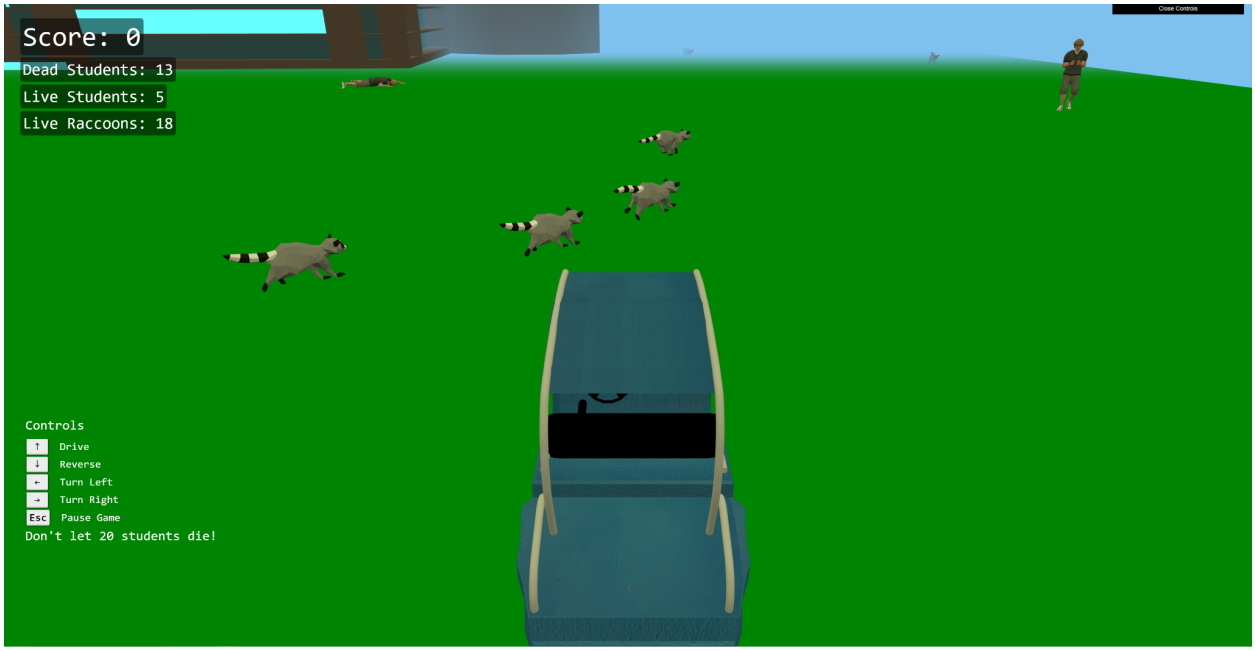
The Great Princeton Raccoon Invasion Written Report

Ilay Furman, Louis Larsen, Everett Shen

Game link: https://everett-shen.github.io/COS-426-Final-Assignment/
Video: https://www.youtube.com/watch?v=FxKWsLpdb3M&ab_channel=EverettShen
Github: https://github.com/Everett-Shen/COS-426-Final-Assignment.git



Abstract

In this paper, we introduce "The Great Princeton Racoon Invasion," a 3D single-player game inspired by recent campus events built primarily in three.js. The player plays as a sentient golf cart, whose goal is to run over as many raccoons as possible before too many of the Princeton students walking around the game map are killed. We provide a complete physics system featuring collisions, gravity, and spatial sound. In this paper we will dive deeper into the game's implementation.

Introduction

On December 5th, 2023, Kathy Lee was peacefully walking home from late night studies at Firestone. The crisp winter air filled her lungs as she strolled along the dimly lit path, the only sound being the echo of her footsteps against the silent night. As Kathy ambled through a quiet residential area, she heard a rustling in the bushes. Ignoring it, she figured it was just some nocturnal creature enjoying the nightlife. But out of nowhere, a raccoon flew down from the sky

and bit her leg. Passersby, attracted by the commotion, began to gather, some offering assistance and others dialing emergency services. Kathy was later treated in the hospital for rabies.

In times of such tragedy, we took upon ourselves to create a game of payback against the raccoons. In “The Great Princeton Raccoon Invasion,” the player drives around with a golf cart running over every raccoon they see. For every one they kill, the player receives one point. However, they have to be careful to not run over students, or make the raccoons get to them either. When 20 students die, the game is over.

Goal

Our primary goal in developing "The Great Princeton Raccoon Invasion" was to provide the Princeton community with an entertaining and engaging outlet to collectively overcome the unexpected and peculiar incident involving Kathy Lee and the airborne raccoon.

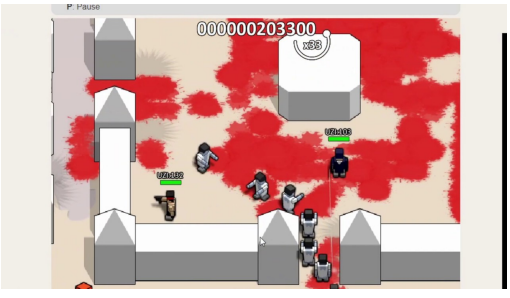
With a focus on simplicity and accessibility, our aim was to design gameplay that would appeal to a broad audience. The mechanics of driving a golf cart, strategically eliminating raccoons while avoiding harm to virtual students, were crafted to be both intuitive and entertaining. By incorporating a point system for every raccoon successfully defeated, we sought to create a competitive yet lighthearted atmosphere that would encourage players to challenge themselves and each other.

Moreover, we recognized the importance of infusing the game with a sense of strategy and skill. Players must navigate the virtual environment with care, balancing their quest for raccoon revenge with the imperative to protect the simulated students from harm. Moreover, they must examine their own moral stances as they must resist the urge to run over their fellow students. This dual challenge adds an extra layer of complexity, ensuring that players must employ a combination of quick thinking and precision to succeed.

In essence, our goal was not merely to create a game but to provide a communal experience that transcends the boundaries of the unexpected incident that inspired it. "The Great Princeton Raccoon Invasion" was conceived as a testament to the community's ability to turn challenges into opportunities for creativity, humor, and shared resilience, fostering a unique sense of unity among its players.

Previous Work

For our project, we were inspired by “Boxhead: The Zombie Wars,” an old school flash game where the player survives waves of zombies by shooting at them, sometimes with automated turrets. However, our game is in 3D rather than 2D. Additionally, instead of killing the enemies with guns, we kill them using a golf cart.



We were also inspired by the Javascript Github Library, Sketchbook. Sketchbook is a “simple web based game engine built on three.js focused on third-person character controls and related

gameplay mechanics.” It aspires to recreate popular gameplay mechanics in modern games, such as real-life physics, a third person camera, and driving. Since Sketchbook seemed to be very complicated to work with, we decided to attempt to implement our own versions of what was found on Sketchbook.

Approach

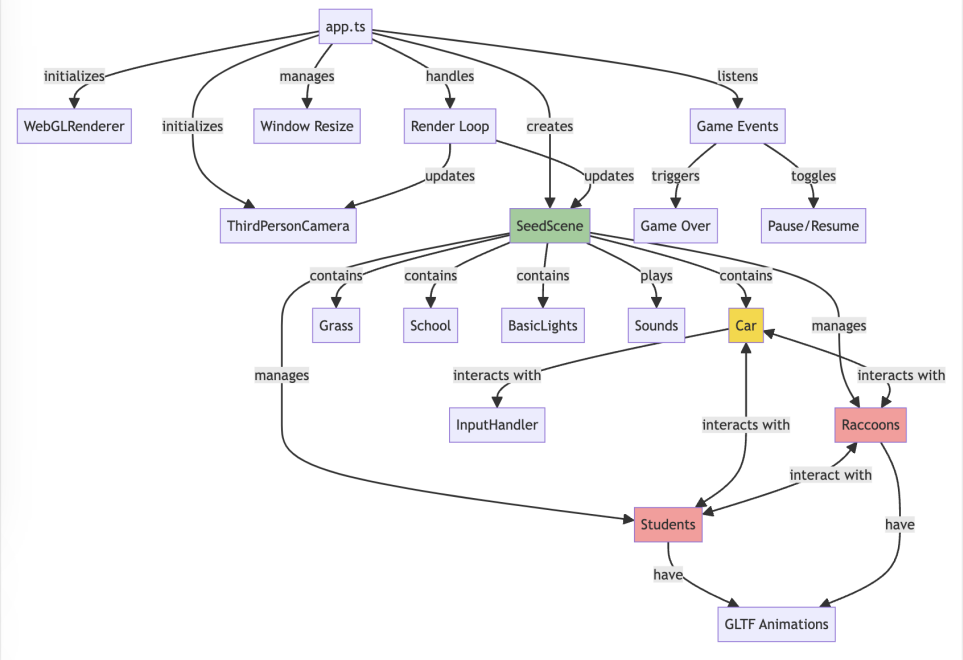
Our approach in developing "The Great Princeton Raccoon Invasion" was two-fold: drawing inspiration from the classic "Boxhead: The Zombie Wars" and attempting to recreate the functionalities offered by the Javascript Github Library, Sketchbook.

As far as our inspiration from “Boxhead,” our approach was to capture the essence of a beloved flash game while infusing it with a fresh and whimsical twist. We aimed to create a 3D gaming experience that retained the simplicity and addictiveness of its 2D predecessor. The driving force behind our approach was the belief that the fundamental appeal of surviving waves of enemies, could be recreated in the backdrop of a Princeton University Setting.

As far as “Sketchbook,” Our approach was to attempt to implement the features seen in the library, such as driving and a third person camera. We did this to make the game more immersive and to relate to the character of whoever’s driving around in the golf cart more.

Under circumstances where players sought a lighthearted and competitive outlet to collectively overcome an unexpected incident, we believed our approach would work exceptionally well. The strategic element of avoiding harm to virtual students while pursuing raccoon revenge added depth to the gameplay, making it suitable for both casual and more serious gamers. Additionally, through adding a 3D camera and driving mechanics, we felt that the game would become more realistic.

Methodology



Architecture graph

1. Car

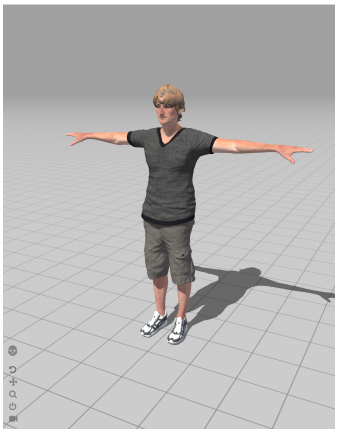
The car is a player-controlled entity and the focus of the camera. While brainstorming, we thought about enhancing the physics of the car's movement by using the cannon library. However, we decided to not proceed with this idea due to the increased complexity and development time it would entail. Instead, we opted for a more straightforward approach where the car's movement is primarily controlled through keyboard strokes, influencing the vector of movement.

The car's acceleration and deceleration are manipulated by altering its velocity vector in the z-component, and turning is handled by rotating around the y-axis in response to left or right arrow key inputs.The car's state updates at each timestamp to ensure responsive and dynamic interaction with the game environment.

When it came to collision handling, we drew inspiration from the way cloth particles interacted with the box. All collisions are detected in the main render loop in SeedScene. The collision with the arena boundaries is handled by pushing the car back within the playable area. For collisions with the school, we faced a challenge with the school's bounding box extending beyond the intended limits. To resolve this, we created custom bounding boxes using three.js' Box3 class that more accurately reflected the school's layout. This method ensured that the car's velocity was temporarily reversed only upon a realistic collision with the school structure.

Collisions with NPCs, specifically raccoons and students, are another critical aspect. At every game update, we check for these interactions. When a collision occurs, the game reacts appropriately with sound effects, animations, and updates the score, increasing the 'Dead Student Count' or the overall score based on whether the car hits a student or a raccoon, respectively.

2. Student



For the student component in our game, we used a student model with animations from Mixamo. We downloaded separate animations for walking and dying and combined them in Blender. To enhance the realism and unpredictability of the game, students are spawned at random locations across the map, with their appearance locations confined within the map's width.

The movement of the students is simple- constant velocity in a straight line. However, to prevent students from wandering off the playable area, we implemented a boundary collision system. When a student approaches the arena's edge or collides with the school's bounding box, their velocity is temporarily halted, and they are moved back into the playing field, much like the car's interaction with boundaries. This feature ensures that the students remain within the player's reach and the game's scope.

A critical aspect of the student component is their interaction with the car and raccoons. When a student is hit by either, they are marked as 'dead,' triggering a falling animation and playing corresponding sound effects. Once declared dead, the student is removed from the list of active NPCs, yet their animation remains on the screen.

3. Raccoons

For the raccoons, we used an animated model purchased from CGTrader and made minor modifications in Blender, like making the eyes red. We chose to use a pre-made model so we could focus our time on the gameplay and also to give the game a more polished look. Raccoons were rendered in a random location within the map, with the constraint that the positions had to be a minimum distance away from any students on the map to prevent students from instantly being killed. For pursuit behavior, we calculated the closest student to each raccoon and changed their direction vector to point in their direction. We also had a function to spawn a new raccoon which fell from the sky every 5 seconds. To create the “flattened” look of the raccoons after they were run over, we switched to the crawling animation provided with the model and paused the animation, which created the effect we desired.

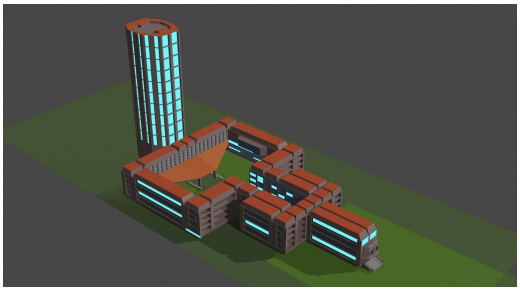
4. Third person Camera

For the third-person camera in our game, we developed a system that ensures a dynamic and immersive viewing experience for the player. The camera's design was influenced by a detailed tutorial we found on YouTube, which provided the principles behind effective third-person camera movement in a 3D gaming environment.

The implementation of our third-person camera was focused on achieving a seamless and responsive connection between the camera and the player-controlled car. The camera is designed to fluidly follow the car, maintaining an optimal distance and angle .

In terms of implementation details, we focused on algorithms that calculate the ideal position and orientation of the camera based on the car's current state. This involves determining the best offset position from the car and the point at which the camera should be directed. These calculations are constantly updated and applied to the camera, ensuring that it smoothly transitions between positions and angles, thus avoiding any jarring or disorienting movements.

5. Map



For our game, the creation of the map was a pivotal aspect, as it sets the stage for the entire gameplay experience. The map consists of two primary segments: the grassy areas and the university campus. The grassy part of the map is essentially a flat plane designed in Blender, and the university segment of the map, placed at the center of the grassy plane, is a basic university model from an online repository. The model, shown to the left, came with grass that had to be removed to add our own green plane.

6. Welcome / Game Over Pages

In our game, the Game Over page is triggered when the number of dead students exceeds a predetermined threshold, in our case, 20. Upon this event, the Game Over page overlays the game screen, effectively halting the ongoing animation and gameplay. Additionally, it includes the game over page and provides the option to press the ‘Enter’ key to restart the game. As for the Welcome page, its design was created by DALL-E. As the central message on the page indicates, the game begins by the user pressing the spacebar.

Results

The end result is a simple but complete game, offering engaging gameplay and a humorous player experience. We fine tuned the game parameters (speed, win conditions, spawning) for the best balance of excitement but to also make the game just the right difficulty. Our goal was to have a complete physics system, while avoiding any artifacts like clipping, spawning off map etc. We also wanted the game to feel as responsive as possible, adding things like acceleration/deceleration and sound effects triggered by user action. To enhance realism, we adjusted the volume of sounds based on the distance to the player. We added a HUD with the score and controls for a better user experience. We also obviously wanted the game to be as performant as possible to ensure a flawless player experience, and were able to achieve an average frame rate of 50+ FPS, with the framerate frequently reaching 60 FPS, as tested through Stats.js.

Discussion

The approach of combining humor with action in a familiar setting proved to be engaging. We had a great experience developing the game and learned a lot about making improvements through incremental changes. We faced several challenges along the way, including troubleshooting Blender when adapting the raccoon/school models. Since none of us were particularly experienced with Blender, designing and working with the 3D models proved to have a high learning curve. Additionally, handling collisions with the school and map edges also proved to be a headache, as there were many edge cases to handle and we weren't sure what the correct behavior was upon collisions. We spent a lot of time fine-tuning the code for bounding box intersection, which in the end proved worthwhile. We had lots of fun during the development of this game and hope others in the class and at Princeton find it to be fun to play.

Conclusion

"The Great Princeton Raccoon Invasion" successfully achieved its goal of creating an entertaining and humorous game. Future directions could include multiplayer capabilities, more complex terrain, and varied game modes. We would also like to add mobile support, which would require the addition of a touchscreen joystick. We would also like to spend more time on improving performance. Our render loop essentially consisted of a nested for loop of all the objects in the scene that ran each loop. In this implementation, we chose to implement our own physics to allow for greater control and flexibility, which in this short time span we think was the right approach, but in the future we could consider using third party libraries like Ammo.js or Cannon.js. Finally, it would be nice to add more Princeton related assets and flavortext to make the game's worldbuilding more compelling.

Contributions

Ilay Furman - In this project, I was primarily responsible for developing the third-person camera feature, creating the golf cart and its functionalities, creating the control panel, and creating all interactive pages (Welcome, Pause, and Game Over). This includes handling all the event listeners and calculations for the camera, the car, and the functionalities of the pages.

Louis Larsen - I helped design the map as well as the collisions with the golf carts, students, and racoons with the out-of-bounds area of the map. I also helped Everett with other collisions with the model of the university.

Everett Shen - I implemented the raccoon and student models, including all behaviors, movement, spawning, and animation handling. I also worked on collisions between the car, students, school, and raccoons, as well as all game sounds.

Works Cited:

- Third person camera tutorial by SimonDev: https://www.youtube.com/watch?v=UuNPHOJ_V5o&t=510s
- Css code for control panel: https://github.com/swift502/Sketchbook
- Pixel Peeker Polka by Kevin Macleod (music)
    - https://www.chosic.com/download-audio/39321/
- Racoon Blender file
    - https://www.cgtrader.com/3d-models/animals/mammal/raccoon-4c3774a2-dff3-4d6a-97bc-d2c544db274f
- University Blender File
    - https://www.cgtrader.com/free-3d-models/architectural/other/university-campus-a447b531-5c82-4398-bb7e-69d1acd4b0c1
- Golf Cart Blender File
    - https://sketchfab.com/3d-models/golf-cart-5637f77fc2fd42439b006802741c1cc0
- Mixamo character model (Remy)
    - https://www.mixamo.com/#/?page=1&query=remy&type=Character
- Libraries
    - threeJS
    - Stats.js
- Sound effects from Pixabay
- ChatGPT and StackOverflow for debugging and code consultation
- DALL-E for title screen generation