**UNITY EDITOR (GAME DEV CLASS)**

1. **Rigidbody in Unity:**

* **Mass:-** if collides with an object how far the object will move. depends on mass of object itself.
* **Drag:-** will stop the object. linearly.
* **Angular Drag:-** will stop the object angularly.
* **Use Gravity:-** if gravity should be present or not.
* **Is Kinematic:-** static. not moving.
* **Interpolate:-** if *none*, rendering and physics will not be synced. if *interpolate*, then the rendering and physics will be synced at intervals. *extrapolate*:- 100% effort. if you want smooth movement, then choose extrapolate.
* **Collision Detection:-** if *discrete*, checks after a while (after FixedUpdate) if collided. if *continuous*, checks at every frame. if *continuous dynamic*, very expensive, as it checks itself and other rigidbodies and colliders around it + at every frame. (dependent on situation).
* **Constraints:-** freezes an object on its position.

\*MeshCollider is a bit expensive as it makes a mesh according to the object's shape. For a quad or plane, use box collider.

1. **BoxCollider in Unity:**
   * **IsTrigger:-** if ticked, collider gets removed. you know you have reached the object. used for pickups usually.
   * **Centre:-** respective to the object it is attached to.
   * **Size:-** respective to the object it is attached to.
   * **Material:-** defines what kind of object it should be.

**\***An object will move through another object, even if a collider is attached to it, unless otherwise scripted.

\* A Rigidbody is moved when a ***force*** is applied to it.

\* When referencing an object, use this syntax :- Type.name (example, rigidbody, player type etc)

\* In local transform, only forward and right and up exist in unity. If you want to go backward or left or down, multiply by -1.

\*All objects that are interacted should have Rigidbody attached, as otherwise they won’t have a defined mass. Their mass will be infinity and won’t be able to move.

\*We avoid using Rigidbody, as when objects collide with obstacles, they bounce back according to the physics applied to it, and depend a lot on phycics. Wherever physics is present, we won’t be able to control objects. We use CharacterController instead.

1. **Materials:**
   * A renderer shows an object. But how will we know what type of an object is it? We use materials.
   * Example, a table will have a wooden material to show it’s made out of wood.
   * **Shader:-** Defines the view of a material on an object in scene/game view. How? It has different types, like standard, unlit etc. Then accordingly, it changes the look and feel of the object. We can add multiple textures with one shader on an object.

\* A quad is optimized that’s why we use it. Multiple triangles make up one shape. A quad is made up of two triangles. 4 vertices. So, a quad doesn’t have any normals on its downside, that’s why you’re unable to see it.

1. **CharacterController:**

* **Skin Width:** if 0, the collision will be right on its surface/skin. If more, the object will stop before it touches the character.
* **Radius:** defines collision area.
* **Min Move Distance:** when a rigid body is static, its sleep. Consider it like the mass of the character. More value, more force required to move it.

\*Go to Edit>Project Settings>Input. Axes. Negative button and Positive button in Horizontal, corresponds to the keyboard keys for the same game. It gives controls for joystick and keyboard games.

\*Pooling:- objects are created and kept in a pool, and when needed they are used. Saves memory and better performance.