Soumya Gottipati
Kyra Acquah

3D-Pool: Intermediary Written Report

3D-Pool is a working implementation of an interactive, online pool game in which users can view and virtually "move" around a customizable three-dimensional pool table. The ultimate goal is to allow users to adjust the power and aim of cue ball strikes in order to direct computer-generated pool balls towards pockets in a manner that simulates the real-world forces that one would expect to see in a physical game of pool. The current implementation of the game involves the rendering of a three-dimensional pool table as well as moving pool balls on the surface of the table. In the coming days, customizable themes, more accurate physics, as well as user control of the strength and direction of strikes will be implemented in order to better model interactions like collisions as well as provide a more fun, customizable interface for users.

Introduction

The game of pool, also known as billiards or cue ball, has roots dating back to medieval Europe, where players utilized hammer-like objects to strike balls in a lawn game resembling croquet (Shamos, n.d.). Since then, pool has evolved to become an indoor game, often involving a pool table on which a cue stick is used to direct balls into pockets placed periodically around the edge of the table. In more recent years, pool has also been adapted into a virtual setting in which players can very similarly aim and strike computer-generated balls through mouse clicks or finger taps on their screen.

Goal

The current project, 3D-Pool, is one such adaptation computer-based virtual pool in which two players can compete against one another in a game of three-dimensional billiards. The aim of the project is to create an interactive online environment in which players can view a virtual 3D pool table and take turns directing the angle and power of a strike on the cue ball in order to pocket other balls. In a completed implementation, this would be done in a physically realistic manner, in which collisions between balls and with the pool table, friction during rolling, and gravity when falling into pockets is properly accounted for. Another key goal of 3D-pool is customizability of visuals, in which players can choose among a dropdown menu of various "themes" for the scene. In different themes, different background colors and various themed images on the pool table surface are rendered in order to make the game more fun and interactive for anyone looking to play pool in an easy to access, virtual setting that requires only a computer.

Previous Projects

In terms of related work, there are many existing virtual pool games on the market from which consumers can choose from. One very popular game is 8-Ball, an iOS-based game included in the GamePidgeon application in which users can manipulate the angle of the cue as well as the power of the shot through finger taps and swipes (Zlotskii, n.d.). The physics of the game are very smooth and realistic, with collisions and frictional forces closely modeling real

life. The game does not, however, offer a “three-dimensional” view of the pool table; the user is restricted to a top-down, 2D view, which while effective in displaying the progress of the game, does not closely simulate the scene and angles one would experience when playing pool in real life.

A screenshot of 8-Ball in GamePidgeon, demonstrating the two-dimensional top-down view (Zlotskii, n.d.)

In addition to mobile games, there are also many computer games that allow players to engage in a virtual game of pool. The top two Google search results for “online pool game” are 8-Ball pool games by Arkadium and Miniclip, respectively (*Arkadium*, 2016; “8-ball pool – A free sports game,” 2011). Like 8-ball GamePidgeon, both games offer smooth and realistic physics during gameplay but are limited to a top-down, two-dimensional view of the table rather than an interactive side view.

The same can be said for games demoed through sites like GitHub; henshimi’s “Classic-Pool-Game” offers very clean, streamlined visuals, nice sound effects, an intuitive interface for choosing the angle and strength of the cue strike, and even features a player vs. computer option in which artificial intelligence is utilized to simulate a two-player game (Schmilovich, 2020). Although a very well-made game, it too offers solely a top-down view of the table, which demonstrates that a project like 3D-pool has something unique to offer within the virtual pool game space. Additionally, it can also be noted that none of the aforementioned games offer the customizability of scene (i.e. colors in the background and design of the pool table) that is a part of the overall goal for 3D-pool in order to make the experience more interactive and unique for users. This again shows that despite the myriad virtual pool games available online, there are many opportunities to contribute a unique aspect of user experience.

Approach

We decided to use ThreeJS to implement this project as that was the framework we were most comfortable with given the previous assignments in the course (“three.js,” n.d.). As for the minimum viable product, we intended to develop a scene with a pool table mesh as well as ball components. More specifically, the goal was to get a simple implementation in which

balls (plain spheres without a design), could sit on the pool table and potentially roll as well. At the time of writing this report, we have been able to generate a pool table as well as rudimentary pool balls and have worked towards the rolling of such balls across the pool table. From here we plan on adding proper physics to the components of the scene, user interface components and/or event listeners for setting the power and angle of the cue stick, as well as improved graphics and customizable themes for the user to choose from.

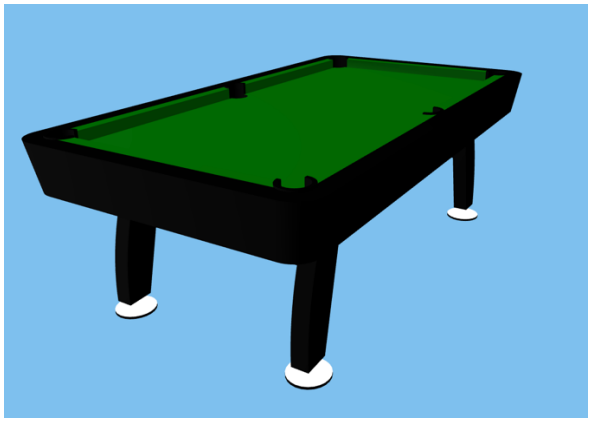
Methodology

The project can be split into multiple phases as listed by the subsection headers below. Some phases we have completed; for those phases, the corresponding sections detail our approach, obstacles, and solutions for that phase. Others, however, are yet to be completed, the sections for which we describe potential approaches and plans for implementing those aspects of the project.

Pool Table Mesh

One issue we faced quite early on in the project initiation phase was the addition of a proper pool table to the scene. Many of the meshes we found on websites like poly.google.com were pool tables that came with various details on them (like balls, triangle racks, etc.) that were difficult to separate from the table mesh and utilize as distinct components. We then tried to create our own pool table using some of the techniques we learned in class, like beveling and extruding meshes. Although this was somewhat successful in terms of creating a table-like structure, we found it difficult to add more detailed components like curved pocket holes that are needed to make a pool table more realistic.

Although we kept our original pool table mesh as a backup, we went back to trying to edit the meshes we found online in order to remove any extra components that weren't part of the pool table itself. To accomplish this, we utilized Blender to remove the unwanted objects, generating .mtl and .obj files that we could then load in our scene as a pool table.



Pool table generated using Blender

Movement and Physics

Generating moving objects abiding by physical principles like friction was a challenging portion of the project development thus far as it involved multiple iterations of learning and executing different methods to understand the best course of implementation for our project. We started by trying to code the physics for the project from scratch, but eventually realized that this was not a plausible option given the time limit and scope of the final project. We then searched for potential physics engines that could be utilized for the pool game; we came across Physijs, a plug-in for ThreeJS built on ammo.js, but after going through the steps of adding this the project we realized that there weren't enough documentation notes and relevant examples for us to be able to learn how to use Physijs thoroughly in order to implement our project (Prall, 2015).

As we have already familiarized ourselves with it through study of Physijs, we have decided to continue using ammo.js and have looked into utilizing enable3D, which offers a ThreeJS plug-in based on ammo as well (Yandeu, 2021). We are currently utilizing enable3D to help code the rolling movement of the ball in the MVP. As we move forward in coding the rest of the physics of our game, especially in terms of handling collisions between balls as well as between balls and the pool table, we are planning on using the ammo.js physics engine directly in order to specify how the collisions should occur.

Customizable Themes

In order to add customizable themes from which users can choose from, we are going to add a GUI element in the scene that allows users to pick one theme from a set list of approximately four to five themes, as this is the most straightforward approach to adding a level of customizability while also limiting the possibilities in order to keep scene rendering efficient. Based on the user's choice out of the list, we will load a different pool table mesh into the scene (altered using Blender to match the theme). If possible, we will also change background colors and/or patterns (altering the color directly in the scene constructor or downloading an appropriate a texture image) to match the theme as well in order to provide a more interesting and stimulating visual experience while interacting with the pool table in the three-dimensional scene.

User Input and Striking

An integral component of the pool game is the user input for the angle and power level at which the cue ball should be struck. Our minimum viable goal for this component would be to add GUI portions where the user can adjust a slider or type in numbers within a given range to specify the angle and strength, eventually pushing a button such as the space bar to generate an event upon which the cue ball will be pushed forward in the specified direction and with the specified magnitude of impulse. We realize that it would be very helpful to visualize the angle at which the cue ball will move before striking, so our next goal after achieving the minimum viable product would be to determine the angle based on the position of the user's mouse, drawing a line so that the player can see the direction in which the ball might travel before actually striking it. A further stretch goal would then be to implement a visualization for the strength of the strike as well, perhaps in the form of a meter that changes values from "weak" to "strong" based on user input.

Two-player Mode

Finally, we need to implement a two-player modality for the game. This could come in many forms, one of which would be to do a split screen type interface in which one player sees and manipulates the scene on the left half of the screen while another player does the same on the right. Given time constraints, however, we believe the best approach might be to add a scoreboard to the game, keeping track of the score for every alternate player. In terms of a backup plan for implementing this over the next few days, if we are not able to keep track of whether the player put their assigned balls in the holes, we may alter the game slightly such that the objective is to pocket as many balls as possible in consecutive strikes, with any ball counting as a point regardless of a striped or solid pattern.

Results

Much of the success of the project can be evaluated through visualization of the scene. Thus far, our goals were to create a three-dimensional, realistic pool table to add in the scene, as well as include moving components such as balls that can roll across this pool table. These goals can be easily verified by observing the scene through demonstration/local hosting after coding. The same can be said for some of the components of the project that are currently being developed – the customizable themes, for example, must all be modeled to ensure that the colors and images are changing appropriately.

As for the physics aspect of the project, it is important to check corner cases and test as many possible situations before determining that the physics are coded properly. These checks can be broken down into different scenarios, such as collision of a ball with a ball, collision of a ball with the edges of the table, pocketing of the ball, etc. which can then also be sequentially performed to make larger sequences (such as a ball bouncing off of the table edge and then colliding with a ball). It's important to be very detailed in this checking process, testing as many sequences as possible, as the physics are meant to simulate real-world mechanics. A very similar experimentation process must also be conducted for user inputs when striking the cue ball – it is important to try out many combinations of angles and strengths in order to ensure that the ball is moving as expected in any given pairing and the user interface is intuitive and easy to maneuver.

Discussion and Conclusion

In terms of the progress made on the project thus far, we have achieved our vision for the minimum viable product and are currently making progress on the next important steps, including adding more realistic physics to the game mechanics as well as formulating different themes that the user can apply to the interface. Although there is still quite a lot of work to be done in terms of the final project we had originally planned (see Methodology section above), the approaches we have now set in place are more effective in accomplishing our goals than in previous iterations. Much of the early phase of development was spent understanding how to go about tackling such a large-scale project through much trial and error (especially in terms of familiarizing ourselves with ThreeJS, generating a pool table mesh using Blender, as well as understanding how to code for physics using libraries that are well-suited to our project goals). Overall, we attained our first goal of getting acquainted with tools available to us to use within this project and developed a strong base that we can build upon over the next few days.

Contributions

Kyra generated the pool table mesh and initial scene, while Soumya researched movement and physics. Currently, both Kyra and Soumya are continuing work on physics, and Soumya has started working on customizable themes.

References

8 ball pool - A free sports game. (2011, April 19). Miniclip. https://www.miniclip.com/games/8-ball-pool-multiplayer/en/#privacy-settings

(2016, October 14). Arkadium. https://www.arkadium.com/games/free-8-ball-pool/

Prall, C. (2015, October 19). *Physijs*. Physijs GitHub Pages. https://chandlerprall.github.io/Physijs/

Schmilovich, C. (2020). *henshmi/classic-pool-Game*. GitHub. https://github.com/henshmi/Classic-Pool-Game

Shamos, M. (n.d.). *A Brief History of the Nobel Game of Billiards*. Billiard Congress of America. https://bca-pool.com/page/39#:~:text=The%20history%20of%20billiards%20is,Europe%20and%20probably%20in%20France

three.js. (n.d.). three.js – JavaScript 3D library. https://threejs.org/

Yandeu. (2021, May). *Enable3d/enable3d*. GitHub. https://github.com/enable3d/enable3d

Zlotskii, V. (n.d.). *GamePigeon*. App Store. https://apps.apple.com/ae/app/gamepigeon/id1124197642