Seyi Oderinde & Dresden Lubic
Adam Finkelstein
COS426 Final Project
Friday, December 13th

Dash3D

I. Abstract

This project explores the development of an interactive 3D platformer game, designed using the Three.js library, focusing on dynamic gameplay mechanics, audio-reactive visual effects, and player-driven interactions. The core gameplay involves a player navigating obstacles and activating orbs for strategic jump boosts to traverse challenging terrains. Key features include manually placed obstacles for precision-designed levels, audio synchronization for immersive environmental effects, and different interactive elements that enhance player mobility.

The project demonstrates the integration of real-time physics, collision detection, and custom game objects to try and create a familiar yet new engaging user experience. This report outlines the design process, implementation challenges, and future enhancements for the game.

II. Introduction

The goal of this project was to develop a platformer game inspired by Geometry Dash, featuring a first-person 3D perspective, immersive gameplay mechanics, dynamic player-environment interactions, and audio-synchronized effects. The player aims to navigate through obstacles, strategically use jump pads to enhance mobility, and enjoy responsive visuals that react to the game's soundtrack. This type of game could benefit enthusiasts of casual and rhythm-based games.

A. Previous Work

Interactive platformer games have long been a staple in the gaming industry, with notable examples such as *Geometry Dash* and *Super Mario Bros.* Rhythm-based games, like *Beat Saber* and *Dance Dance Revolution*, have also effectively merged gameplay with music. Prior work in game design often focuses on level procedural generation, precise collision mechanics, and visually captivating worlds. However, these approaches sometimes fall short in offering true player interactivity with the environment or fail to incorporate dynamic visual feedback effectively. By building upon these ideas, this project introduces a unique combination of spatial and temporal mechanics, blending music-responsive visuals and manual obstacle placement for a tailored gameplay experience.

B. Approach

This project was developed using Three.js, leveraging its capabilities for real-time rendering, physics simulations, and object interactions. The core approach involved designing a player-object system from a first-person perspective, incorporating a "ship" mode for gameplay variety, and implementing custom logic for collision detection, responsive jump boosts through jump orbs, and dynamic obstacle placement. Audio-reactive visual effects were integrated by analyzing the game's soundtrack in real time, enhancing the synergy between the player's perspective, the ship's movement, and the music. By enabling manual obstacle placement and incorporating music-driven feedback, the project aims to overcome some limitations seen in fully procedural or overly automatic level designs.

III. Methodology

This project leveraged Three.js to create a rhythm-based platformer game with a first-person perspective and ship-based movement, focusing on player interaction, obstacle challenges, and audio-synchronized visuals. The foundation of the game was a rule-based player-object system that allowed the user to control the player and strictly restart the game when the collision-detection was enabled with the in-game obstacles. This rule-based approach prioritized simplicity and efficiency while maintaining precise control over movement and jump mechanics. The system also supported the integration of ship-based dynamics, adding variety to the gameplay experience.

For obstacle placement, manual positioning was chosen over procedural generation to achieve precise synchronization with the game's soundtrack. Initially, we attempted to build an in-game editor to manually place objects, which would have allowed for a more interactive level design process. However, this approach introduced unnecessary complexity and lacked the precision required for rhythm-based gameplay. Instead, we chose to manually place obstacles directly through the code, which provided greater control over timing and alignment with the music. This method allowed for deliberate and finely-tuned obstacle placement along the player's path. Utility functions were implemented to streamline the manual placement process for each object, and reusable obstacle patterns were introduced to maintain consistency and efficiency. By foregoing the in-game editor, this approach ensured precise synchronization with the soundtrack, ultimately delivering a more polished and rhythmically aligned gameplay experience.

Audio-driven visual effects played a significant role in enhancing the game's immersion. The project employed real-time frequency analysis to dynamically adjust elements such as lighting and background hues, creating an environment that reacted to the soundtrack. Additionally, low-level beat tracking was integrated to synchronize visual effects with the music's tempo, amplifying the connection between the gameplay and audio. Collision-based orb interactions added another layer of responsiveness, providing immediate jump boosts with visual feedback to guide the player. By combining these elements with the first-person perspective and ship mode, the gameplay experience became more rhythmically cohesive and engaging.

IV. Results

We measured success by evaluating the synchronization of obstacles with the soundtrack, the responsiveness of player mechanics, and the cohesion of audio-visual effects. Testing focused on the accuracy of manually placed obstacles in aligning with musical beats, the responsiveness of player actions such as jumping and orb interactions, and the impact of audio-reactive visual effects on immersion. Results showed that direct obstacle placement through code, chosen over an in-game editor, provided better precision for obstacle rhythm matching. Player mechanics, including movement and orb jump boosts, provided an instant, smooth response time. The first-person perspective and ship mode added depth and variety to the level. Additionally, the audio-driven visual effects dynamically synchronized with the soundtrack added to the immersion. While the project achieved its goals, a more extensive level design with mpre advanced effects would further improve the experience.

V. Discussion

The approach we took—manually synchronizing obstacles and incorporating audio-responsive visuals—proved to be a promising method for enhancing gameplay immersion and interaction. The combination of rhythmic challenges and dynamic visual effects created an engaging experience. However, a more automated system for beat detection and obstacle placement could streamline development and improve scalability for varied music tracks. Future iterations could explore machine learning models or signal-processing techniques to identify beats and adaptively generate gameplay elements.

A. Ethical Concerns

Two ethical concerns arise in the context of this project. First, the accessibility of the game could be objectionable to individuals with disabilities, such as those with visual impairments or limited dexterity. This aligns with principles of inclusivity and fairness in the ACM Code of Ethics, which emphasizes ensuring "equal access to computing resources" and "respecting the rights of all affected parties." Mitigation strategies could involve adding accessibility features, such as customizable controls, visual cues, or auditory alternatives to visual effects.

Second, potential over-reliance on real-world music could lead to copyright issues, raising concerns about intellectual property. The ACM's principle to "give proper credit for intellectual property" underscores the need to respect the rights of content creators. To address this, the game could prioritize the use of royalty-free tracks, partnerships with artists, or user-uploaded content with appropriate licensing checks.

B. Follow-Up Work

Future work could focus on automating obstacle placement through real-time beat analysis to enhance scalability and flexibility. Reintroducing the in-game editor for manual obstacle placement could provide designers with an intuitive tool for fine-tuning levels, balancing precision with ease of use. Additional research into advanced audio-visual synchronization methods, such as more sophisticated beat detection algorithms or adaptive lighting effects, could further improve the cohesiveness of the experience. Finally, testing the game with a broader audience would help identify opportunities for refinement and gather feedback on gameplay mechanics and immersion.

C. Lessons Learned

This project highlighted the importance of modularizing code for different objects and their interactions, ensuring that each component—such as the player-object system, obstacles, orbs, and audio-driven effects—was independently manageable yet still integrated with each other. This modular approach not only streamlined development but also allowed for easier debugging, scalability, and future enhancements. The project also underscored the need to balance manual control with automation, as precise obstacle placement benefited from direct coding, while automation could enhance scalability in future iterations.

VI. Conclusion

Overall, the project successfully met its goal of creating a first-person version of a rhythm-based platformer inspired by Geometry Dash, combining synchronized obstacle placement, responsive player mechanics, and audio-driven visuals. By integrating modular systems for the player-object interactions, obstacle challenges, and real-time audio effects, the game delivered an engaging experience that captured the core appeal of Geometry Dash while innovating through a first-person perspective. These synchronized elements created an immersive and dynamic environment, balancing challenge and enjoyment, though further refinements are needed to expand scalability and accessibility.

The next steps focus on automating obstacle placement through real-time beat analysis to streamline level creation while preserving musical precision. Reintroducing an in-game editor could provide designers with intuitive tools to fine-tune levels in a way that remains faithful to the first-person Geometry Dash vision. Additionally, refining gameplay mechanics to support seamless transitions and expanding accessibility features—such as customizable settings or enhanced visual feedback—will make the game more inclusive and engaging. Testing with varied music tracks and larger user groups could be crucial for future iterations.

Key improvements to address include optimizing the balance between manual control and automation, further modularizing components for easier updates and scalability, and ensuring accessibility and intellectual property compliance. By focusing on these areas, the project can evolve into a polished, innovative rhythm-based game that builds upon the iconic Geometry Dash experience, reimagined through the lens of a first-person perspective.

References

1. Association for Computing Machinery. (2018). *ACM Code of Ethics and Professional Conduct.* Retrieved from https://www.acm.org/code-of-ethics
2. Unity Technologies. (2023). *Game Design with Unity: Audio-Visual Synchronization Techniques.* Retrieved from https://unity.com
3. Brown, J. C., & Zhang, Y. (2019). "Beat Detection and Audio Synchronization in Rhythm Games." *Journal of Game Development Research,* 12(3), 45-58.
4. Hartmann, B., & Bernstein, M. S. (2016). "Design Patterns for Rhythm-Based Game Mechanics." *Proceedings of the ACM Conference on Interactive Entertainment Design,* 85-92.
5. Three.js Documentation. (2024). *Three.js Library Reference.* Retrieved from https://threejs.org/docs/
6. Szalavári, Z., & Thomas, B. H. (1997). "Interaction Techniques for 3D Object Manipulation in Virtual Environments." *ACM Transactions on Graphics,* 16(3), 173-196.
7. Wilson, G., & Kearney, G. (2021). "Enhancing Immersion in Video Games Using Dynamic Audio-Visual Feedback." *International Conference on Game and Media Development,* 256-265.
8. MIT OpenCourseWare. (2023). *Principles of Game Design.* Retrieved from https://ocw.mit.edu