Pacman 3D

Jasamrit Rahala, Joseph Rupertus, Itoro Ekpenyong, Iniabasi Ekpenyong

---

Abstract

We created the classic video game Pacman in 3D using Three.js. This implementation offers a first-person experience where the player follows Pacman as he navigates a maze, collecting pellets and avoiding ghosts. Pacman scores points by consuming pellets, and the game ends if a ghost catches him. However, eating a power pellet gives Pacman a temporary ability to eat the ghosts, increasing the score.

Introduction

PacMan remains one of the most iconic arcade games of all time. Thus, we decided to take from this popular 2D game and make our own rendition of it in a 3D environment. Our goal was to recreate the feel of Pacman but from a first-person perspective. Instead of looking at the maze from a birds-eye-view, the perspective sets the player inside the maze from Pacman's point of view, this helps to add a new layer of difficulty and enjoyment for the player, as this viewpoint is much more immersive and terrifying

---

Previous Work

The majority of Pacman adaptations use the original 2D implementation to preserve gameplay and simplicity of the camera view. The original implementation of the game uses a set of rules to determine how the ghosts will chase Pacman around the maze. An article online contains most of the mechanics. This is very useful as we are emulating the original game, but brings up different challenges.

In short, the ghosts may move around the maze in the four cardinal directions and only change direction at an intersection point. That is a point where they have a choice to turn left or right. A ghost may never do a 180 and reverse direction unless its state is changed. Ghosts have three states: chasing, scattered, and frightened. In all of these states, the ghosts have a target position they want to get to. This is updated for each frame. If a ghost is not at an intersection they continue to walk in their current direction. If they are at one, they choose the up/down/left/right tile that brings them closest to the target square. We use the euclidean distance to reach the target square. The ghosts may still not turn back, so we only ever have three choices. The ghost AI is very short-sighted and can be viewed as a very greedy algorithm.

In the chasing state, the ghosts have different metrics for finding their target. Traditionally the red ghost uses the exact position of the ghost as its target. The pink ghost uses the ghost's position and displaces it by 4 units in the direction it's facing. The blue ghost uses the displacement between the red ghost and the player scales this displacement by 2 and adds it to the red ghost's position, to set its target. The orange ghost does the same thing but with the player's position displaced by 2 in its facing direction.

If the orange ghost is close to the player, it sets its chasing state target to its scatter state target, which I will talk about.

Each ghost also has a scatter state target, which is a fixed square, in the top left, right or bottom left, and right of the maze. The squares these targets are set to are unreachable (e.g. the coordinates don't exist -1, -3), therefore the ghosts are stuck in a loop trying to reach these areas. This leads to the ghosts wandering around the four corners.

Finally, each ghost has a frightened state, which means that the ghost makes random direction choices each time they walk. This can be thought of as setting a random target position as well.

Pushing the project to 3D has come with its challenges:

Our first-person implementations limit the field of view, making ghost evasion nearly impossible, especially around corners. A key mechanic of the original Pacman is knowing where all the ghosts are at all times to plan ahead. This becomes a lot more complicated when the ghosts are blocked by the 3d walls. As a result, we have to rely on graphic effects to get across the point that ghosts are approaching.

Another mechanic that is lost from the game is that it becomes hard to tell when the ghosts are edible by Pacman. This is a hurdle we had to overcome by swapping out meshes/colours. Using effects like the bloom on certain colors especially helped.

A big issue with importing this game to 3D was that the ghost movement is relatively discrete and collision checking boils down to calculating 2D grid coordinates. However, in 3D, this is a lot more complicated. Animating the movement involves storing a variable "walkingProgress" for the ghosts and lerping between start and endpoints. During this movement, the ghosts also have to check for collisions. Tests of these mechanics are shown in the /test/ folder of GitHub.

Despite these shortcomings, successful adaptations have retained elements such as ghost behavior patterns or creative tweaks to maintain the game's balance of strategy and unpredictability.

Approach

Our implementation preserves Pacman's core mechanics while transforming the visuals and player perspective. The game environment was built in Three.js, where the maze is a 3D representation derived from a simple 2D grid. Also, we recreated the original game's ghost behavior, including chase, scatter, and frightened modes. The player's motion, rendered in 3D, interacts with the ghost set patterns to create the 3D-Pacman game. For the ghost interactions, we consistently checked the distance between the ghost, and the spherical mesh of Pacman to tell whether or not it's game over. Under the circumstances that the player has optimal awareness and prediction skills to predict the ghost movements, the approach works well. To facilitate the player's vision and predictions, when positioned the camera has a partial overview of the maze in first person perspective. Also, if our player has played any first-perspective type game, like Halo, the movement is similar and if they are also familiar with Pacman 2D the gameplay and strategies for beating the ghosts are fairly the same. We think it works in these circumstances because we

aren’t really changing any of Pacman’s core mechanics essentially then the circumstances in our implementation for this game would work are the instances in which the camera always look around, and over the maze to see what's coming and whether or not you've played a game like this previously.

Methodology

Maze Design

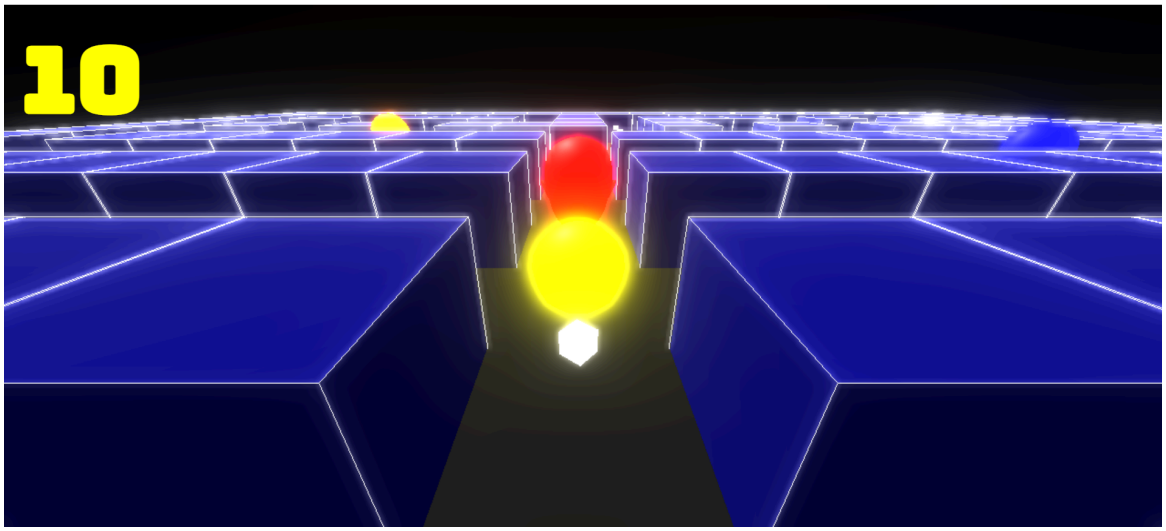
The maze is implemented using primitive boxes in ThreeJS and is represented internally by a 2D array of states (i.e., wall, pellet, power pellet, empty). The allowed motion of Pacman is determined by these states. Originally, we placed the camera above the maze for an overhead view, similar to the way classic 2D Pacman is played. We realized it would make for a more unique and interesting experience having the player take the place of Pacman and feel like they are really in the maze. However, this approach made the game extremely difficult and almost impossible to avoid ghosts, especially coming around corners. The compromise we landed on is having the player follow PacMan slightly behind, with a partial view of the surrounding maze area. This wider field of view still makes the game feel unique and immersive while also being playable. Furthermore, to make it look more aesthetically pleasing, and give the game a neo-retro vibe we added white lines to border the meshes of the wall groupings that were created.

Ghosts

For the mesh of the ghosts, we borrowed the mesh online (link is referenced below) and added free meshes found online by importing an obj loader. We created a separate file called Ghost.js for the ghost mesh, initialized each mesh within the Game.js file, and loaded the original colors (red, blue, pink, orange).

We implemented ghost algorithms based on the original Pacman game:

- **Red Ghost:** Directly targets Pacman’s current position, creating constant pressure.
- **Pink Ghost:** Aims for a position four tiles ahead of Pacman, leveraging predictive behavior.
- **Blue Ghost:** Targets a position derived from both the red ghost and Pacman’s location, introducing variability.
- **Orange Ghost:** Combines proximity-based logic, fleeing to a safe distance if close, but pursuing when farther away.



Ghosts switch between chase, scattered/frightened, and defeated modes, ensuring varied pacing within gameplay. When the ghost is frightened, the ghosts themselves alternate between blue and white before returning to their original colors at the end of the duration. This flashing and slight viewpoint zoom-out and back to it essentially tells the player that they are ready to be killed. The ghosts also take into account the way they are facing after every turn, ensuring the ghost is always front-facing when chasing the player mesh, ensuring a more thrilling experience to gameplay. One disadvantage to this implementation of utilizing the original PacMan algorithms is the reuse and lack of originality that comes with it. An alternate implementation at least in regards to the pathfinding algorithms for the ghosts is that we could have had the ghosts follow a predetermined path; however this would take away from the variability and overall enjoyment of the game, hence why we opted for a more complex implementation of the ghost algorithms.

Player Mechanics

The player is represented as a point with a spherical mesh. To begin with, we implemented a 2d version of Pacman with a 2d grid. The controls were arrow key based and we had to keep track of Pac-Man's discrete grid cell position. This was a simple starting point. We also implemented the ghost AI as described above. Then we ported this to 3d by scaling everything by a factor of z. The left and right arrows control the camera view/perspective of the player, and the player just moves where the camera is facing, thus essentially there are only two controls for the player's movement(the up and down arrow keys). Furthermore, “V” can be pressed to change the field of view. Also added generic Pacman sounds as background music, and pellet sounds each time a player intersects pellets.

Lacking Features (Ones we did not Implement)

Unfortunately due to the lack of time as well as the scope of other areas of the project, some of these extensions were unable to be implemented.

Ghost Implementation

Ghost Sound: ties audio to ghost movement; the closer the ghosts are the louder

their noise grows(adds more suspense).

Wacky powerups

Transparency: power-up that allows you to see through the walls of the maze.

Invisibility: powerup that makes the player undetectable/unhittable to ghosts.

Game Modes

Blind Mode: The hardest mode makes the ghosts completely invisible and undetectable without sound

---

Results

To evaluate the success of our game, we focused on two key factors: immersion and playability. Immersion was a standout feature, with players reporting that they felt "inside" the maze, experiencing a sense of tension reminiscent of the original Pacman. The combination of the first-person perspective and dynamic ghost behaviors contributed to this heightened sense of engagement. Playability, despite the challenges posed by the first-person visuals, was maintained through balanced difficulty. The ghost AI delivered a mix of predictable and unpredictable movements, preserving the strategic elements of the original game while adapting to the 3D environment.

We conducted two types of tests to gauge gameplay effectiveness:

1. **Blind Test:** Players unfamiliar with the original Pacman reported high levels of engagement but found ghost evasion challenging, particularly in the constrained 3D environment.
2. **Veteran Test:** Players familiar with the classic Pacman appreciated the blend of nostalgia and innovation. They noted that the ghost behaviors closely resembled the original game's mechanics while benefiting from the added complexity of a 3D perspective.

Strengths

- Immersive Experience: The "follow-behind" camera provided a unique balance between first-person immersion and gameplay functionality. This perspective allowed players to feel part of the maze while retaining a partial overview for strategic planning.
- Faithful Ghost AI: The ghost behaviors effectively captured the tension and excitement of the original Pacman. Adapting these patterns to 3D without losing their core mechanics ensured both authenticity and a challenging gameplay experience.

Weaknesses

- Visibility Challenges: While the camera design offered some peripheral awareness, certain maze configurations created blind spots that limited players' reaction time, making ghost evasion particularly difficult.
- Simplistic Maze Design: The static maze layout reduced replayability. Introducing procedural or dynamic maze generation could significantly enhance the game's long-term appeal.

Ethical Concerns

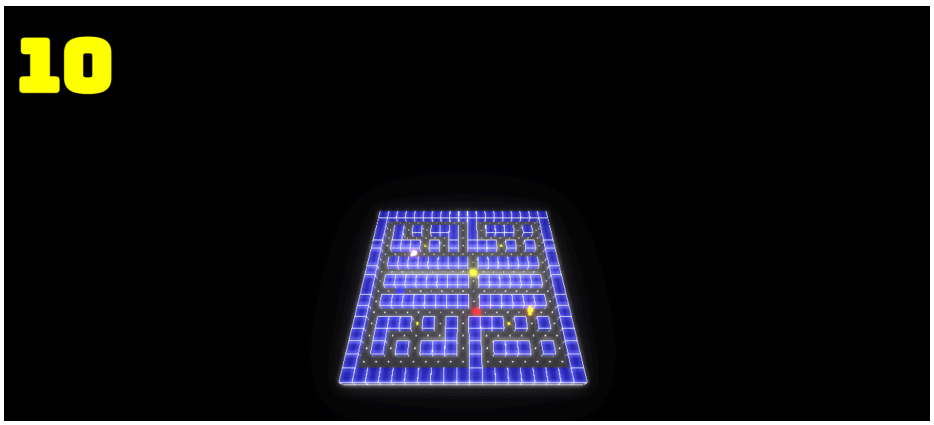
Our 3D Pacman project, while an academic exploration, raises two key ethical concerns: intellectual property infringement and accessibility and inclusivity challenges. Both aspects have implications for fairness, respect, and social responsibility, as outlined by the ACM Code of Ethics and Professional Conduct.

Pacman is a trademarked intellectual property owned by Bandai Namco. Replicating its mechanics, characters, and visual design risks violating copyright and trademark laws. While the project is intended as an educational homage and not for commercial purposes, the use of original elements like ghost behaviors and visual aesthetics could still be deemed problematic.

To mitigate this, we can ensure that the project credits the original creators explicitly within our documentation and clarifies its non-commercial, educational intent. Additionally, we could replace recognizable assets, such as character designs and names, with original creations inspired by the mechanics of Pacman but distinct in identity. This aligns with the ACM guideline of *avoiding harm* and *respecting intellectual property*.

The immersive, first-person perspective and keyboard-based controls may exclude certain groups of users, particularly those with motion sensitivity or physical disabilities that hinder interaction with standard input devices. This reduces the inclusivity of the project and could alienate players who cannot experience the game as intended.

To address this, we could include customizable control schemes and provide alternative perspectives, such as an overhead view option, as shown below, to accommodate different player needs. Following the ACM principle of *ensuring fairness* and *acknowledging the diverse needs of all users*, these modifications would make the game more accessible while maintaining its immersive aspects.



Future Work

Potential next steps include:

1. Procedurally generated mazes for enhanced replayability.
2. Enhanced visuals, such as dynamic lighting or maze textures.
3. Online leaderboard integration to increase competition.
4. The refined player controls for even greater fluidity.

Lessons Learned

This project highlighted the challenges of translating 2D designs into immersive 3D experiences. It's hard to strike the right balance between nostalgia and innovation, especially for iconic games.

Conclusion

Overall, our implementation of PacMan successfully marries the nostalgia of the original Pacman with the immersiveness of 3D gameplay. While maintaining classic gameplay mechanics, we introduced novel perspectives and dynamic ghost behaviors. The result is an engaging, immersive experience that respects its roots. However, further enhancements to maze design and controls can elevate the game. Future iterations should build upon this foundation, with the potential to redefine how classic arcade games are experienced in modern contexts.

References

- Original Pacman arcade algorithms:https://gameinternals.com/understanding-pac-man-ghost-behavior
- Three.js documentation for 3D modeling
- Ghost Mesh: https://free3d.com/3d-model/femalehead-smiling-v3--251270.html