3D Breakout

Abstract

Welcome to 3D Breakout, a single-player game based on the already-existing Breakout made by Atari, Inc. The player uses arrow keys to control a moving platform to prevent the ball from hitting the ground. At the start of the game, the player is given four lives and one life is lost each time the player lets the ball hit the ground. At the same time, the player must attempt to break bricks with the ball. The game gets progressively harder: at each level, the player has to clear more bricks, and the ball moves more quickly. The goal of this game is to clear as many levels as possible with the four lives the player is given. There are currently 7 levels implemented.

Introduction

Goal

3D Breakout is a single-player game based on traditional arcade games such as Breakout. As with these traditional games, the player attempts to break bricks by bouncing a ball off a movable platform. In creating 3D Breakout, we wanted to capture the essence of traditional arcade games and bring about nostalgia with some retro fun for people who played these games while growing up. In addition, because of the current circumstances where social distancing is crucial, having an activity that a person can do on their own such as a single-player game can be very helpful for relieving boredom. We also hoped to demonstrate our ability to apply concepts learned in Computer Graphics (COS426) in the process of creating this game. We do this by introducing novel elements such as 3D visuals through the use of polygonal meshes and particle collision.

Previous Work

We took inspiration from many classic games, most obviously Atari Breakout and Blackberry's Brick Breaker. More generally, we tried to replicate the aesthetic of an older arcade game, with more updated modern visuals.

- Atari Breakout: The game begins with 8 rows of bricks lining the top third of the screen. The player must knock down as many bricks as possible by using the walls and a moving platform to ricochet the ball against the bricks. The player has three turns to do this and loses a turn each time the player fails to catch the ball's

rebound with the paddle. Ball speed increases at specific intervals based on the number of hits.
- Brick Breaker: This game is essentially a Breakout clone that came preloaded on the BlackBerry. The player must smash down walls of bricks by bouncing a ball off a paddle controlled by the BlackBerry's trackwheel.

Our 3D Breakout has the same game mechanics as these classic games. We believe the simple gameplay of bouncing a ball off a paddle to break bricks is what made the game successful and why it has inspired so many clones. However, the majority of these clones maintain the 2D visualization of the original game, while our version differs in that we use polygonal meshes to recreate the game with 3D visuals. This allows us to bring the game to life, while still introducing enough new visual elements to provide a compelling source of entertainment.

Approach

High-level game design:

- Levels: Deviating from the original Breakout game we implemented multiple levels of the game (as opposed to a single level), each level spawning a different number of bricks. We decided to do this to make the game more consumable so that players can take breaks between clearing levels. Increases in difficulty occur from level to level. Furthermore, because we wanted to customize the design of each level, we decided to make the game finite.
- Game mechanics: We use keyboard input for all input pertaining to gameplay, such as moving the platform, pausing, starting the game, etc. However, we delegated camera movement/changes in perspective to mouse input, separating it from gameplay input so as to not confuse the player.

Low-level game design:

- We used the starter code provided for us as it provided a good framework to start our project. The majority of 3D Breakout was coded using Javascript and we used the Three.js library in particular. This is because we are familiar with its usage and documentation from our assignments this semester and because it allows for efficient implementation of simple meshes and camera control. Elements of user interface such as the instructions menu were created using HTML and CSS.

Methodology

The most important pieces of 3D Breakout are the meshes and the functionalities attached to them. We used the following meshes:

- **Brick:** This is a rectangular prism mesh created with the built-in Three.js BoxGeometry. Each level is populated with a different number of bricks (the number of bricks increases for each subsequent level). When the ball comes into contact with a brick, the brick disappears with a tweening animation. Once all of the bricks of a level have disappeared, the level is cleared.
- **Ball:** This is a spherical mesh created with the built-in Three.js SphereGeometry. The player attempts to bounce this ball off a platform to prevent it from hitting the bottom of the border (i.e. the "ground") and make bricks disappear. Each time the ball bounces off an object, a sound is played.
- **Platform:** This is the only mesh that the player has control over. Like the Brick, the Platform is a rectangular prism mesh created with the built-in Three.js BoxGeometry. We applied tweening to this mesh: when given input from arrow keys, the Platform flattens and when there is no input, the Platform reverts to its original shape.
- **Border:** This is simply a frame that defines the boundaries of the game. We created the Border by merging four Box meshes. The platform, ball, and all bricks are spawned inside the border and cannot go out of the Border's bounds.
- **Heart:** This is a flat heart-shaped mesh. We referenced code from threejs.org to create it. Each heart represents a life that the player has remaining. The player starts the game with four lives and thus four hearts on the screen. Each time the player lets the ball hit the bottom of the border, the player loses a life and one heart disappears with a tweening animation.

Because the classic Breakout game simply consists of a ball and bricks, we deemed it not necessary to create any additional complicated meshes for 3D Breakout in spirit of the original game.

Besides the meshes, we also worked with HTML/CSS to create the screens for instructions and gameplay, event listeners, sound effects, and other functionality:

- **Screens:** We used HTML/CSS to create an instruction screen that shows at the beginning of the game or on pauses, a screen for when the user wins a level, a screen for when a user loses the game, and a screen for when a user wins the game.

Additionally, we added a screen prior to the beginning of every level. These screens were created using JS code to build HTML elements, then enhanced with a CSS file.

- **Event Listeners:** The game has several event listeners that allow the user to interact with it:
    - <u>Arrow Keys:</u> Note, the listener for the left and right arrow keys listens for both key up and key down to ensure smooth movement of the platform.
        - Up - starts the ball
        - Left - moves the platform left
        - Right - moves the platform right
    - <u>Spacebar:</u> The spacebar listener is used to transition between states of the game. Therefore, pressing the spacebar can be used to begin the game, begin a level, or return to the main menu to play again.
    - <u>Pause:</u> The "p" key serves as a pause/resume button. When it's pressed, the game will pause, and the instructions screen will appear. When it's pressed again, the game will resume, and the instructions screen will disappear. The pause will only work when gameplay is occurring (the ball is moving).
- **Sound Effects:** As mentioned earlier, the ball causes a sound effect when it collides with objects other than the bottom border. Additionally, the game produces appropriate sound effects when a level is won, the game is won, or the game is lost.
- **Camera Movement:** Notably, you can reposition the camera in this 3D Breakout game, so that you can view the game from different angles. This will enable the viewer to try out interesting angles to play the game from, and also allows the viewer to appreciate the 3D nature of the game build.

Results

Our primary way of measuring success was playing the game, and assessing what features we would like to be added. However, we also asked several friends and family members to play the game to get feedback. Notable feedback included:

- <u>Making the movement of the platform smoother:</u>
    - Initially, we had the platform's position move in increments. However, users commented that this detracted from the game experience.
    - As such, we used both key down and key up events to enable smooth movement of the platform while an arrow key was being pressed.
- <u>Increasing speed of ball with every level:</u>

    - Some users thought the game was too easy when the main difference between levels was the number of bricks.
    - Thus, we also increased the speed of the ball as levels progressed to make the game more difficult for users.
- Maintaining the number of hearts across all levels:
    - Originally, we had the number of hearts restart every level and the number of hearts decrease with each level. However, a user commented that this was less intuitive than just beginning with a number of hearts, and having only that many hearts across all levels.
    - As such, we did not restart the number of hearts per level. We also feel that this balanced the difficulty of the game better and helped prevent user confusion.
- Adding sounds:
    - Some users commented that it would be helpful to have auditory effects, as well as visual effects.
    - As such, we added effects for when the ball had a collision, and sound effects for losing the game, winning a level, and winning the game.

Discussion

We feel that the approach we took for this project was very promising. We ended up accomplishing most, if not all, of the functionalities we decided on when we initially planned out 3D Breakout and then some. The majority of our game is coded from scratch as we intended. However, we did run into some problems attempting to import some code. For instance, we wanted to outsource the code for a confetti animation upon beating all of the levels. However, we found difficulty in locating code that we could easily adapt. Hence, the majority of our project is still coded from scratch.

As for follow-up work, we feel that we could add more graphics components to “juice” the game and make it feel more interactive. Such components include the following:

- When a break disappears, making it look as though the brick is shattering. Currently, we have a tweening effect where the brick shrinks to nothingness.
- Creating a trail of smoke or light behind the ball.
- Having our scene drop into place from the top of the screen.
- Using a different material for the bricks.

We also feel that we could include gameplay by adding components such as the following:

- Power-ups such as having boosters that enable:
    - The ball breaking through multiple bricks
    - The platform expanding in width
    - The platform speed increasing
- Further improving the realism of our collision mechanism.

In doing this project, we learned how important graphics are to making a game feel alive and responsive. In particular, we discovered the usefulness of “tweening” to make our game more interactive and interesting to play. Additionally, we learned about how important it is to set reasonable expectations and focus on the minimum viable product, as building the minimum viable product helped us identify the most viable reach goals.

Conclusion

Our goal in creating 3D Breakout was to recreate a game that would inspire nostalgia and provide entertainment to people in these unprecedented circumstances. Based on our own experience of this game and the feedback of friends and family members, we feel that we were successful in this endeavor. Furthermore, we feel that this project also successfully demonstrates our ability to apply concepts learned in COS426 as we made use of meshes, particle collision, animation, and other ideas from the course material. Some next steps would be to implement the follow-up work described above and bring this game to other platforms such as smartphone apps.

Overall, this project was a very fun experience and provided a wonderful opportunity to apply the knowledge we have gained throughout the semester.

Contributions

For this project, all three members worked equally to brainstorm and contribute ideas to the game. Justin handled the majority of the particle collision mechanisms and incorporated tweening into the game interface. AnneMarie was responsible for creating different levels, working with meshes, creating the aesthetic for the project, and directing gameplay. Karen contributed to the platform animation mechanisms, handling user input, and designing the user interface. We feel very satisfied with the cooperation, contributions, and communication among the members of this group.

Acknowledgements

Special thanks to TA Julian Knodt for directing us to this Youtube lecture (https://www.youtube.com/watch?v=Fy0aCDmgnxg) on how to “juice” a game. We ended up using several suggestions from the video, such as making the platform stretch when it moved.

We also wanted to thank the friends and family members that donated their time free of charge to try out our game. They provided us with valuable user feedback.

References

Assets

- Sound Libraries:
    - Free Music Archive
    - Mix Kit
- Gifs were created by Usagyuun

Meshes

- Heart Mesh: https://threejs.org/docs/#api/en/geometries/ShapeGeometry

Screens

- We took inspiration for our screen code from the Driver’s Ed project: https://github.com/karenying/drivers-ed

Libraries

- Three.js
- Tween.js