UNITY ENGINEER INTERVIEW.

**Unity**

**Start** is called on the frame when a script is enabled just before of the Update method is called the first time.

**Awake** is called when the script instance is being loaded, after all the objects are initialized. regardless of whether or not the script is enabled.

Awake and Start are called only once in the Script lifetime.

-------------------------------------------------------------------------------------------------------------

-**Update**-> called every frame

-**Fixed Update**->called every Fixed framerate . For RigidBody operations.

-**Late Update**->After every update, for Camera position.

------------------------------------------------------------------------------------------------------------

**AssetsBundles**-> exported files for downloading content purposes. To stream content such as textures audio clips, etc.

**Resources Folder**, to allow content to be supplied in game file yet not loaded until requested. Collections of assets included in the build of unity, but are not necessary linked to a gameObject in the inspector. All Assets in “resources folder” can be acccesed via Resources.Load.

**Streaming Asstes Folder**: Asstes stores in the normal filesystem on target machine.

-------------------------------------------------------------------------------------------------------------------------

**Batching**: Techniques to address the great amount of computing resources needed for drawing objects on screen. The draw call to the graphics API (Such as OpenGL or Direct 3D) are often resource-intense. This is mostly caused to the state changes between draw calls (such switching to different materials).

**Dynamic Batching**: for small enough Meshes, this transform their vertices on the CPU groups many vertices together and draws them all in one go.

**Static Batching**: combines static (not moving) GameObjects into big Meshes and renders them in a faster way.

------------------------------------------------------------------------------------------------------------------------------

**Occlusion Culling** is a feature that disable rendering of objects when they are not currently seen by the camera because they are obscured (occluded) by other objects to save GPU processing.

**C##**

**Interface 1**

interface IContenedor

{

void Meter();

void Quitar();

}

class Contenedor : IContenedor

{

//The required method of the interface must be public

public void Meter()

{

Debug.Log("Metiendo");

}

public void Quitar()

{

Debug.Log("Quitando");

}

}

class OtroContenedor : IContenedor

{

//The required method of the interface must be public

public void Meter()

{

Debug.Log("Metiendo");

}

public void Quitar()

{

Debug.Log("Quitando");

}

private void OtherFunction()

{

Debug.Log("Another func");

}

}

class Test1Interfaces : MonoBehaviour

{

void Start()

{

IContenedor ic = new Contenedor();

Contenedor c = new Contenedor();

OtroContenedor oc = new OtroContenedor();

Test1(ic);

Test1(c);

Test1(oc);

}

public void Test1(IContenedor c)

{

c.Meter();

c.Quitar();

Debug.Log("Whatever");

}

}

**Interface2**

interface IVehiculo  
{  
 void Acelerar(int kmh);  
 void Frenar();  
 void Girar (int angulos);  
}  
  
interface IVehiculoVolador : IVehiculo  
{  
 void Despegar();  
 void Aterrizar();  
}

In this example Interface inherits form interface

A class implementing IVolador must inherit IVehiculo.

One Class can implements several interfaces at the same time. class Tanque : IVehiculo, IArmaDeGuerra

**Automated Tests**

Test your code frequently in less time, Catch bugs, refactor with confidence….

-----------------------------------------------------------------

Unit-test to test quickly

Integration-Test before commiting our code to repository.

------------------------------------------------------------------------------------------------------

Test method convention name:->MethodToBeTested\_Scenario\_ExpectedBehaviour

Example: UnitTest1.cs

[TestMethod]

public void CabBeCancelledBy\_ScenarioUserIsAdmin\_ReturnsTrue()

{

}

**Patterns and architectural concepts**

Factory method (No Refactoring which means changing structure of code without changing behaivour).