**Most Important Points:**

1. **Mesh:** Mesh makeup a large part of your 3D world.unity provides a modeling tool called proBuilder & there are also some assets store modeling plugins,such as Mesh Deformer,UModeler,Mesh Editor.
2. **Mesh Filter:** Mesh filter takes a meshes from your assets and pass it to "Mesh Render" for rendering on the screen.
3. **Mesh Rernder:** Mesh Render takes the geometry from "Mesh filter" & render it at the position defined by the GameObject's Transform and component.
4. **Material and Texture:** Firstly material apply to mesh and then texture apply on material.
5. **Global/Local:** In global mode arrow are aligned in world axes and local mode arrow shows the object direction.
6. **Pivot/Center:** x,y,z is a pivot and its origin is center.
7. **Tag & layer:** We can set tag at every object for identify and layer for camera rendering.
8. **Model import:** "FBX" is the format.all imported files in project window. it is up to you how to import(drag & drop,from file menu option etc.)
9. **For build:** ctrl+shft+b
10. **Game object:** set of collection (transform and component).
11. **MonoBehaviour:** Monobehaviour is the base class and all functions and variables can call from that class to another class.
12. **Methods:**

> Initialize:

\* "Awake()" is called varable or game state etc.

\* "OnEnable()" is called after awake method.

\* "OnDisable()" is called in the end method.

\* "Reset()" is called when the script is enabled and not palymode.

\* "Update()" method working on fps(frame per second).For Example: update works on 60 frames(pictures/images) per second.

\* "Start()" is called on the frame when script is enabled just before update method are called first time.

1. **Delta Time and Space.World/Space.Self:**

\* "Delta Time(DT)" is the completion time in the second since last frame.For Example:Frame 1 executing,the DT will be 0.Then game continue & frame 2 executing.so,the DT between 2 frames is 0.05 seconds.

***NOTE:***  FPS(frame per second),DT value is changing continuously,for this example used 0.05.

Detail Example: suppose, we use fps = 20,

speed = 10,

Vector3.forward = 1,

DT is approximately erqualto 1/fps so,DT=0.05 (Here)

totalMovement = fps\*Vector3.forward\*speed\*DT = 10 Ans.

\* "Space.World" is the parameter which is used to say to move the object global/scene/world axis, not for self/local axis.

\* "Space.Self" is the parameter which is used to say to move the object local axis, not for world axis.

1. **Writing method of Vector:**

\* new Vector(0,0,0)

\* Vector.forward

1. **Gizmo:** box collider or structures gizmo
2. **Scene Manager/Load scene/LoadSceneMode:**

\* scene management at run time.

\* load the scene by its name or index which is defined in build setting.

\* Used when loading a Scene in a player.Use LoadSceneMode to choose what type of Scene loads when using SceneManager.LoadScene. The available modes are Single and Additive.

1. **Scene loading/Additive loading:**

\* Adds the Scene to the current loaded Scenes is called Additive.

\* Closes all current loaded Scenes and loads a Scene.

1. **List and Dictionary:**

\* A Dictionary is similar to a List. However, instead of accessing a certain element by index value,

we use a string called key.

\* A list is an object which holds variables in a specific order. The type of variable that the list can

store is defined using the generic syntax.

1. **Delegate:** There are 2 type of delegate "single delegate" and multicast delegate".A Delegate is a reference pointer to a method. It allows us to treat method as a variable and pass method as a variable for a callback.When it get called ,it notifies all methods that reference the delegate.

\* **single:** It can reference to only single method at a time which represent (=).

\* **multicast:** It can store the reference of multiple methods at a time which represent (+=).

**How to use:**

> //delegate define;

delegate void myDelegate(int num);

myDelegate myDelegated;

> //single delegate

myDelegated = printNum;

> //calling delegate

myDelegated(5);

> // create method

void printNum(int num){ print("print num: " + num); }

1. **Custom Event:** "Custom Event" same as "Delegate". custom events has only multicast like delegate has.

**How to use:**

> // Custom Event define

public UnityEvent player2;

> // calling custom event

void playerOnTrigger() {

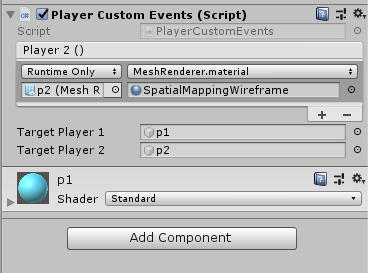
.

.

.

player2.Invoke();

Debug.Log("Player1 Deactivated & Player2 Activated Custom-Event call at Player2"); }



1. **Common Method Unity:**

* **Instantiate:**  is a function which is used to create clones of game object. This is commonly used in context of cloning a prefabs which is work at run time.
* **Destroy:** Removes a GameObject, component or asset. This immediately remove from update and if you declear time then it remove after certain time.
* **LookAt:** Rotate the camera every frame so it keeps looking at the target.
* **Coroutine: “**IEnumerator” is the signature of coroutine. couroutine allow us to pause execution at one point give control back to unity until we tell it otherwise and then it continue the execution afterward from where it left off and based on the time we might have asked.
* **How to use couroutine:**

IEnumator yourfunctionName(){yield return new waitforsecond(3);}

StartCoroutine(yourfunctionName());

* **UnityWebRequest:** The UnityWebRequest object is used to communicate with web servers. Return **UnityWebRequest** An object that retrieves data from the uri.
* **Enuemrator:**  Enumerations allow you to create a collection of related constants.
* **Invoke Repeating:** Work as coroutine “Invoke repeating” takes method name,when run,delay run.
* **Invoke:** Invoke function allow us enable you to schedule a function call after some specified time delay.

1. **Mouse Events:**

//MOUSE EVENTS

private void OnMouseEnter()

{Debug.Log("Mouse Enter");

}

private void OnMouseDrag()

{Debug.Log("Mouse Drag");

}

private void OnMouseOver()

{Debug.Log("Mouse Over");

}

private void OnMouseDown()

{Debug.Log("Mouse Down");

}

private void OnMouseUp()

{Debug.Log("Mouse Up");

}

private void OnMouseUpAsButton()

{Debug.Log("Mouse Up As Button");

}

private void OnMouseExit()

{Debug.Log("Mouse Exit");

}

1. **Keyboard Events:**

/\* void objectMovingWithSimple()

{

if (Input.GetKey(KeyCode.LeftArrow))

{

transform.Translate(Vector3.left \* this.speed);

}

else if (Input.GetKey(KeyCode.RightArrow))

{

transform.Translate(Vector3.right \* this.speed);

}

else if (Input.GetKey(KeyCode.UpArrow))

{

transform.Translate(Vector3.forward \* this.speed);

}

else if (Input.GetKey(KeyCode.DownArrow))

{

transform.Translate(Vector3.back \* this.speed);

}

else if (Input.GetKey(KeyCode.Space))

{

transform.Translate(Vector3.up \* this.speed);

}

}\*/

1. **RayCasting:**

* **Ray:** Representation of rays. A ray is an infinite line starting at [origin](https://docs.unity3d.com/ScriptReference/Ray-origin.html) and going in some [direction](https://docs.unity3d.com/ScriptReference/Ray-direction.html).
* **RayCast:**  **bool** Returns true if the ray intersects with a Collider, otherwise false. Casts a ray, from point origin, in direction direction, of length maxDistance, against all colliders in the Scene.