Rhythm Game

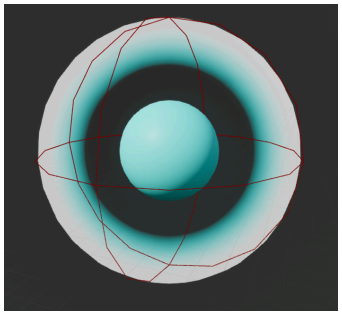
Project Overview

The game is a music rhythm game that has mixed combat elements. Final goal is to have a cinematic musical fight that is controlled by the player. The development process was planned to be a project for learning unreal engine and its blueprint.

Currently the project serves as a proof of concept with the following most crucial elements of the game mechanics developed: music sync, motion wrapping, spawning notes, mouse click get location, camera tracking and rail system. And with additional features that makes the game more complete and closer to the final product such as Fracture explosion and translucent indicator.

The following documentation is separated by three different key elements of the game, Ball(music node), player, and Ball Spawner. Each is explained with the features it has.

Ball(music node)



Hit Feedback:

The fracture, and explosion effect is activated once the player hits the ball to create a satisfying feedback.

It deletes the original ball unfractured model at the moment the player hits the ball at the same and same location and replaces it with a fractured ball. Give it an impact based on the player's previous distance before dashing and angle and launch the fracture to create impact feedback. A fire effect and explosion sound is also added to enhance this hit feedback.

Technical Details:

Fracture Setup: Created using Chaos Destruction in Unreal (available in newer engine versions).

Blueprint Steps:

1. Detect Hit: On mouse click that overlap event, call a custom event in the Ball Blueprint.

2. Spawn Fractured Mesh: Replace the static mesh with a Geometry Collection mesh at the same transform.
3. Apply Impulse: Use the Add Impulse node to launch fractured pieces.
4. Effects: Trigger a Particle System for fire/explosion.
5. Sound: Play a Sound Cue (explosion) at the impact location.

Hit timing indicator:

The outer layer ‘s color shifts from blue to yellow to indicate the perfect time to hit the ball on the beat. And suggest a clickable area without obstructing readability node and view.

This is done with a shield effect around the ball that spawns with the ball. An additional layer of yellow shield slowly expands from the center of the ball and will overlap the outer blue layer to suggest perfect time to hit the ball.

Technical Details

Shield Effect: Niagara system that scales over time.

Timeline for Animation: Use a Blueprint Timeline to smoothly transition color and size from blue (start) to yellow (perfect moment).

Ball Movement

The ball is attached to an invisible target in the center of the platform and moves and same constant speed towards it.

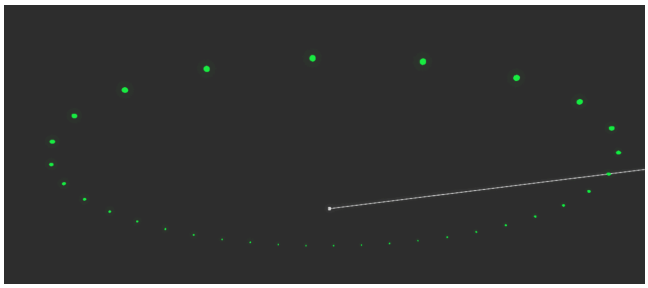
Technical Details

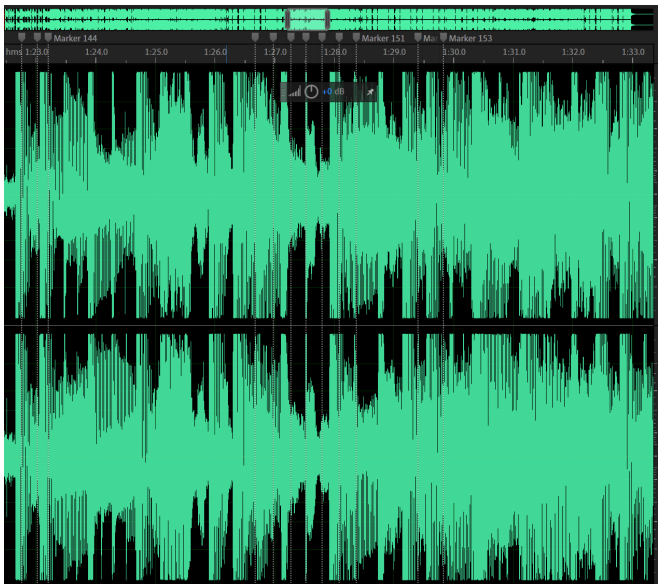
- Attach to Actor: The ball is attached to a hidden dummy object..
- Constant Speed: Set velocity in the Event Tick function.
- Collision Handling: Set up Collision Channels so that the ball detects player hits.

Spawner

Balls are spawned at the green dots around the platform at fixed time at random dots. This process of setting spawn time is simplified so I can repeatedly set spawn time for different music very easily.

Following a tutorial, I have made the process of setting the spawn time of the balls very easy. I can mark the music in adobe audition while playing the music and the spawner will read the maker accordingly and spawn the ball in the same timing as the marks in the music.





Technical Details

- Spawn Logic:
    1. Data Reading: Each marker in Adobe Audition export audio file.
    2. Blueprint Read: Parse those timestamps in a Blueprint.
    3. Spawn at Mark: At the specified time, call SpawnActor at a random or specific green dot's location (stored as an array of transforms).
- Ease of Editing: By centralizing all spawn times in a single data file or table, different music tracks can be quickly tested or updated without rewriting spawn logic.

Player

The player will be doing free flow combat onbeat to create a musical fight.

The mouse with get the location of each ball, and the player's character will perform motion raping towards the targeted location. Because this is a rhythm, to make the game play satisfying, its very important to have responsive feedback and unnoticeable delay between when the ball is clicked and when the character finishes dash and combat moves to hit the ball. While showing the moment and combat move during this time. A couple of tricks are used to achieve this goal. First, a sound of explosion is played the moment the player clicks on the ball. This creates instant feedback. The combat animation is edited so that it skips the translation from default pose to raise arm and leg and straight to moment before the punch happens and keeps the animation after the punch happens where the swing after the punch suggests that combat has happened. The fracture impact mentioned earlier also helps suggest the impact and the punch.

Technical Details

1. Click Detection:
    - Use Player Controller to detect a Left Mouse Button event.
    - Line Trace to find the clicked ball's location.
2. Motion Wrapping / Dash:
    - Character Blueprint handles a custom dash function.
    - Set Actor Location and Launch Character to quickly move the player to the ball.
    - Adjust the dash speed so that the arrival time aligns with the beat.
3. Instant Feedback:
    - Sound: Immediately play an explosion sound when the click is registered to mask any small delays in dash/animation.
    - Animation Montage: Use a shortened and tweaked montage that skips transitional frames (e.g., raise arm/leg) and goes straight into the punch.
4. Impact Confirmation:
    - The Ball triggers its fracture event upon the same signal (click + overlap detection).
    - Both the dash animation and ball fracture occur almost simultaneously to reinforce the sense of impact.

Camera

The camera moment creates a cinematic feeling for the game play while adding difficulty and variation for play style.

Technical Details

- Camera Rail:
    - Used the Cine Camera Actor on a Spline.
- Dynamic Tracking:
    - Attach to Look At pre-defined cinematic viewpoints.