POOL PHYSICS SIMULATION BY EVENT PREDICTION II: COLLISIONS

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

Department of Electrical and Computer Engineering, and School of Computing
Queen's University, Kingston, Ontario, Canada

ABSTRACT

A predictive event-based method to simulate the physics of the game of pool is described, including event time prediction for collisions between balls and between balls and the rails and pockets of the table. The method uses the vector parameterization of the equation of motion of a moving ball to analytically predict the time of occurence of collision events. The method is both accurate, since it returns exact analytical solutions for ball trajectories using no linear approximations, and efficient, since it requires a minimal number of floating point operations for trajectory solution and collision prediction. It is suitable for use within a game tree search, which requires a great many potential shots to be modeled efficiently, and within a robotic pool system, which requires high accuracy in predicting shot outcomes.

1. INTRODUCTION

Computational pool requires an explicit physical model of the balls, cue, and table, and their interactions. In previous work [Leckie and Greenspan (2005)], the foundation of a numerically accurate and computationally efficient event-based pool physics simulator was described. The event-based method was motivated by two requirements on the simulator: that it be numerically accurate, and that it be as efficient as possible. This simulator has been developed for use within the gaming component of the Deep Green robotic pool playing system [Long *et al.* (2004), Lam *et al.* (2006), Greenspan (2006)] and is distributed as the physics model behind the Computational Pool Tournament of the International Computer Olympiad [Greenspan (2005)].

In this paper, the event-based simulator is described further by detailing the method required to solve for the times of collision events and the post-collision state variables. Collisions are a fundamental part of the game and occur between balls and between balls and table elements (i.e, rails and pockets). The event-based method predicts the times at which collision events occur in order to advance the state variables for a ball forward in time to the instant of the collision and apply the appropriate physics governing the event.

The paper continues in Section 2 with a review of the foundation of the event-based algorithm and the solution for collision event times. Section 3 follows with a discussion of the overall efficiency of the simulator. The paper concludes in Section 4 with a description of future work.

2. EVENT-TIME PREDICTION FOR COLLISIONS

A standard approach in physics simulation is numerical integration, in which the state variables for an object's position $\vec{r}(t)$ and velocity $\vec{v}(t)$ are advanced forward in time by a small discrete amount Δt, using a discrete numerical integration technique [for example, Zeigler, Kim, and Praehofer (2000)]. The object's acceleration is integrated to obtain the velocity and the velocity is integrated to obtain the position. The integral is computed

[1]email:will.leckie@ece.queensu.ca
[2]email:michael.greenspan@queensu.ca

by assuming a constant acceleration and velocity for the object during the tiny Δt. Computational efficiency is increased as a result of these linear approximations to the actual equations of motion; however, there is a trade-off between efficiency and the physical accuracy of the solution. A smaller time step Δt results in a more accurate solution but many more operations will be required to simulate the motion, and vice versa.

In modeling pool physics, 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, an integration-style 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. In this case the simulator must somehow back the dynamics up to the time of the event in order to apply the appropriate physics for the event, for example computing the post-collision velocities and angular velocities of the balls involved in a collision.

For example, assume that two balls in motion 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 of the centers of the balls varies with time, and is denoted as a scalar function $D(t)$. At the time of collision t_c, the separation is exactly twice the ball radius R, i.e. $D(t_c) = 2R$. Within the same shot and prior to t_c, their separation $D(t) > 2R$. Provided Δt is small enough relative to the linear velocities of the balls, then $D(t) < 2R$ within the period $t \in (t_c, t_{if}]$. Upon inspection at time t_{if}, the numerical integration simulator would realize that a collision must have occurred sometime during s_i, because $D(t_{if}) < 2R$. One possiblity would be to ignore the fact that the collision occurred at some time within the time step, and to simply apply the state change from t_{if} on, which would lead to inaccuracies in the resulting motion states. An alternative approach would be to have the simulator back up in time to t_c in order to apply the appropriate equations governing the collision from that time on. 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. A further 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 fewer approximations. This option adds complexity and decreases efficiency.

An alternative to numerical integration for this problem is event-based simulation, wherein the equations governing ball motion are used to analytically predict the time of an event. An earlier paper developed an event-based solution to the problem of simulating the movement of a single ball across the table surface [Leckie and Greenspan (2005)]. By solving the equations of motion exactly, with no linear approximations (as exist in numerical integration), the event-based simulation of ball motion is as accurate as possible. The event-based method can also improve the efficiency of the pool physics simulator because far fewer operations are required for detecting events such as collisions between balls, or between a ball and the rails of the table.

The two types of event that can occur in pool are *motion transitions* and *collisions*. A motion transition event occurs simply through the passage of time as a ball's speed and spin evolve under the influence of table friction. The various motion states and event times for motion transitions of a moving ball were presented in [Leckie and Greenspan (2005)]. Collision events occur when two objects impact, resulting in a change of a ball's linear and angular velocities and therefore a transition to a different motion state. The cue striking the cue ball is also considered a collision event.

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.

During the motion of a ball across the surface of the table, the following types of MOTION-TRANSITION events are possible [Leckie and Greenspan (2005)]:

- SLIDING to ROLLING
- SLIDING to STATIONARY
- ROLLING to SPINNING
- ROLLING to STATIONARY
- SPINNING to STATIONARY

There are therefore a total of nine different types of events that can occur on a given shot. Prior to the start of a shot, all balls are stationary. The shot commences at time $t = 0$ with a CUE-STRIKE event, and thereafter balls in motion can collide with other balls (moving or stationary) or with the rails or pockets of the table. The simulated shot ends when all balls come to rest in the STATIONARY motion state.

Under this framework, the simulation problem is equivalent to predicting the time at which the next event occurs, rather than discovering an event after it has occurred, as in numerical integration. When modeling a shot, the time τ_E at which the next event E occurs is calculated and each ball's state variables $\vec{r}\,(t)$, $\vec{v}\,(t)$ and $\vec{\omega}\,(t)$ are advanced forward in time to $t = \tau_E$ using the appropriate equations from [Leckie and Greenspan (2005)]. The appropriate physics are applied at the time of the event, for example calculating the post-collision velocities of two balls following a collision between two balls. This process is repeated while any balls are still in motion for a given shot, and the simulation is complete when all balls are stationary. A summary of the algorithm is illustrated in Figure 1. This is an attractive solution method because it is both accurate and efficient. The dynamics of the moving balls are calculated completely analytically, with no linear approximations, and are subject only to floating point (double precision) round-off error. Collision detection does not need to be performed for all pairs of objects at each time step as in the integration method, which increases the efficiency of the simulation.

The prediction of motion transition events was described in [Leckie and Greenspan (2005)]. The prediction of ball-ball collision events is based upon a parametrization of the separation of two balls as a function of time. This parametrization gives rise to a quartic polynomial which can be solved analytically to yield the collision time. Ball-rail and ball-pocket collisions follow a similar construct.

2.1 Ball-Ball Collisions

A collision event between two balls each of radius R, with one or both of them in motion, will occur at time τ_E at which the centers of the balls are separated by a distance $D(\tau_E) = 2R$. The vector parameterization for the position of a ball in the SLIDING state is [Leckie and Greenspan (2005)]:

$$\vec{r}_B\,(t) = \begin{bmatrix} x_B\,(t) \\ y_B\,(t) \end{bmatrix} = \begin{bmatrix} v_0 t - \frac{1}{2}\mu_s g t^2 \hat{u}_{0x} \\ -\frac{1}{2}\mu_s g t^2 \hat{u}_{0y} \end{bmatrix}. \tag{1}$$

The subscript $_B$ indicates that the position vector is 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. To find the position $\vec{r}\,(t)$ of the ball in the table frame as a function of time, $\vec{r}_B\,(t)$ is expressed in the table frame using a simple coordinate rotation:

$$\vec{r}_T\,(t) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \vec{r}_B\,(t) \tag{2}$$

where ψ is the angle between the axes of the ball frame and the table frame (*i.e.* the ball's heading).

The position of the ball in the table frame is therefore

$$\vec{r}\,(t) = \vec{r}_0 + \vec{r}_T\,(t) \tag{3}$$

where $\vec{r}_0$ is position of the origin of the ball frame, expressed in the table frame (*i.e.* the initial position of the ball).

For a ball in the ROLLING state, the position as a function of time expressed in the table frame is [Leckie and Greenspan (2005)]:

$$\vec{r}\,(t) = \vec{r}_0 + \vec{v}_0 t - \frac{1}{2}\mu_r g t^2 \hat{v}_0. \tag{4}$$

The separation of two balls, denoted respectively by subscripts 1 and 2, is a time-dependent vector $\vec{d}\,(t)$ given by:

$$\vec{d}\,(t) = \vec{r}_2\,(t) - \vec{r}_1\,(t) = \left[\begin{array}{c} a_x t^2 + b_x t + c_x \\ a_y t^2 + b_y t + c_y \end{array}\right] \tag{5}$$

where the coefficients a_i, b_i and c_i, $i \in \{x, y\}$, are obtained by collecting terms appropriately from the expansion of $\vec{r}_2\,(t) - \vec{r}_1\,(t)$ using the appropriate parameterization for $\vec{r}\,(t)$ depending on the motion states of each ball. The expansion is written in component form in Eq. 5 with a_x denoting the $\hat{\imath}$-component of the t^2 coefficient of $\vec{d}\,(t)$, for example. Note that if one of the balls is STATIONARY or SPINNING, its values of $\vec{v}(t)$ and $\vec{u}(t)$ are zero and both Eqs. 3 and 2 reduce to the ball's position in the table frame.

The coefficients a_i, b_i and c_i, $i \in \{x, y\}$ are therefore given in general by

$$\begin{array}{l} a_x = \frac{g}{2}\left[\mu_2\left(\hat{u}_{2y}\sin\psi_2 - \hat{u}_{2x}\cos\psi_2\right) - \mu_1\left(\hat{u}_{1y}\sin\psi_1 - \hat{u}_{1x}\cos\psi_1\right)\right] \\ a_y = \frac{-g}{2}\left[\mu_2\left(\hat{u}_{2x}\sin\psi_2 - \hat{u}_{2y}\cos\psi_2\right) - \mu_1\left(\hat{u}_{1x}\sin\psi_1 - \hat{u}_{1y}\cos\psi_1\right)\right] \\ b_x = \vec{v}_{2x}\cos\psi_2 - \vec{v}_{1x}\cos\psi_1 \\ b_y = \vec{v}_{2x}\sin\psi_2 - \vec{v}_{1x}\sin\psi_1 \\ c_x = \vec{r}_{2x} - \vec{r}_{1x} \\ c_y = \vec{r}_{2y} - \vec{r}_{1y} \end{array} \tag{6}$$

where μ_j is set depending on the motion state of ball $j \in \{1, 2\}$, and ψ is the heading of a given ball in the table frame (*i.e.* the angle between the ball's velocity vector and the $\hat{\imath}$-axis of the table frame).

The time of collision τ_E corresponds to the solution of $D(\tau_E)^2 = |\vec{d}\,(t)\,|^2_{t=\tau_E} = 4R^2$. Expanding Eq. 5 the following quartic polynomial is obtained:

$$\begin{array}{l} \left(a_x^2 + a_y^2\right)t^4 + \left(2a_x b_x + 2a_y b_y\right)t^3 + \\ \qquad \left(b_x^2 + 2a_x c_x + 2a_y c_y + b_y^2\right)t^2 + \left(2b_x c_x + 2b_y c_y\right)t + c_x^2 + c_y^2 - 4R^2 = 0 \end{array} \tag{7}$$

This quartic polynomial can be solved either analytically in closed form [for example, Mathworld (2006)] or iteratively using standard numerical methods [for example, Gnu Scientific Library (2006)]. The four unique and possibly complex roots of Eq. 7 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. 7 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 (i.e. a MOTION-TRANSITION, RAIL-COLLISION, or POCKET-COLLISION) must precede the collision.

In the development of Eqs. 3 and 4, it was assumed that all balls are constrained to motion on the surface of the table. There are situations, such as the flamboyant jump shot but also including other interactions that occur in ordinary play, where balls can lift off the surface of the table. Relaxing this assumption by including vertical $\hat{k}$-components of position $\vec{r}\,(t)$ and velocity $\vec{v}\,(t)$ would result in a different functional form of the coefficients a_i, b_i and c_i in Eqs. 5 and 7. These coefficients would still be time-independent, and as a result Eq. 7 would remain fourth-order and a solution for the time of collision could still be obtained in the same way.

Once it has been established that a BALL-COLLISION event is the next event to occur, the state variables for all balls are advanced to the time of the event τ_E, and the physics governing the ball-ball collision are applied to the two balls involved to obtain their post-collision state variables. Determining the post-collision state variables involves considering the geometry of the interaction and conservation of both linear and angular momentum, as described in Shepard (1997) or Marlow (1995) for example.

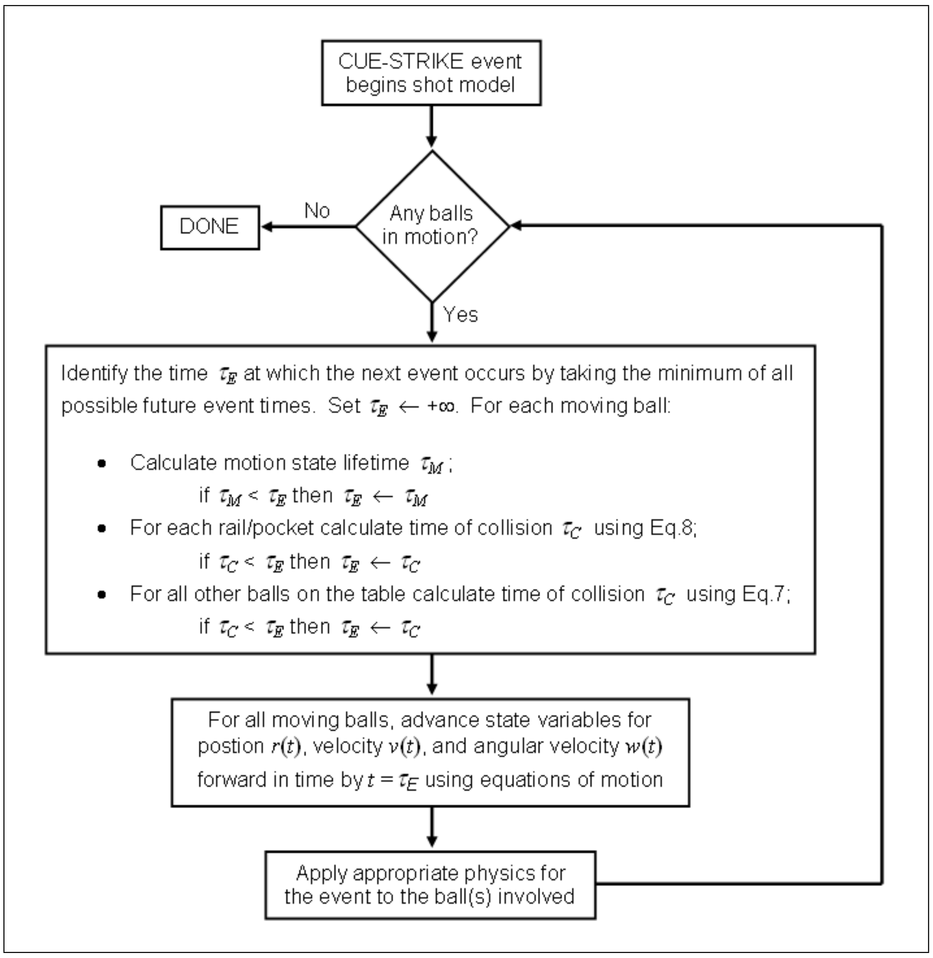
2.2 Ball-Rail and Ball-Pocket Collisions

A similar approach is used to determine the time at which a given ball collides with a rail or a pocket. The ball trajectory is parameterized as a function of time t and a straight line representing a rail or a pocket mouth is parametrized as a function of distance s. Since the ball's position vector $\vec{r}(t)$ refers to its center, when predicting rail collisions the look for the intersection of the ball's center and a horizontal line parallel to the rail but one ball radius toward the center of the table. For example, for a horizontal rail the following would be solved:

$$\begin{bmatrix} s+R \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{r}_x(t) \\ \vec{r}_y(t) \end{bmatrix} \tag{8}$$

where the left hand side is the parameterization of the line representing the rail and the right hand side is the parameterization of the ball's trajectory $\vec{r}(t)$ using either Eq. 2 or 4 depending on the ball's motion state. Other rails and the six pockets are parameterized as lines in a similar way, giving a generic framework for the solution of the event times for RAIL-COLLISION and POCKET-COLLISION events. A ball is considered to be pocketed if its center crosses the line that defines the mouth of a pocket in a direction away from the table center.

Eq. 8 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 interpreted to determine whether either correspond to a physically meaningful solution.


CUE-STRIKE event begins shot model
Any balls in motion?
No
DONE
Yes
Apply appropriate physics for the event to the ball(s) involved

Identify the time τ_E at which the next event occurs by taking the minimum of all possible future event times. Set $\tau_E \leftarrow +\infty$. For each moving ball:

- Calculate motion state lifetime τ_M;
  if $\tau_M < \tau_E$ then $\tau_E \leftarrow \tau_M$
- For each rail/pocket calculate time of collision τ_C using Eq.8;
  if $\tau_C < \tau_E$ then $\tau_E \leftarrow \tau_C$
- For all other balls on the table calculate time of collision τ_C using Eq.7;
  if $\tau_C < \tau_E$ then $\tau_E \leftarrow \tau_C$

For all moving balls, advance state variables for postion $r(t)$, velocity $v(t)$, and angular velocity $w(t)$ forward in time by $t = \tau_E$ using equations of motion

Figure 1: Event-based Algorithm for Modeling Pool Physics.

3. DISCUSSION

Since the event-based algorithm presented is completely analytical and uses no approximations in determining the time of the next event or in 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 the event-based method are the accuracy of the quartic polynomial solver, the accuracy of values of physical constants such as μ used, and any approximations used to model the physics of a collision event. The limitations due to values of physical constants and collision physics are also present in the integration method.

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. The quartic polynomial can be solved analytically [for example, Mathworld (2006)] or iteratively using standard numerical methods [for example, Gnu Scientific Library (2006)]. Average execution time for the GNU Scientific Library (GSL) quartic solver is approximately ten microseconds on a Pentium IV 3 GHz machine, while for the analytic method it is approximately eight microseconds, which results in average shot simulation times on the order of a few milliseconds for the event-based simulator. Tests have shown that the differences between the roots returned from the analytic method and those returned from the GSL's quartic solver are usually equal to within machine precision for nominal values of the coefficients of the polynomial, but that the GSL method is likely more robust to very large or small values of the coefficients in the polynomial because it chooses the order in which to compute the roots using more sophisticated error-cancellation considerations. This was confirmed in a test of 100000 randomized shots, in which the simulator using the analytic solver returned an incorrect or physically impossible result approximately 50 times (0.2%), while the simulator using the GSL solver always returned a physically valid result. The slightly larger execution time of the GSL is worth it for the increase in accuracy. Another challenge lies in the interpretation of the roots returned. A fourth-order equation often has at least one pair of complex-conjugate roots. Determining how "real" or how "complex" a given root is requires some care, since floating-point numbers are never exactly equal to zero. The set of roots returned by the GSL polynomial solver is searched to eliminate complex-conjugate pairs, then a simple thresholding on the imaginary part is used to determine if any remaining roots are purely "real".

In purely simulated games, the lack of accuracy that would result from the numerical integration method would likely not be an issue. An experienced player may sense some lack of verisimilitude to the simulation, but this would more likely be attributed either to the perceptual limitations resulting from the graphical interface, or to the particular settings of the physical parameters of the table (e.g. value of μ), rather than with a limitation of the underlying method of simulation. Where accuracy does become important is in robotic play, where the ability of the machine to accurately predict the state of the table following a shot is essential to strategic play. Indeed, it has been shown in [Leckie and Greenspan (2006)] that the ability to expand a game tree and simulate table states resulting from possible future shots increases the effectiveness of a computational 8 Ball method that is based upon game tree expansion. It was also shown that as shot accuracy degrades, so too does the ability to predict future table states, and therefore play effectively. It stands to reason that limitations in simulation accuracy would have the same degrading effect on shot prediction, and therefore strategic play.

The efficiency of the event-based method is another of its advantages. During many billiards shots, relatively long time periods can pass in which 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 $n_{max} = 16$ for 8 Ball), 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 N 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 the event-based approach is not very heavy, even when modeling more complicated dynamics.

4. CONCLUSION

We have described a predictive event-based method to simulate the physics of collisions between balls and between balls and table elements (rails and pockets). The method is based upon a parametrization of the ball trajectory, which allows motion transition and collision event times to be obtained analytically. The solution requires no granular time step, no linear approximations to the ball trajectory, and is accurate to floating point precision in both time and space. It is also efficient since collision detection for all pairs of objects does not need to be performed, requiring a much smaller number of floating point operations than a more traditional integration-style simulation method.

In future work, the model will be extended to handle three dimensional ball motion, allowing the simulation of jump shots. In addition, the pockets will be modeled more accurately by including prediction of collisions with the horns (points) on either side of the jaw of the pocket. The development of a game strategy program that uses the event-based simulation to model shot outcomes and automatically select shots in order to play a game of 8 Ball will be continued. Following that, the physical parameters and error model of the Deep Green robotic pool 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.

ACKNOWLEDGMENTS

The authors thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their support of this work.

5. REFERENCES

Greenspan, M. (2006). Toward a Competitive Pool Playing Robot. submitted to *AAAI-06: 21st Nat. Conf. Art. Int.*

Greenspan, M. (2005). Pool at the 10th Computer Olympiad. *ACG 2005: Int'l Comp. Games Assoc. Advances in Computer Games Conference.*

Lam, J., Long, F., Roth, G., and Greenspan, M. (2006). Determining Shot Accuracy of a Robotic Pool System. to appear in *CRV 2006:* 3^{rd} *Can. Conf. Comp. Rob. Vis.*

Leckie, W. and Greenspan, M. (2006). Monte Carlo Methods in Pool Strategy Game Trees. to appear in 5^{th} *Intl. Conf. Comp. Games.*

Leckie, W. and Greenspan, M. (2005). Pool Physics Simulation by Event Prediction 1: Motion States and Events. *ICGA Journal.*

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.

Gnu Scientific Library (2006). http:www.gnu.orgsoftwaregsl.

Mathworld (2006). http:mathworld.wolfram.comQuarticEquation.html.

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

Zeigler, B., Kim, T. G., and Praehofer, H. (2000). *Theory of Modeling and Simulation, 2d ed.* Academic Press.