Name of Game

Bionic Kickball (we call it Pixelball though...)

Team Members

Justin Mancusi (jem2dc)
Jeremy Weng (jhw2np)

Github Link

https://github.com/UVA-CS4730-F17/game-project-bionic-kickball

Game Pitch

- Pixelball is a 2D competitive arena action game where 2 players fight to the death. The only way that they can be killed is getting hit in the head with a ball that has been kicked by another player. It is a fast-paced game with a low skill floor and a high skill ceiling.

How to Play

- This game requires two players and is controlled using Xbox gamepads and runs on Windows.
- The main menu requires use of the mouse
- Controls
    - Left joystick moves the player
    - A button: jumps
    - X button: slides if on the ground, dives if in the air
    - Select Button: restart stage
    - Start Button: Go back to title screen
- Setup
    - The only setup needed is to specify the number of rounds you'd like to play. This is done in the main menu after clicking play.
- Objective
    - The objective of this game is to outscore your opponent in the number of rounds that have been specified. A score is made if one player kicks the ball into the head of the other player. A player cannot get hurt if the ball trail is their color and hits their head. If the ball trail is neutral or the opposite color at hits your player this will count as a hit and the other player will get a point. Kicking is done by running, sliding, or diving into the ball.

Amount of Content

There are 9 arenas that are randomly selected from each round.

Playtesting Report

When we got our game playtested out of class, our friends had quite a bit of fun, but the playtesting revealed a number of bugs that we hadn't noticed. We had a stopgap in place to reload the main menu in case there was a bug that rendered the game unplayable, but doing so would fail to reset the score counter, so to reset the score we would have had to restart the entire game.

Two other big bugs were noticed during playtesting. One of these was that the player could slide out of the map entirely and drop forever into the void. This was easily fixed by just tightening up the

colliders. Another one was that the divekick animation used the wrong triggers and colliders, as we had forgotten to change the colliders with the animation; they still used the standing animation colliders.

Next, one of our level designs was flawed, specifically the “volleyball” map, which was a map that was divided by a wall that was just higher than what the player could jump over, meaning both players were separated and could not cross over. This became an issue, as only one player had access to the ball at any time, meaning if the player with the ball is struggling to get the ball back over, the other player would only be able to stand there until play continues, meaning the pace of the game was slowed substantially and the game could become boring. We ended up just deleting the map.

One of the most commonly brought up issues was the pace of the game. Many players found that the game was a bit slow, and noted that it’d likely be more fun if the game was faster. Because we thought the kicking was already faster, our fix ended up as just increasing gravity on the ball by a noticeable, but not huge, amount. We chose this fix because the game itself wasn’t really that slow, but the ball would often take awhile to fall back down to the ground and therefore made the game feel much slower. We also increased the run speed to help with this.

Players also noted that because there was no round-selecting system and no game over screen, the game didn’t have a very satisfying end for the more competitive playtesters.

Overall, playtesting was very helpful as it revealed some game-breaking bugs and some balancing issues. It pushed us to create a more balanced, polished version of the game, and we felt much more satisfied with the game after resolving all the playtesting issues.

Lessons Learned

The biggest thing we learned is just how difficult game development is. Even doing something like relatively simple like making a fading transition required a couple dozen lines of code, and could often break other parts of the game or create more bugs. Also, unlike other programming projects we’ve worked on, game development requires a lot of artistic and design skill which we really did not have going into the project. On top of realizing how hard a job game designers have, we’ve also gained a newfound appreciation for graphics, because without any graphic designers on a game project, finding assets that fit the visual aesthetic a game is striving for can be very difficult.

Game design also forces upon us very difficult problems, especially when it comes to balancing the game. By solving one balancing issue, one or more other balancing issues will often pop up, forcing balancing to become a horrible game of whack-a-mole. As a result, balancing games required us to often make many small adjustments to fix a bigger problem. Players will often see the game much more differently than us as designers, and playtesters would suggest fixes that simply didn’t fit our vision of the game. In fact, the biggest balancing problem of all was balancing player enjoyment and our design goals and vision.

Lastly, we learned the importance of playtesting and prototypes. Through the use of prototypes and such we managed to find issues with our design goals and solve them early, rather than finishing most of the coding for an idea only to find out it’s impossible to do. Many of the goals we had originally were thrown out because it was difficult or impossible to do, or it was actually detrimental to the game itself, and we only found them out through prototypes and playtesting. Designing our game gave us a weird tunnel vision when reviewing our game, and the biggest improvements in our game almost always happened after playtesting and showcasing our game with other people.