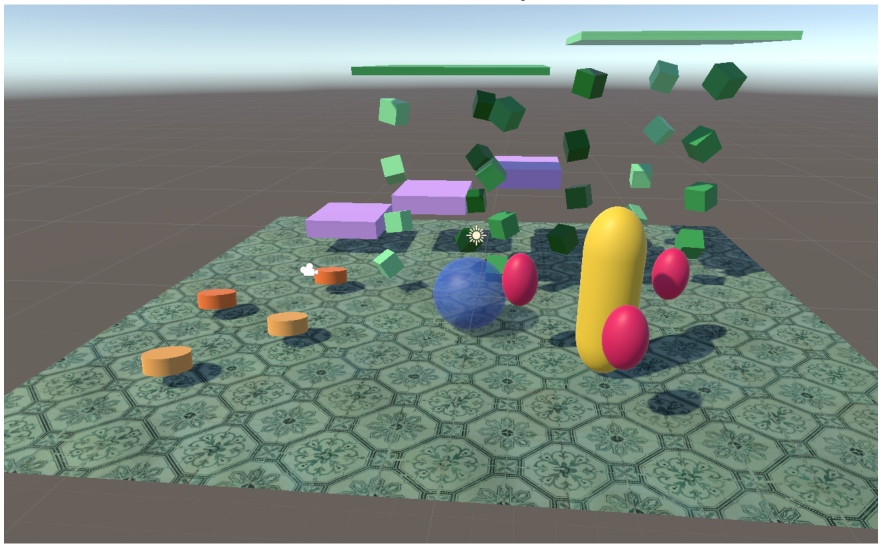
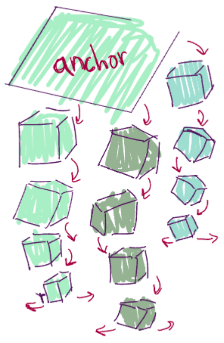
Interactive Installation to Explore Joint Behavior

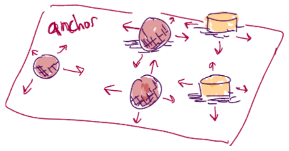


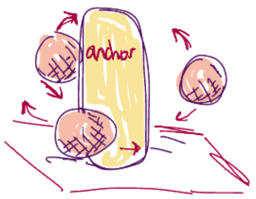
The result of this project is a virtual interactive installation. The goal of the installation is to create an open-ended environment to learn about joints in Unity through play. Joints are a type of connection between objects in Unity that follow special physics rules. They are often used in character construction and are challenging because of the intricacies they add to collisions. In this installation, the user can interact in 3rd person using the WASD keys and space bar. The installation is made up by a hanging cube apparatus, a capsule apparatus, and floor objects. These are made by fixed joints, hinge joints, and spring joints.

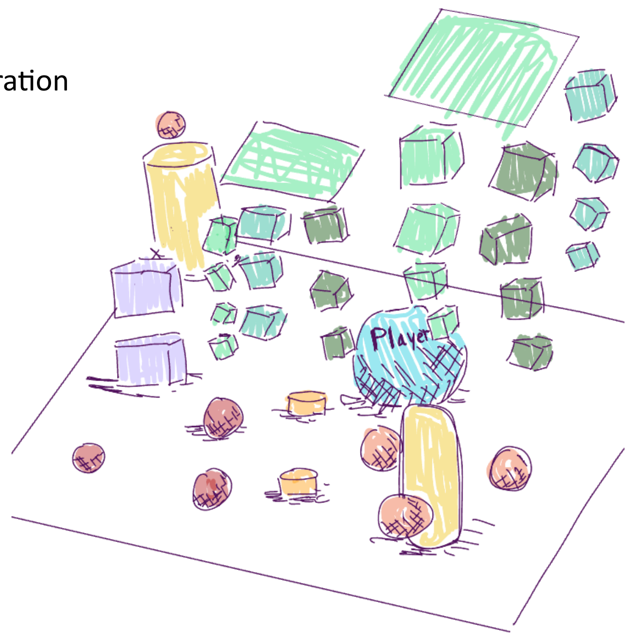
Design Process

I began this project by sketching out ideas for apparatuses that could inspire interesting exploration in the functionality of each joint type.

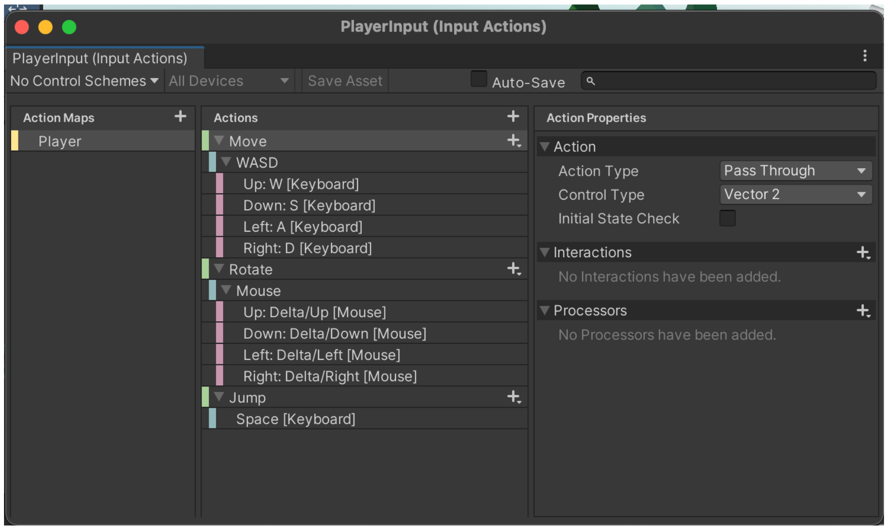








The player is represented by a 1.5m radius sphere and has an empty game object representing the camera position.



Player

- Move
  - WASD
    - Up: W [Keyboard]
    - Down: S [Keyboard]
    - Left: A [Keyboard]
    - Right: D [Keyboard]
- Rotate
  - Mouse
    - Up: Delta/Up [Mouse]
    - Down: Delta/Down [Mouse]
    - Left: Delta/Left [Mouse]
    - Right: Delta/Right [Mouse]
- Jump
  - Space [Keyboard]

Action

Action Type	Pass Through
Control Type	Vector 2
Initial State Check	

Interactions

No Interactions have been added.

Processors

No Processors have been added.

Figure 1: Input Action Scheme

A new input action scheme is made for the player. The move action returns a Vector2 based on the WASD keys. The rotate action returns a Vector2 based on the mouse. The jump action is connected to the space bar.

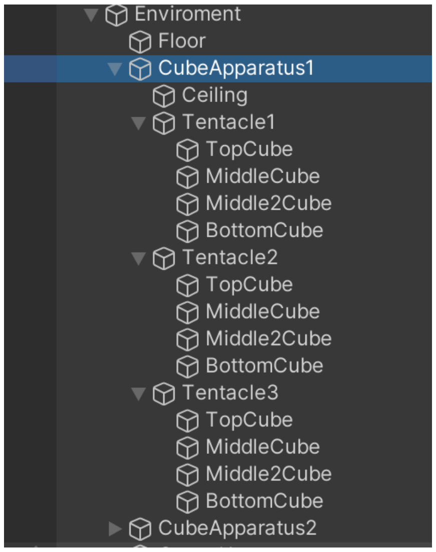
An empty game object for the input manager is connected to the input manager script. Similarly, the movement controller script and rotation controller script are connected to empty game objects. The movement controller and rotation controller are serialized with the player and the empty game object representing the camera position in the player.

A camera follow script is connected to the main camera and the camera position from the player is target field. A rigid body is added to the player and rotation is frozen in all directions. To allow the player to interact with objects in the scene they need their own rigid bodies. The mass of the player is set to 30 where the mass of the objects in the scene are set to 1 to ensure correct interactions.

The jump capabilities are added to the player with the Jump Controller script, which extends mono behavior and is connected to the empty game object jump controller. The initialize method subscribes to the JumpAction_Performed method. Initially, the player was just teleported up using MoveTowards(), but that results in jerky jumping. So, a jump counter was introduced to make the jumping more natural. When the space bar is pressed, the jump counter is started by setting it to one. Then, if the player is jumping, the FixedUpdate() method translate the player up by using transform.Translate and increases the counter. Once the max value of the counter is reached, the counter is reset to zero.

Unfortunately, since the CameraFollow() code from a previous project was made for a first person perspective, transitioning to 3rd person raised some issues. With the 10m offset, the 3rd person camera lost the player when rotating a large amount. To fix this the offset vector must change with respect to the player rather than be 10m in the negative z direction. The offset vector is calculated from the target transform.forward and a serialized offset scalar. In the

LateUpdate() method, the camera is moved to the composite of the players position and the offset vector. Based on research, I changed FixedUpdate() to LateUpdate() to avoid jumpy visuals.



The cube apparatus is organized into a hierarchy. Each tentacle is made up of 4 cubes. Every cube gets a rigid body. The ceiling tile has a rigid body as well and is frozen in position and rotation as it serves as the anchor for the tentacles. A spring joint is added to the ceiling tile and the top cube is connected to it. Then the top cube gets a spring joint and that's connected to the middle cube. This pattern is repeated all the way to the bottom joint.

Figure 2: Cube Apparatus Hierarchy

There was a lot of experimentation associated to establishing the anchors and the spring constants for each joint to get the tentacles moving correctly. Initially the tentacles would all get tangled together. To fix this the anchors of the springs needed to be set further apart. The springs would also collapse onto the floor. To fix this, the spring constants had to be increased so the springs have more resistance. What ended up working was setting the spring constants to 80, 75, 70, and 60 from top to bottom. The higher spring constant at the top allow the tentacles to not collapse to the floor, while to lower ones at the bottom makes the tentacles interesting to interact with.

The capsule apparatus is made up by a stationary capsule and 3 spheres evenly spaced around it. The capsule and the spheres each have a rigid body. The spheres' y movement is frozen, and the capsule position and rotation is frozen. Three hinge joints connect the capsule to each sphere. To achieve the correct behavior, the axis of the rotation is the y axis and the anchor is the center of the capsule. The three spheres freely rotate about the capsule and bump into each other and that causes more rotation.

To create more interesting behavior, a fixed joint connects one sphere to another. This forces the spheres to be separated by a fixed amount, but the third sphere can move freely between the first and second sphere. A fixed joint is the most basic joint and stiffly connects two objects. The capsule's mass is set to 50 so that the player cannot pass through the capsule since the position is frozen.

The floor objects are cylinders. It was difficult to figure out the correct anchor to get the right spring behavior. First the floor was used as an anchor, but this was problematic since then the floor had to have a rigid body. This caused the player to fall through the floor. I also tried making a second floor to anchor the objects, but this also failed as the spring joint functioned vertically instead of horizontally. Adding an empty game object that floats slightly off the floor as the anchor fixed the issues. A rigid body and spring joint are added to the anchor and the cylinder floor object was attached to the anchor. The spring constant was set to 6, a low value, so the floor objects have more movement range. Each floor object got its own empty game object anchor.

Finally, stairs were added so the user can interact with the tentacle cubes from a higher level. Then materials were added to color the scene. A tile texture from 9t5 PBR Textures Pack Freebies was added to the floor, as texture was needed to tell movement better. A transparent material was made for the player sphere. The color was set to blue, with an alpha value of about 0.5.

Results

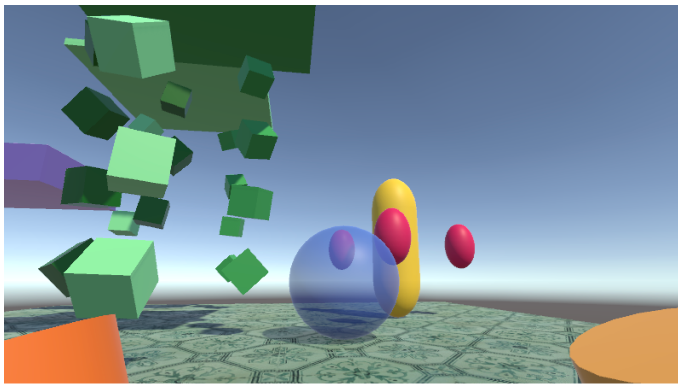


Figure 3: 3rd Person Player View

The interactive installation includes a hanging cube apparatus, a capsule apparatus, and floor objects. The user, represented by a sphere, can interact with the installation by moving around with the WASD keys. In addition, the user can jump by pressing the spacebar. The mouse can be used to rotate the player's view.

The hanging cube apparatus consists of tentacles hanging from the ceiling tiles. The tentacles are made up by cubes connected by spring joints. They oscillate and bounce as the user hits them. There are stairs the user can jump up to achieve different vertical heights to further interact with the apparatus.

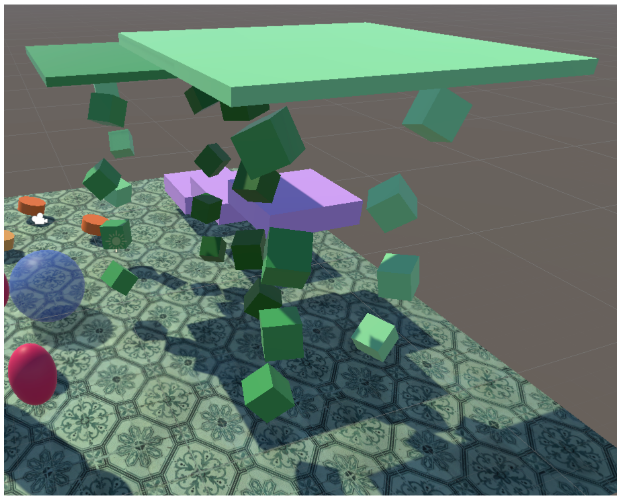


Figure 4: Cube Apparatus

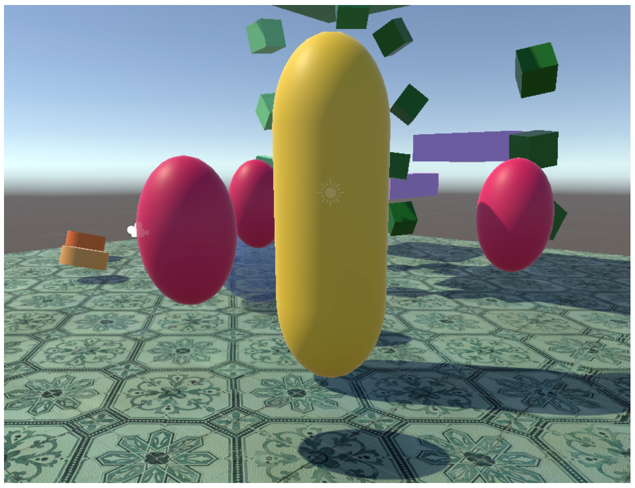


Figure 5: Capsule Apparatus

The capsule apparatus has a stationary capsule surrounded by 3 spheres. The spheres rotate around the capsule when the user bumps into them.

The floor objects are anchored on the floor and move around parallel to the floor as the user interacts with them.

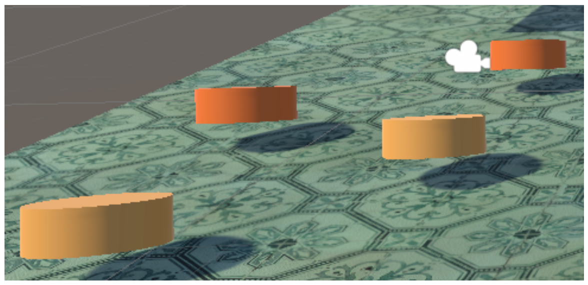


Figure 6: Floor Objects

References:

https://www.youtube.com/watch?v=MElbAwhMvTc

https://code.tutsplus.com/tutorials/unity3d-third-person-cameras--mobile-11230

https://docs.unity3d.com/ScriptReference/MonoBehaviour.LateUpdate.html