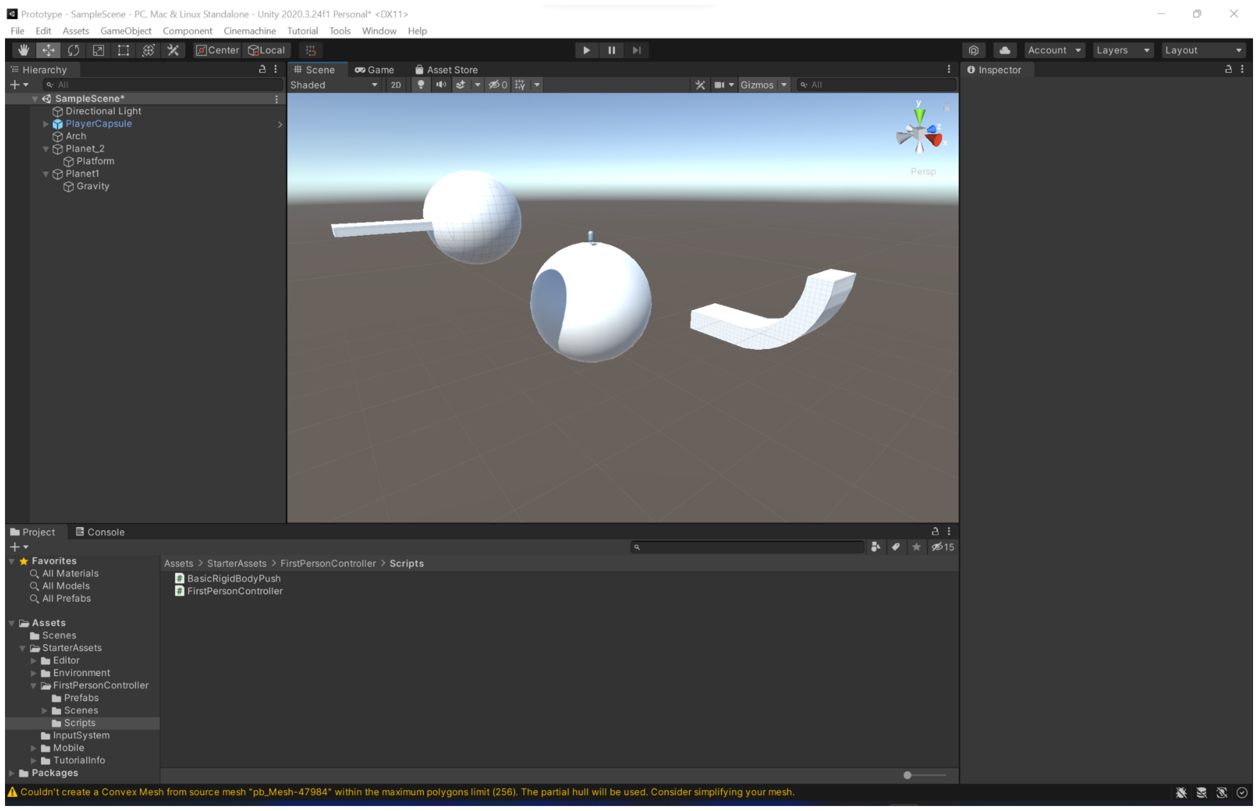
First Person Orbital Gravity

Super Mario Galaxy's introduction of multi-directional gravity opened avenues for a variety of creative and exhilarating gameplay mechanics and allowed for level/environment designers to build relatively unique and playful worlds. This system of gravity, specifically the orbital gravity that pulled Mario to the centers of small 'planets' would have been extremely disorienting for players if not for how tight the controls were, as well as how meticulously designed the 3rd person camera which always kept Mario at the center of the player's screen.

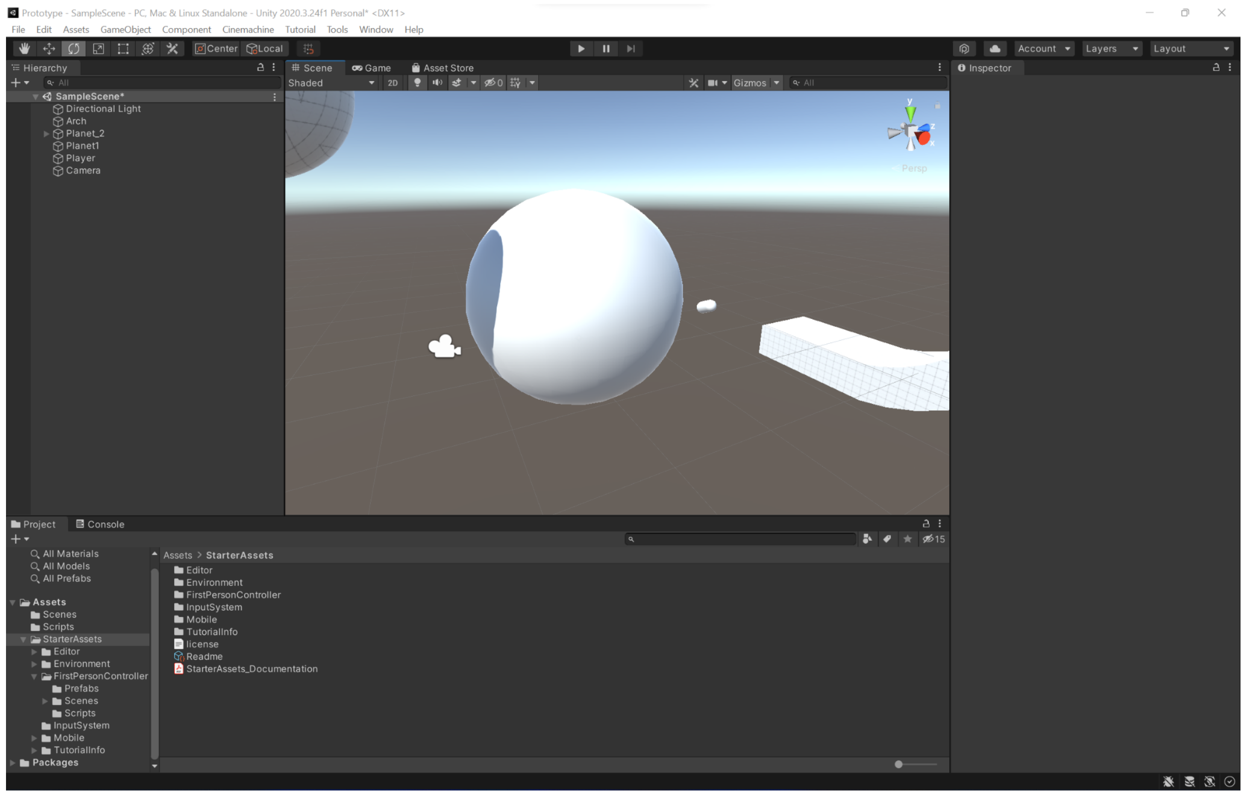
This left me wondering if this mechanic had a place in games with a first person camera. I'm envisioning the possibility of a third person controller having to navigate a topsy turvy environment only made possible by a multi directional/orbital gravity system. However, I'm worried that this system might feel too janky or disorientating for the player to control. This prototype aims to see if that can be worked around, or played into for added effect.

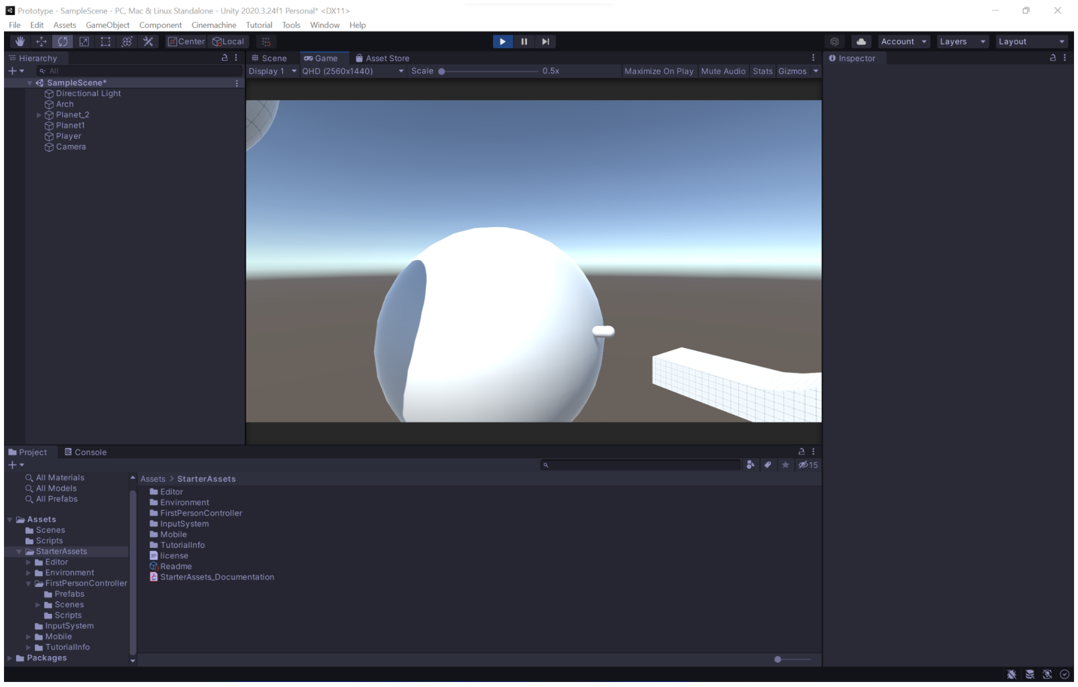


Couldn't create a Convex Mesh from source mesh "pb_Mesh-47984" within the maximum polygons limit (256). The partial hull will be used. Consider simplifying your mesh.

I set up this world in Unity using primitive gameobjects and probuilder for the arch. I ambitiously set myself the goal to have functioning gravity for the player to traverse each of these surfaces with unique gravity. I used the prefab from class for the first person controller. This ended up causing plenty of issues. Examples I found online would rely on the player object containing a basic rigidbody.

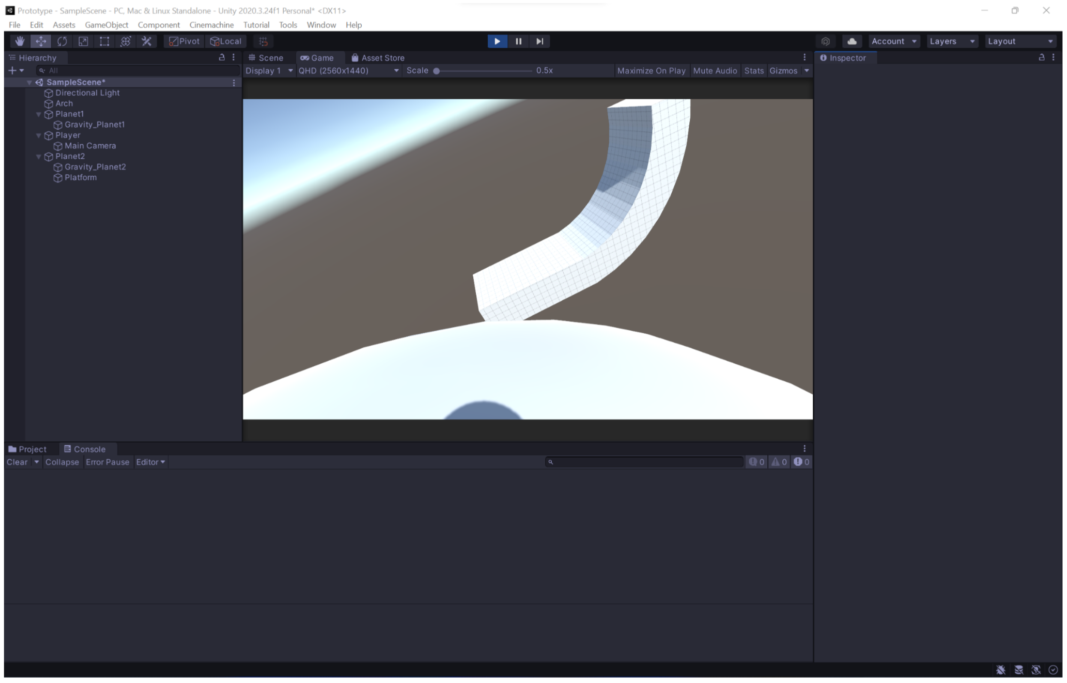
However, the prefab had its own built in mass and gravity, which would result in errors extremely difficult to debug.
I then decided to focus later on the player controller, and instead to try to get a capsule gameobject attracted to a planet, with the capsule standing upright on its base. The tutorial I found involved using one script for the objects/player that needs to be attracted (GravityBody), and a script for each planet/surface to pull in those objects/player. The gravity attractor script would store a variable for the direction in which the object should be facing by subtracting its transform position (center) from the body's transform position A separate variable stores the body's up position by using body.up. Using the Quaternion FromToRotation command, the body is then rotated from its default up position to the target direction stored in the first variable. Finally, a downward force is created using AddForce to simulate gravity opposite to the new up direction.





And we had success!

I then began to work on creating a new first person player controller which relied on using a rigidbody. I made a few errors along the way, but it wasn't long before I had a first person controller with jumping functionality as well I learnt about how a first person controller works by rotating just the game object with horizontal mouse movement, and rotating just the camera with vertical mouse movement.


Prototype - SampleScene - PC, Mac & Linux Standalone - Unity 2020.3.24f1 Personal* <DX11>
File Edit Assets GameObject Component Cinemachine Tutorial Tools Window Help
Pivot Local
Account Layers Layout
Hierarchy
SampleScene*
Directional Light
Arch
Planet1
Gravity_Planet1
Player
Main Camera
Planet2
Gravity_Planet2
Platform
Scene Game Asset Store
Display 1 QHD (2560x1440) Scale 0.5x
Maximize On Play Mute Audio Stats Gizmos
Inspector
Project Console
Clear Collapse Error Pause Editor
0 0 0

The GravityBody script would use FindGameObjectWithTag to access the Gravity Attractor script to create a downwards force on the gravity body. This was working fine until I gave the other planets the same ‘planet’ tags. This led to me creating a new array for the variable which was using FindGameObjectsWithTag, and then accessing the attractor script for a specified index within that array.

I was then able to set up colliders around each planet with ‘is trigger’ set to true. I then used OnTriggerEnter to change an int var which would determine which planet the gravitational force would be applied towards. This allowed the player to be able to jump between planets being pulled in by whichever planet they were closer to. However, since I was using Quarternion.ToFromRotation to angle the player according to the planet, it would change the angle instantly when jumping between planets. This is extremely disorientating, and I have not been able to figure out how to make this transition smooth.

Unfortunately that is as far as I was able to get in a week. I really enjoyed making my first unity prototype. Although a lot of the code was taken from the link below, I made sure to thoroughly understand what was being done at each line, mostly Debug.Log, before commenting it next to each line. I thought the movement came out really nicely in the end. Using artificial gravity did not make the jump work strangely, or the directional movement feel janky. If the transition between planets is made to feel smoother, this prototype can definitely be built upon further. Really excited to see what adding other objects with the gravity body script which the player can interact with would look like.

Questions for Playtesters

1. What type of game do you imagine this prototype developing into? Perhaps a trippy disorientating puzzle game like portal, or a dynamic, colorful action/platformer like Super Mario Galaxy?
2. Are there any other cool possibilities and mechanics you can envision this prototype containing down the line?
3. For a prototype, how could this experience be made more clearer to you?
4. Did the commenting of the code make understanding how this prototype works any easier? Any recommendations for future //ing?

Bibliography

https://www.youtube.com/watch?v=TicipSVT-T8