Lewis Bailey

Journal

Getting the gravity itself to work, was easy to do as it was just using forces and had to get the position of certain colliders to calculate rotations, but when it came to creating a character controller that interacted with the custom gravity was a little harder, as I couldn't use a standard controller, as I wasn't just working with X and Y to move, but X, Y and Z, and having to calculate the forward direction and how much to move by.

I managed to figure it out by using the function, AddRelativeForce which adds a force in the local transform rather than in the world. But i also needed to get the forward vector from the player, so I know which way they are facing when they are moving.

```
Vector3 direction = Vector3.forward * _direction.y;
_rigidbody.AddRelitiveForce(direction * _rigidbody.mass * _speed, ForceMode.Acceleration);
```

But when moving this will just keep adding force and you will just fly off forever, i also needed to clamp the velocity to a specific number so that the player will only move so fast, and not fly off into the distance.

```
float maxForce = (Mathf.Max(0.0f, _maxSpeed - _rigidbody.velocity.magniude)
* _rigidbody.mass) / Time.fixedDeltaTime;
_rigidbody.AddRelativeFoce(Mathf.Min(maxForce, _speed) * direction);
```

I wanted to make it so you had a preview of the inventory before playing the game inside of the editor, so you know the size of it without having to test the game. The hard bit was to draw images to the editor in a grid pattern. Which me and Paul manage to do after a couple hours as we had to create a rect and then set the starting position of the grid to the rect, so it was in the right place. Once we had done that, we could then draw the grid inside of this.

Unity was throwing error that it couldn't find a property in a custom editor, but it was solved my serialising the field.