are confined to movement on the surface of the table, which is true for the vast majority of shots encountered in competitive play.

In addition to linear velocity, the collision with the cue also imparts an angular velocity to the ball whenever a, b or θ are nonzero. The ability to control the angular velocity (i.e. *spin*) of the ball is an essential skill in mastering the game. Players conceptualize the following 3 types of spin:

- Follow: i.e., *top spin*. Follow is achieved by striking the ball above center ($b > 0$), so that there is a component of spin around the ball's positive $\hat{\imath}$-axis. The usual desired effect of follow is to have the cue ball continue forward motion after collision with an object ball.
- Draw: i.e., *back spin*. Draw is the opposite of follow, and is achieved by striking the ball below center ($b < 0$). After an object ball collision, the cue ball reverses direction.
- English: i.e., *side spin*. Left or right English is achieved by striking the ball slightly left ($a > 0$) or right ($a < 0$) of center respectively, resulting in a rotation around the ball's $\hat{k}$-axis. There are a number of possible effects of English, including altering the angle of incidence of the cue and/or object ball following a collision. English can also be communicated to object balls.

Given the moment of inertia of a solid sphere as $I = \frac{2}{5}mR^2$, the post impact instantaneous angular velocity of the ball within the ball frame is:

$$\vec{\omega} = \frac{1}{I}\left(-cF\sin\theta + bF\cos\theta,\ aF\sin\theta,\ -aF\cos\theta\right) \tag{3}$$

It is interesting to note that instances of the above itemized spins can all be achieved with a horizontal cue (*i.e.* $\theta = 0$), in which case the $\hat{j}$-component of $\vec{\omega}$ vanishes. When $\vec{\omega}_y = 0$, the ball travels with straight rectilinear motion[4]. There is a fourth type of spin, usually attempted only by more advanced players, wherein the cue is elevated at an extreme angle nearly perpendicular to the table ($\theta \sim 90^o$). This causes a large $\vec{w}_y$ component resulting in a curvilinear ball motion and is the basis of *curve* and *massé* shots.

2.2 Ball Motion

As the ball moves across the table, the only unbalanced external force acting on it is friction with the table. This friction acts at the point of contact between the ball and the table surface, in a direction opposite to the ball's motion. We approximate this as a single point of contact, although in reality the ball compresses the fibers of the table cloth and sinks slightly into the cloth so that the point of contact is actually a small portion of the ball's spherical surface.

[3]Marlow empirically determined the collision duration to be $\sim 200\mu$sec at a speed of 1 m/s [Marlow (1995), p45]

[4]Many people believe that the use of English necessarily causes the cue ball to travel with curvilinear motion, which is incorrect. The ball travels in a straight trajectory unless struck with an elevated cue stick.

When a ball is struck by the cue, it begins its motion by sliding across the surface of the table. After some time, the interplay between the table friction and the ball's linear and angular velocities causes the ball to begin rolling. It takes a very specific combination of the ball's linear and angular velocities to cause it to transition from the sliding to the rolling state. Suppose the ball is moving in the $\hat{\imath}$ direction. Intuitively, the ball will be rolling only when it makes one full revolution about the $\hat{\jmath}$ axis for every distance $2\pi R$ that it travels along the table, equal to the circumference of the ball.

The concept of *relative velocity* was introduced by Marlow [Marlow (1995)] for explaining the ball motion and classifying it as either sliding or rolling. The relative velocity $\vec{u}(t)$ of the point P at the bottom of the ball that is in contact with the table surface is:

$$\vec{u}\left(t\right) = \vec{v}\left(t\right) + R\hat{k} \times \vec{w}\left(t\right) \tag{4}$$

Note that the relative velocity $\vec{u}(t)$ has no $\hat{k}$-component since $\hat{k} \times \hat{k} = 0$. As a result, the angular velocity about the vertical $\hat{k}$ axis does not couple into the translational motion of the ball and must be treated separately as we shall see below.

From Newton's Laws we have the following equations governing the sliding ball's state variables for position $\vec{r}\left(t\right)$, velocity $\vec{v}\left(t\right)$ and angular velocity $\vec{\omega}\left(t\right)$ as a function of time t, expressed in a frame of reference attached to the center of the ball such that the $\hat{\imath}$ axis of this frame is along the ball's direction of motion:

$$\vec{r}_B\left(t\right) = \left[\begin{array}{c} x_B\left(t\right) \\ y_B\left(t\right) \end{array}\right] = \left[\begin{array}{c} v_0 t - \frac{1}{2}\mu_s g t^2 \hat{u}_{0x} \\ -\frac{1}{2}\mu_s g t^2 \hat{u}_{0y} \end{array}\right] \tag{5}$$

$$\vec{v}_B\left(t\right) = \vec{v}_0 - \mu_s g t \hat{u}_0 \tag{6}$$

$$\vec{\omega}_B\left(t\right) = \vec{\omega}_{\|}\left(t\right) = \vec{\omega}_0 - \frac{5\mu_s g}{2R} t \left(\hat{k} \times \hat{u}_0\right) \tag{7}$$

The $_B$ subscripts in the above equations indicate that the equations are expressed in the ball frame. The subtractive terms represent the effect of the table friction slowly bleeding off the ball's speed and spin as time progresses. The $_0$ subscripts denote the initial values, and $_x$ represents the $\hat{\imath}$-component of a vector, for example. The gravitational constant is represented by g and μ is the dimensionless coefficient of friction, which takes on values between 0.0 and 1.0 and results from the material properties of the ball and table cloth surfaces. Larger values of μ correspond to larger frictional forces which reduce the ball's speed and spin more quickly.

The $\hat{k}$ components of $\vec{r}\left(t\right)$ and $\vec{v}\left(t\right)$ are assumed to be zero at all times as a result of our earlier assumption that the ball is constrained to movement on the surface of the table. Eq.7 is a three-dimensional vector equation for the angular velocity, but since the $\hat{k}$-component of $\hat{k} \times \hat{u}_0$ is zero, the $\hat{k}$ component of the angular velocity given by Eq.7 does not evolve with time. This is a result of our earlier assumption of a single point of contact between the ball and the table cloth. We denote $\vec{\omega}_B\left(t\right)$ as $\vec{\omega}_{\|}\left(t\right)$ since it always lies in the plane of the table, the $\hat{\imath} - \hat{\jmath}$ plane. We use the simple model given by Eq.8 to calculate the vertical component of the ball's angular velocity as a function of time, which we denote by $\omega_\perp$:

$$\omega_\perp\left(t\right) = \omega_{\perp 0} - \frac{5\mu_{sp} g}{2R} t \tag{8}$$

In practice the coefficients of friction governing the sliding and the spinning motion of the ball are not equal and are denoted by μ_s and μ_{sp}, respectively.

To find the position of the ball in the table frame as a function of time, $\vec{r}\left(t\right)$, we first express $\vec{r}_B\left(t\right)$ in the table frame using a simple rotation:

$$\vec{r}_T\left(t\right) = \left[\begin{array}{cc} cos\left(\theta\right) & -sin\left(\theta\right) \\ sin\left(\theta\right) & cos\left(\theta\right) \end{array}\right] \vec{r}_B\left(t\right) \tag{9}$$

The position of the ball in the table frame is then $\vec{r}\left(t\right) = \vec{r}_0 + \vec{r}_T\left(t\right)$.

By combining Eqs.4, 6 and 7, the relative velocity evolves with time according to:

$$\vec{u}(t) = \vec{u}_0 - \frac{7}{2}\mu_s g t \hat{u}_0 \tag{10}$$

The sliding state lasts until $\vec{u}(t) = 0$, so from Eq.10 the duration τ_S of the sliding state is:

$$\tau_S = \frac{2|\vec{u}_0|}{7\mu_s g} \tag{11}$$

The rolling state is defined by the condition $\vec{u}(t) = 0$, whence $|\vec{v}(t)| = R\left|\vec{w}_\parallel(t)\right|$. When $\vec{u}(t) = 0$ the ball travels one circumference in the $\hat{\imath}$ direction for every rotation about the $\hat{\jmath}$-axis as shown in Fig. 2.

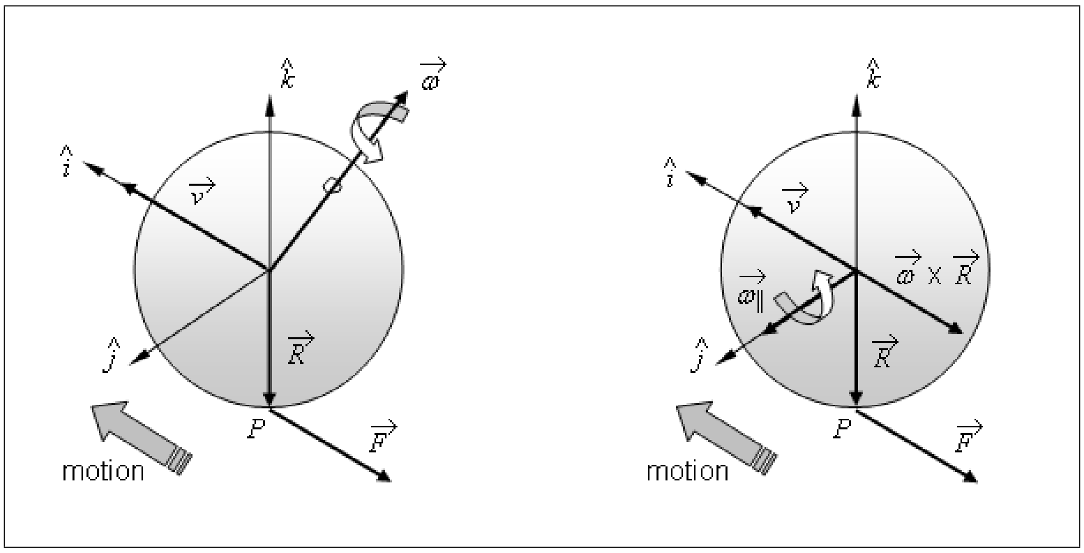

Figure 2: Ball velocity $\vec{v}$ and angular velocity $\vec{\omega}$ during the sliding state (left) and the rolling state (right). During the sliding state the angular velocity $\vec{\omega}$ is arbitrary and the relative velocity $\vec{u}$ is non-zero. During the rolling state the relative velocity is zero and the angular velocity lies in the $\hat{\imath}$-$\hat{k}$ plane with $|\vec{\omega}_\parallel| = |\vec{v}|/R$. The force of friction acting on the ball is denoted by $\vec{F}$ in the figure.

Once the relative velocity has vanished and the ball has started rolling, it continues rolling in a straight trajectory because there is no angular velocity around the axis along which the ball is moving. During this *natural roll* state the ball's state variables evolve according to:

$$\vec{r}(t) = \vec{r}_0 + \vec{v}_0 t - \frac{1}{2}\mu_r g t^2 \hat{v}_0 \tag{12}$$

$$\vec{v}(t) = \vec{v}_0 - \mu_r g t \hat{v}_0 \tag{13}$$

$$\left|\vec{\omega}_\parallel(t)\right| = \frac{|\vec{v}(t)|}{R} \tag{14}$$

Once again, the subtractive terms in Eqs.12, 13 and 14 represent the effect of table friction, where μ_r is the coefficient of rolling friction, in general different from μ_s[5]. By definition, during the rolling state the direction of $\vec{\omega}_\parallel$ is perpendicular to the direction of the ball's motion as shown in Fig.2. During the rolling state the vertical component of the angular velocity, $\vec{\omega}_\perp$, continues to evolve according to Eq.8.

The natural roll state lasts until the table friction reduces the ball's velocity to zero, so from Eq.13 the duration of the rolling state is:

$$\tau_R = \frac{|\vec{v}_0|}{\mu_r g} \tag{15}$$

[5]Marlow experimentally measured μ_{sp}=0.044, μ_r=0.016 and thereby calculated μ_s=0.2 [Marlow (1995), p237]

Since $\mu_s \neq \mu_r$, the motion of a moving billiard ball is quantitatively different during the sliding and rolling states. During the natural roll state, the ball always travels in a straight trajectory because the component of spin about the ball's axis of motion is zero by definition. During the sliding state, however, any component of spin about the axis of motion, resulting from $\theta > 0$, causes the ball to move in a curved path, as evident in the non-zero y-component of the ball position as expressed in the ball frame from Eq.5. A computer model attempting to accurately predict the dynamics of the game, including the curvilinear motion during the sliding state, must therefore treat the sliding and rolling motions separately.

2.3 Collisions

Once a ball is in motion, there are two types of collisions that can occur: a collision with another ball, and a collision with a rail (*cushion*). A simple model of the dynamics of a collision between two balls begins by assuming that there is no friction between the balls and therefore no transfer of spin/English between them. The transfer of linear momentum between the balls is simple to calculate and the post-collision velocities of the balls easily obtained.

Experienced pool players know that in reality a small amount of friction exists between two balls in contact with one another, adding a key dimension to the game. This friction is the source of the important effect known as *throw*, in which the spin of the cue ball is communicated to the object ball during the short time of the collision, causing a deflection of the object ball from the intended post-impact trajectory. The effect of throw is small but must be taken into account for any measure of success on long shots down the table.

Marlow treats the most general case of the collision by including ball-ball friction using linear and angular momentum conservation laws as a starting principle. This results in a system of 16 equations in 16 unknowns which is solved to obtain four equations for the post-impact velocities and angular velocities for the two balls [Marlow (1995)]. Our model uses Marlow's equations in order to solve the dynamics of the ball-ball collision.

Models of the ball-cushion interaction often involve some model of a damped-spring system. Some of the ball's kinetic energy is absorbed by the cushion, and the angular velocity of the ball is altered by the collision. We use Marlow's damped-spring model to calculate the post-impact linear and angular velocities for the ball-rail collision.

3. EVENT-BASED SIMULATION

Using the equations presented in the Sec.2, we now consider the problem of simulating the physics of the game. The chosen simulation method must be computationally efficient because it will be used in a game strategy algorithm to expand a search tree, which necessitates invoking the method for a large number of potential shots. We also strive for physical accuracy in the simulation, both to make a simulated game more realistic and to allow its use within a robotic system, wherein slight inaccuracies can result in missed shots.

3.1 Discrete Numerical Integration

The obvious first option for a simulation method is numerical integration. With full knowledge of the state variables for a ball's position $\vec{r}\,(t)$, velocity $\vec{v}\,(t)$ and angular velocity $\vec{\omega}\,(t)$ at a given time t, these state variables are advanced forward in time by a small discrete amount Δt, using some discrete numerical integration technique. The simplest approach equates the acceleration of the ball to the time-derivative of its velocity, and integrates the acceleration equation assuming that the acceleration is constant over Δt. This gives the velocity after Δt in terms of the previous time step's velocity and the (constant) acceleration: $\vec{v}\,(t + \Delta t) = \vec{v}\,(t) + \vec{a}\Delta t$. Similarly, integrating the velocity and assuming it is constant over the tiny Δt yields the position $\vec{r}\,(t + \Delta t) = \vec{r}\,(t) + \vec{v}\Delta t$. This method is the standard approach for modeling physics in human-interactive computer games of all kinds, where the emphasis is more on apparent realism than on physical accuracy.

In pool, use of the integration method boils down to the problem of the discovery of events *after* they have occurred, and then handling them using the appropriate physics. After each time step, the simulator must consider all ball positions, velocities and angular velocities to determine whether any events such as ball collisions or rail collisions have occurred during the previous Δt. Due to the continuous measurement domain, it is highly improbable that an event occurs *exactly* at the beginning or end of a time step; it is much more likely that an event occurs sometime within the time step, i.e. at a fractional Δt.

For example, assume that two moving balls collide at time t_c within time step s_i, where the duration of s_i is $[t_{i0}, t_{if}]$; $t_{if} - t_{i0} = \Delta t$. The separation distance D of the centers of the balls at t_c is exactly equal to $2R$. Prior to t_c, their separation $D > 2R$, and provided Δt is small enough relative to the linear velocities of the balls, then $D < 2R$ within the period $[t_c, t_{if}]$. Upon inspection at time t_{if}, the simulator would realize that a collision must have occurred sometime during s_i, because $D < 2R$. The simulator would then have to back up in time to t_c in order to apply the appropriate equations governing the collision. Backing the model up by a fraction of Δt to calculate the exact values of the state variables at t_c adds complexity and the possibility of lost accuracy. Calculating the exact value of t_c without making any simplifying assumptions that may limit accuracy is itself a challenging problem. Another option is to iteratively return to t_{i0} and simulate forward with a smaller time step, iterating until the time step has been reduced such that $\Delta t \approx t_c - t_{i0}$, in order to advance forward from t_{i0} to t_c with no approximations. This option adds complexity and decreases efficiency.

In addition, a common situation in pool is that of *frozen balls*, where two or more balls are stationary and in contact before being struck by another moving ball. The discovery-based integration method would have difficulty simply recognizing and handling the events in the correct order. Modeling a break shot this way is difficult, and computer pool games often use either a heavily-approximated solution, or some randomized selection from an encyclopedia of pre-computed break shots, instead of attempting to generate a new break shot each time.

There are therefore several intertwined challenges associated with the integration method applied to modeling pool physics. Choosing a reasonable time step Δt is a difficult trade-off between speed and accuracy, and dealing with the fact that events usually occur within a time step necessitates complexity and approximations. A smaller time step results in a more accurate model for two reasons; the assumption that the speed of the ball is constant over the time step is more valid as Δt is reduced, and with a smaller time step any approximations needed to back the model up by a portion of a time step will be more accurate. However, with a larger time step the model can predict the eventual outcome much faster since fewer computations are necessary. Given these challenges and drawbacks, we looked to another approach to model the outcome of a shot.

3.2 Continuous Event-based Simulation

Since the equations developed in Sec.2.2 provide the value of the ball state variables at any time t, we chose to take an event-oriented approach that allows us to solve the ball dynamics problem exactly, in the continuous domain in both time and space.

3.2.1 Motion States and Events

The approach is based upon the observation that a ball can be in one of four possible *motion states* at any time t:

- SLIDING, defined by $\vec{v}(t) \neq 0$ and $\vec{u}(t) \neq 0$, during which the ball slides across the table;
- ROLLING, defined by $\vec{v}(t) \neq 0$, and $\vec{u}(t) = 0$, during which the ball rolls across the table;
- SPINNING, defined by $\vec{v}(t) = \vec{u}(t) = 0$ and $\vec{\omega}(t) = \pm|\vec{\omega}(t)|\hat{k}$, during which the ball spins in place about the vertical axis;
- and STATIONARY, defined by $\vec{v}(t) = \vec{u}(t) = \vec{\omega}(t) = 0$, when the ball is completely stationary.

An *event* is defined as the transition of a ball from one motion state to another. A *motion transition* event occurs simply through the passage of time as a ball's speed and spin evolve under the influence of the table friction. In addition, there are a number of *collision* events that occur when two objects impact, resulting in a change of the ball's linear and angular velocities and therefore a transition to a different motion state. Events are classified into the following five categories:

- CUE-STRIKE, in which the cue ball is struck by the cue stick;

- BALL-COLLISION, in which two balls collide with each other;
- RAIL-COLLISION, in which a given ball collides with a cushion on the table;
- POCKET-COLLISION, in which a given ball is pocketed;
- and MOTION-TRANSITION, in which a given ball transitions from one motion state to another through the passage of time.

3.2.2 Event Time Prediction

Using this framework, the simulation problem boils down to one of *prediction* of when the next event occurs. When modeling a shot, we calculate the time τ_E at which the next event E occurs and advance each ball's state variables $\vec{r}\,(t)$, $\vec{v}\,(t)$ and $\vec{\omega}\,(t)$ forward in time to $t = \tau_E$ using the appropriate equations from Section 2.2. We then apply the appropriate physics for the event, for example calculating the post-collision instantaneous linear and angular velocities of two balls following a BALL-COLLISION event. This is an attractive solution method because the dynamics of the moving balls are calculated completely analytically, with no approximations, and are subject only to floating point (double precision) round off error.

The problem of predicting τ_E for the collision of two balls with one or both of them moving is one of the more interesting problems that the event-based method must handle. The event will occur at the time τ_E at which the pair of balls have a distance of separation D between their centers equal to $2R$. The separation of the two balls, denoted respectively by subscripts 1 and 2, is a time-dependent vector $\vec{d}\,(t)$ given by:

$$\vec{d}\,(t) = \vec{r}_2\,(t) - \vec{r}_1\,(t) = \begin{bmatrix} a_x t^2 + b_x t + c_x \\ a_y t^2 + b_y t + c_y \end{bmatrix} \tag{16}$$

where the coefficients a, b and c are obtained by collecting terms appropriately from the expansion of $\vec{r}_2\,(t) - \vec{r}_1\,(t)$ using the parameterization for $\vec{r}\,(t)$ given by Eq.5 or 12. In the expansion, the value of μ used is either μ_s if the ball is in the SLIDING state, μ_r if it is in the ROLLING state, or 0 otherwise. The expansion is written in component form in Eq.16 with a_x denoting the $\hat{\imath}$-component of the t^2 coefficient of $\vec{d}\,(t)$, for example.

The time of collision τ_E that we are interested in finding corresponds to the solution of $D^2 = |\vec{d}\,(t)\,|^2 = 4R^2$. Expanding Eq.16 we obtain:

$$\begin{aligned}&\left(a_x^2 + a_y^2\right) t^4 + \left(2a_x b_x + 2a_y b_y\right) t^3 + \\ &\qquad \left(b_x^2 + 2a_x c_x + 2a_y c_y + b_y^2\right) t^2 + \left(2b_x c_x + 2b_y c_y\right) t + c_x^2 + c_y^2 - 4R^2 = 0\end{aligned} \tag{17}$$

This quartic polynomial is solved in closed form using standard analytical methods. The four unique and possibly complex roots of Eq.17 represent possible values of τ_E and must be carefully interpreted to determine which among them is physically possible. If the balls do indeed collide, then the correct root corresponding to τ_E is the smallest positive real root of Eq.17 that results in both ball centers lying somewhere on the table surface (i.e. within the rail bounds) within the lifetime of both balls' present motion states. If no roots are found that meet these criteria, then it is concluded that this pair of balls do not collide on the table surface within their current motion states and that some other event (such as a MOTION-TRANSITION or RAIL-COLLISION) must precede the collision.

At this point it is worth noting that relaxing our earlier assumption that all balls are constained to move on the surface of the table and including $\hat{k}$-components of position $\vec{r}\,(t)$ and velocity $\vec{v}\,(t)$ would result in a different functional form of the coefficients a, b and c in Eq.16 and Eq.17 but that, importantly, these coefficients would still be time-independent. As a result, Eq.17 would still be fourth-order and a solution for the time of collision could still be obtained using analytical methods.

We follow a similar approach to determine the time at which a given ball collides with a rail or a pocket. We parameterize the ball trajectory as a function of t and the straight line representing the rail or pocket mouth as a function of s. Since the ball's position vector $\vec{r}\,(t)$ refers to its center, when predicting rail collisions we look for the intersection of the ball's center and a horizontal line parallel to the rail but one ball radius toward the table center. For example, for a horizontal rail and a ball in the natural roll state, we would solve:

$$\begin{bmatrix} s + R \\ 0 \end{bmatrix} = \begin{bmatrix} x_0 + v_{0x} t - \frac{1}{2}\mu g t^2 \hat{v_{0x}} \\ y_0 + v_{0y} t - \frac{1}{2}\mu g t^2 \hat{v_{0y}} \end{bmatrix} \tag{18}$$

where the left hand side is the parameterization of the rail and the right hand side is the parameterization of the ball's trajectory $\vec{r}(t)$ in the rolling state, as given by Eq.12. The right hand side is written in component form and the variables are expressed in the table frame. Again, we use the appropriate value of μ depending on the ball's motion state. Other rails and the six pockets are parameterized in a similar way.

Eq.18 is easily solved using simple algebra and the quadratic formula, which returns two possibly complex roots corresponding to the time of collision τ_E. Again, the two roots must be carefully interpreted to determine whether either correspond to a physically meaningful solution.

3.2.3 Summary of the Algorithm

A summary of our event-based algorithm is presented in Figure 3.

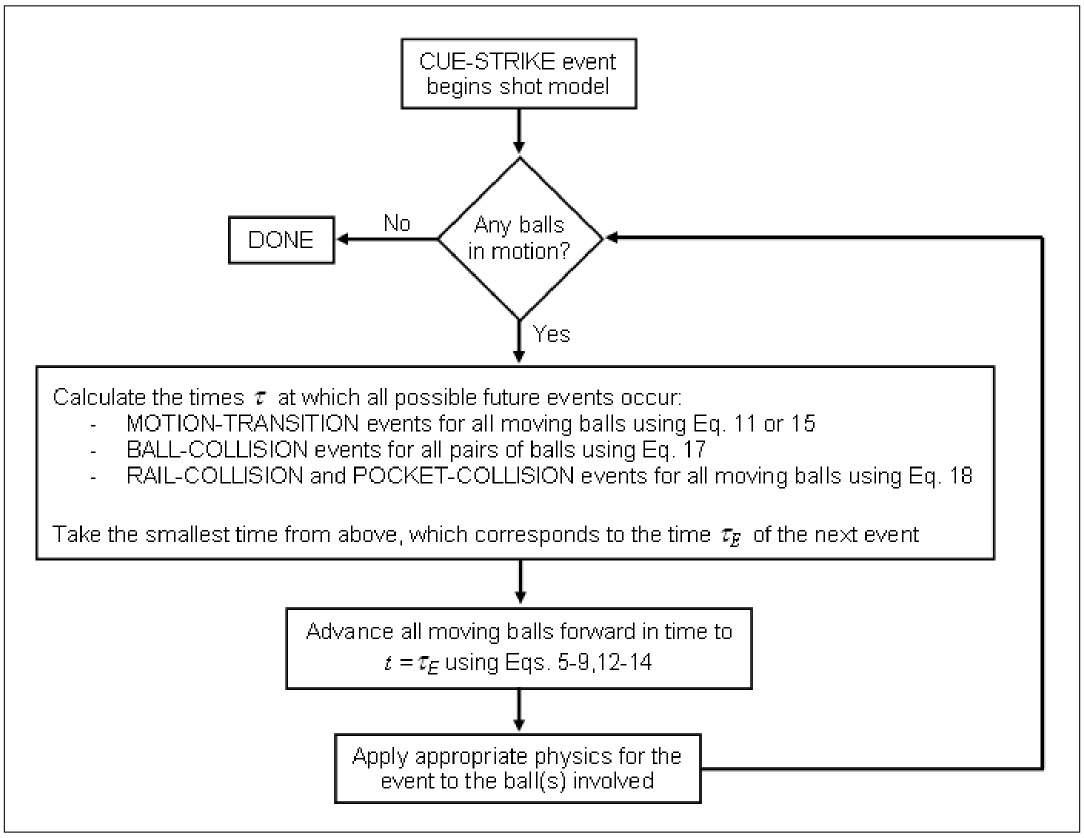
CUE-STRIKE event begins shot model
Any balls in motion?
No
DONE
Yes
Calculate the times τ at which all possible future events occur:
- MOTION-TRANSITION events for all moving balls using Eq. 11 or 15
- BALL-COLLISION events for all pairs of balls using Eq. 17
- RAIL-COLLISION and POCKET-COLLISION events for all moving balls using Eq. 18
Take the smallest time from above, which corresponds to the time τ_E of the next event
Advance all moving balls forward in time to $t = \tau_E$ using Eqs. 5-9,12-14
Apply appropriate physics for the event to the ball(s) involved

Figure 3: Event-based algorithm for modeling pool physics.

4. DISCUSSION

Since the event-based algorithm we have presented is completely analytical and uses no approximations in determining the time of the next event and advancing the state of the table forward in time to the next event, it has the potential to be more accurate than the numerical integration method. There is no need to carefully choose the time step Δt and there is no loss of accuracy arising from approximations to back the model up by a fraction of a time step in order to handle an event. In addition, there is no loss of accuracy due to the approximation that the velocity is constant during the time step. The only limitations on the accuracy of our method are floating-point (double) precision, the values of physical constants such as μ used, and any approximations used to model the physics of a collision event. All of these limitations are also present in the integration method.

The efficiency of the event-based method is another interesting topic. During many billiards shots relatively long time periods can pass where no events occur at all. For example, on a long cue ball-object ball shot down the table, one or two seconds can pass while the cue ball rolls down the table before it strikes the object ball, during which time no events occur whatsoever. Suppose that there are n balls in motion at some time $t = \tau$ (where $n_{max} = 16$ since there are at most 16 balls on the pool table), and suppose that time $\Delta\tau$ passes during which none of the moving balls transition to another motion state or collide with another ball or rail. Let the integration-style method use a time step Δt such that there are $N = \Delta\tau/\Delta t$ time steps in $[\tau, \tau + \Delta\tau]$. To simulate the event-free ball dynamics for this time period $\Delta\tau$, the integration-style model must perform:

- $3nN$ operations to step the moving balls' state variables ahead, since there are N time steps, n moving balls, and 3 state equations for each moving ball;
- $12nN$ operations to test for ball-rail/pocket collisions, since there are 6 rails, 6 pockets, N time steps, and n moving balls;
- and $N\left(n\left(n-1\right)/2 + n\left(n_{max} - n\right)\right) = N\left(31n - n^2\right)/2$ operations to test for ball-ball collisions, since there are $n\left(n-1\right)/2$ tests for pairs of moving balls, $n\left(n_{max} - n\right)$ tests for moving-stationary ball collisions, and N time steps.

The total number of operations required is $N\left(61n - n^2\right)/2$, with the number of operations increasing linearly with decreasing time step Δt.

Contrast this result with the requirements for the event-based method:

- n operations to predict the next motion state transition for all of the moving balls;
- 24n operations to predict the next ball-rail/pocket collision, since there are 6 rails, 6 pockets, and 2 operations are required for each quadratic solution;
- $19(n\left(n-1\right)/2 + n\left(n_{max} - n\right)) = 19\left(31n - n^2\right)/2$ operations to predict the next ball-ball collision, since there are $n\left(n-1\right)/2$ tests for pairs of moving balls, $n\left(n_{max} - n\right)$ tests for moving-stationary collisions, and 19 operations needed to solve each quartic equation;
- and $3n$ operations to advance all of the moving balls' state variables ahead to the time of the next event.

The total number of operations required is $\left(645n - 19n^2\right)/2$, independent of the amount of time $\Delta\tau$ until the next event.

With three moving balls and $\Delta\tau = 1$ second, for example, the integration method with a rather coarse time step of $\Delta t = 1$ millisecond needs 87000 operations to solve this one second of the dynamics, while the event-based method needs just 882. A typical shot sees perhaps five to ten events occur, while a perfectly executed shot involving only the cue ball, one object ball and a pocket may see even less than five events occur. It is clear that the workload of our event-based approach is not very heavy even when modeling more complicated dynamics.

There are several other advantages to using the event-based approach. The dynamics and end result of a given shot can be completely and precisely stored by recording only the initial table state ($i.e.$ ball locations) just before the shot as well as a time-stamped list of the events that occur during a shot. A shot can then be played back/animated after the fact using only the recorded initial table state and the list of events, with no need to re-model all of the physics of the shot.

Since the event-based approach needs to solve polynomial equations up to order four, its accuracy depends heavily on the numerical accuracy of the polynomial solver and the interpretation of the results. Closed form analytical formulae exist for the solutions of polynomials up to fourth-order, but care must be taken when implementing these formulae. For example, a fourth-order equation often has at least one pair of complex-conjugate roots. One challenge lies in determining how "real" or how "complex" a given root is, since floating-point numbers are never exactly equal to zero. We search the set of roots returned by our quartic solver to eliminate complex-conjugate pairs, then use a simple thresholding on the imaginary part to determine if any remaining roots are really "real".

5. CONCLUSION

We have described a predictive event-based method to simulate the physics of pool. The method is based upon a parametrization of the separation of balls, which allows event times to be obtained analytically. The solution requires no granular time step and is accurate to floating point precision in both time and space. It is also efficient, requiring no iterative numerical methods.

In future work, we will develop a game strategy program that uses the simulation to model shot outcomes and automatically select shots in order to play a game of 8 Ball. Following that, the physical parameters of the Deep Green system will be measured and calibrated, and the physics simulation and game strategy will be integrated into the Deep Green system to compete against a human opponent.

ACKNOWLEDGEMENTS

We would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their support, and David Levy and Jonathan Schaeffer for their suggestions and encouragement of this work.

6. REFERENCES

Alian, M. E., Shouraki, S. B., Shalmani, M. M., Karimian, P., and Sabzmeydani, P. (2004). Robotshark: a gantry pool player robot. *ISR 2004:* 35^{th} *Intl. Sym. Rob.*

Berger, F. (2000). http:foobillard.sunsite.dk.

Cheng, B., Li, J., and Yang, J. (2004). Design of the Neural-Fuzzy Compensator for a Billiard Robot. *IEEE Intl. Conf. Networking, Sensing & Control*, pp. 909–913.

Chua, S., Wong, E., and Koo, V. (2003). Pool Balls Identification and Calibration for a Pool Robot. *ROVISP 2003: Proc. Intl. Conf. Robotics, Vision, Information and Signal Processing*, pp. 312–315.

Chua, S., Wong, E., Tan, A. W., and Koo, V. (2002). Decision Algorithm for Pool Using Fuzzy System. *iCAiET 2002: Intl. Conf. AI in Eng. & Tech.*, pp. 370–375.

Koehler, J. H. (1989). *The Science of Pocket Billiards.* Sportology Publications.

Lin, Z., Yang, J., and Yang, C. (2004). Grey Decision-Making for a Billiard Robot. *IEEE Intl. Conf. Systems, Man and Cybernetics*, pp. 5350–5355.

Long, F., Herland, J., Tessier, M.-C., Naulls, D., Roth, A., Roth, G., and Greenspan, M. (2004). Robotic Pool: An Experiment in Automatic Potting. *IROS 2004: IEEE/RSJ Intl. Conf. Intell. Rob. Sys.*, pp. 361–366.

Marlow, W. C. (1995). *The Physics of Pocket Billiards.* Marlow Advanced Systems Technologies.

Petit, R. (1997). *Billard Theorie du Jeu.* Chiron.

Shepard, R. (1997). *Amateur Physics for the Amateur Pool Player.* self published.

Shu, S. W. (1994). *Automating Skills Using a Robot Snooker Player.* Ph.D. thesis, Bristol University.