**ASSIGNMENT**

* **PARTICLE SYSTEM:**
* A particle system simulates and produces many small images or meshes, known as particles. Which are used to create a visual effect.
* Each particle in a system represents an individual graphical element in the effect.
* The system simulates every particle collectively to create the impression of the complete effect.
* Particle systems are useful when you want to create dynamic objects like fire, smoke, liquids, bullet effects because it is difficult to depict this kind of object with a Mesh(3D) or Sprite(2D). Meshes and Sprites are better at depicting solid objects such as a house or a car.
* **Built-in Particle System:**
* Unity’s Built-in Particle System allows you to create effects.
* The Built-in Particle System simulates particle behavior on the CPU which allows for the following main benefits:
* You can use C# scripts to interact with a system and the individual particles within it.
* Particle systems can use Unity’s underlying physics system and thus interact with Colliders in your Scene.
* **Varying properties over time in built-in particle system:**
* Constant: The property’s value is fixed throughout its lifetime.
* Curve: The value is specified by a curve/graph.
* Random Between Two Constants: Two constant values define the upper and lower bounds for the value; the actual value varies randomly over time between those bounds.
* Random Between Two Curves: Two curves define the upper and lower bounds of the value at a given point in its lifetime; the current value varies randomly between those bounds.
* **COLLIDERS:**

Collider components define the shape of a GameObject for the purposes of physical collisions. A collider, which is invisible, does not need to be the exact same shape as the GameObject’s mesh.

* Compound colliders:

Compound colliders approximate the shape of a GameObject while keeping a low processor overhead. To get further flexibility, you can add additional colliders on child GameObjects. For instance, you can rotate boxes relative to the local axes of the parent GameObject. When you create a compound collider like this, you should only use one Rigidbody component, placed on the root GameObject in the hierarchy.

* Mesh colliders:

These colliders are much more processor-intensive than primitive types, so use them sparingly to maintain good performance. Also, a mesh collider does not collide with another mesh collider (i.e., nothing happens when they make contact).

* Static colliders:

You can add colliders to a GameObject without a Rigidbody component to create floors, walls and other motionless elements of a Scene. These are referred to as static colliders. On the other hand, colliders on a GameObject that has a Rigidbody are known as dynamic colliders. Static colliders can interact with dynamic colliders but since they don’t have a Rigidbody, they don’t move in response to collisions.

* **Triggers:**

The scripting system can detect when collisions occur and initiate actions using the OnCollisionEnter function. However, you can also use the physics engine simply to detect when one collider enters the space of another without creating a collision. A collider configured as a Trigger (using the Is Trigger property) does not behave as a solid object and will simply allow other colliders to pass through. When a collider enters its space, a trigger will call the OnTriggerEnter function on the trigger object’s scripts.

.

## 

.