**Box Colliders, Physics System, Joints, and Is Kinematic**

Colliders

* If you want an object to have a shape in the physics system, give it a collider.
* If you want to an object that has a shape to be simulated by the physics system (mass, gravity, collisions), give it a RigidBody Component
* If you want an object that you are controlling its position in space (via transform in script), and you don’t want it to be simulated in the physics system. However, you still want the collision callbacks, so that it registers a collision when an object with a rigidbody hits it, then you make your rigidbody Is Kinematic. (Floating platforms, doors, windows, ect…)
* If two objects pass into each other, at least one of the objects must have a Rigidbody for a collision to happen.
* To be able to move an object or have an object be moved by a collision, that object must have a Rigidbody component attached to it.
  + Not True: The first part at least. You can move an object with the transform without a rigidbody on it, but that is much more cpu intensive to do so without a Rigidbody attached to it. So, it might as well be true, since the CPU difference makes it silly not to attach a Rigidbody to a moving object. It is true that an object won’t be moved by a collision unless a rigidbody is attached to it. (Also, it’s not set to Is Kinematic).
* Collision calls only happen via the physics system,
  + So, if you have a physics timestamp of more than .02, and you have a fast-moving object, there’s a chance the object will be inside or through the object before the physics timestamp fires. The result being no collision will be detected.
    - To combat this, you can set your collision detection to continuous. This doesn’t force the physics system to update faster but it does project the path of the moving object, and checks to see if there is another static object in it’s path between physic system calls. If there is, then the object will stop when colliding with the other object

Is Kinematic

* I want to hook up a box collider on this object, and I want a rigidbody on the object, but I don’t want it to be continually updated via the Physics system. (aka Forces, collision, and joints will not affect the rigidbody directly). The Rigidbody will be under full control of an animation or script.
* Kinematic Rigidbody objects are like a static collider object except a moving Kinematic Rigidbody will apply friction to other objects and will ‘wake up’ other rigidbody objects that make contact.
* If two Objects with Kinematic Rigid Bodies pass into each other, the collision detection will not happen. It will move right through it despite both of them having Rigidbodies
* When you push an object with IsKinematic turned on, it will collide with other rigidbody object (NOT Kinematic) and move it straight off. If you had Is Kinematic flagged as off, then physics system handles the collision from the player and will be begin to rotate both objects for an accurate collision simulation.
* If you use a transform.positiion to move a player with a RigidBody (not Kinematic) on it in the update function, then the physics system is fighting with the update function, which will cause jitteriness.

Performance issues with Physics System in Unity

* Moving an object via the transform in script, and then not putting a rigidbody on it (IsKinematicj)
* Moving an object via the transform in script, putting a RigidBody on it, and not setting it to not Kinematic.
* Using Mesh Colliders
* RigidBodies inside of other Rigidbodies (a child object also has a rigidbody just like its parent object)
  + Two physic system equation fighting each other, makes for awkward results

Rotate Objects with RigidBodies

* Set the angular velocity directly, or use AddTorque
  + Rb.AddTorque(10, 0, 0) // be sure to account for mass of object
  + Rb.angularVelocity = new Vector3(10, 0, 0);

Moving Platforms

* Crank up the friction
* Implement custom logic that asks what the player is standing on. Is that object moving? If that object is moving, you want to move the player along with it.

