




Pooltool: A Python package for realistic billiards simulation

Evan Kiefl ¹

¹ Independent Researcher, Canada

DOI: [10.21105/joss.07301](https://doi.org/10.21105/joss.07301)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Daniel S. Katz](#) 

Reviewers:

- [@danielskatz](#)

Submitted: 22 September 2024

Published: 28 September 2024

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Billiards, a broad classification for games like pool and snooker, supports a robust, multidisciplinary research and engineering community that investigates topics in physics, game theory, computer vision, robotics, and cue sports analytics. Central to these pursuits is the need for accurate simulation.

Pooltool is a general-purpose billiards simulator crafted specifically for science and engineering. Its core design principles focus on speed, ease of visualization, and fine-grained analysis. It features customizable physics, an interactive 3D interface, a robust API, and extensive documentation, enabling users to easily simulate, visualize, and analyze billiards shots for generic research and engineering applications. Bolstered by a growing community and active development, pooltool aims to be a systemic tool for billiards-related research.

Statement of need

Billiards simulation serves as the foundation for a wide array of research topics that collectively encompass billiards-related studies. Specifically, the application of game theory to develop AI billiards players has led to simulations becoming critical environments for the training of autonomous agents ([Archibald et al., 2010, 2016](#); [Chen & Li, 2019](#); [Fragkiadaki et al., 2015](#); [Silva & Prada, 2018](#); [Smith, 2007](#); [Tung et al., 2019](#)). Meanwhile, billiards-playing robot research, which relies on simulations to predict the outcome of potential actions, has progressed significantly in the last 30 years and serves as a benchmark for broader advancements within sports robotics ([Alian et al., 2004](#); [Bhagat, 2018](#); [Greenspan et al., 2008](#); [Mathavan et al., 2016](#); [Nierhoff et al., 2012](#); [Sang, 1994](#)). Billiards simulations also enrich computer vision (CV) capabilities, facilitating precise ball trajectory tracking and enhancing shot reconstruction for player analysis and training (for a review, see [Rodriguez-Lozano et al. \(2023\)](#)). Additionally, through augmented reality (AR) and broadcast overlays, simulations have the potential to extend their impact by offering shot prediction and strategy formulation in contexts such as personal training apps and TV broadcasting, creating a more immersive understanding of the game.

Unfortunately, the current billiards simulation software landscape reveals a stark contrast between the realistic physics seen in some commercially-produced games (i.e., *Shooterspool* and *VirtualPool4*) and the limited functionality of open-source projects. Commercial products have little, if any, utility in research contexts due to closed source code and a lack of open APIs. Conversely, available open source tools lack realism, usability, and adaptability for generic research needs. The most widely cited simulator in research studies, *FastFiz*¹, is unpackaged, unmaintained, provides no modularity for custom geometries or for physical models, offers restrictive 2D visualizations, outputs minimal simulation results with no built-in capabilities

¹<https://github.com/ekiefl/FastFiz>

for introspection, and was custom built for hosting the Association for the Advancement of Artificial Intelligence (AAAI) Computational Pool Tournament from 2005-2008 ([Archibald et al., 2010](#)). Another option, *Billiards*², offers a visually appealing 3D game experience, realistic physics, and supports customization via Lua scripting. However, as a standalone application, it lacks interoperability with commonly used systems and tools in research. Written in Lua, an uncommon language in the scientific community, it has limited appeal in research settings. The lack of Windows support is another drawback. *FooBilliard++*³ is a 3D game with realistic physics, yet is not a general-purpose billiards simulator, instead focusing on game experience and aesthetics. Other offerings suffer from drawbacks already mentioned.

The lack of suitable software for billiards simulation in research contexts forces researchers to develop case-specific simulators that meet their research requirements but fall short of serving the broader community as general-purpose simulators. This fragments the research collective, renders cross-study results difficult or impossible to compare, and leads to wasted effort spent reinventing the wheel. Pooltool fills this niche by providing a billiards simulation platform designed with speed, flexibility, and extensibility in mind.

Implementation

Pooltool is implemented as a Python package, and thus can be utilized within Python scripts, Jupyter notebooks, other Python packages, or any environment that supports Python.

Pooltool employs an event-based simulation algorithm that significantly increases computational efficiency compared to traditional methods that rely on small, discrete time steps ([Leckie & Greenspan, 2006](#)). By utilizing analytical formulations of the equations of motion in billiards, pooltool advances the system state directly to the next significant event—such as a collision or a change in a ball's motion—by precisely calculating when these events occur. To further increase efficiency, all computationally intensive portions of the shot evolution algorithm are accelerated with just-in-time (JIT) compilation using Numba ([Lam et al., 2024](#)), which compiles Python code to machine code at runtime.

Pooltool includes an interactive 3D interface written with the Python game engine, *Panda3D*⁴. The interface is a central feature of pooltool and can be launched either from the command line or directly through the Python API. It offers a controllable camera for visualizing shot trajectories in a realistic 3D environment, along with a comprehensive set of playback controls—including options to pause, slow down, rewind, and fast-forward shots. Beyond visualization, users can also interactively simulate shots in real time, utilizing game-like controls to stroke the cue stick via keyboard and mouse inputs. Additionally, shots can be programmatically generated and visualized, making it a flexible tool for both interactive play and scripted simulations.

²<https://www.nongnu.org/billiards/>

³<https://foobillardplus.sourceforge.net/>

⁴<https://www.panda3d.org/>

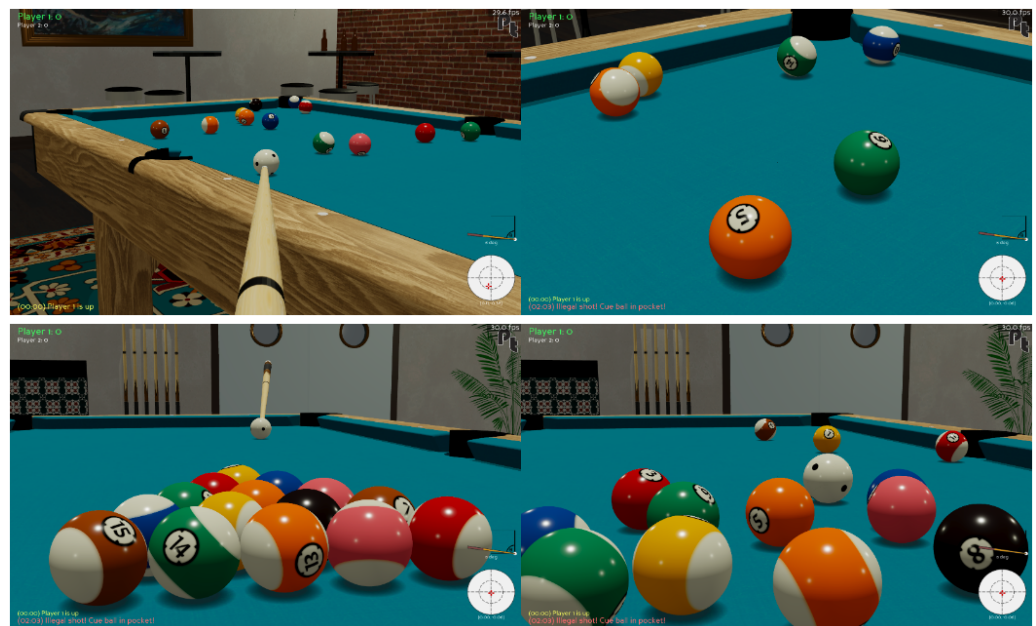


Figure 1: Screenshots from the interactive interface.

Usage

Pooltool's API enables precise control over billiard system construction, simulation, and analysis. Up-to-date tutorials and examples can be found in the official documentation: pooltool.readthedocs.io.

References

- Alian, M. E., Shouraki, S. B., Shalmani, M., Karimian, P., & Sabzmeydani, P. (2004). Roboshark: A gantry pool player robot. *Thirty-Fifth International Symposium on Robotics (ISR 2004)*, Paris.
- Archibald, C., Altman, A., Greenspan, M., & Shoham, Y. (2010). Computational pool: A new challenge for game theory pragmatics. *AI Mag.*, 31(4), 33–41. <https://doi.org/10.1609/aimag.v31i4.2312>
- Archibald, C., Altman, A., & Shoham, Y. (2016). A distributed agent for computational pool. *IEEE Trans. Comput. Intell. AI Games*, 8(2), 190–202. <https://doi.org/10.1109/tciaig.2016.2549748>
- Bhagat, K. H. (2018). *Automatic snooker-playing robot with speech recognition using deep learning* [Master's thesis]. California State University, Long Beach.
- Chen, Y., & Li, Y. (2019). Macro and micro reinforcement learning for playing nine-ball pool. *2019 IEEE Conference on Games (CoG)*, 1–4. <https://doi.org/10.1109/cig.2019.8848113>
- Fragkiadaki, K., Agrawal, P., Levine, S., & Malik, J. (2015). *Learning visual predictive models of physics for playing billiards*. <https://doi.org/10.48550/arxiv.1511.07404>
- Greenspan, M., Lam, J., Godard, M., Zaidi, I., Jordan, S., Leckie, W., Anderson, K., & Dupuis, D. (2008). Toward a competitive pool-playing robot. *Computer*, 41(1), 46–53. <https://doi.org/10.1109/mc.2008.33>
- Lam, S. K., stuartarchibald, Pitrou, A., Florisson, M., Markall, G., Seibert, S., Self-Construct,

- E., Anderson, T. A., Leobas, G., rjenc29, Collison, M., luk-f-a, Kaustubh, Bourque, J., Meurer, A., Oliphant, T. E., Riasanovsky, N., Wang, M., densmirn, ... MattyG. (2024). *Numba/numba: 0.60.0* (Version 0.60.0). Zenodo. <https://doi.org/10.5281/zenodo.11642058>
- Leckie, W., & Greenspan, M. (2006). An event-based pool physics simulator. *Advances in Computer Games*, 247–262. https://doi.org/10.1007/11922155_19
- Mathavan, S., Jackson, M. R., & Parkin, R. M. (2016). Ball positioning in robotic billiards: A nonprehensile manipulation-based solution. *IEEE/ASME Trans. Mechatron.*, 21(1), 184–195. <https://doi.org/10.1109/tmech.2015.2461547>
- Nierhoff, T., Heunisch, K., & Hirche, S. (2012, May). Strategic play for a pool-playing robot. *2012 IEEE Workshop on Advanced Robotics and Its Social Impacts (ARSO)*. <https://doi.org/10.1109/arso.2012.6213402>
- Rodriguez-Lozano, F. J., Gámez-Granados, J. C., Martínez, H., Palomares, J. M., & Olivares, J. (2023). 3D reconstruction system and multiobject local tracking algorithm designed for billiards. *Applied Intelligence*, 53(19), 21543–21575. <https://doi.org/10.1007/s10489-023-04542-3>
- Sang, W. C. S. (1994). *Automating skills using a robot snooker player* [PhD thesis]. University of Bristol.
- Silva, D., & Prada, R. (2018). *MiniPool: Real-time artificial player for an 8-ball video game*. <https://doi.org/10.34627/rcc.v12iespecial.13>
- Smith, M. (2007). PickPocket: A computer billiards shark. *Artif. Intell.*, 171(16), 1069–1091. <https://doi.org/10.1016/j.artint.2007.04.011>
- Tung, K. G., Wang, S. W., Tai, W. K., Way, D. L., & Chang, C. C. (2019). Toward human-like billiard AI bot based on backward induction and machine learning. *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 924–932. <https://doi.org/10.1109/ssci44817.2019.9003085>