Games-Middleware Breakdown

## Part 1: Physic Simulation

The SpherePhysics.cs Script is a multi-step approach to manually simulating collisions between multiple spheres as well as planes.

**Sphere Movement Simulation**

* **Update Velocity and Position** (FixedUpdate):

Gravity (gravityForce) and drag (dragForce) Vectors are applied to calculate acceleration.

**Collision with Planes**

* **Checks Distance to Plane** (ResolveCollisionWith):

Calculates the current and previous distances between the sphere and the plane.

* **Computes Time of Impact (TOI)**:

Determines the precise time when the sphere contacts the plane.

* **Resolves Velocity at Impact**:

Splits velocity into components relative to the plane's normal and resolves the new velocity, with the coefficient of restitution.

* **Adjusts Position**:

Calculates the sphere's new position after accounting for the remaining time.

**Sphere-to-Sphere Collision**

* **Detects Collision** (isCollidingWith):

Checks if the spheres overlap by comparing the distance between their centres with the sum of their radii.

* **Computes Time of Impact (TOI)** (ResolveCollisionWith):

Calculates the time when the spheres first contact based on their current and previous distances.

* **Determines Collision Normal**:

The collision normal is the direction of the vector connecting the centres of the two spheres.

* **Splits Velocities into Components**:

Separates velocities into parallel and perpendicular components relative to the collision normal.

* **Applies Conservation of Momentum**:

Updates parallel velocities based on masses and the coefficient of restitution.

* **Updates Position and Velocity**:

Adjusts the spheres positions and velocities for the remaining time after collision.

**Post-Collision Validation**

* **Detects Overlaps**:

Checks if the spheres overlap after resolution, indicating edge cases or errors.

## Part 2: Animation

The animation portion of the project has a few different sections.

* The first part is using the animations from one model from the Unity asset store (supercyan character pack) on the model of another model (banana man).
* The character controller provides basic movement with WASD and animations with a blend tree to animate the walking based on speed.

A screenshot of a computer

Description automatically generated

* There is also a mask on the animation layer for upper body actions such as throw (T key) and wave (J key) so the walking is not affected by those actions.

A screenshot of a computer

Description automatically generated

* An animation event is used to play a “whoosh” sound during the throw animation.

A screenshot of a computer

Description automatically generated

Figure : AudioManager on character prefab

A screenshot of a video editing

Description automatically generated

Figure : AnimationEvent on throw animation(I used attack animation)

A screenshot of a computer

Description automatically generated

Figure : Add the method from AudioManager onto Animation Event

A screenshot of a computer program

Description automatically generated

Figure : AudioManager Script & PlayThrowSound Method

## Part 3: Networking

The networking is done using Unity’s Netcode for GameObjects.

Important things worth noting are:

A few overriding scripts such as ClientNetworkAnimator are used to allow the client and hosts of the network session to animate and transform themselves without having to ask the server to do it.

A screen shot of a computer program

Description automatically generated

Network object, rigidbody, animator and transform components are added to the character prefab to allow network synchronising.

A workaround for picking up an object had to be used due to complications with how network object can reparent objects during runtime.

Effectively, a network object can only reparent a network object to another object with a direct network object component. In my use case: this made parenting a snowball directly to the characters hand impossible as the hand itself did not have a network object, even if it was the child of a network object.

As such a workaround was used where a separate game object with a network object component is set to follow the characters hand, then the snowball can be parented to that game object without issue.