Topics for discussion

**Particle systems**

Water, smoke, fire, sparks can be simulated with large amounts of particles that interact with each other.

A particle system is an effective way of managing many objects that interact with each other, lots of objects responding to forces. But these particles don’t have to interact with each other to create an effect like smoke or liquid in games and can just have a lifetime or despawn after a condition.

Involves making a class of individual particles with variables like location velocity and acceleration, and creating lots of those particles

**Fluid dynamics**

Fluid dynamics is the study of the flow of fluids(liquids and gases)

It is used in physics engines and games to simulate how fluids move, like ocean waves, smoke, and fire.

There are two ways of simulating fluid, through a grid or particle system. Particle systems are a collection of many objects that move and interact with each other. Grid systems calculate the average of the attributes of particles that would be in each square. The most accurate methods use both together.

**Brownian motion**

Erratic random motion of particles continuously hitting and applying forces in random directions to a larger object in a fluid medium (ie water molecules hitting pollen)

Used in high precision and realistic simulations. Not used in game engines to reduce computational power required

**Constraints**

Any rotational or positional constraints

Hinges, ragdoll

Springs

**Collisions**

Spheres, cubes, arbitrary meshes

**Continuous vs Discrete collision detection**

Discrete collision detection check if objects are colliding every physic frame and is the cheapest way computationally but can cause object to warp through each other (tunnelling) or has weird behaviour if speeds are too high

Continuous collision predicts where the object is going to be next frame, and if the path of the object causes it to collide with something, a collision is detected. This method is expensive computationally but ensures objects are hit.

**Different Algorithms for detecting collisions in a particle system or many object in one area**

Sweep and Prune Algorithm: Sorting all objects in one axis and check if they are intersecting

Uniform Grid Space Partitioning: Dividing the space were all objects are into uniform grid and only checking collisions inside each grid

Smarter Space Partitioning: Similar concepts to Uniform Grid but more evenly divides objects for efficiency (K-D Trees)

Object Partitioning: Similar to Space partitioning but instead of spaces, it is dependent on object, one example is bounding volume hierarchies

**Ridgid vs Soft body physics**

A rigid body or rigid object is an object where there is zero deformation, the distance between two points on a rigid body never changes regardless of the forces applied to it. Most commonly used but a real rigid body does not exist in the real world

A soft body is similar to a rigid body but can deform while retaining its shape to some degree(like a fluid) when forces are applied to it. Used in video games and film for many applications. Created using nodes or point masses connected with ideal springs between them.

**Real-time and high-precision physics engines**

How does real-time and high-precision physics engine differ?

Real-time physics engines are the most common and useful for video games and simulations, it ignores a lot of realistic calculations in order to reduce the computational power/time needed. It only has to be good enough for gameplay to feel smooth and fun, not precise like the real world.

High-precision physics engines are used mostly for scientific research or simulations that require a high degree of accuracy and implement a lot of topics like fluid dynamics, Brownian motion, friction, and plasticity that require the most computational power and time to calculate

**Friction models**

How to add and calculate friction between objects

Add friction to colliding spheres (rotation after collision)

**How do servers for a real game like rocket league works**

**Fixed tickrate vs variable tickrate**

How tick rate affects physics in a real game like rocket league

* pros and cons for faster or slower tick rate (computation, networking, collision consistencies)
* Fixed tick rate vs variable tick rate

**Client-side vs server-side calculations (latency, cheating, computation cost)**

* Pros and cons of each

[Rocket League ball physics - Rocket Science #4 - YouTube](https://www.youtube.com/watch?v=9uh8-nBlufM)

Hitboxes are cubes, but collisions between car and ball are calculated like spheres in order to make collisions feel more natural and fun. When the ball collides with the cube hitbox, a normal is drawn from the centre of mass of the car to the collision point and that is used for the collision.