The Tetromino Trials

Abani Ahmed, Teddy Feig, Jalen Johnson
COS 426 – Professor Adam Finkelstein
Fall 2024

Abstract

The Tetromino Trials is a game where players are trapped inside a GameBoy hand-held console, and have to avoid the falling blocks as they use them as platforms to climb higher up within the console in order to escape. However, there is no escape, and so the player has to climb up as high as possible or avoid falling blocks as long as possible. The player gets a score, which is determined by how high they can get, so that way they don't try to stay at the bottom of the game.

Introduction

We set out to create a retro horror platformer that combines nostalgia with dread. The Tetromino Trials immerses players in a GameBoy-inspired world where the familiar charm of falling blocks is distorted into an oppressive and eerie experience.

On *itch.io* there are various types of Tetris platformers, but most of them are in 2D. For our game, we wanted the player to be immersed inside the actual game, as opposed to playing it from the outside. Some of the games let the player move the tetris blocks, but we wanted that to be out of our player's control to add to the feeling of helplessness.

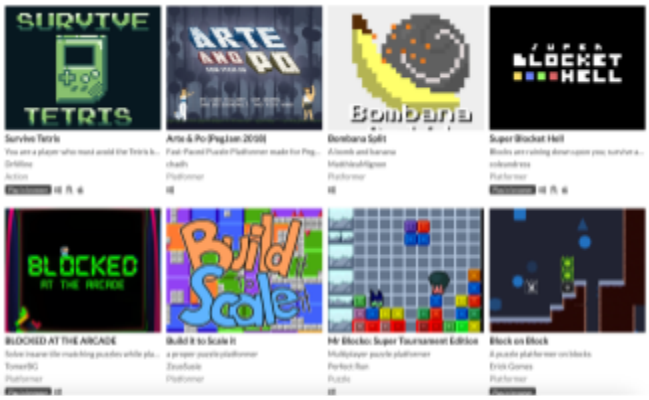

Figure 1: Various other Indie Tetris Platformers on Itch.io that are in 2D. *Survive Tetris* was similar to what we wanted to achieve, but in 2D, and a few different mechanics.

Our approach was to model a GameBoy and have the player spawn inside of it. Then, we randomly generate Tetris pieces that could fall on top of the player. If a piece lands on top of the player, then they die and the game resets. This approach relies on the tension of survival mechanics paired with atmospheric horror elements. The unpredictability of falling blocks keeps players on edge, while the visual and audio design enhances the sense of confinement and inevitability.

The game is most effective when players are immersed in its audio-visual environment. The looping nature of the gameplay amplifies the eerie tone, making every attempt feel like another step deeper into an inescapable nightmare.

Methodology

We first tried to understand the example code provided, with the flower on a piece of land. We understood that we could import our own assets into the game, so we decided to make those in Blender.

Step 1: Importing a Cube and a Floor

We first imported a cube from assignment 3 (Raytracer) to make sure that we could properly load objects as .obj files. Next, we created a floor by elongating and thinning the original cube. We had to understand what coordinate plane that the game was running on in order to do so.

Step 2: Generating Random Pieces

Our next step was to generate random Tetris pieces. We did this by writing a generatePiece() method. The generatePiece() method creates a random Tetris-like piece consisting of four cubes in a 3D space. It starts by selecting a random piece type (e.g., T-piece, L-piece, Zigzag, Square, or Straight) and a random color from a predefined set. Four cubes are then instantiated, with the first cube positioned randomly and the remaining three cubes arranged relative to it based on the selected piece type and rotation logic. The cubes are added to the 3D scene and stored in a falling list to track their movement and interactions within the game. This method enables the creation of dynamic, visually consistent Tetris pieces with varying shapes, orientations, and colors.

Step 3: Dropping Pieces Randomly

After randomly generating a piece, we needed to have them randomly fall. The dropPieces() method manages this behavior: For each falling piece, it attempts to shift its cubes downward by one unit while checking for collisions with the ground, other cubes, or the player. If a cube collides with the height map or an existing structure, the piece is added to the resting list, and the height map is updated to reflect the new maximum height. If a falling cube collides with the player's position, the method returns true to indicate that the player has been hit and the game ends. Otherwise, the updated cubes are added back to the falling list, and the game continues. This method ensures realistic piece behavior, manages the transition from falling to resting states, and integrates collision detection to maintain a dynamic and engaging gameplay experience.

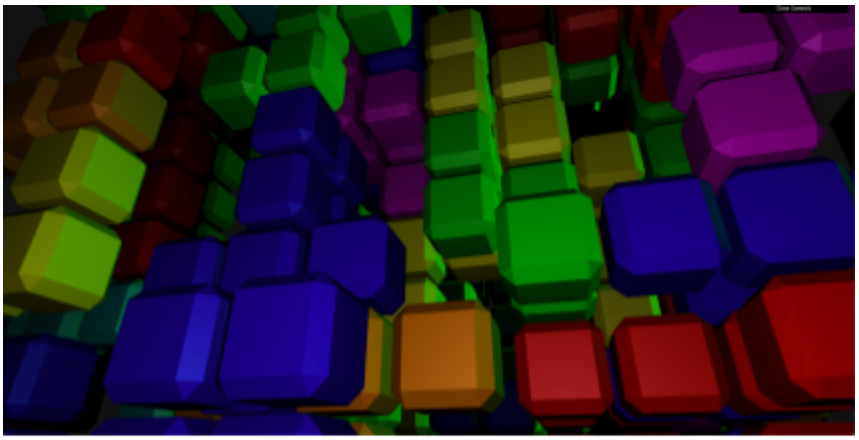

Figure 2: The Random Generation and Falling of the Blocks

Step 4: Collision

Now that we had the pieces falling randomly, we needed to make sure that if the player collides with one from any of its sides (such as jumping and bumping their head on a cube, running into one from the side, etc.) the player does not go through the cube.

1. handleFallCollision(cameraPosition)

This method detects when the player lands on a cube or is inside a cube while falling. It checks the player's position (cameraPosition) against all cubes in the game (both falling and stationary). The method determines whether the player is directly above or inside a cube by comparing their position to the cube's boundaries. If the player is inside a cube, it immediately returns the necessary displacement to correct their position. Otherwise, it calculates the closest distance to the top of the cube beneath them to determine if they should stop falling. This ensures that the player's movement stops naturally when they land on a cube.

2. bumpHeadCollisions(cameraPosition)

This method detects when the player bumps their head against a cube from below. It checks if the player is inside the boundaries of a cube by comparing their position to the cube's center and size. If a collision is detected, it calculates the vertical displacement needed to push the player out of the cube and returns it. This prevents the player from unrealistically clipping through cubes when jumping or moving upward.

3. handleSideCollisions(cameraPosition, previousCamera)

This method handles side collisions, ensuring the player doesn't move through cubes when walking or running into them. It compares the player's position (cameraPosition) and their previous position (previousCamera) to determine if they are inside a cube. If so, it calculates the direction the player was moving and determines the minimal adjustment needed to keep them outside the cube. The method accounts for movement along the x and z axes while ignoring vertical movement. Additionally, the method checks if the player has gone out of bounds and returns a correction vector to keep them within the playable area.

Together, these methods create a collision handling system that ensures realistic player movement and interaction with the game environment.

Step 6: Importing Assets & Lighting

With the game functioning with our test assets, we created a GameBoy and a beveled cube object in Blender and imported those into the game. We then worked on lighting, trying to add a soft green light that represented light coming through the GameBoy scene, and adding a spotlight to the camera, so that way the player can see clearer as they move closer to the blocks, like human vision.

Step 7: Start and End Screen

Finally, we wanted an end screen to appear for the player once they died. We did this by creating an End Scene object in Blender, as we originally did not know how to show text. However, once we figured that out, we got the score to show, as well as a

little Title screen. The player can click anywhere to start the game on the start screen and on the end screen to restart the game.

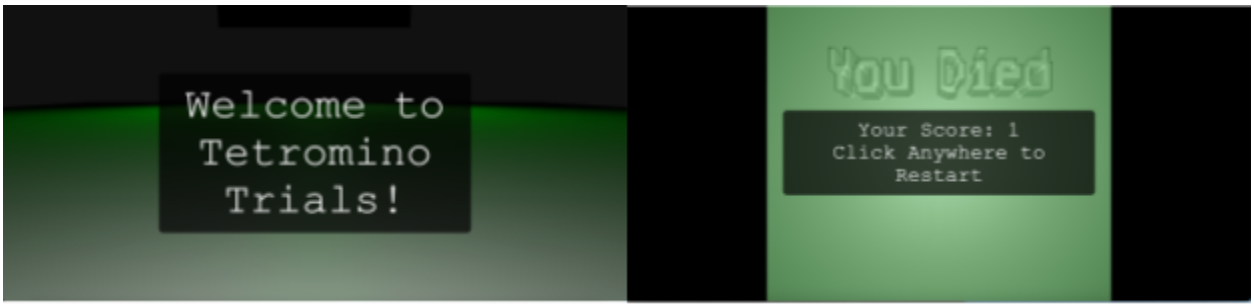

Figure 3: Start and End Screens

Things we did not implement:

There were many things we wanted as stretch goals, but we were unable to implement given our time constraints including importing materials, having a camera that starts outside the screen and moves you into the GameBoy, a white light at the top of the scene so that way the player tries to move towards it, and having the walls of the Gameboy move up as you move up.

One feature that was a longer stretch goal was to have the pieces fall with more planning, so that they could actually form line clears and the player is even more forced to make it to the top of the GameBoy, or else a piece will definitely fall on their head.

Results

Overall, we achieved significant progress beyond our initial expectations for The Tetromino Trials. In addition to finalizing the core gameplay mechanics, we refined the visual and audio design to align with the game's eerie, retro theme. We experimented with falling block patterns and randomized player challenges, which added unpredictability to gameplay and kept players engaged. Through iterations, we fine-tuned the difficulty curve to balance fairness and tension, ensuring that the game felt challenging but not overly punishing.

We were able to find many solutions to problems that we were facing. For example, we wanted to have a black border around the blocks for the Tetris vibe, however we had significant problems importing custom materials from Blender. We decided that because our lighting was “spooky” and would allow us to see shadows, we could bevel the cubes so that way the border is based on the lighting. This not only worked, but gave it a much stronger “Tetris” feeling.

Our experimentation involved intensive playtesting with ourselves and a few peers. Many of our peers found the game to be really fun to play, with a lot of mentions of “Minecraft Parkour” and genuine disappointment if they fell to the bottom of the game. While there is still room to develop additional features, such as advanced visual effects,

the current state of the game showcases its potential as a suspenseful, retro-inspired survival platformer.

Discussion

A few parts of our approach need to be changed for further work. For one, we may switch from .obj files to .gltf files as shown in the flower example. We chose .obj based on familiarity, but we struggled to load the proper materials.

Our code also needs to be much more modularized. We would like to move our collision detection into a separate file, as well as our camera movement and generation methods.

Finally, we would love to implement the features above that we were unable to. We believe that our game is not as frightening as we would like it to be, and implementing those features would not only make it more realistic to the game of Tetris, but would make it spookier and more unnerving.

Ethical Concerns

The Tetromino Trials presents three notable ethical concerns: the psychological impact of its themes, the accessibility of its design, and the use of creative elements inspired by Tetris.

1. Psychological Impact

The game's focus on themes of confinement and inevitability creates an intentionally tense and spooky atmosphere. While this adds to the immersive experience, it may inadvertently cause distress for some players, especially those prone to anxiety, claustrophobia, or frustration. According to the ACM Code of Ethics, Principle 1.2 emphasizes the importance of "avoiding harm," which includes psychological harm. To address this, the game could include clear content warnings about its themes and tone, allowing players to make informed decisions about whether to play. Additionally, implementing adjustable difficulty settings or a less intense mode would help make the game more accessible to players who may find the core experience overwhelming.

2. Accessibility of Design

The game's retro grayscale aesthetic and precision platforming mechanics may unintentionally exclude certain players, such as those with visual or motor impairments. The ACM Code of Ethics, Principle 1.4, highlights the importance of "ensuring equitable access," which involves accommodating diverse user needs. To mitigate this, the game could offer accessibility features such as adjustable contrast settings, alternative color palettes, and customizable controls. Including a mode with slower-paced gameplay or reduced platforming difficulty would make the game more inclusive for a wider range of players.

3. Use of Creative Elements Inspired by Tetris

The game's falling block mechanics and overall concept draw heavily from Tetris, one of the most iconic and recognizable intellectual properties in gaming. While creative inspiration is essential, the ACM Code of Ethics, Principle 1.5, stresses the importance of "respecting intellectual property." Without sufficient differentiation, the project risks being perceived as derivative or infringing upon the original work. To mitigate this concern, we should clearly distinguish ourselves through unique mechanics, narrative elements, and visual design. Additionally, acknowledging the influence of Tetris in project documentation or promotional materials demonstrates transparency and respect for the original creators.

By addressing these three ethical concerns, we can align more closely with professional standards of ethical game development. These measures not only foster inclusivity and player well-being but also ensure that the game is recognized as a distinct and innovative work that respects the contributions of its predecessors.

Conclusion

We successfully achieved our goal of creating a retro-inspired survival platformer that combines engaging gameplay with a spooky and nostalgic atmosphere. Through our use of eerie visuals and challenging mechanics, we were able to immerse players in our unsettling world. Playtesters responded positively to the core concept, appreciating the blend of tension and platforming. However, we recognize that some aspects, such as balancing the difficulty and refining player controls, need further attention to fully realize the game's potential.

Moving forward, we plan to expand the game's mechanics to deepen the player experience. We want to develop new game modes, like structured levels or progressively harder waves, to reduce randomness while maintaining the core tension. Accessibility is another key focus—implementing features like adjustable difficulty, alternative color palettes, and customizable controls will help make the game more inclusive. Finally, adding elements such as leaderboards or rewards for high scores will encourage replayability and long-term engagement.

We need to revisit the randomness of block spawning, which occasionally leads to unavoidable player deaths and frustration. Refining the spawning logic and incorporating clearer visual or auditory cues for falling blocks will make the gameplay feel fairer. Player controls also require further optimization to ensure smooth and responsive movement. Additionally, we need to address the ethical concerns of distinguishing our work from its Tetris inspiration and ensuring the game's themes remain accessible without causing undue distress to players. By tackling these issues, we believe we can take *The Tetromino Trials* to the next level as a polished, immersive, and ethically responsible game.

Contributions

Teddy's primary focus was implementing the collision mechanics of the game, as well as the camera movements. Jalen's primary focus was generating pieces, randomizing their falling, and sound. Abani's primary focus was creating the assets and colors and importing them into the game. All three of us worked on lighting and changing scenes as well as the writeup.

Works Cited

- ACM Code of Ethics: https://www.acm.org/code-of-ethics
- Three.js: https://threejs.org/
    - Discussion Forums: https://discourse.threejs.org/
- Tetris Platformers: https://itch.io/games/genre-platformer/tag-tetris
- Halloween Tetris Music: https://www.youtube.com/watch?app=desktop&v=eunhYtd8agE&t=0s
- Tetris: https://tetris.com/