# C174C: Pool Game

**Benet Oriol Sabat Haniyeh Ehsani Oskouie**
**Avalon Vinella Mohsen Fayyaz**
Computer Science Department, University of California, Los Angeles
{benet, haniyeh, avinella, mohsenfayyaz}@cs.ucla.edu

## Abstract

In this project, we present a comprehensive simulation of a pool game wherein users are empowered to manipulate shot parameters such as angle and velocity. The system employs various techniques, including articulated body dynamics and inverse kinematics, to accurately replicate real-world shot execution. Furthermore, our simulation integrates advanced physics dynamics, encompassing aspects such as ball movement, rotation, and collision behavior. Notably, when a ball successfully lands in a pocket, it triggers a visually captivating explosion effect driven by meticulously designed velocity fields. Throughout this report, we examine each aspect of our simulation, providing detailed insights into its design and implementation. Our project code is publicly available on Github[1].

## 1 Introduction

In this project, we present a comprehensive simulation of a pool game wherein users are empowered to manipulate shot parameters such as angle and velocity. The system employs sophisticated techniques, including articulated body dynamics and inverse kinematics, to accurately replicate real-world shot execution. Furthermore, our simulation integrates advanced physics dynamics, encompassing aspects such as ball movement, rotation, and collision behavior. Notably, when a ball successfully lands in a pocket, it triggers a visually captivating explosion effect driven by meticulously designed velocity fields. Throughout this paper, we meticulously examine each aspect of our simulation, providing detailed insights into its design and implementation.

## 2 Models

The table is a combination of basic geometric shapes transformed to mimic a pool table. The balls have been modelled as a sphere with a custom shader that gives the striped pattern. The cue is modelled as a scaled cylinder.

[1]https://github.com/benoriol/billiards

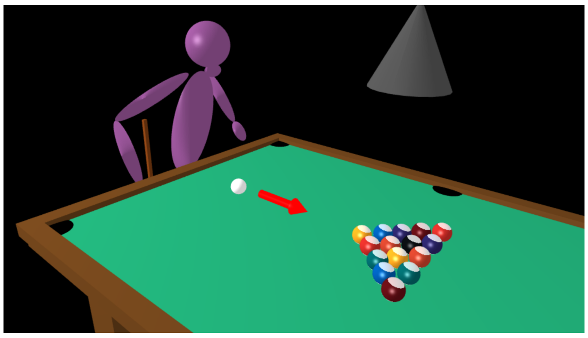
Figure 1: The game of pool with the control arrow and the articulated body ready to move the cue.

## 3 Controls

The player is able to control the desired angle and speed of the white cue ball by changing the orientation and length of a trajectory arrow. This gives the player a visual indicator of the ball's future velocity and informs the cue stick's resulting motion, as described in Section 5.1.

## 4 Physics based dynamics

Each ball $\mathbf{b}_i = \{\mathbf{p}_i, \mathbf{v}_i, \mathbf{a}_i, r_i, \mathbf{rot}_i\}$ is described by a position, velocity, acceleration, radius and object-space rotation matrix. We use symplectic integration to update the values of $\mathbf{p}_i$ and $\mathbf{v}_i$ where the only force that affects the acceleration is the friction force. The friction force $\mathbf{f}_i$ is computed as:

$$\mathbf{f}_i = -k_{friction}\frac{\mathbf{v}_i}{||\mathbf{v}_i||}$$

, where $k_{friction} > 0$ is a friction coefficient parameter and in order to compute the acceleration we assume the mass is 1 unit. Moreover, we update the rotation of the ball around the axis that is orthogonal to the velocity and to the up $(0, 1, 0)$ vector. if

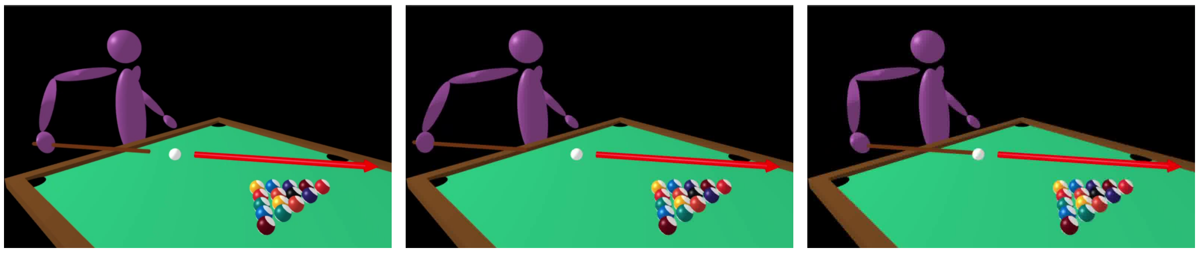

Figure 2: The human and cue movement following the anticipation principle (Lasseter, 1987). (Left to right.)

the position of the ball increases by $\Delta\mathbf{x} = \Delta t\ \mathbf{v}$, we rotate the ball on that axis by $\Delta\theta = ||\Delta\mathbf{x}||/r$, being $r$ the radius.

### 4.1 Collision detection and resolution

We detect collisions between the balls and: i) the walls, ii) other balls and iii) the table's pockets. We resolve the first two with perfect elastic collision and the last on by explosion of the ball. Assuming that the board can be described an axis-aligned, zero-centered bounding box, collision detection between the balls and the walls simply consists in checking $-W/2 < x < W/2$ and $-H/2 < z < H/2$, being $W, H$ the width and height of the pool table. If any of these two conditions is not true, we detected a collision with one of the walls. We then resolve the collision by flipping the sign of the velocity component that is orthogonal to the wall (or parallel to the collision direction) and leave unchanged the component of the velocity that is parallel to the wall or orthogonal to the collision detection.

Similarly, when two balls collide, we can define the *collision direction vector* as the vector between the center of the two balls at the moment of collision. We can decompose the velocity of each ball into two components: the component that is parallel to the collision direction vector and the one that is orthogonal. For perfect elastic collision of same-mass elements, the velocity component that is orthogonal to the collision direction remains unchanged but the component of the velocity that is parallel to it is swapped between the two balls. [2]

Finally, we detect the collision with the pockets of the table by checking that the distance between position of each ball and the position of the pocket

## 5 Articulated-Body Kinematics

### 5.1 Cue Movement

In crafting the cue's movement, we adhere to the renowned **ANTICIPATION** principle in animation, as outlined by Lasseter (1987). This principle dictates a preparatory motion before the actual action, enhancing the realism of the interaction. Thus, we animate the cue to execute a distinct pull-back motion followed by a smooth forward movement towards the ball. This movement is shown in Figure 2. Notably, the extent of this motion is intricately determined by the desired final velocity, with higher velocities necessitating a more pronounced pull-back. To achieve this nuanced motion, we employ the Hermite spline. Specifically, the final control point of the spline aligns with the position of the white ball, ensuring seamless integration with the game's dynamics. The speed of the cue's motion corresponds directly to the user's input, ensuring intuitive control. Furthermore, we consider the velocity and curvature of the spline in determining the cue's movement. As the cue traverses along the spline, it adjusts its orientation to align with the intended angle of impact specified by the user, thereby enhancing the overall realism and immersion of the simulation.

### 5.2 Inverse Kinematics

In order to support the human model with the ability to grasp and manipulate the cue in a lifelike manner along its designated spline, we employ articulated-body kinematics techniques. Specifically, we leverage inverse kinematics, which involves computing the Jacobian matrix and dynamically adjusting joint angles along an interpolated path to achieve the desired cue-holding posture. However, relying solely on the degrees of freedom of the shoulder, elbow, and wrist joints fails to yield

[2] `https://en.wikipedia.org/wiki/Elastic_collision#Two-dimensional`

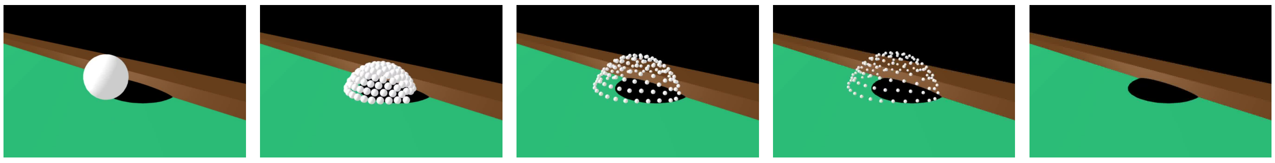

Figure 3: A ball explodes outwards into particles and disappears. (Left to right.)

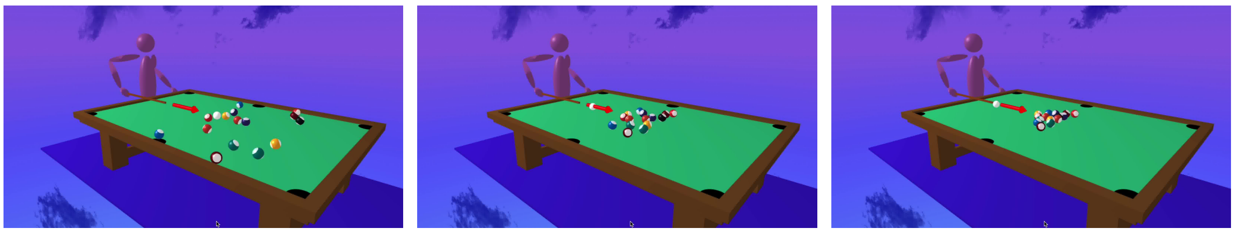

Figure 4: Balls gathering during the game restart. (Left to right.)

truly realistic results. Given the inherent limitations in hand mobility and the standardized technique employed in wielding a cue during a pool game, we found it necessary to impose more strict constraints on the allowable angles for each joint. This ensures that the human model employs its hands in a manner consistent with real-world expectations, enhancing the overall believability of the simulation.

## 6 Spherical Explosion

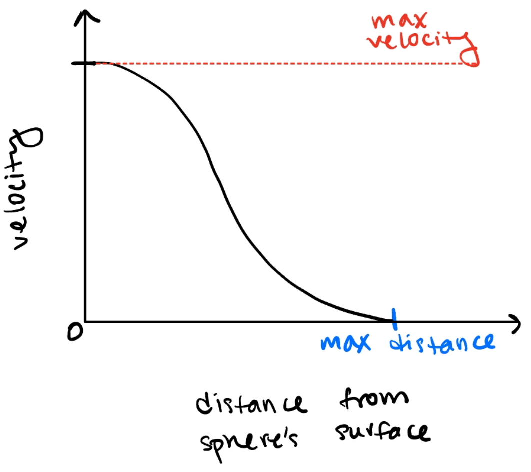

Figure 5: The sinusoidal function defining the velocity of exploding particles.

When a ball is detected to have landed in a hole, it explodes and disappears; this is simulated by replacing the ball with particles in a radial velocity field around the surface of the ball (Figure 3). First, particles are equally distributed in a spherical shape by enforcing equal local distances between particles (Markus, 2004). The velocity field is then initiated using a sinusoidal function relative to the particle's distance from the sphere's center in order to create an "eased" motion where the particle's velocity is greatest at its starting point and slows as it reaches a specified distance away from the center. The period of the easing function is dictated by the desired maximum distance away from the sphere a particle will reach, and the amplitude is determined by the maximum (initial) velocity (Figure 5). To strengthen the illusion of the ball disappearing, the size of the particles decrease linearly based on its distance to the center. Finally, once the velocity falls below a certain threshold, the explosion animation is completed and the particles are no longer rendered.

## 7 Game Restart

When the "e" button is pressed, an intriguing behavior is triggered: the balls gradually return to their initial positions through the utilization of multiple splines. Each ball is assigned a specific spline, carefully designed to guide it from its current position back to its original location. To create a visually appealing effect, the velocities of the balls are controlled using a sinusoidal function. Initially, the velocity is set to a higher value, creating swift movement, but as time progresses, the speed gradually decreases, resulting in a slower and more deliberate motion. This combination of splines and varying velocities adds an engaging and dynamic element to the overall animation. You can see the balls gathering in (Figure 4).

| Task | Assignee |
|---|---|
| • Table Design<br>• Game Restart<br>• Initial ball positions and debugging | Haniyeh |
| • Ball dynamics/physics<br>• Collision detection<br>• Collision resolution | Benet |
| • Explosion Simulation<br>• Trajectory Arrow<br>• Skybox | Avalon |
| • Cue Movement<br>• Articulated Body<br>• Inverse Kinematics | Mohsen |

Table 1: Task Assignments

## 8 Conclusions

In this project, we developed a realistic pool game simulation, incorporating advanced techniques like articulated-body dynamics and inverse kinematics. Our work enhances gameplay realism through accurate physics dynamics and lifelike cue movements. Additionally, we implemented a visually appealing spherical explosion effect. Our project's success is attributed to the collaborative efforts of the team members. Task assignments and responsibilities for the project are outlined in Table 1.

## References

John Lasseter. 1987. Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 35–44, New York, NY, USA. Association for Computing Machinery.

Deserno Markus. 2004. How to generate equidistributed points on the surface of a sphere.