Brianna Royer & Justin Yi

Abstract

A single-player game was created to emulate the mechanic of cup-pong games. More specifically, a game was created in the likeness of the game “Ruits” played by the members and alumni of the Princeton Quadrangle Eating Club, or Quad. Created with Three.js and cannon-es.js, a player controls the trajectory and power of a ball launcher with the aim of launching the ball into several cups. Importantly, the physical table present in Quad, with LED lights, has been recreated. As the player successfully gets the balls in the cups, the cups are deleted and the player earns points.

Introduction

We sought to recreate a game called “Ruits”, a variant of cup pong played by the members of the Princeton Quadrangle Club. During the pandemic, many of our members and alumni missed out on critical bonding offered by being within the clubhouse. By creating this game, we hope to give everyone affiliated with Quad the chance to have reminiscent fun wherever they are, or if a player doesn’t know anything about the origins of this game, they enjoy the game for the mechanics.

Two-dimensional cup pong games are commonly available as mobile apps. For instance, the “Cup Pong” created by GamePigeon for iOS devices, allows a user to drag their finger across the screen to indicate speed and direction of launch. Another game called “Cup Pong Challenge” available on desktop at https://www.playit-online.com/game/cup-pong-challenge/ and other websites has users hold the left mouse button down and drag across the screen to throw a ball. Both of these applications involve parsing movement to extract ball parameters with different levels of freedom. In Cup Pong, movement of the ball and landing in the cup–or bouncing off of it–somewhat resembles the freedom of a real-world system. In contrast, Cup Pong Challenge appears to limit the number of possible ball launch vectors, and the game is easy to complete.

Cup pong games are a subtype of projectile launching games, like golf, basketball, darts, or skee ball games. In “Street Shot”, available on desktop at https://www.ducksters.com/games/street_shot.php, the player cannot control the launch vector and must instead click their left mouse button when an oscillating power arrow reaches the desired value. Compared to the aforementioned two games, this launching mechanism is more predictable. However, mouse movement launching mechanisms are more difficult (in the case of fast bottom-up moving on a trackpad, one might even consider the movement unergonomic). Between these extremes of launch freedom, there are a variety of mechanisms which may fix parameters of the projectile motion or constrain them within certain ranges. For instance, the vertical angle of a projectile may be set while the initial speed is not, or the speed may be fixed while axes of the initial vector are manipulable. These are ultimately the key to making such games fun by providing an intermediate amount of control for the right difficulty.

Approach

We elected to create a three-dimensional cup pong game using Three.js and physics engine Cannon-es.js in order to realistically simulate interactions between cups, balls, and the table top. Cannon-es allows for objects to be assigned a basic hard body geometry and mass. This works well when the ball is simulated as a low-mass object and the cups, which would usually have a small amount of liquid, are given a higher mass to limit cup movement (which is typically minimal, but possible, in real life). For ergonomic ease of control, the launching mechanism was approached with the intent that it be controlled using keys, but to introduce an element of difficulty, we needed to consider in what way the precision of control could be limited.

Methodology

Cannon-ES - This is a maintained fork of Cannon.js, a physics engine compatible with Three.js. Developers use the physics engine by first initializing a world object from the library. The world instance takes Cannon-ES rigid bodies and calculates the physics, collision detection, and animation on all bodies. While the physics of the world is customizable, for the sake of realism and simplicity, we stuck with the default settings and set y = -9.81 for the gravity.

Scene - The main scene is responsible for loading all components, as well as tracking cup counts, cup deletion status, ball instances and deletion, and whether the game is first or second person. When playing the game, the camera is fixed on either side of the table object. It alternates between sides after each ball launch and deletion.

Table - The mesh for the table is Simple Table (Low Poly) by Berk Gedik , licensed under Creative Commons Attribution. The texture has however been modified to show the table top of the Ruits table in Quad. It is imported using a GLTF loader, and associated with an invisible Cannon-ES box the dimensions of the mesh. This was found to be better than manually creating a table out of simple polygons–although the game was always intended to be unrealistic–because of the visual appeal of the Ruits table texture and provided roughness map.

Cups - Cups were originally going to be objects with an imported GLTF mesh that maps to cylinder Cannon-ES hard bodies. The roughness maps of the imported meshes greatly increased loading time, and mapping between the Cannon-ES rotation quaternion and Three.js Euler rotation plus with difficulty retrieving mesh dimensions produced unrealistic cup movement and clipping. Instead, a cylinder geometry with Three.js MeshStandardMaterial and perfect mapping between mesh and body was used.

12 Cup objects are loaded as children of the object "Rack" for appropriate construction with initial positions and color designation "blue" or "yellow". Rack is itself not updated but is useful for easy access of individual cups and their states, as opposed to loading cup objects individually.

The act of "getting the ball in the cup" is handled with the "collide" Cannon-ES event handler in the Cup class. If the colliding object is a ball, the y-position of the ball is greater than the cup height, and the Euclidian xz-plane distance from the ball to the y-axis of the cup is less than the radius of the cup cylinder top radius, a "successful" hit is recorded, initiating the removal of both the collided cup and the ball instance. For a cup to be removed, it is pushed after a successful collision event within the collision handler on the scene state "cup_to_delete" array, and a scene state boolean "cup_needs_delete" is activated. On the next update, if cup_needs_delete is true, the scene update function pops the cup from the state array, its mesh properties are disposed and its world and scene representations unlinked. Importantly, the cup object name is changed from "cup" to "dead", which signals the scene to unlink the object and remove it from the update list. Afterwards, the scene state count of each cup type is decremented depending on cup color. The only remaining access to the cup object is through the cup's Rack object.

<u>Lights</u> - The lights on the table used in real life are a series of LED strip lights functioning somewhat as point light sources. Each point source could be recreated with Three.js, but each light source results in material recalculations for the objects in the scene. Especially for the table, which has a roughness map, 100+ lights incurs excessive loading times. Each strip light section was instead simplified and visualized using RectAreaLight and RectAreaLightHelper, a light source visualizer.

There's a subset of lights hosted in "RackLightYellow" and "RackLightBlue", meant to illuminate the positions of the cups. The lighting can be switched between two side racks and one central rack with the gui options "rerack_blue" and "rerack_yellow".

<u>Ball</u> - To calculate the physics of the ball, a Cannon-es Sphere was used for the rigid body. The rigid body has a special property to allow bouncing against the table and cup rigid bodies. The ball was graphically represented with standard Three.js mesh material shaped as a sphere. Calling the update function of the ball object updates the position of the mesh by setting the ball's mesh position to the rigidbody position.

<u>Launching Mechanism</u> - Players must hold down the space bar to determine the velocity of the ball and release to shoot the ball. The velocity value oscillates between a set ceiling and floor as the player holds down the space bar. The main skill in Digi-Ruits is releasing the space bar at the right moment to not under/overshoot the ball. When a ball is shot, it travels along a specified direction represented by a Vector3. Cannon-es applies the velocity to the x, y, z-coords of the ball. To rotate the shooting direction, players press either the "a" key for left or "d" key for right. Pressing either one rotates the shooting direction vector around the y-axis.

“Rerack” - Part of Ruits is the ability to “rerack”, or bring six remaining cups into a triangle configuration at the center of the table. Rather than disposing of the remaining 6 cup instances of a rack and creating 6 new ones, which would disturb the intended structure of the Rack parent, the Rack children array is iterated over to identify the 6 children with the name “cup” instead of “dead”. Each is moved to the 6 central rack positions. Although reracking is triggered by the GUI options tabs rerack_blue and rerack_yellow, which will always change the rack lights, a rerack can only alter cup positions when exactly 6 cups are present, and this can occur only once. Admittedly, players can ask for any cup that has shifted away from their initial position to be corrected, but this feature was omitted to avoid the complications of moving cups to nearest rack positions.

Results

During development, we first tested the physics of our game world. We attempted every combination of object collisions to confirm that behavior was realistic and accurate. After development, we had a couple of students, specifically Quad members, to playtest our game. As they played, we took notes of any bugs or unintuitive aspects of the GUI. For example, we realized that we had accidentally allowed users to launch more than one ball in a single turn. Despite the bugs and minimal GUI though, the users all reported having fun.

There are a couple of things we could have tried differently. First, we could have tried to incorporate a new launch mechanic for the game. A pulling mechanic, such as the one in *Angry Birds*, may have been more engaging than the current spacebar approach. Second, we could have tried to incorporate more dynamic cameras. Instead of abruptly positioning the camera at the opposite end of the table when a player takes their turn, we could have tried to slowly rotate the camera around. Perhaps a camera to follow the ball when it is launched.

Future development should be focused on improving the GUI. We currently use the dat.GUI to display the original two options from the starter code, the rerack buttons, and the power slider that displays how hard the ball is launched. The project is still missing game controls and rules and the score total for each player.

What did we learn by doing this project?

That graphics is fun and super dope and the coolest class ever.

Conclusion

This project attains the goal of making a cup pong game with the main visual elements of Ruits, invoking excitement from members and alumni. However, there are higher-order elements of Ruits that were not recapitulated in the assignment, such as the ability to move the final cup to a center position, and the fact that such a game gets increasingly difficult to play as one consumes from the cups. We would like to add increasing visual distortions to a player as their cups are consumed. Furthermore, this game involves chants and exciting music that can be added as sound effects. Finally, a good GUI needs to be implemented prior to the live demo.

The primary issue is the representation of cups as solid bodies, which entails collisions and mesh intersections that detract from the visual appeal of the game. Furthermore, it is a completely normal case in real Ruits for a ball to enter and then bounce out of a cup, something that adds infuriation and chance to the game. Representation of cups as concentric cylinders, the inner of which does not count as a hard body, would create more appropriate physics.