The Ripening

Chandler Ault and Will Drury

Abstract

It is no secret that the gaming industry mass produces games depicting violence, and these games are often disseminated to children who take in the violent imagery at a young age. Perhaps no game series embodies this stereotype more than the Call of Duty series. One of the most popular game modes in this series is the Zombie game mode in which players fight hoards of zombies and try to survive. Given its popularity in youth, we wanted to design a zombie-like game that met the desires of children to play a shooter game while maintaining a less serious tone and not depicting acts of violence against humans. Thus, we introduce The Ripening. The Ripening is a shooter game where players play as a clock and shoot produce (e.g. avocados and bananas). Once the produce ripens, it will expire and the player receives points for ripening the fruit. After one fruit expires, two more spawn to take its place. As more enemies spawn, they give the impression of a zombie-like produce hoard approaching the player. This game satisfies our desire for a light-hearted zombie shooter game experience for kids and others who do not wish to play a game promoting violence against humanoids.

Introduction

Goal

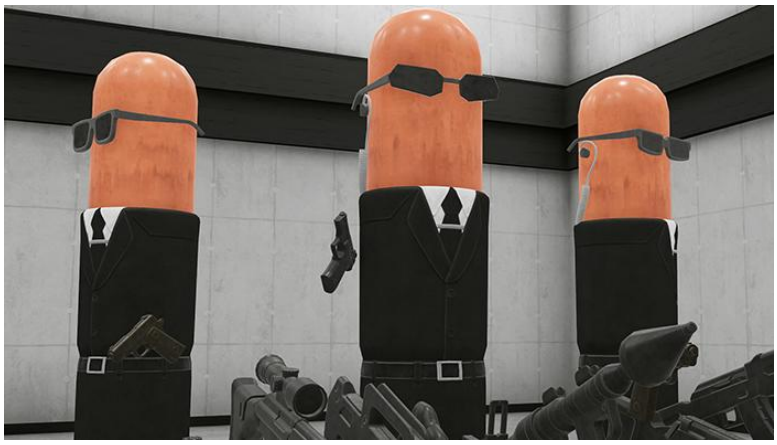
The primary goal of this project was to develop a game that could be played by all ages without age restrictions due to violence. Thus, we sought to create a goofy, creative game with simple mechanics and low difficulty. However, with limited time and resources, we recognized that we would only be able to implement basic features.

The realization of this goal will benefit twofold. First, kids will benefit by having a colorful, fun game to play in place of grim and violent video games. Additionally, our game will benefit parents who do not want to expose children to the violence in many shooter games but still want to allow their kids to play and feel involved in the gaming community. We believe our focus on creating a silly aesthetic will attract children and give parents a viable alternative to violent shooters.

Previous Work

There are a variety of games that implement similar non-violent shooter gameplay. One such work was the recipient of 2020's COS 426 Gameplay Award. This project was called Pac-Atac and was created by Jerry Zhu, Daniel Wey, Michael Fletcher and Hollis Ma. In this project, the player plays as Pacman and can shoot a variety of objects at large Pacman ghosts. Every ghost killed adds to the player's overall score. This is a great example of a non-violent shooter game, but it had some flaws as well. First, the enemies only appear to vary in color and have no variation across attributes such as speed, damage, or score when destroyed. This creates a somewhat stagnant, unengaging gameplay. Additionally, it is a very dark game with the only colors being the characters and projectile objects. Even the characters in the game, though varying in color, are not vibrant. However, vibrancy is necessary for creating an engaging game for children since they perceive bright colors better and are therefore drawn to brighter colors (Kids and Color). Overall, Pac-Atac is an excellent game with many brilliant details, but it does not deliver on some key details necessary for making an engaging children's game.

Another shooting game which avoids depicting violence towards humans is the virtual reality (VR) game: Hot Dogs, Horseshoes, and Hand Grenades. This game prides itself on having 0 human targets and has the player shoot at hot dogs instead (Hot Dogs, Horseshoes, and Hand Grenades). Once again, this game does not meet some of the goals laid out previously. First, the game does not shoot at humans, but it does shoot at humanoids. Often the wieners appear to have human characteristics such as clothing and sunglasses as can be seen in the image below. This creates a false sense of not depicting violence when, in reality, it still does depict violence towards humanoids. Furthermore, the controls of this game are tailored for a slightly older audience which makes it untenable as a game for young children. Thus, though Hot Dogs, Horseshoes, and Hand Grenades provides engaging gameplay and has 0 human targets, it is not sufficient in its avoidance of depicting violence and is not suitable for children.



Although there is plenty of previous work to draw on, we believe the aforementioned inspirations do not sufficiently meet our goals. To remedy this, we sought to fix the issues in gameplay and aesthetic that we identified.

Approach

We sought to create a game that met our desires for non-violence, child engagement, and general wackiness. Thus, across all aspects of our approach, we kept these desires in mind. To meet the non-violence criterion, we settled on making various fruits as the enemy NPC's. We settled on this because making the enemies fruits not only met our desire for non-violence, but it is also whacky and an engaging concept to children due to the colorfulness and personification of nature. To address the enemy issues identified in the game, Pac-Atac, our approach was to give variability across different attributes (e.g. speed). The decision to make the player a clock naturally followed in the vein wackiness. Furthermore, the depiction of the contention of food with time was made with desire to create a subliminal message concerning food safety. Thus, the approach to the design of the playable character as well as the enemies met our design desires across the board.

For projectiles, our approach was to make a projectile that did not appear like a bullet or any other weapon's projectiles. To do this, our approach was to use a die which signaled the randomness of the damage of time upon fruit. We wanted to make the process of shooting engaging, so we wanted to give a small amount of kickback to any enemy hit by a projectile. Overall, our approach to projectiles achieved the non-violent design goals as well as made the game more engaging via pseudo-physical interactions.

Finally, for general aesthetic, our approach was to leverage color to make the game more engaging. One way this was achieved was through the fruits visually ripening by becoming darker as they took damage. Another approach was to give the walls texture. The game Pac-Atac only had mesh walls which limited the engagement of players with their surroundings. By mapping an image onto our walls, our approach gave some visual intricacies to our map. Unfortunately, the texture mapping gave us difficulties on the deployed version, but we utilized texture mapping in our personal version. Therefore, our approach to the general aesthetic of the game made it engaging for players since they can visually see fruit ripening and the detail of the textures mapped onto the walls.

Methods

Player and Camera Movement

During gameplay users can navigate the scene as the clock using the WASD keys on the keyboard and move the camera by clicking and dragging the mouse. The clock is always oriented away from the camera and the keys move the player based on the direction the clock is facing (ie, 'W' moves the player away from the camera). The camera is also bounded by a

maximum and minimum height to make sure players cannot view underneath the scene or directly above it. During the menu and defeat screen, all movement and camera controls are locked.

Scene Layout

On the title and defeat screens, the scene consists of four walls, the floor, the player and text. We chose the color scheme of the walls and floors based on the meshes we fetched for our objects in order to give the game a cohesive feel, and we added images to the wall to give the arena a bit more detail. The font and color of the text was chosen to fit the theme of a zombie apocalypse with a more lighthearted feel. In contrast with the pastel colors and simple models, the horrific font is meant to create an atmosphere of humor. Once the game starts, the text objects are removed from the global ‘text’ list and enemies begin to spawn. Thereafter, the player can create new dice objects which serve as projectiles by pressing spacebar.

Enemy Spawning

Two enemies spawn immediately at the start of the game and after one is killed, two more spawn randomly on the map. Enemies are initialized with a pre-determined health and damage output depending on what type of object they are. They are also given a random speed bounded by different values for each object. After taking the requisite amount of damage, (aka, after enemies are sufficiently ripened), they will de-spawn.

Since this is a universally agreed upon and well known fact that apples are nature’s cannons, we decided to give them high health, medium-high damage, and slow movement. Everyone also knows that bananas are highly aerodynamic and wimpy since they ripen quickly and don’t hurt when being whacked. They therefore have low damage and health, but high speed. Finally, avocados are nature’s glass canon, and as such are given high damage, low health, and medium speed.

Enemy Behavior

Certain enemies are adjusted to always face the player by calculating the direction from the enemies position to the clock position. This is meant to give the appearance of chasing the player. As enemies take damage the color of the material gets closer to black representing ripening of the produce. This was implemented by multiplying the ratio of current health to max health by the color of the original material. Since each object contained multiple meshes each with a different material and color, we had to loop over each mesh to recalculate the new appearance.

Shooting

During each round, players can press the spacebar to shoot dice. After being shot each die is moved forwards on every timestep and checked for collision with enemies and walls. If a collision occurred on a given timestep, the projectile object is removed from the scene. If a collision occurs with a wall nothing happens, but if collision occurs with an enemy, a random number between one and six is generated and subtracted from the enemy's health. Additionally, to avoid giving the player an unfair advantage, there is a cooldown associated with each shot. Until a specified number of frames have passed, the player cannot take another shot. To give the player a fighting chance as more enemies appear, this cooldown is reduced after achieving certain benchmark scores. This causes the fire rate to increase and therefore more enemies to be destroyed.

Collisions

There are a few different collisions implemented in the game. First, to give a visual cue that the player has taken damage, we apply a knockback effect when an enemy collides with the player. The direction of this knockback is calculated by taking the direction the enemy was traveling prior to the collision and applying a scaling factor to this vector. This vector is added to the velocity variable associated with the player. On every time step, the velocity is added to the player's position and then a friction factor is applied. Additionally, we added wall collisions to prevent players from clipping out of the map. This acts very similarly to the enemy collisions but with a smaller scaling factor. Finally, to prevent all the enemies from overlapping, we applied a detection method for enemies amongst themselves. Rather than adding a kickback, we apply a small "bump" to the enemy that moved into the other enemy's bounding box. Due to computational cost and issues when applying numerous "bumps" in different directions, we used the heuristic of only applying this interaction to a single other enemy.

Scoring

Each enemy has an associated xp value which players are awarded for defeating. These values are accumulated in a global variable 'xp' along with a global variable 'level'. This level value increases with every 100 xp points and each new level allows players to shoot faster. At the end of the game the score is displayed and reset for the next round.

States

The game state is initialized in the 'start' state which consists of the clock and the title. Once players hit the spacebar the eventhandler changes the state to 'play' and the text is removed. This transition also spawns the first two enemies and allows camera and player movement. After the player loses all their health it triggers the 'defeat' state in which players are returned to

the starting position and shown their final score. This transition also resets all global variables and includes a transition back to the ‘play’ state once players hit the spacebar.

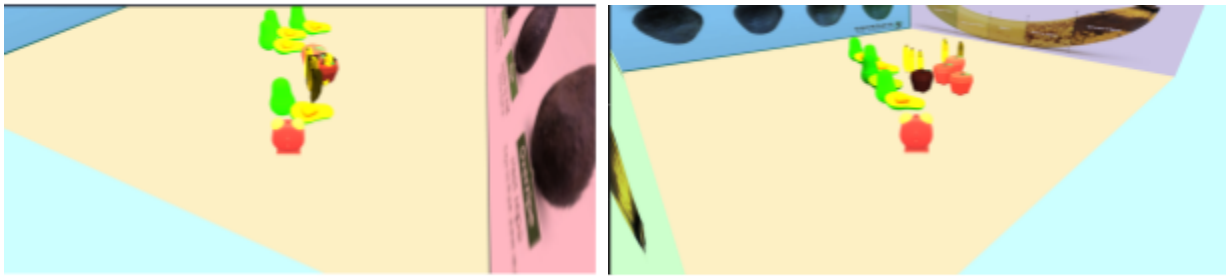
Results

How We Measured Success

We measured the success of the game in a few different ways. First, we had our friends and family play the game and give honest feedback. This gives an idea of the general quality of the game. The feedback was generally positive with most suggestions concerning adding additional features. Unfortunately, limited time did not allow for many features to be implemented, nor did it allow the game to be tested on a wider audience. Second, we observed the score and the gameplay time of those who played it. This was done to measure how the difficulty scaled as players played the game. Ideally, a game scales in difficulty the longer a player plays so that it is not a game of attrition, but rather a game of skill. We noticed a definite critical point in length of gameplay among the limited cases we observed. That is to say, as enemies reproduced, it became increasingly difficult to survive in the game.

Testing

We performed unit testing and observational testing during the production of this game. We utilized unit testing for testing individual utility functions such as collision detection, enemy detection, and spawning. This took the form of inputting values with known outputs from the functions. For instance, we tested wall detection by inputting boxes that should intersect the wall and boxes that should not intersect the wall. Overall, unit testing was very useful for production and development as it demonstrated what functions had errors. However, unit testing is not as useful for real game play scenarios. Thus, to test gameplay we used observational testing. This could take many forms. To give some examples, we would repeatedly make contact with a wall to clip through, trap ourselves in corners and attempt to glitch out of the room when enemies attacked among other things. Another observational method was to just observe how improvements effected the gameplay aesthetic. One piece of feedback we received was that enemies clipped into each other too much. We made some collision detection alterations to NPC’s and observed the changes seen in the images below. Although not quantifiable, it is qualitatively a huge improvement, and we measured this as a success.



Clipping | No Clipping

What This Indicates

Overall, we had positive reviews and we implemented gameplay suggestions that we received to a qualitatively adequate level. As with all games, evaluation is a qualitative endeavor, so given the positive reviews from adult friends and family, we extrapolate that our game would be engaging to a younger audience as well. By observing improvements made such as the one seen in the images above, we conclude that our implementation was a general success.

Discussion

Evaluating Our Approach

Generally speaking, we were very pleased with our approach to this project. Having a clear MVP and stretch goals from day one made it easy to track our progress and determine the best course of action. Our use of Github's project page and issue tracking also allowed us to effectively divide work among ourselves and stay up to date with the progress of the other person. Visual Studio Code liveshare also came in handy during the early stages of development while setting up the initial directory structure.

Follow-up Work

Without a doubt, most of the follow-up work we would do would concern improving the map layout and implementing additional features such as sound effects. Doing this would round out the game experience that this project provides. Additionally, implementing power-ups or other level scaling features would increase the length of playability. It would also be nice to have separate home, pause, and defeat screens rather than simply adding text as an object. Finally, there are a couple bugs such as minor issues with enemy rotation that could be addressed to give a more polished feel.

Conclusion

Evaluating Our Goals

Upon reflection, we believe our goal of providing a non-violent, shooter game that is engaging to children was both reasonable and well executed. The game is light hearted and fun, and it provides an engaging way to pass time. There are undoubtedly area of improvement, but we were at a minimum able to complete the core features of our MVP.

What We Learned

Working on this project provided us a valuable opportunity to apply our knowledge of computer graphics concepts like rendering, manipulating meshes, and simulating physical interactions. Understanding the initial code required us to understand intimate details of the three.js library and how to leverage the API of different objects to our advantage. Learning how to effectively use github also became essential as time went on and we anticipate it being a valuable tool in the future.

Works Cited:

Meshes - https://poly.google.com/
Pac-Atachttps://github.com/MichaelF49/Pacman3D
Kids and Color - https://sciencing.com/do-bright-colors-appeal-kids-5476948.html
Hot Dog, Horseshoes, and Hand Grenades - http://hotdogshorseshoesandhandgrenades.com/
Three.js Resources - https://threejs.org/docs/