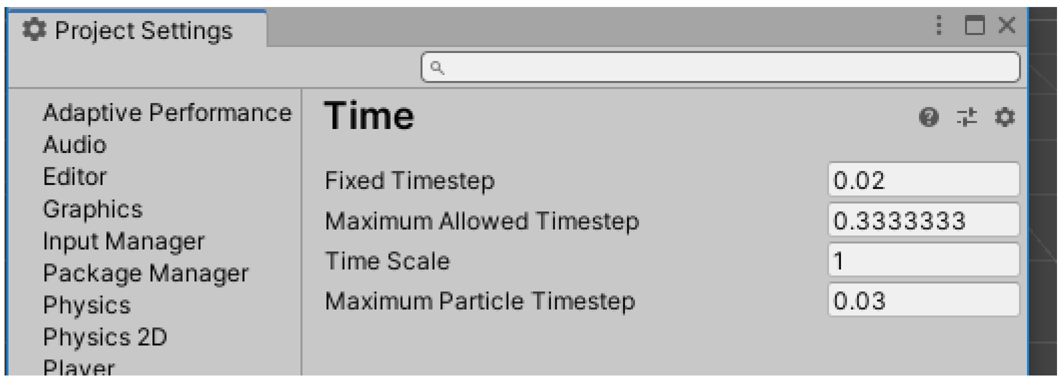
COMP2007 - Game Development

Week 5 - Code session

Physics settings

The **Fixed Timestep** is the speed the **FixedUpdate** method runs in seconds.
You can set the **Fixed Timestep** in the **Time** section of your project settings



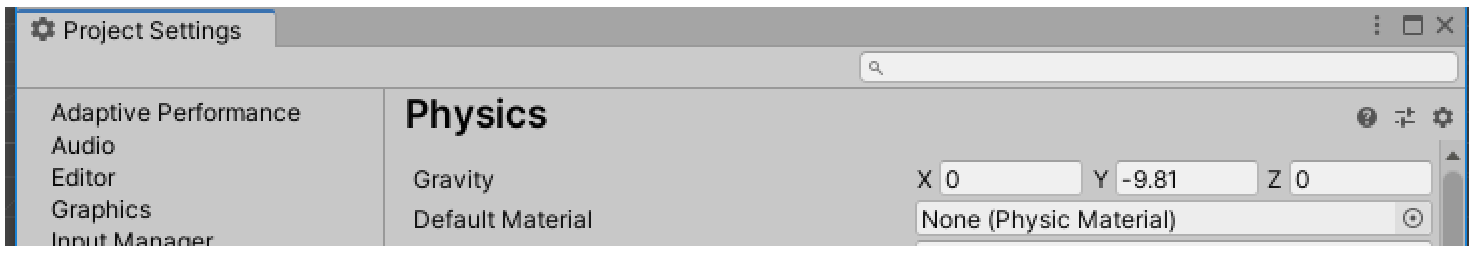
Project Settings

Adaptive Performance
Audio
Editor
Graphics
Input Manager
Package Manager
Physics
Physics 2D
Player

Time

Fixed Timestep 0.02
Maximum Allowed Timestep 0.3333333
Time Scale 1
Maximum Particle Timestep 0.03

World **Gravity** is set in the Physics settings
NOTE: the default is based on earth's gravity (9.81 metres per second)
A default physic material can also be assigned - this sets the bounce/grip/slip of a surface



Project Settings

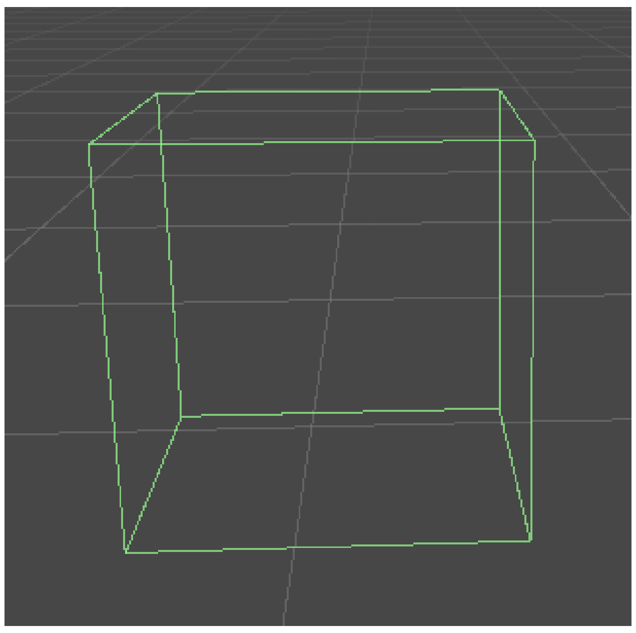
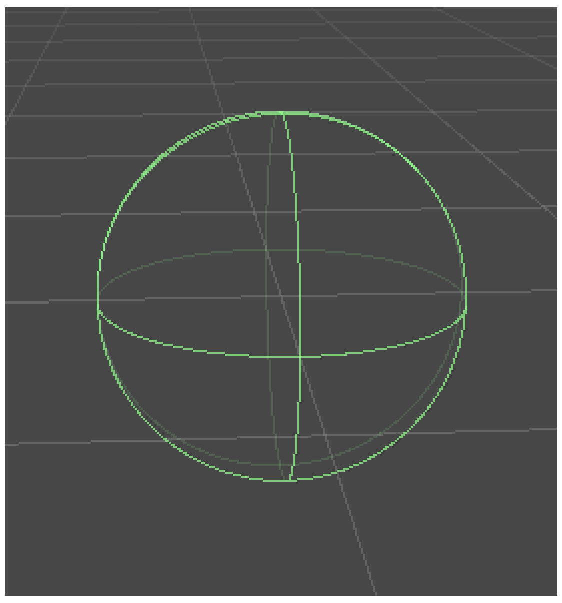
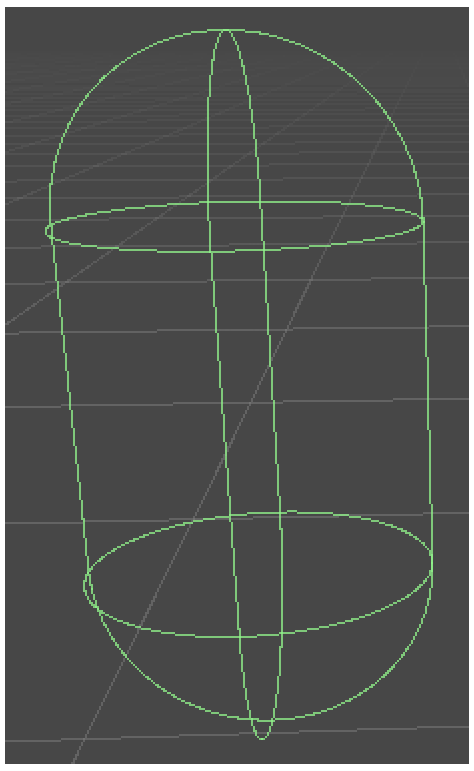
Adaptive Performance
Audio
Editor
Graphics
Input Manager

Physics

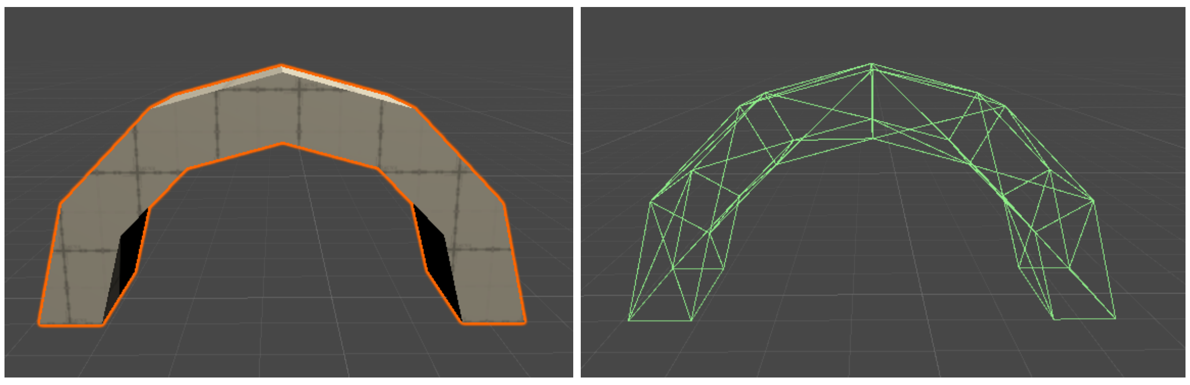
Gravity X 0 Y -9.81 Z 0
Default Material None (Physic Material)

3D Colliders

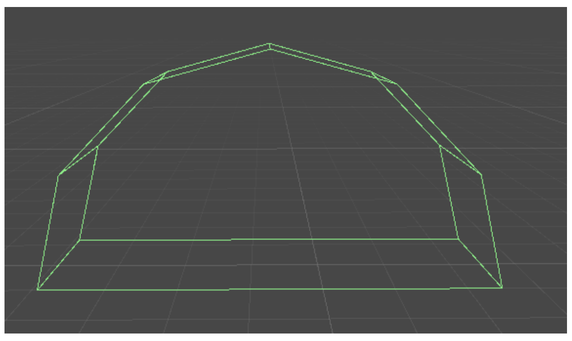
Default Colliders

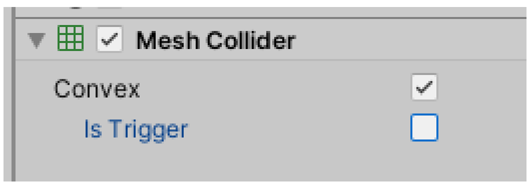
Box Collider	Sphere Collider	Capsule Collider
		

Mesh Collider
NOTE: the Collider can be a **Concave** shape



Mesh Collider: Convex mode
NOTE: to use a Mesh Collider as a trigger, it will need to be converted to a **Convex** Collider





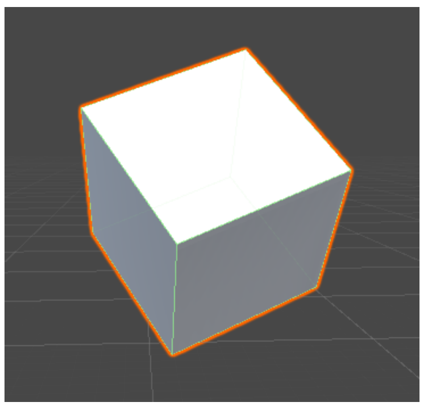
Mesh Collider

Convex ✓

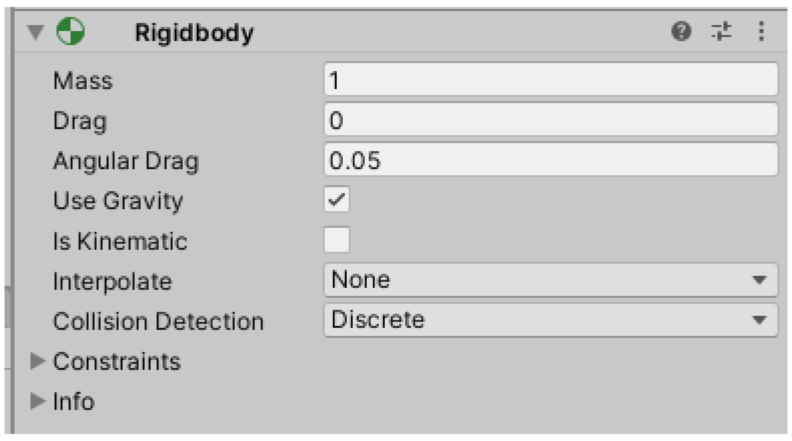
Is Trigger ☐

Dropping a Cube

This cube has a Rigidbody to use the Physics system
A Collider is also used to interact with the world

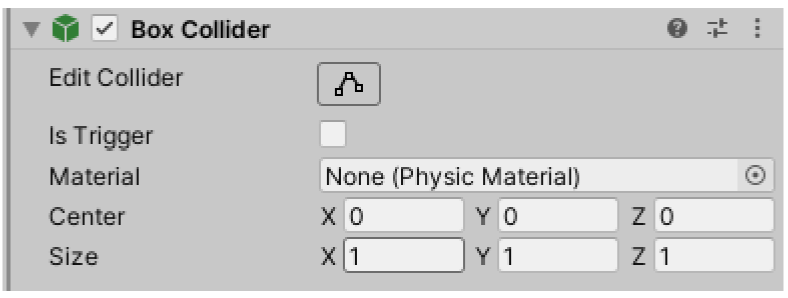


The Rigidbody has settings to use mass (in KG), gravity and updating
The Collider's shape is used to interact with any other colliders in the scene



Rigidbody

Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	
Interpolate	None
Collision Detection	Discrete
Constraints	
Info	



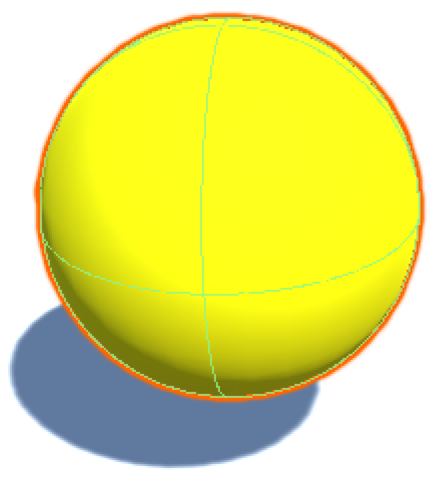
Box Collider

Edit Collider	
Is Trigger	
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Size	X 1 Y 1 Z 1

Movement physics

A Rigidbody can have directional forces applied to it as a Vector3

The sphere Collider shape can roll on a surface with a Rigidbody component



We can create a force using the input from a joystick or keyboard

```
// input values from a keyboard or joystick
float horizontalInput = Input.GetAxis("Horizontal");
float verticalInput = Input.GetAxis("Vertical");

// create a vector3 for the input direction
Vector3 inputDirection = new Vector3(horizontalInput, 0, verticalInput);

// normalise our input direction to keep the values below zero
inputDirection.Normalize();
```

Creating a force vector3 from the input

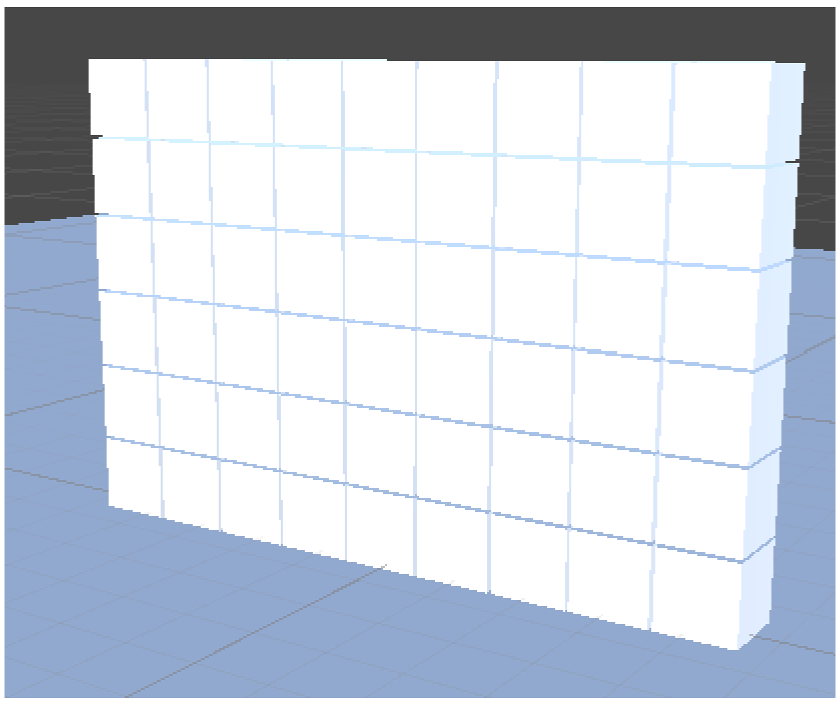
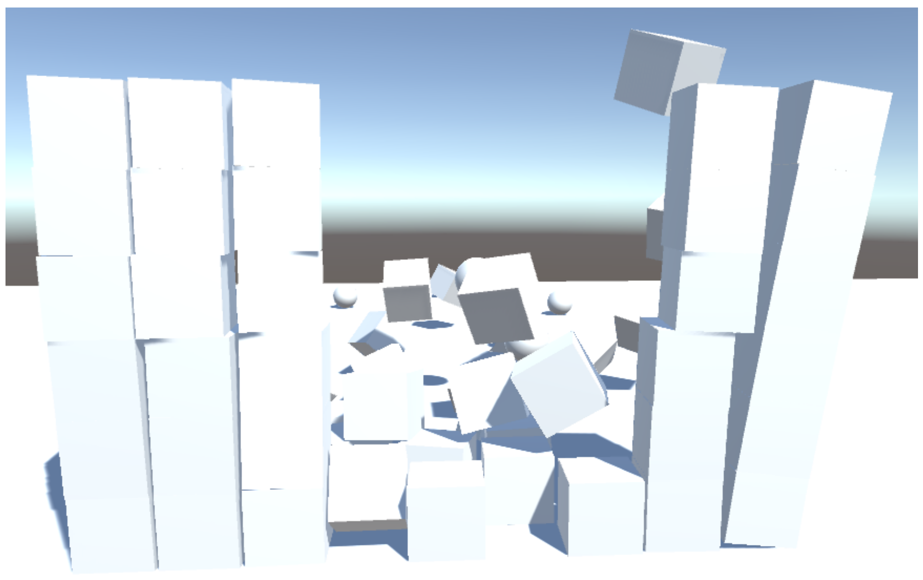
```
// create a force vector3 by multiplying the input direction by the speed field
Vector3 force = inputDirection * speed;
```

Forces can be applied to a Rigidbody by using AddForce

```
// here we use the AddForce method on the Rigidbody to add a force in a direction
// NOTE: we use Time.deltaTime to sync the force to the update
body.AddForce(force * Time.deltaTime);
```

Ball shooter

Shoot spheres at wall of cubes to knock them down

Aiming

SmoothMouseLook, a script from the Unity Wiki provides aiming with the mouse
http://wiki.unity3d.com/index.php/SmoothMouseLook

Gun

The gun spawns a bullet at an offset in front of the gun gameobject

```
// offset in front of the gun
// used with transform.forward to calculate point
[SerializeField]
private float forwardOffset = 3;
```

The spawn position is the forward multiplied by the offset to be in front of the gun

```
Vector3 spawnPosition = transform.position + (transform.forward * forwardOffset);
```

Bullet

The bullet's Rigidbody component has a force applied to it in the start method
When the bullet is spawned the force is instantly applied!

We have a power field for the force multiplier

```
// the amount of power applied to the force
[SerializeField]
private float power = 100;
```

The force is the transform forward multiplied by the power

```
// the force to apply to the Rigidbody
// the forward direction multipled by the power
Vector3 force = transform.forward * power;
```

The Rigidbody's AddForce method will apply the calculated force
The ForceMode.Impulse will apply the force directly - the force is much greater!

```
// use the AddForce method on the Rigidbody
// ForceMode.Impulse will apply the force directly
GetComponent<Rigidbody>().AddForce(force, ForceMode.Impulse);
```

ForceMode

You should use ForceMode.Impulse when you are only applying force once - like in the start method
You should use ForceMode.Force when you apply force within an update method

Physics Materials

A Physic Material is an asset that can be used on Colliders.
A Collider can have grip and bounciness settings when interacting with other Colliders

Dynamic/static friction

- Amount of grip/slip the collider has

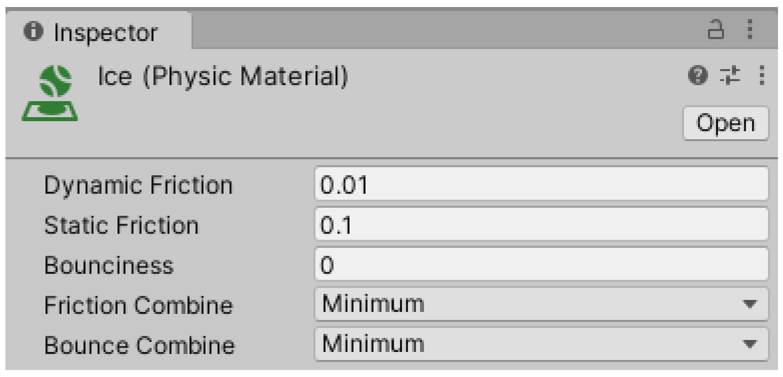
Bounciness

- If a colliding object should bounce

Friction/bounce combine

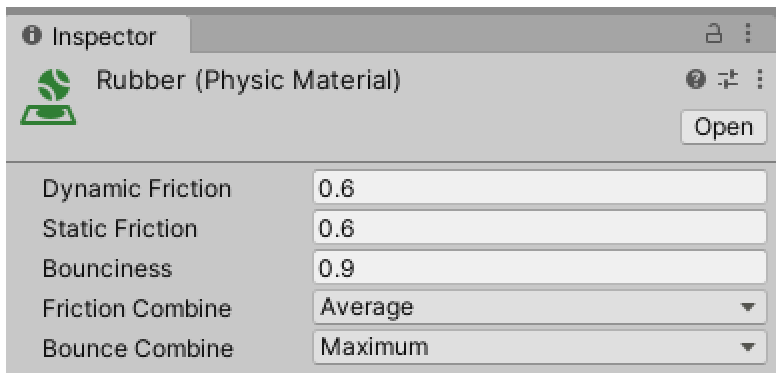
- How to calculate friction/bounce
  - Minimum/Maximum values
  - Average/multiply values

Ice material



Dynamic Friction	0.01
Static Friction	0.1
Bounciness	0
Friction Combine	Minimum
Bounce Combine	Minimum

Rubber material

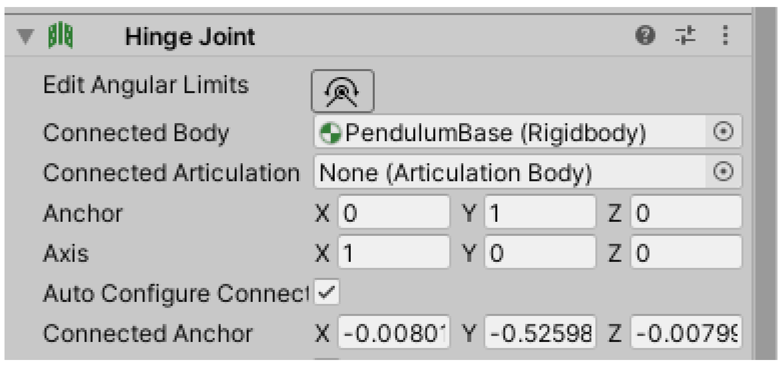


Dynamic Friction	0.6
Static Friction	0.6
Bounciness	0.9
Friction Combine	Average
Bounce Combine	Maximum

Pendulum

Hinge joint component

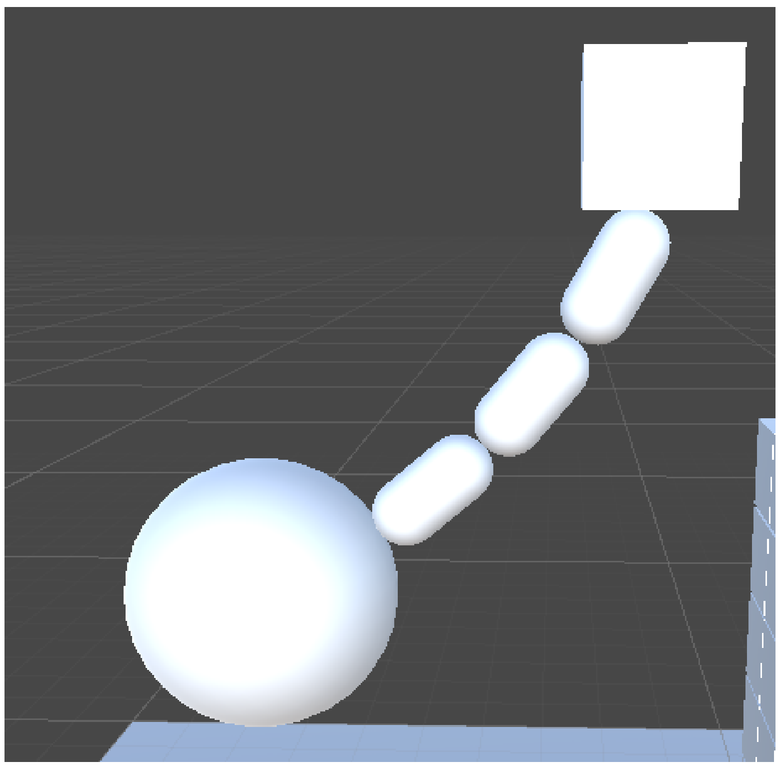
- Provides a connection with an offset between 2 Rigidbodies
- The connection has an axis of rotation



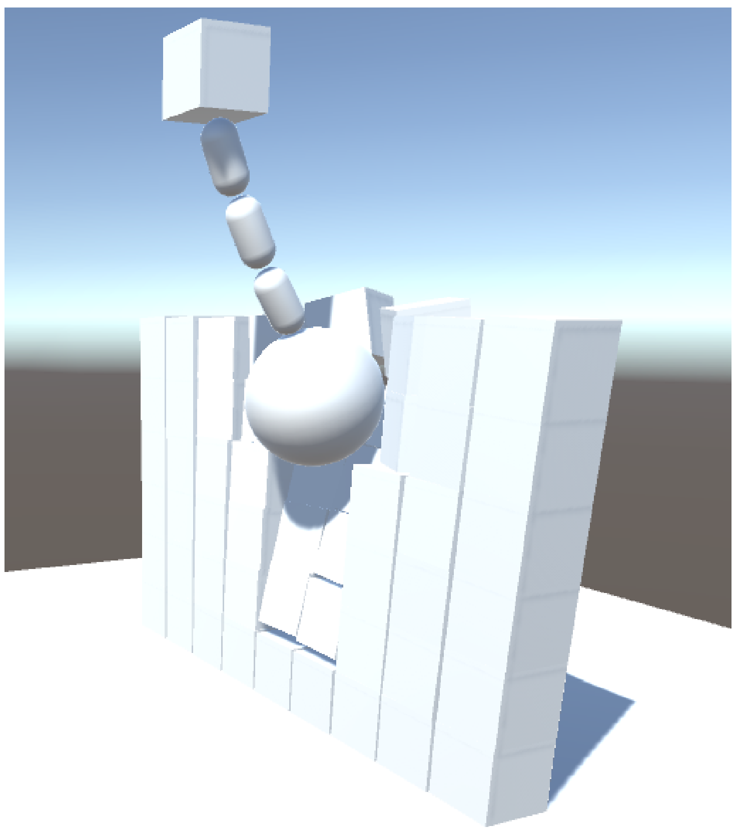
Hinge Joint

Edit Angular Limits	
Connected Body	PendulumBase (Rigidbody)
Connected Articulation	None (Articulation Body)
Anchor	X 0 Y 1 Z 0
Axis	X 1 Y 0 Z 0
Auto Configure Connect	✓
Connected Anchor	X -0.0080 Y -0.52598 Z -0.0079

A set of Rigidbodies with hinges connecting them

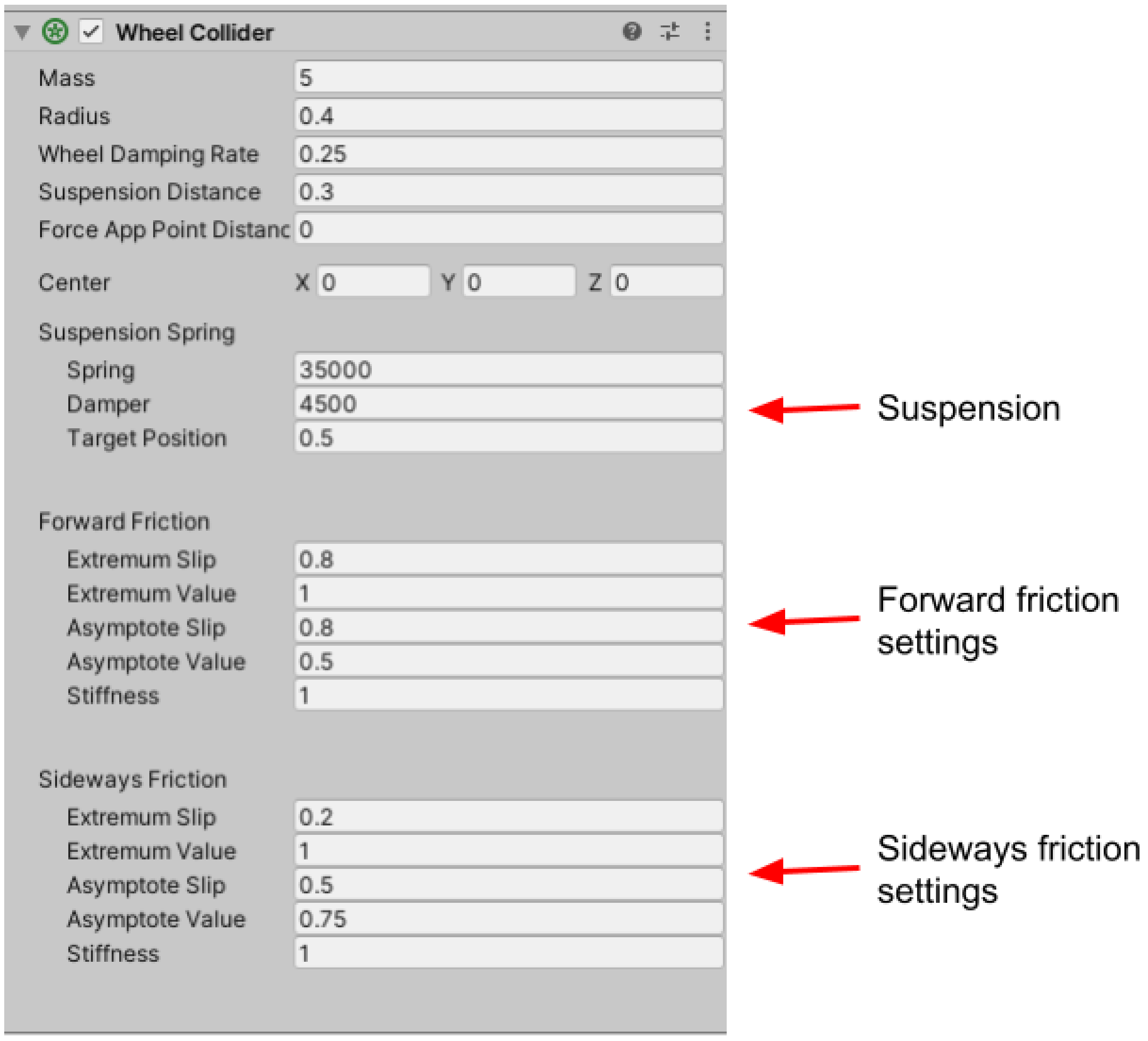


The simulation will chain a ball to a static cube using the hinge component to form a hierarchy



Car

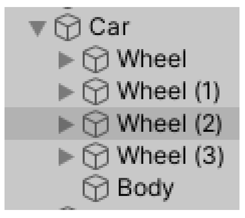
The Wheel Collider is specifically for wheeled vehicles
We can set many different aspects of wheel control, both forward and sideways friction.
We can also adjust the height and strength of the suspension for each wheel



Wheel Collider

Mass	5
Radius	0.4
Wheel Damping Rate	0.25
Suspension Distance	0.3
Force App Point Distanc	0
Center	X 0 Y 0 Z 0
Suspension Spring	
Spring	35000
Damper	4500
Target Position	0.5
Forward Friction	
Extremum Slip	0.8
Extremum Value	1
Asymptote Slip	0.8
Asymptote Value	0.5
Stiffness	1
Sideways Friction	
Extremum Slip	0.2
Extremum Value	1
Asymptote Slip	0.5
Asymptote Value	0.75
Stiffness	1

The Car class controls a set of four WheelColliders, attached as children to the car
Car Hierarchy



```
Car
  Wheel
  Wheel (1)
  Wheel (2)
  Wheel (3)
  Body
```

The car has wheel torque (how fast the wheel turns to drive)
Steer angle max is the maximum angle in degrees the front wheels can turn to steer

```
// the amount of forward rotational force applied to the wheel
public float torque = 200;

// the maximum angle the wheel can turn into corners
public float steerAngleMax = 30;
```

GetComponentsInChildren will gather the WheelColliders from the car hierarchy

```
// search the children of the transform for the WheelColliders
wheels = transform.GetComponentsInChildren<WheelCollider>();
```

When calculating the steering turn, the steerAngleMax is added to the input

```
// turn input applied to the wheels
// multiplied by the steer angle to get the degrees
float turnInput = Input.GetAxis("Horizontal") * steerAngleMax;
```

We calculate the forward turn or acceleration of the wheel

```
// calculate the forward turn to move the vehicle
float forwardTurn = torque * accelInput;
```

WheelCollider has a motorTorque to apply acceleration to the wheel

```
// apply the forward turn or torque input
// WheelCollider has a motor torque setting to apply our current torque
wheels[i].motorTorque = forwardTurn;
```

WheelCollider has a steerAngle in degrees to set the steer angle

```
// WheelCollider has a steering angle in degrees we can use to steer
wheels[i].steerAngle = turnInput;
```

The WheelCollider updates its physics position and rotation, this can be accessed by using GetWorldPose

```
// WheelCollider has a GetWorldPose method that calculates the correct position and rotation of the wheel
// we can use this for our wheel graphics to position them exactly
wheels[i].GetWorldPose(out Vector3 wheelPosition, out Quaternion wheelRotation);
```

Any wheel graphics can be placed in the exact position of the physics wheel

```
// set the position of the wheel
wheelGraphics.position = wheelPosition;
```

And apply any rotation offsets before rotating the graphics to match

```
// store the rotation so we adjust it
Vector3 angles = wheelRotation.eulerAngles;

// we add an offset here to roatate the wheel correctly on the Z Axis
angles.z += 90;

// apply the rotation to the wheel graphic
wheelGraphics.eulerAngles = angles;
```

Links

Box Collider
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/BoxCollider.html

Sphere Collider
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/SphereCollider.html

Capsule Collider
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/CapsuleCollider.html

Mesh Collider
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/MeshCollider.html

Rigidbody
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Rigidbody.html

Rigidbody.AddForce
https://docs.unity3d.com/ScriptReference/Rigidbody.AddForce.html

Time.fixedDeltaTime
https://docs.unity3d.com/ScriptReference/Time-fixedDeltaTime.html

Rigidbody.IsSleeping
https://docs.unity3d.com/ScriptReference/Rigidbody.IsSleeping.html

Rigidbody.Sleep
https://docs.unity3d.com/ScriptReference/Rigidbody.Sleep.html

Physic Material
https://docs.unity3d.com/2020.2/Documentation/Manual/class-PhysicMaterial.html

HingeJoint
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/HingeJoint.html

WheelCollider
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/WheelCollider.html

WheelCollider.steerAngle
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/WheelCollider-steerAngle.html

WheelCollider.motorTorque
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/WheelCollider-motorTorque.html

WheelCollider.GetWorldPose
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/WheelCollider.GetWorldPose.html

GetComponentsInChildren
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/GameObject.GetComponentsInChildren.html

Quaternion.eulerAngles
https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Quaternion-eulerAngles.html

GAME DEV. TECH.

UNIVERSITY OF PLYMOUTH