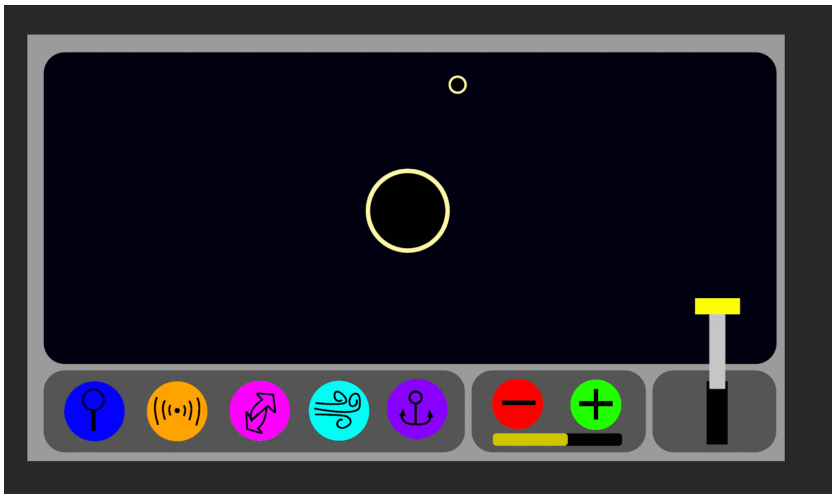
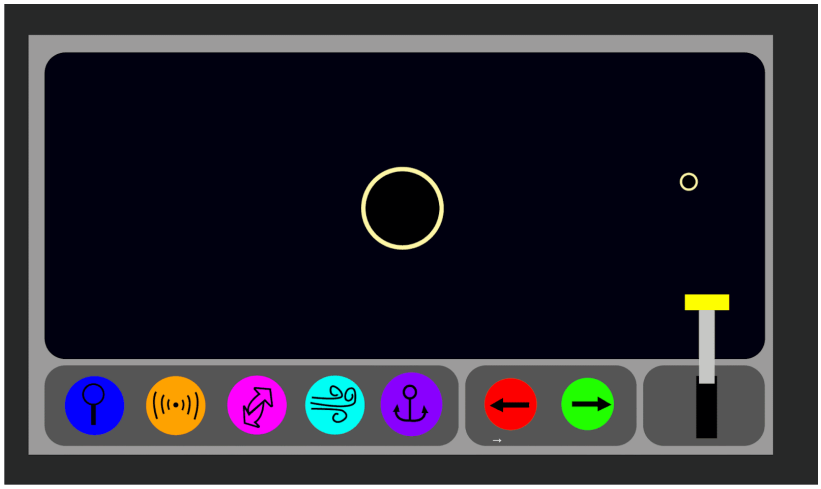
Day 2: Orbit implementation, Orbit speed/direction manipulation

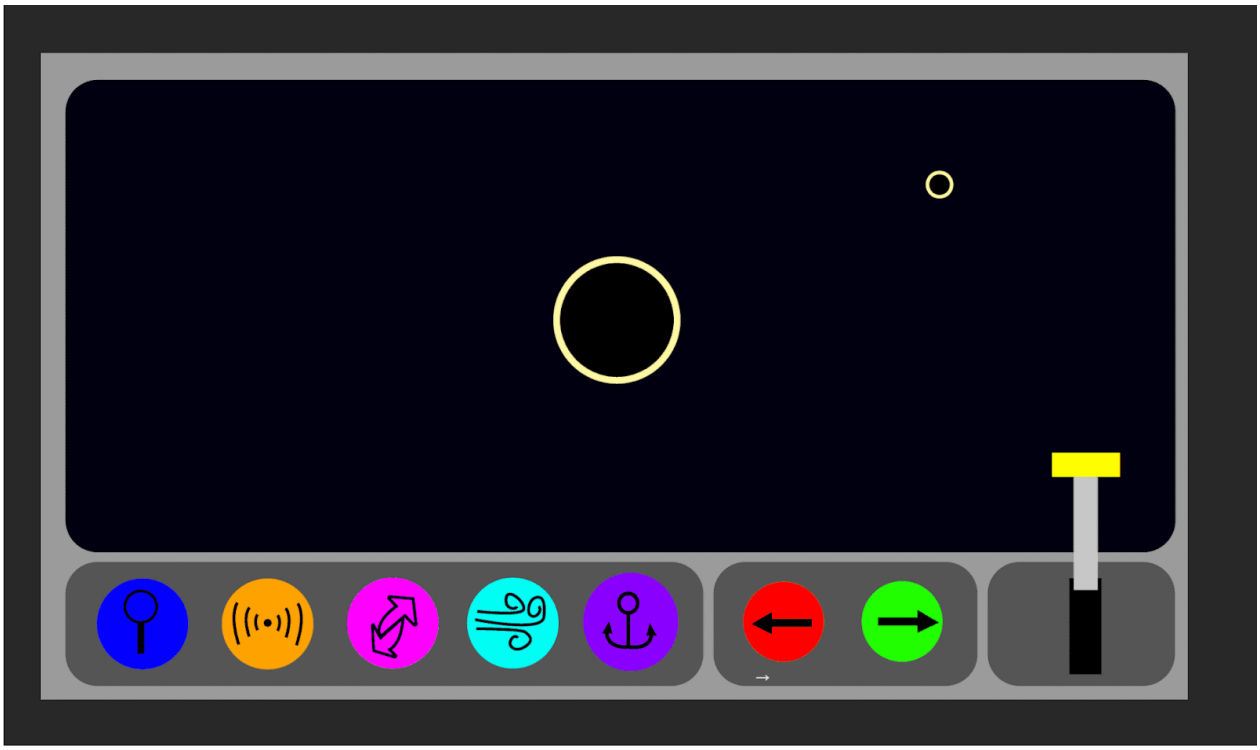
I created a script to handle the planet's orbit around the void. This script looks at a given center point (barycenter) and moves an object around this barycenter. The speed, horizontal and vertical axis are all public float values which can be manipulated from the Unity editor.



I have added to the script to allow the user to add and subtract velocity to the planet, even changing its direction. The speed of the planet is shown in the UI as well as the cooldown to alter the planet's speed again. The script increments the speed of the planet by 0.5f depending on the button clicked. I changed the buttons from a + and - to a right and left arrow to show the direction of the orbit. I think this can still be confusing to the user and I will look for better icons to give a more satisfying feel when changing the planet's velocity. The planet is not currently able to stop, this is by design. When the speed reaches 0 the planet will simply flip it's movement into the direction of the change. Speed is currently capped at 1.5f. I have added a text to the UI that will display arrows indicating the direction and intensity of the planet's movement. 1 arrow per 0.5f.



I have added to the script to implement the anchor. This anchor completely stops the planet's motion for a set amount of time (3 seconds). After this time the planet will continue with its original speed. When the button is pressed a cooldown timer is started (10 seconds). There are UI elements to show both anchor and cooldown timers. I am encountering an issue where changing speed is resuming the planet's motion. This bug has given me the idea to allow the player to change the planet's speed and direction while it is frozen. It will not take effect until after the anchor time is up but this will allow the player to freeze the planet and make precise movements. I am also going to move the ability to change the planet's direction to the pink button. The + and - will return and will only be able to increase and decrease speed without changing direction. There will be a min and max speed. (This was the original design but I got lost in the sauce)



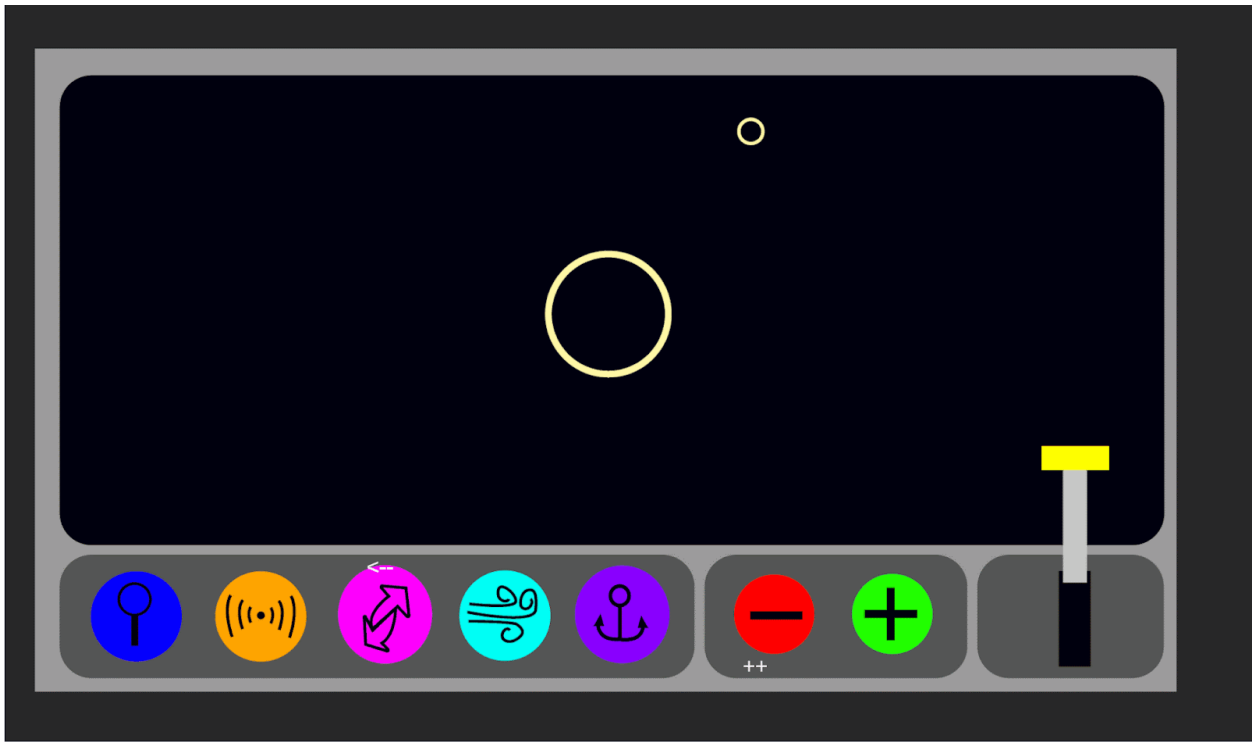
Merged multiple cooldown coroutines into one. Before each cooldown had its own coroutine which was crazy inefficient. By having one coroutine to handle all of our cooldowns we cut down on redundant processes. We track active cooldowns in a dictionary, updating them all at once instead of running separate timers. This approach is way more memory efficient and is easily scalable. If we ever need to add more cooldowns, we just drop them into the dictionary instead of writing a whole new coroutine.

Reworked how speed and direction are handled. Before, speed could be negative, things got messy. Pressing the plus button would slow the planet down if it was traveling in the negative direction. Now speed is always a positive value. We also track direction separately as either -1 (right) or 1 (left). This allows the two to play together better while giving the player clean and predictable movement.

Moved direction control to its own button. Now speed and direction are truly separate.

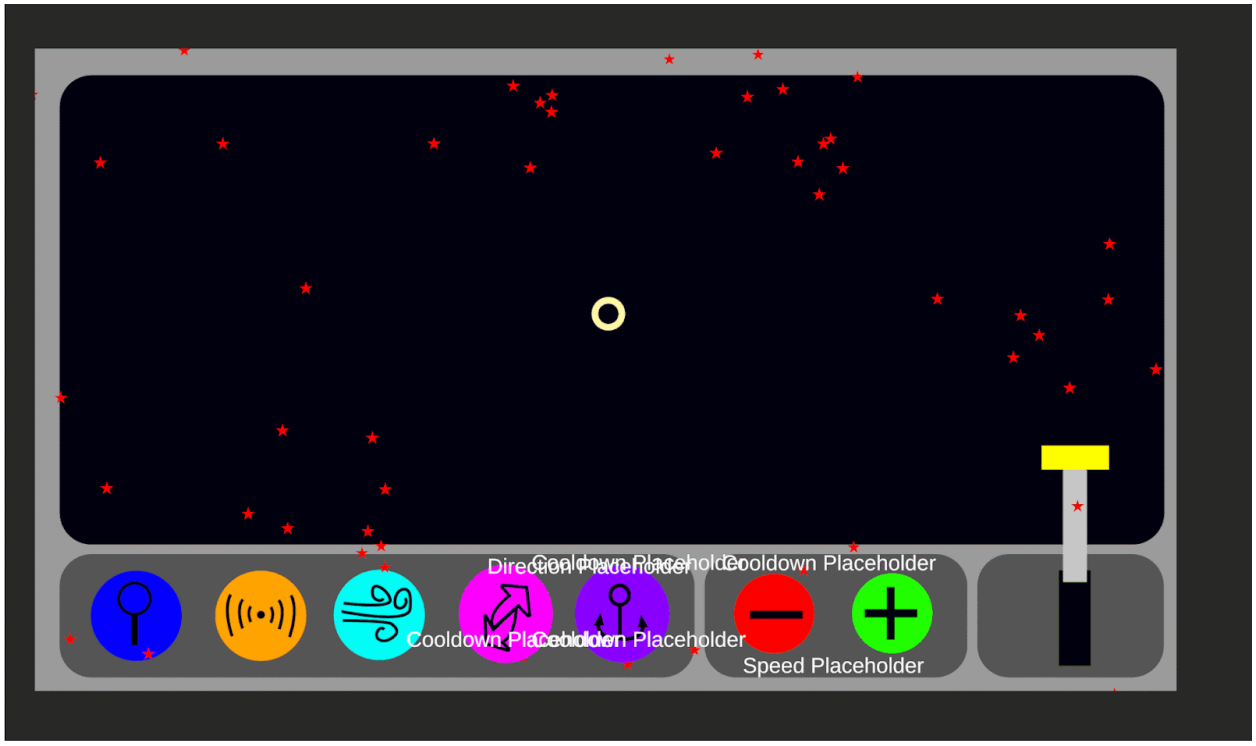
Changed the right and left buttons back to + and - arrows to properly convey their purpose.

Made a minor tweak to stop movement when anchored. The player can still change the planet's speed and direction but this will not reflect until the planet is no longer anchored.

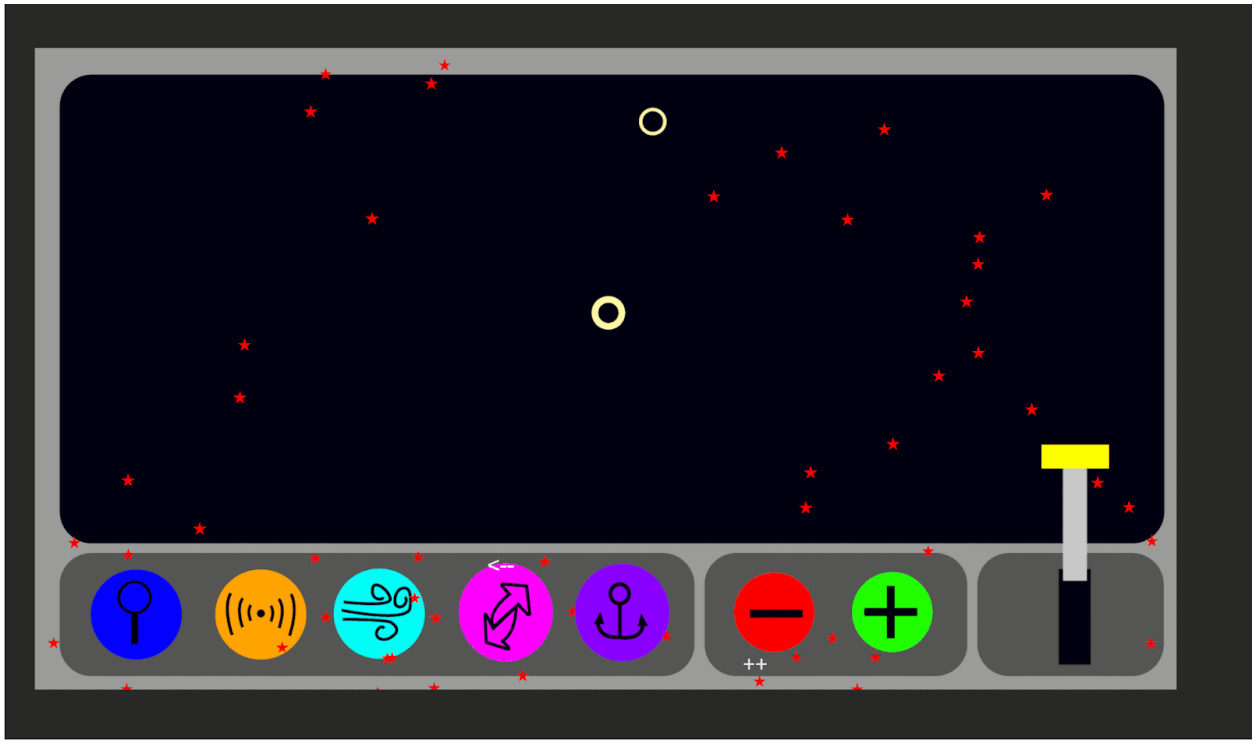


With these changes the orbit manipulation is done for now. We will now begin to add gravity physics as well as debris to test the gravity.

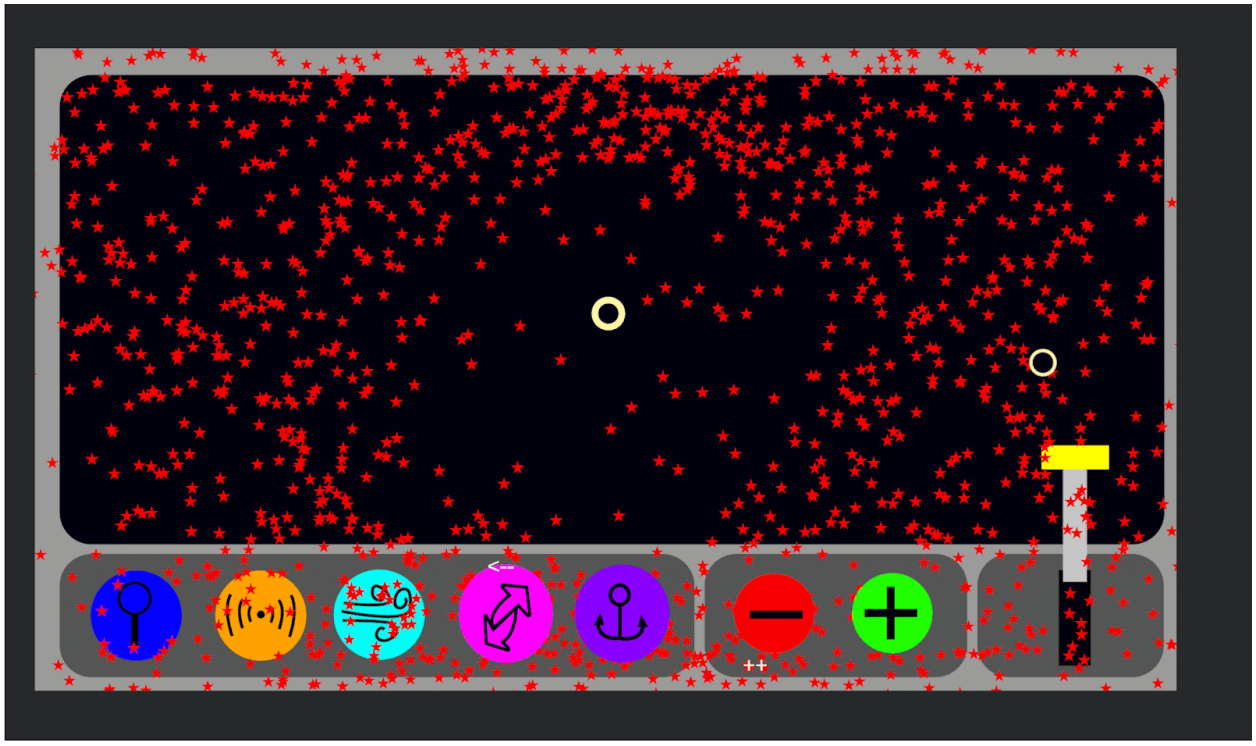
I have added a debris spawner that will take a prefab and spawn that prefab on the edges of the screen. The prefab will be initialized with a direction and speed. There is logic in our spawner script that makes the debris avoid the center of the screen, making sure the player gets no freebies. I have also reduced the size of the void. The spawner pulls from a debris pool, ensuring we do not exceed the max amount of debris. The spawner also controls the debris spawn intervals, the min and max speed as well as the safe radius (how close to the center the debris will naturally travel)



Added a gravity zone of influence onto the planet and void.This gravity zone has a radius and a strength. Created a gizmo to show the radius in the scene editor. Grabbing the debris is very tricky. Seeing the amount of debris and the way the gravity effects it makes me think this game could be cool with simulated fluids. Probably way out of scope for this but would still be fun! I will crank up the number of debris being spawned and see what happens.



MADNESS



I played around with a lot less debris and way slower speeds and I think this is where the fun in the game is found. You really have time to analyze what debris is on the screen, what its speed and approach are. You are able to adjust the planet's speed and direction to try and get these debris to orbit your planet. This is a fun task itself. Once you are able to get the debris to orbit, it is fun to manipulate the planet's speed to launch the debris. There will need to be lots of fine tuning but I can see the fun to be had!

Found this song. Liking the vibe and thinking I'll use it for inspo for the game's music (If we ever get there): https://www.youtube.com/watch?v=g8sX6wZHhD0

Updated Development Timeline

Day 1 (Monday) - Current Day

- ✓ Game design document
- ✓ UI mockup
- ✓ Basic orbital mechanics
- **Implement gravity physics system**
- **Create debris objects and behavior**

Day 2 (Tuesday)

- Complete physics interactions (debris-planet-void)
- Implement debris spawning system
- Basic void interaction detection
- Test core gameplay loop functionality

Day 3 (Wednesday)

- Implement tool/upgrade system architecture
- Add first set of tools (gravity manipulation, anchor, repulsor)
- Create basic "void needs" system
- Implement UI feedback for tools

Day 4 (Thursday)

- Complete all core tools and upgrades
- Add gameplay feedback systems
- Implement basic scoring/progression
- Initial game balancing

Day 5 (Friday)

- Finalize all mechanics and interactions
- Add visual polish (effects, animations)
- Implement simplified narrative elements (if time permits)
- Continue gameplay balancing

Day 6 (Saturday)

- Add sound effects and music
- Final gameplay adjustments
- Bug fixing and polishing
- Final testing

Sunday

- Package game
- Upload to Itch.io
- Submit to game jam

Notes for the future:

Add an ability to freeze all debris on the screen

Soundtrack inspo