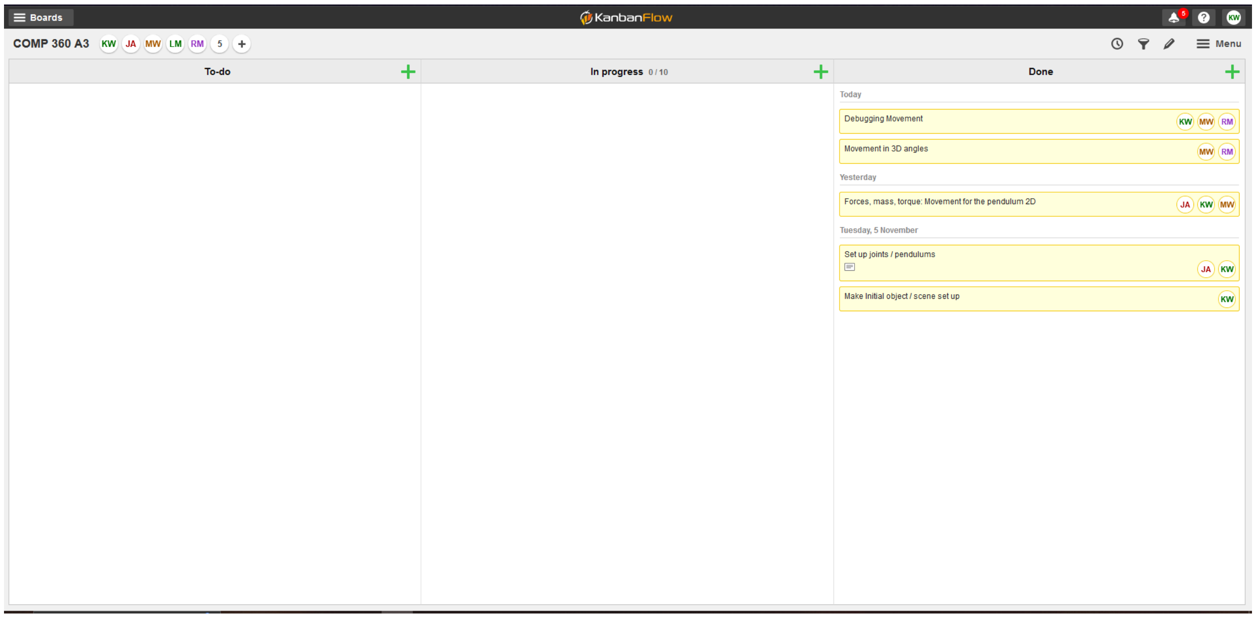
Members:

Jannine Gemmell, Liam Maarhuis, Ryan Morrison, Kaiya Wangler, Markus Webster

Github Link:

https://github.com/MidnightWolfe/COMP360-A3/settings

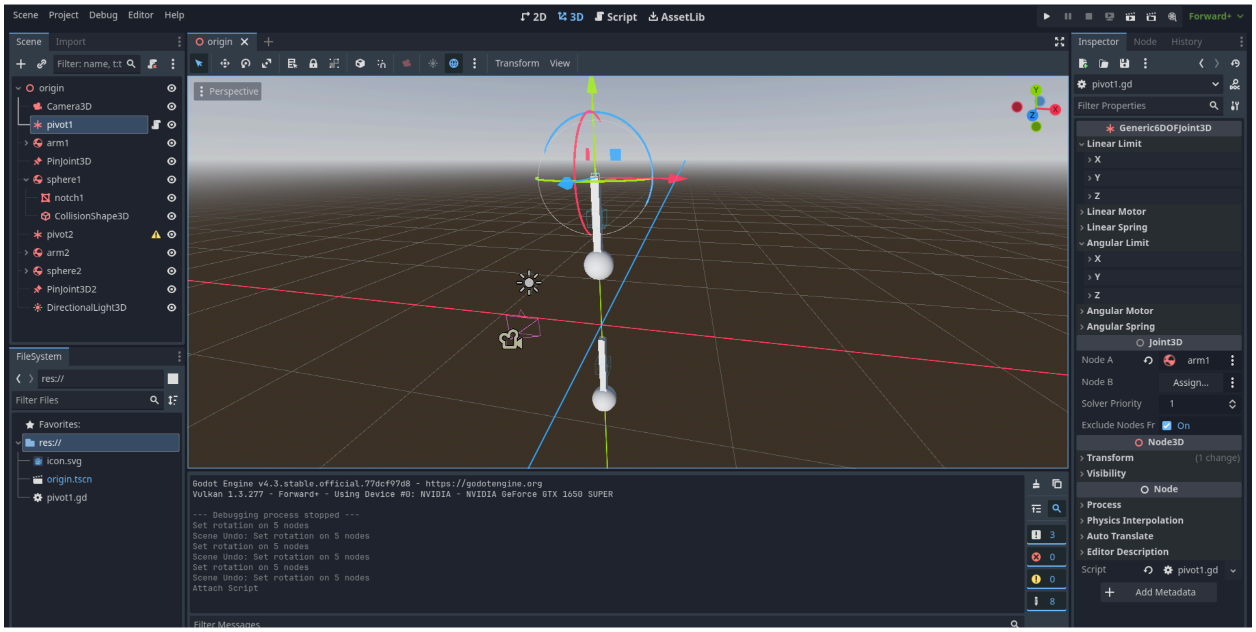
Kanban Board:



To-do	In progress 0 / 10	Done
		Today
		Debugging Movement — KW MW RM
		Movement in 3D angles — MW RM
		Yesterday
		Forces, mass, torque: Movement for the pendulum 2D — JA KW MW
		Tuesday, 5 November
		Set up joints / pendulums — JA KW
		Make Initial object / scene set up — KW

Debug:

Initial setup for the pendulums, using two arms and two spheres. The spheres caused collision issues.

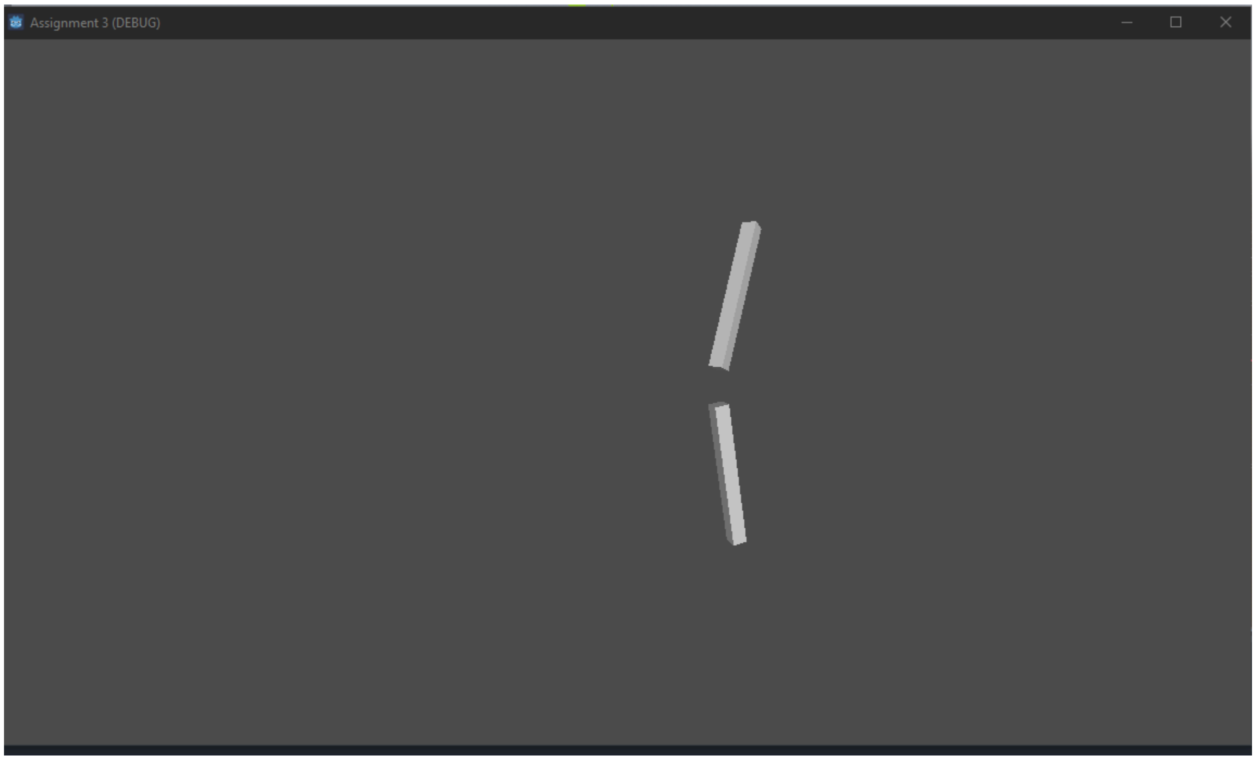


Node A arm1
Node B Assign...
Solver Priority 1
Exclude Nodes Fr On
Transform (1 change)
Script pivot1.gd

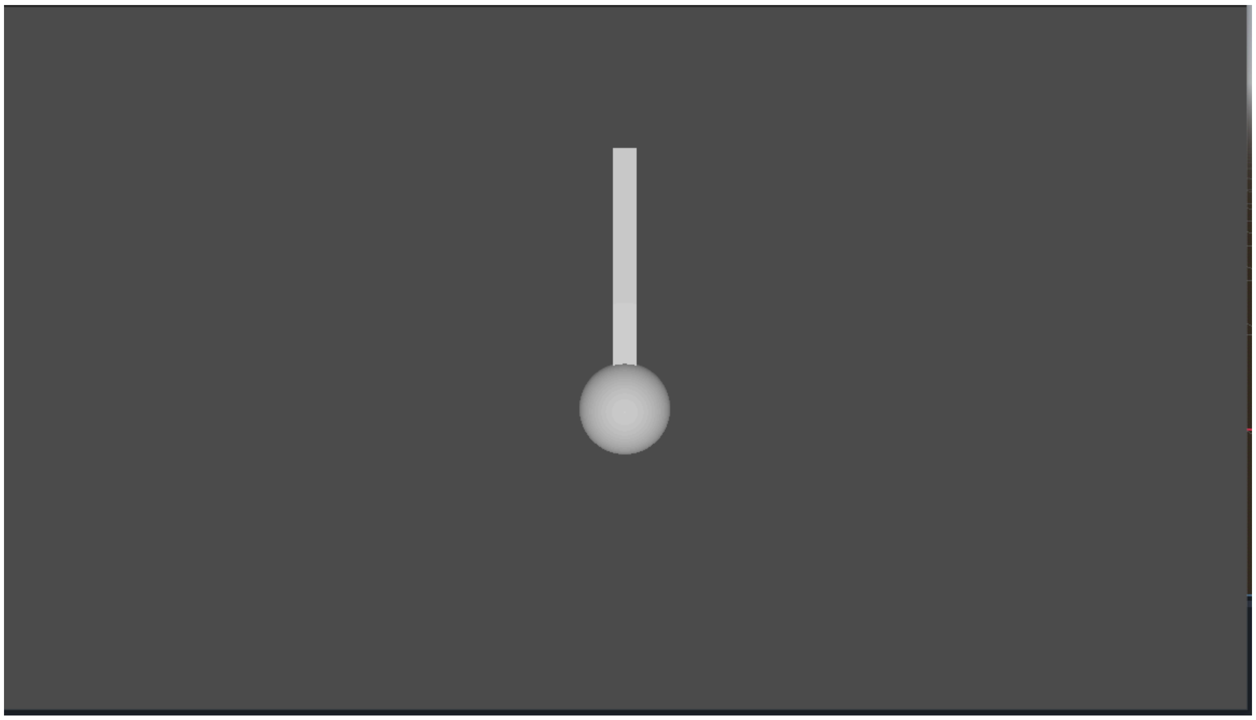
```
Godot Engine v4.3.stable.official.77dcf97d8 - https://godotengine.org
Vulkan 1.3.277 - Forward+ - Using Device #0: NVIDIA - NVIDIA GeForce GTX 1650 SUPER

--- Debugging process stopped ---
Set rotation on 5 nodes
Scene Undo: Set rotation on 5 nodes
Set rotation on 5 nodes
Scene Undo: Set rotation on 5 nodes
Set rotation on 5 nodes
Scene Undo: Set rotation on 5 nodes
Attach Script
```

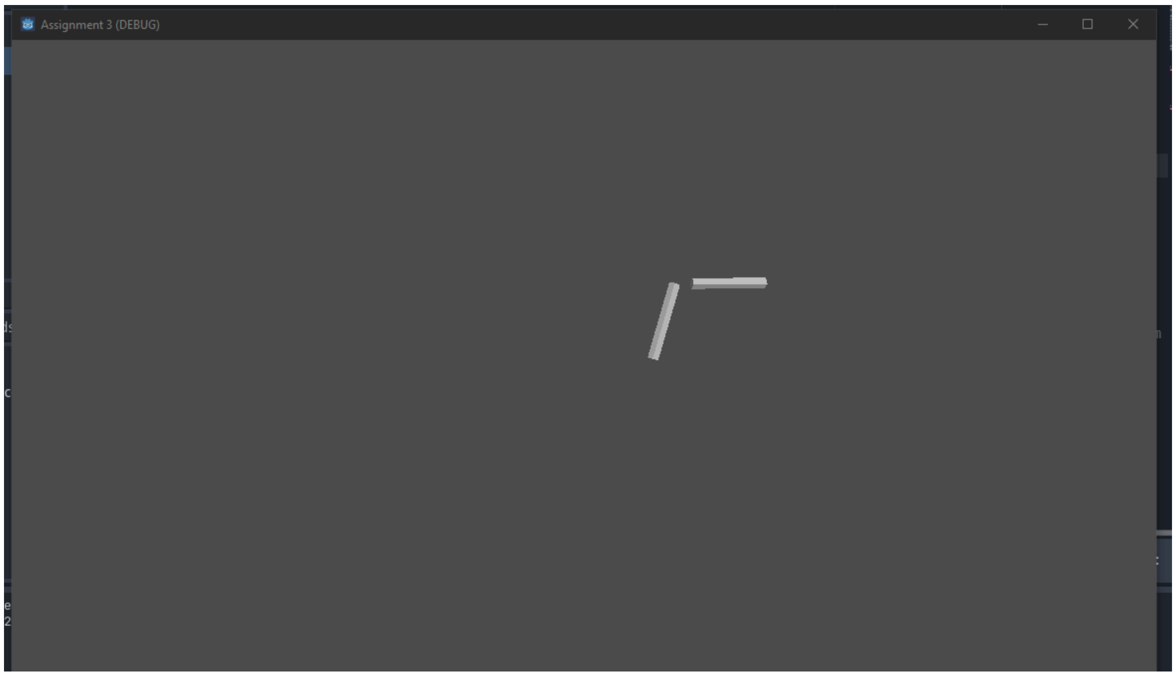
The initial testing of joints. 6DOF joints worked oddly, so pivot joints were used instead.

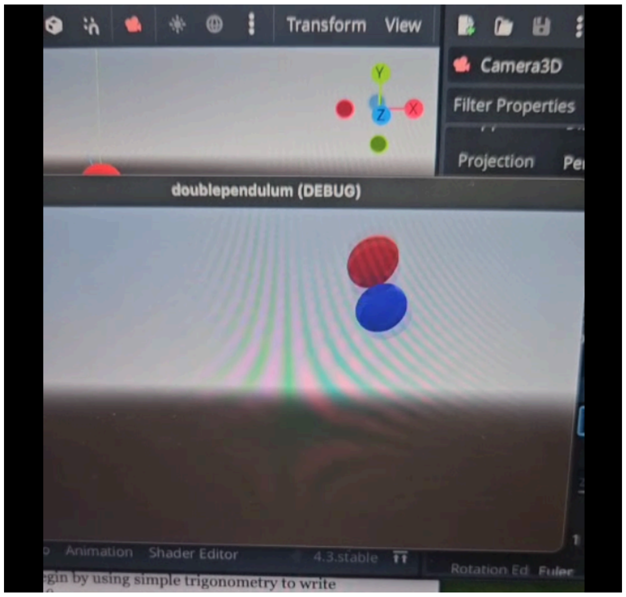


The initial pendulum design with the balls in place. This caused issues as godot does not let you connect two objects as one; so arms is what we were left with.



The first pendulum arm (the one vertical in the picture below) was facing up with the force rather than down. Issue fixed by making the force negative so it followed the same axis as gravity.





First time running the code with physics calculations. The calculations are inspired from https://github.com/Ryan-Mirch/Line-and-Sphere-Drawing. The results are not a pendulum, but at least the first mass is moving in a circle, and the second mass is moving around the first!

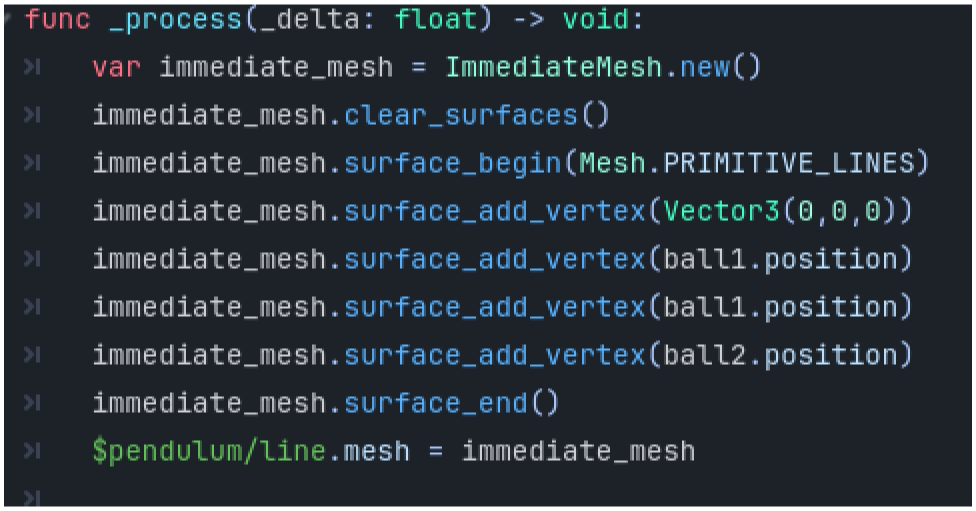
```
var gravity = -9.8
```

```
var gravity = 9.8
```

The error is found. Gravity has the wrong sign and is supposed to be positive. Now the masses are moving as they should, but they are too close together and there is no line between them.

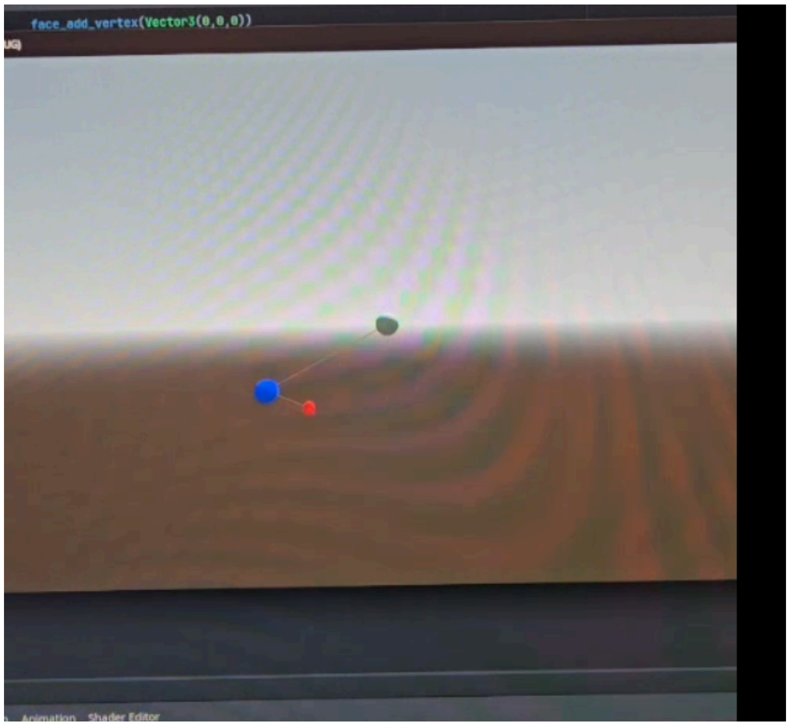
```
# Scales the size of the masses relative to their mass
func _ready() -> void:
	ball1.scale = ball1.scale * mass1
	ball2.scale = ball2.scale * mass2
```

The size of the masses are now scaled to their mass and now they look like a good size.



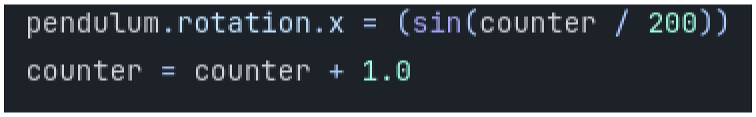
```
func _process(_delta: float) -> void:
	var immediate_mesh = ImmediateMesh.new()
	immediate_mesh.clear_surfaces()
	immediate_mesh.surface_begin(Mesh.PRIMITIVE_LINES)
	immediate_mesh.surface_add_vertex(Vector3(0,0,0))
	immediate_mesh.surface_add_vertex(ball1.position)
	immediate_mesh.surface_add_vertex(ball1.position)
	immediate_mesh.surface_add_vertex(ball2.position)
	immediate_mesh.surface_end()
	$pendulum/line.mesh = immediate_mesh
```

Now solve the problem of no line between the masses. In the process function a mesh is created that is a line from the origin/pivot point to the first mass and then a line from the first mass to the second.



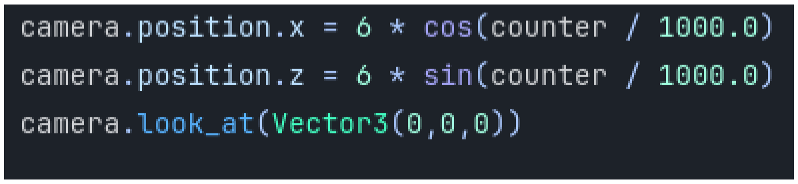
```
face_add_vertex(Vector3(0,0,0))
```

With these changes we now have a very accurate pendulum that moves in the xy-plane.



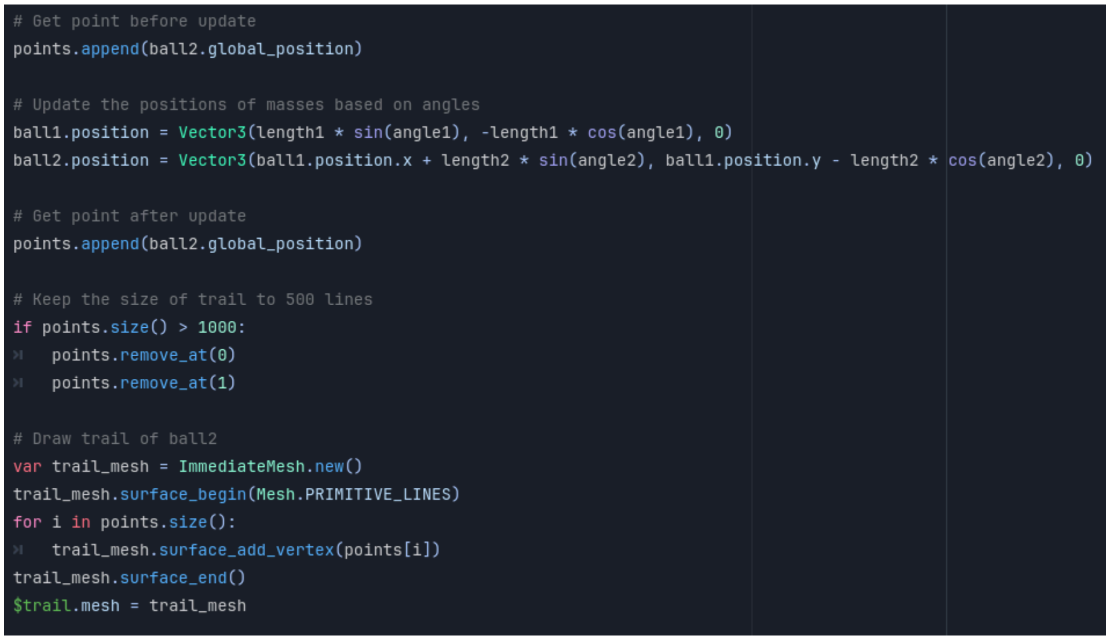
```
pendulum.rotation.x = (sin(counter / 200))
counter = counter + 1.0
```

With this change the system now fully moves in 3 dimensions.



```
camera.position.x = 6 * cos(counter / 1000.0)
camera.position.z = 6 * sin(counter / 1000.0)
camera.look_at(Vector3(0,0,0))
```

Seeing the movement of the pendulum could be made easier. Now the camera orbits around to get a better view of the system.



```
# Get point before update
points.append(ball2.global_position)

# Update the positions of masses based on angles
ball1.position = Vector3(length1 * sin(angle1), -length1 * cos(angle1), 0)
ball2.position = Vector3(ball1.position.x + length2 * sin(angle2), ball1.position.y - length2 * cos(angle2), 0)

# Get point after update
points.append(ball2.global_position)

# Keep the size of trail to 500 lines
if points.size() > 1000:
	points.remove_at(0)
	points.remove_at(1)

# Draw trail of ball2
var trail_mesh = ImmediateMesh.new()
trail_mesh.surface_begin(Mesh.PRIMITIVE_LINES)
for i in points.size():
	trail_mesh.surface_add_vertex(points[i])
trail_mesh.surface_end()
$trail.mesh = trail_mesh
```

To even better see the movement of the pendulum, a trail is now added to the second mass. It is a mesh consisting of line segments where each line has a start point of the second mass position and an end point of the second mass position after the position has been updated. The number of points is limited to 1000 so that there are no more than 500 lines present to prevent the mesh from growing too large.