**Interpolation and Extrapolation**

**Does not update the physics system for collisions, or predict collisions based on current paths. Only creates a smooth set of points to jump to between the current physics update and the next predicted position for the next update. Lerp.**

**Interpolate** – going from the fake position you are currently rendering at to the actual position that the physics system has you at.

* Once we’ve passed B, and we are in the range of B and C, knowing the previous position and bringing it to known position between physics updates.
* I’m catching up to where the last known position that the object is in

**Extrapolate** – predicting forward where you think the object is going to go

* I’m going past that position to predict where the object will be.
* Knowing the last position and bringing it to the next predicted position

Objects that are being extrapolated will be ahead of objects that you are interpolating.

Happens a lot in your physics systems and networking.

Unity/Unreal under the hood will provide some sort of interpolation/extrapolation for a physics engine. It’s not the responsibility of the physics engine to provide these updates, but up to the game engine itself to use interpolation and extrapolation on the physics systems.

How do you find the speed if you know the velocity?

* Figure out the hypotenuse
* Rb.velocity.magnitude

For direction?

* Rb.velocity.Normalize() – sets the size of the vector to 1