Introduction To Unity

(SENG 463 - **Game Programming**)

Dr.Çağatay ÜNDEĞER

Research and Innovation Director

SimBT Inc.

e-mail :

cagatay.undeger@simbt.com.tr

cagatay@undeger.com

Outline

- What is Unity
- Installation and Setup
- Create / Open A Project
- Unity Editor

What is Unity

- Unity is a cross-platform game engine
- Released by Unity Technologies, in 2005.
- The focus of Unity lies in the development of both 2D and 3D games and interactive content.
- Unity now supports over 20 different target platforms for deployment,
- Its most popular platforms are the PC, Android and iOS systems.

What is Unity

- Unity features a complete toolkit for designing and building games, including interfaces for
  - Graphics,
  - Audio, and
  - Level-building tools,
- Requiring minimal use of external programs to work on projects
- Learn Unity from several sources such as:
  - https://learn.unity.com/
  - https://www.tutorialspoint.com/unity/index.htm

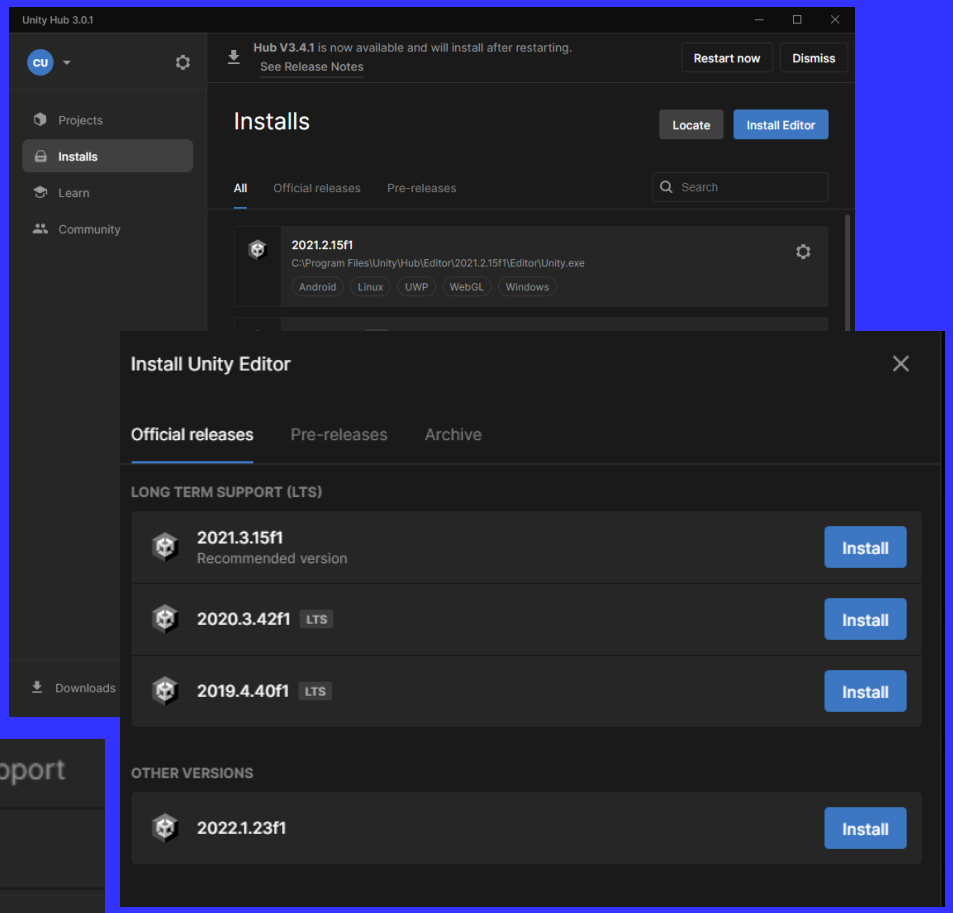
Installation and Setup

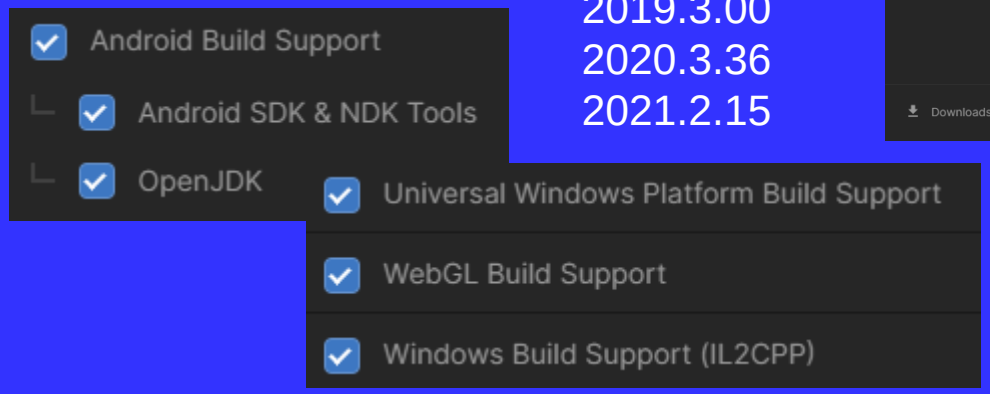
- To use Unity,
  - Download the Unity Hub
  - Than using Unity Hub, download your required Unity Engine version.
  - If not in list, download from archives.
- While setup, along with the core engine,
  - Select optional modules for deploying to various different platforms,
  - As well as tools for integrating Unity scripting into Visual Studio.

Installation and Setup

- From Install Menu
  - Click Install Editor
  - Click Archive
  - Click Download Archive
  - Select Version
  - Click Unity Hub Button

Preferably install:
2019.3.00
2020.3.36
2021.2.15

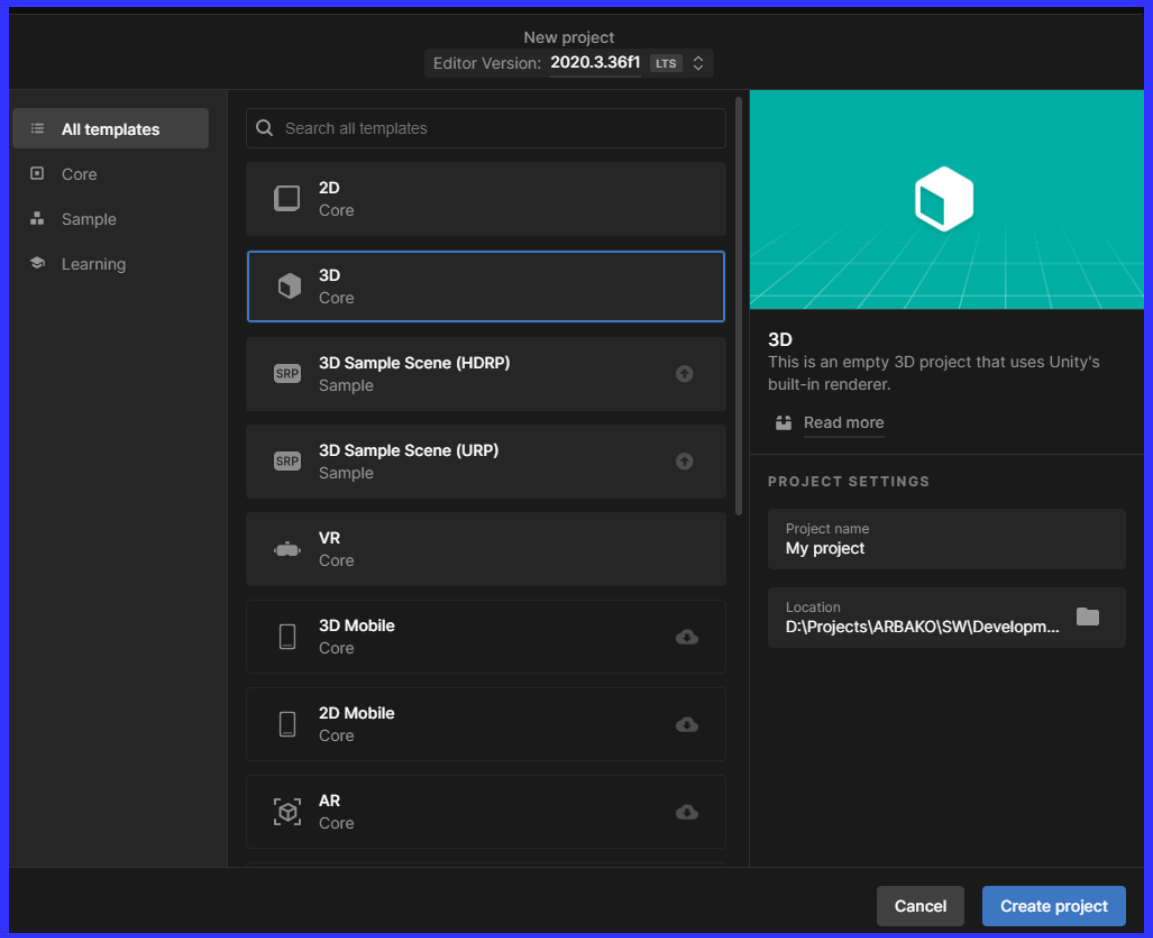




- https://unity.com/releases/editor/archive

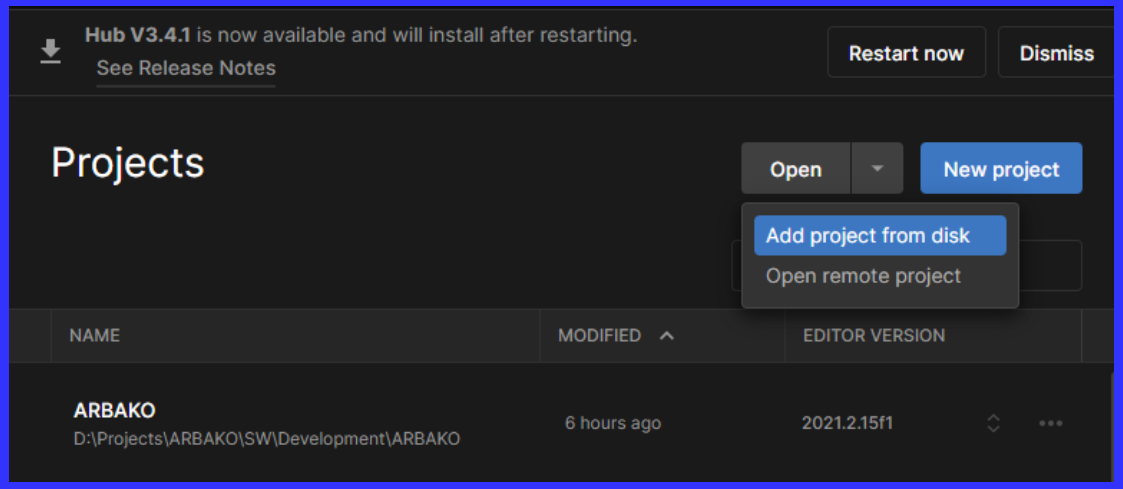
Open a Project

- From Unity Hub
  - Create a new Project



Open a Project

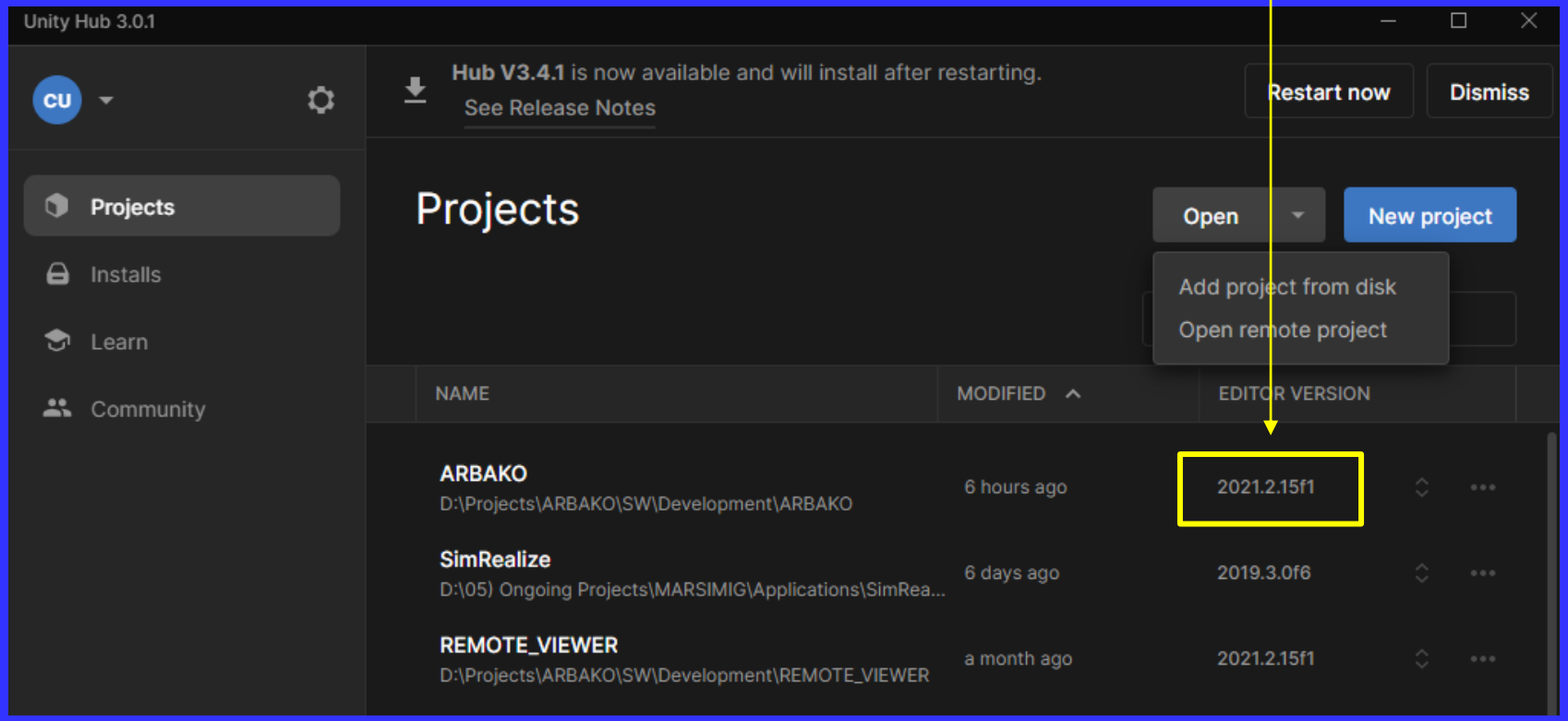
- From Unity Hub
  - Add an existing Project from disk



  - Add / Open a remote Project

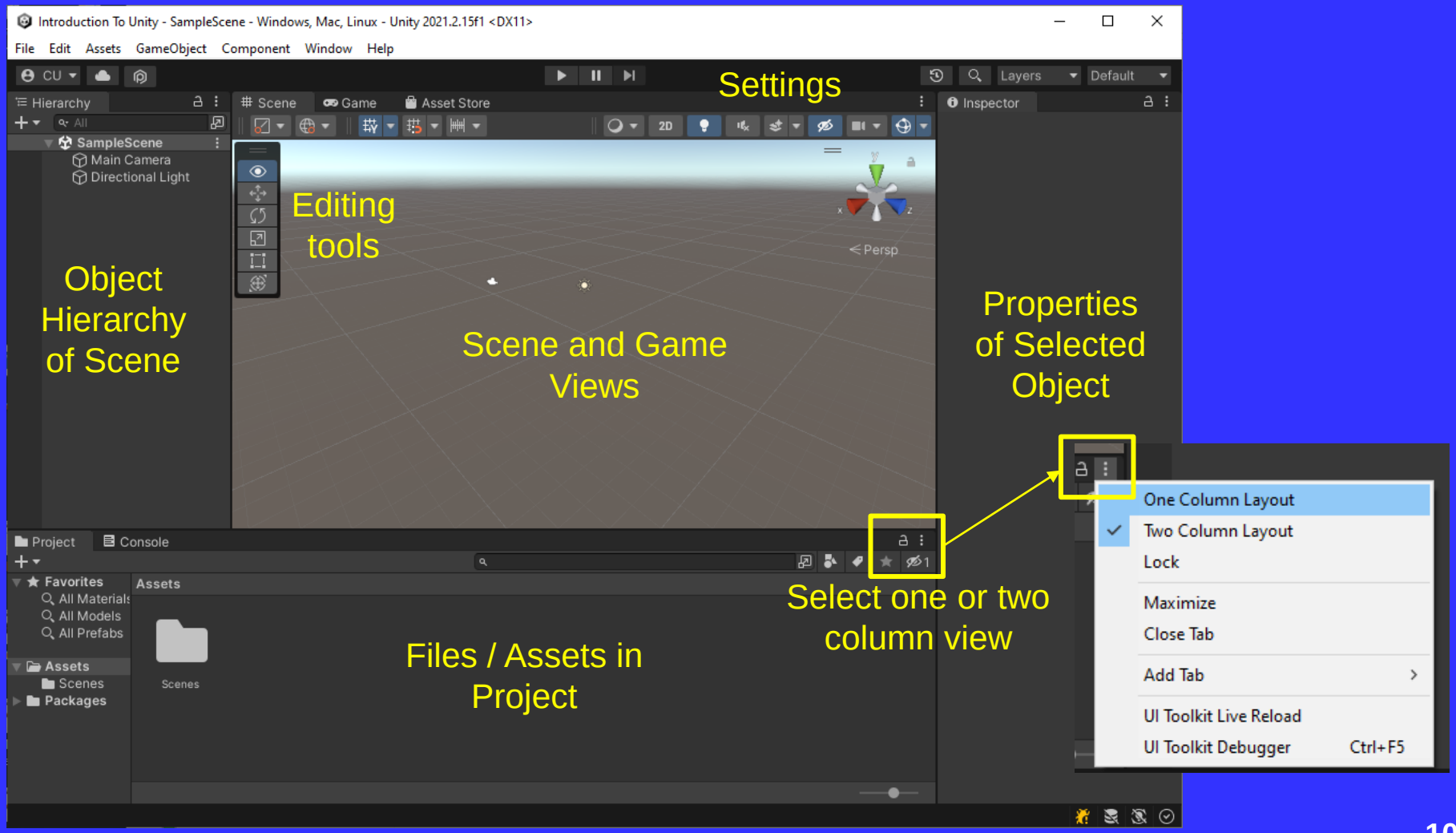
Open a Project

- From Unity Hub
  - Set Version of Unity and
  - Open an existing added project



Editor Window

- Has a customizable editor window



Window Layout

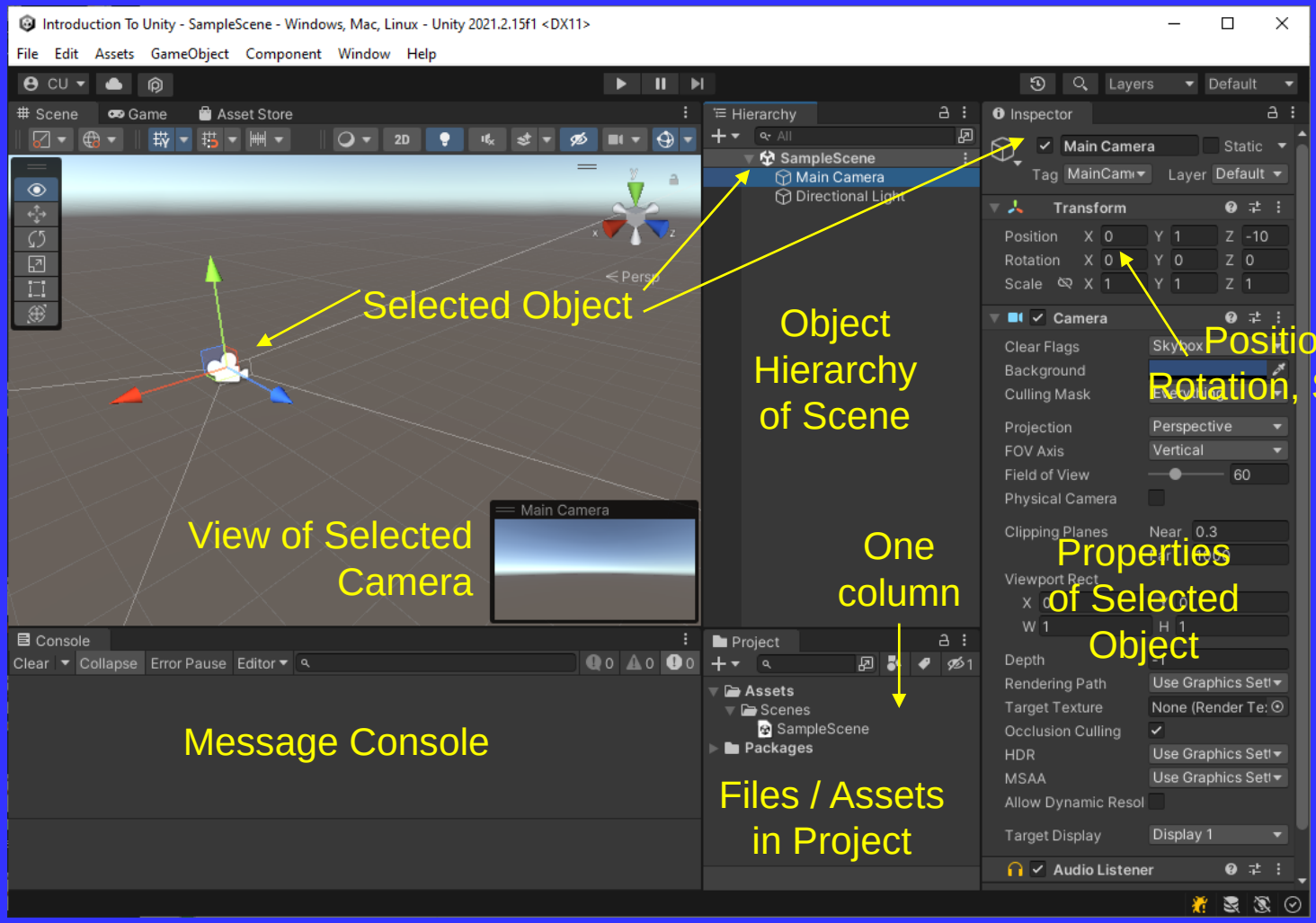
- It is up to you, but I prefer this layout

Left Mouse Select GameObject

Right Mouse Rotate Scene

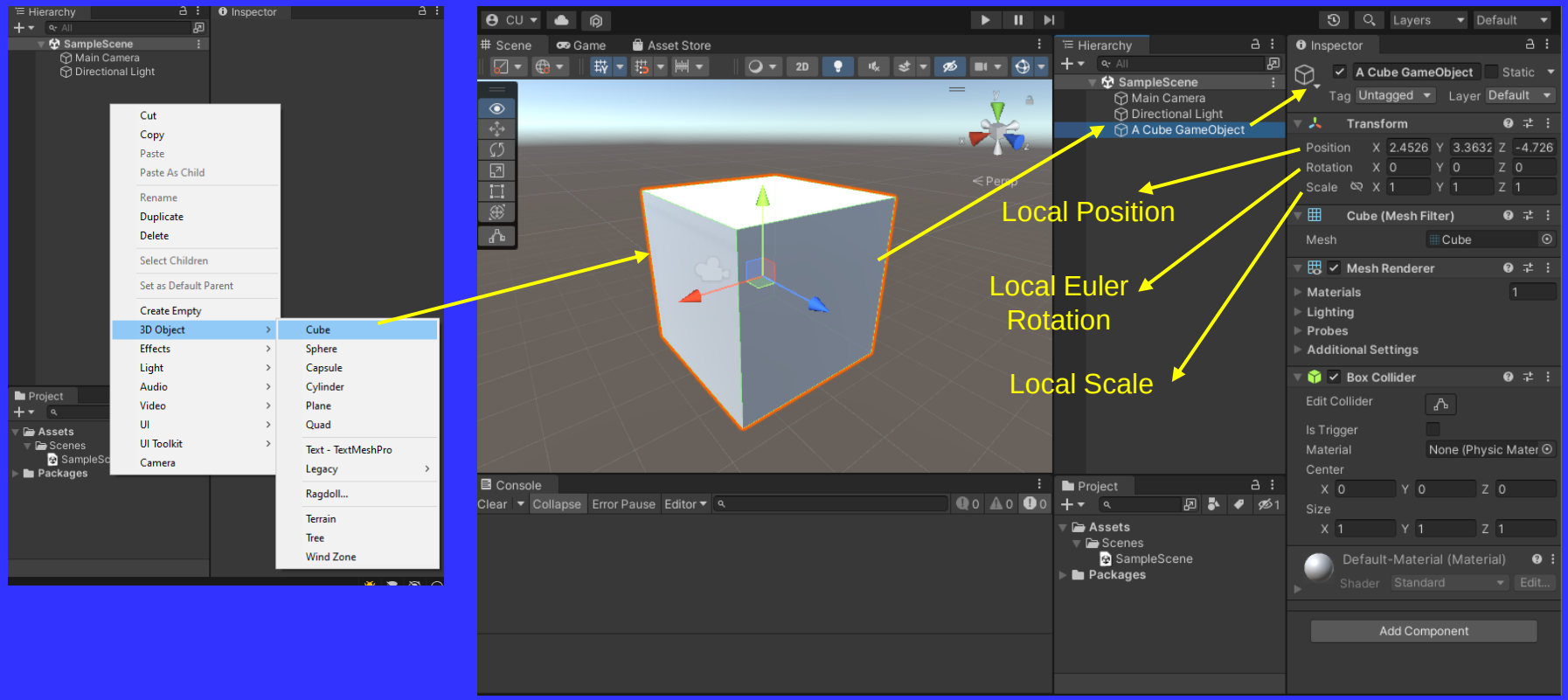
Middle Mouse Pan Scene

Mouse Wheel Forward / Backward



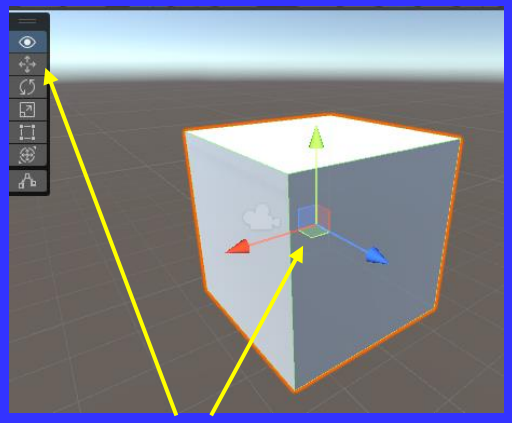
Game Objects

- Create a game object either empty or with a predefined type

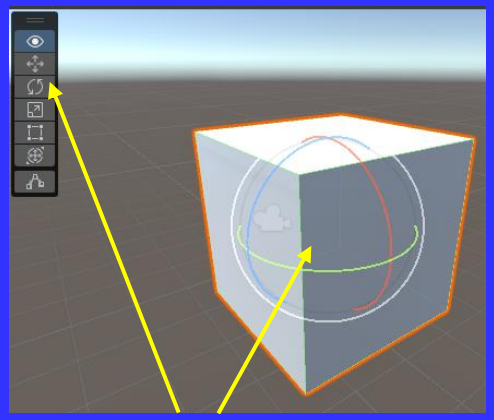


Game Objects

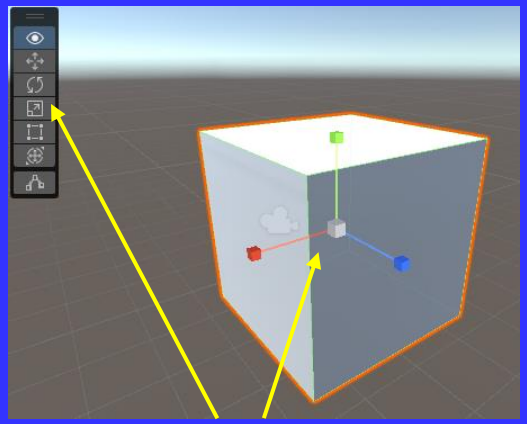
- Edit a game object



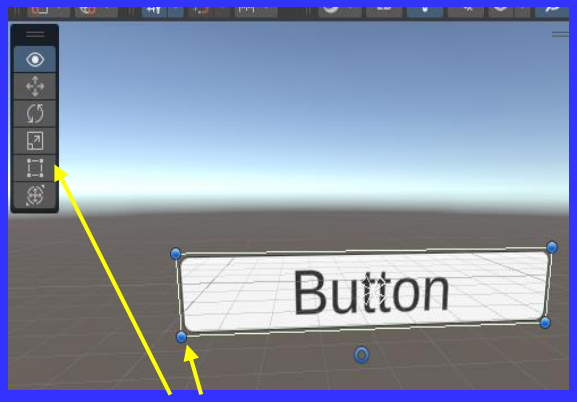
Move Tool



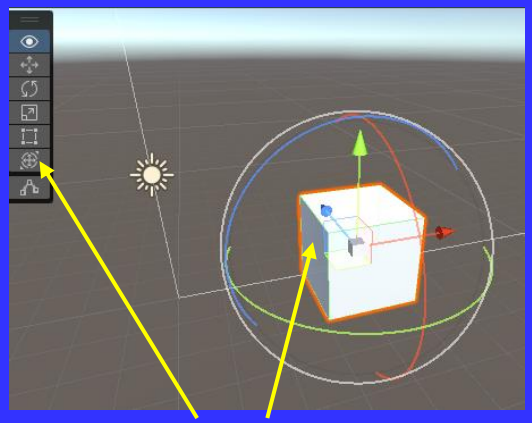
Rotate Tool



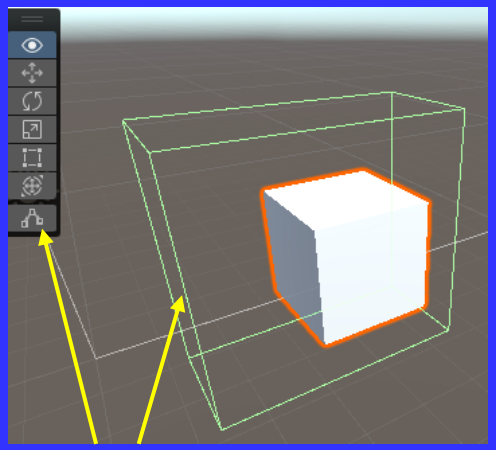
Scale Tool



Rect Tool



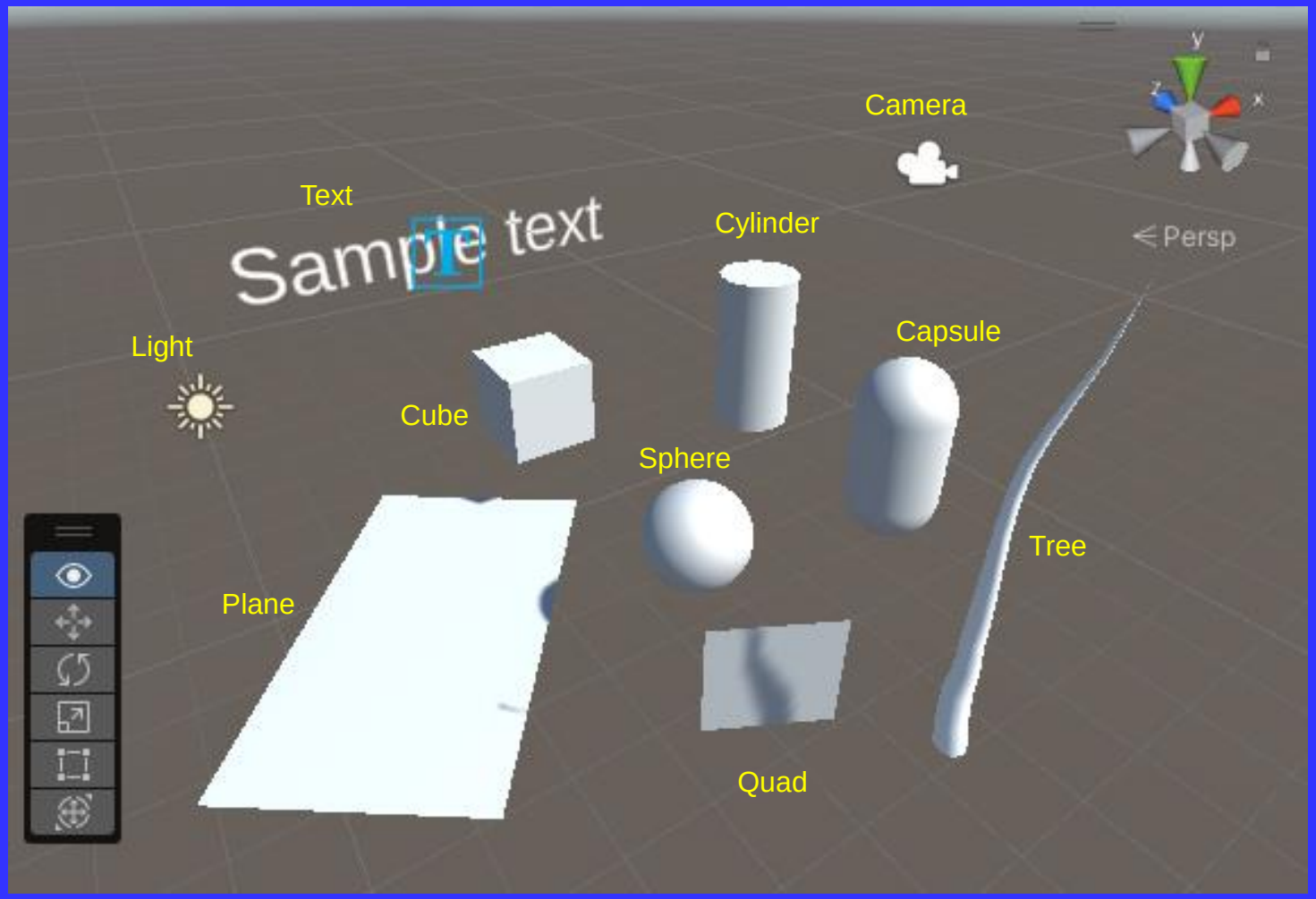
Transform Tool



Bounding Volume Tool

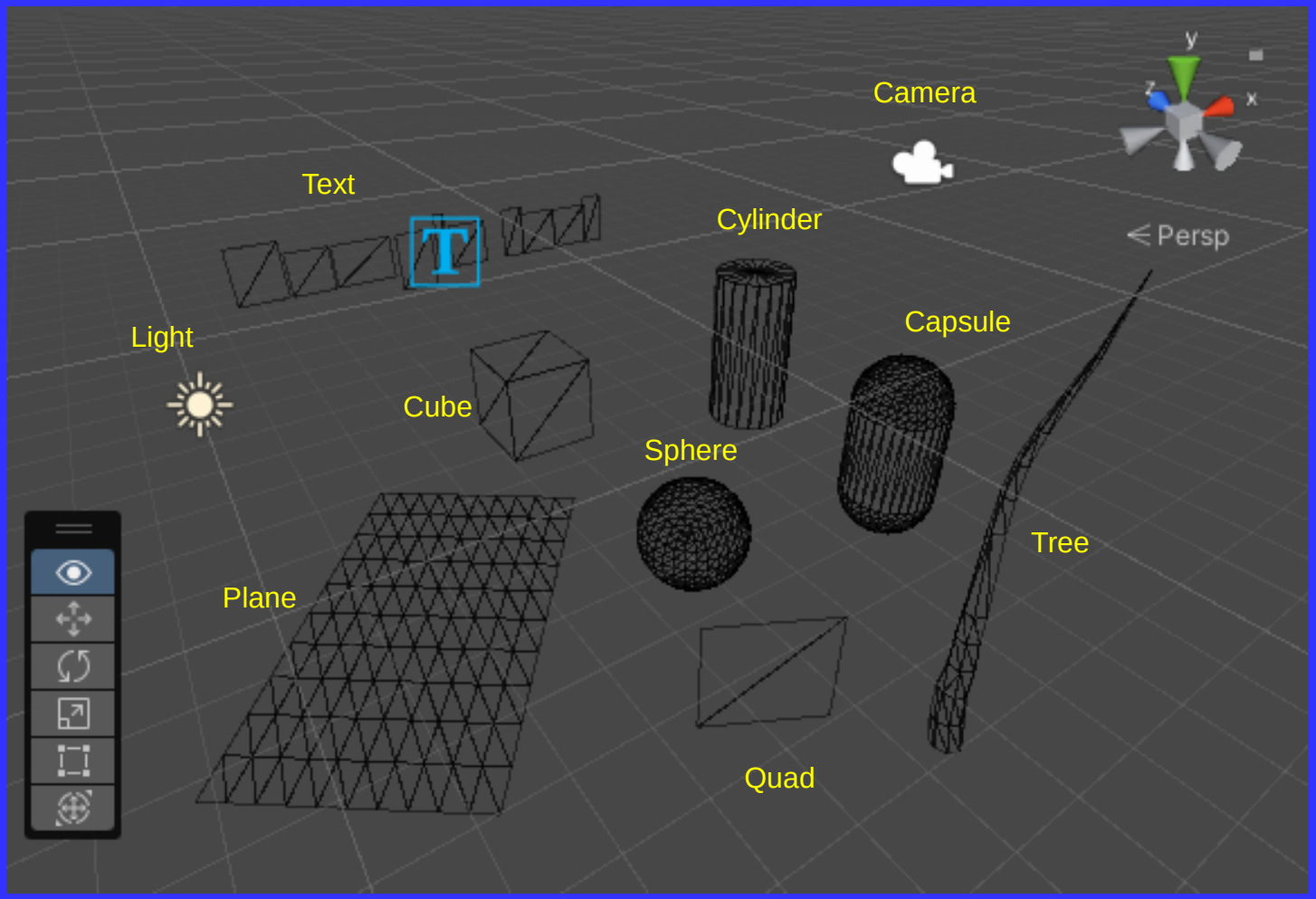
Game Objects

- Some predefined game objects (shaded)



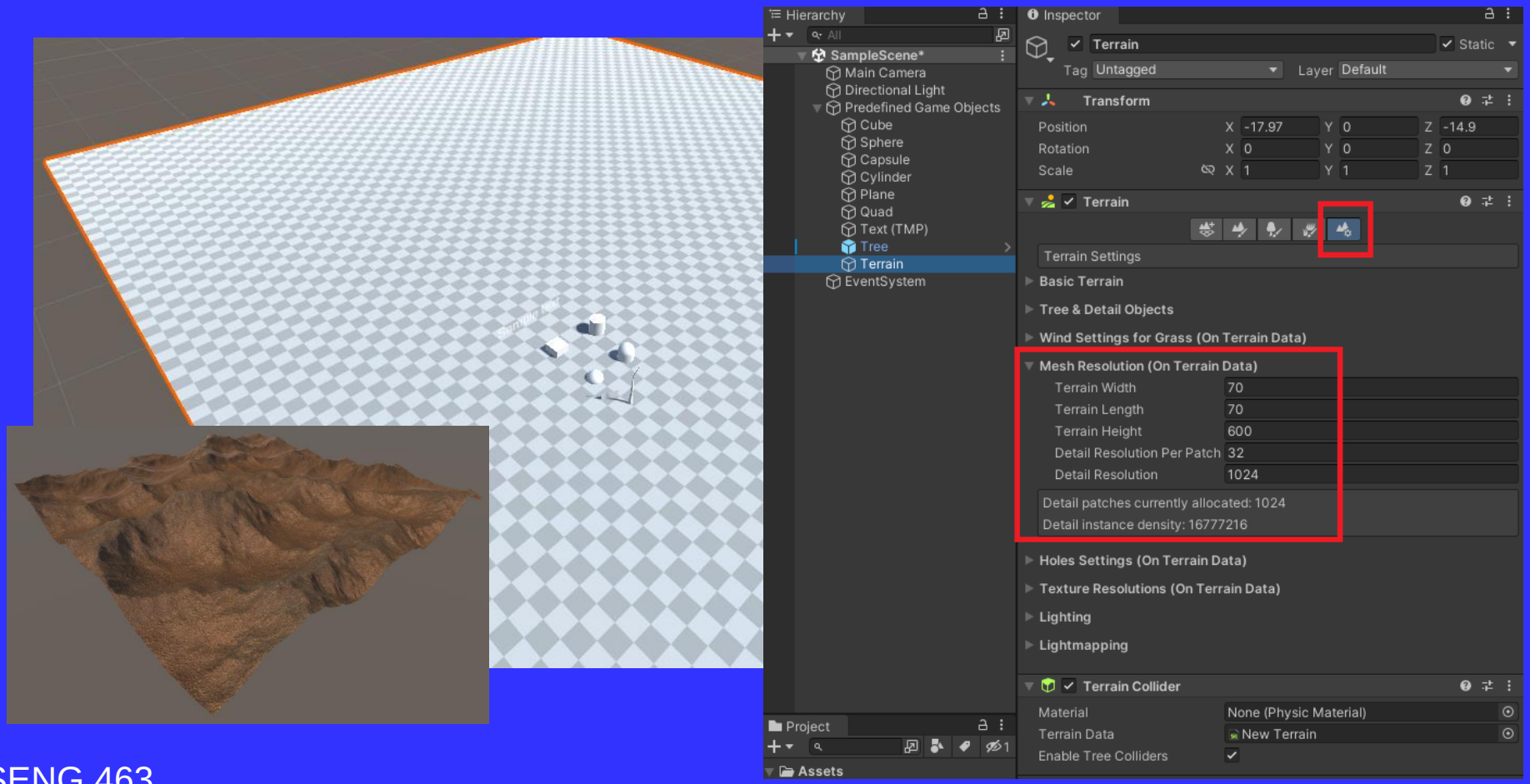
Game Objects

- Some predefined game objects (wireframe)

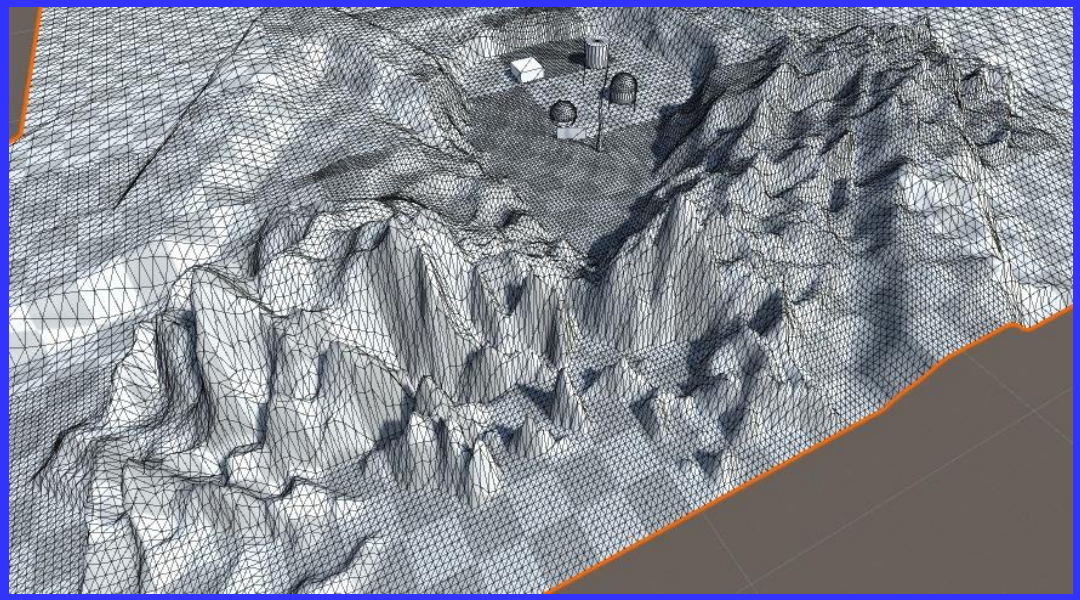


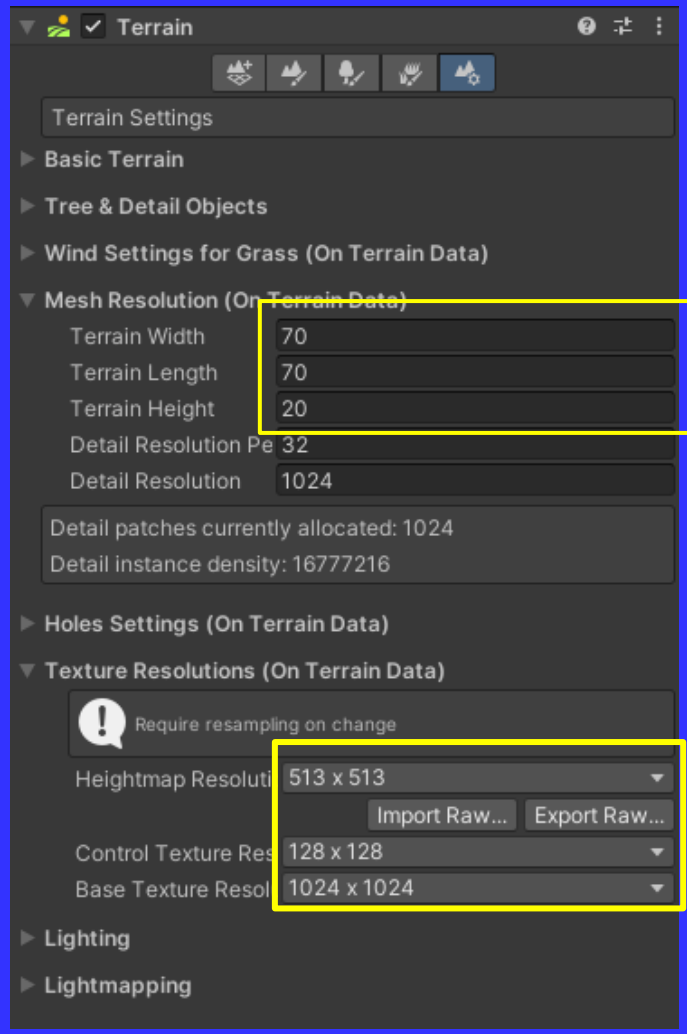
Terrains

- Terrains can be created using height fields
- Height fields: 2D matrix (grid) of elevations



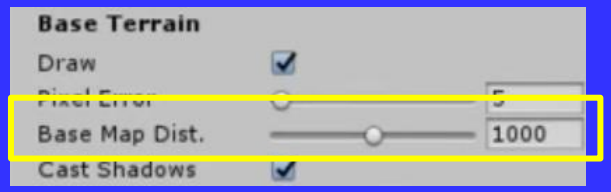
Terrain Topography





Size of the terrain object in its X and Z axis (in world units)

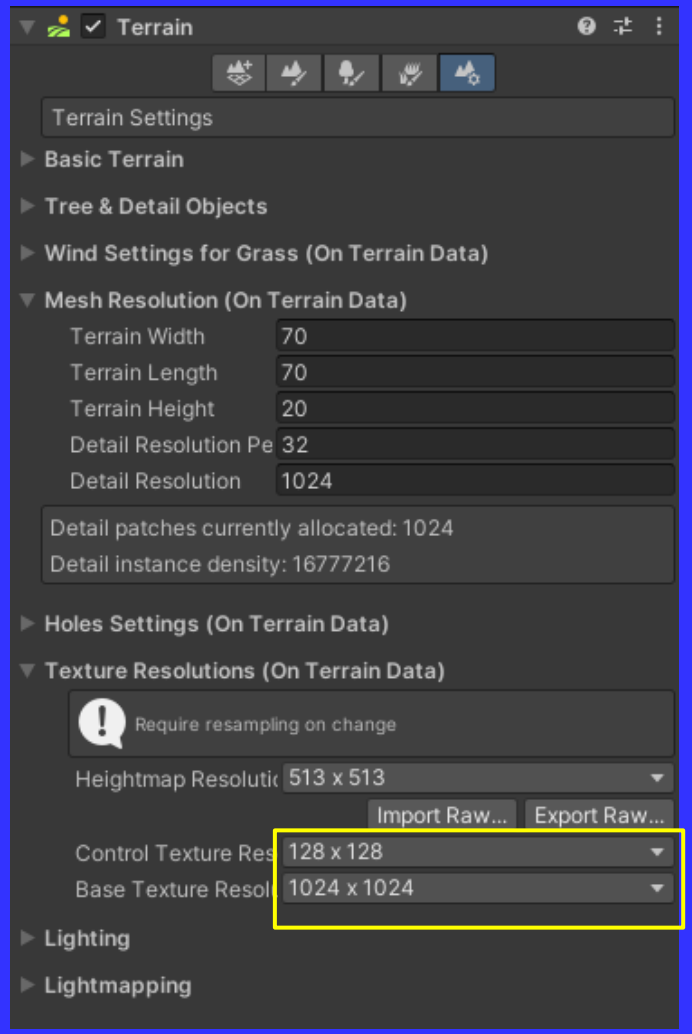
Difference in Y coordinate between the lowest possible heightmap value & the highest

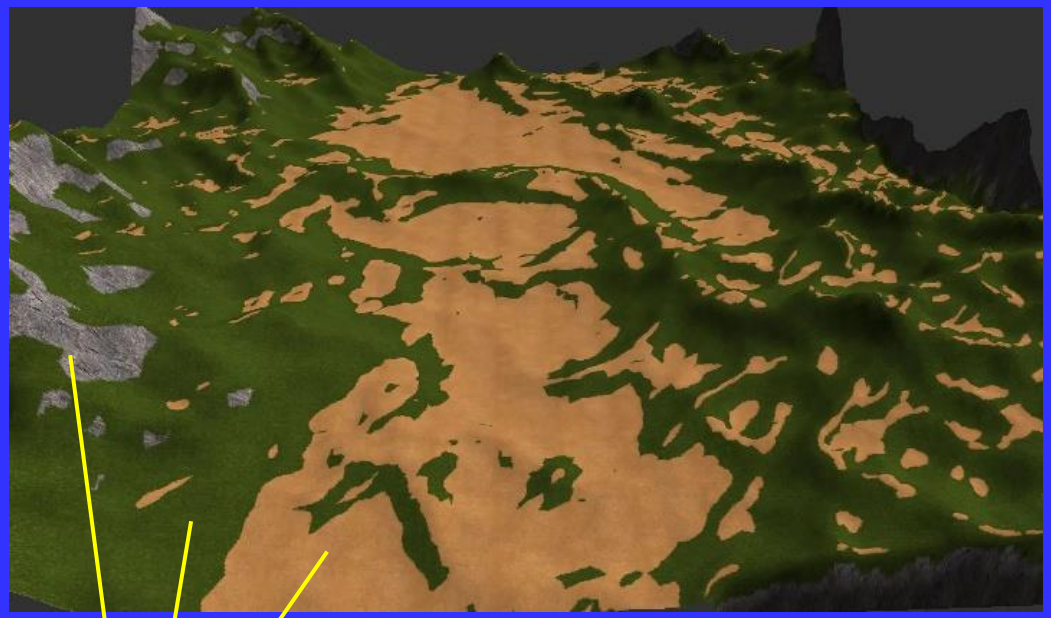


Pixel resolution of the terrain's heightmap (should be a power of 2 plus 1 >> 513 = 512 + 1)

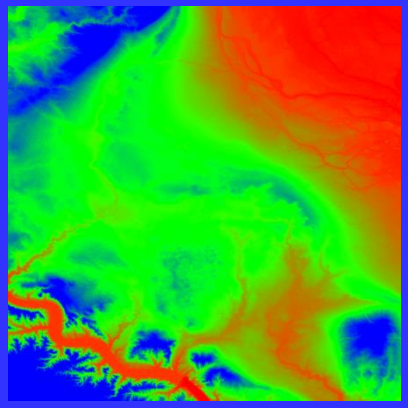
Resolution of the composite texture used on the terrain when viewed from a distance greater than the *Basemap Distance*

Terrain Splats



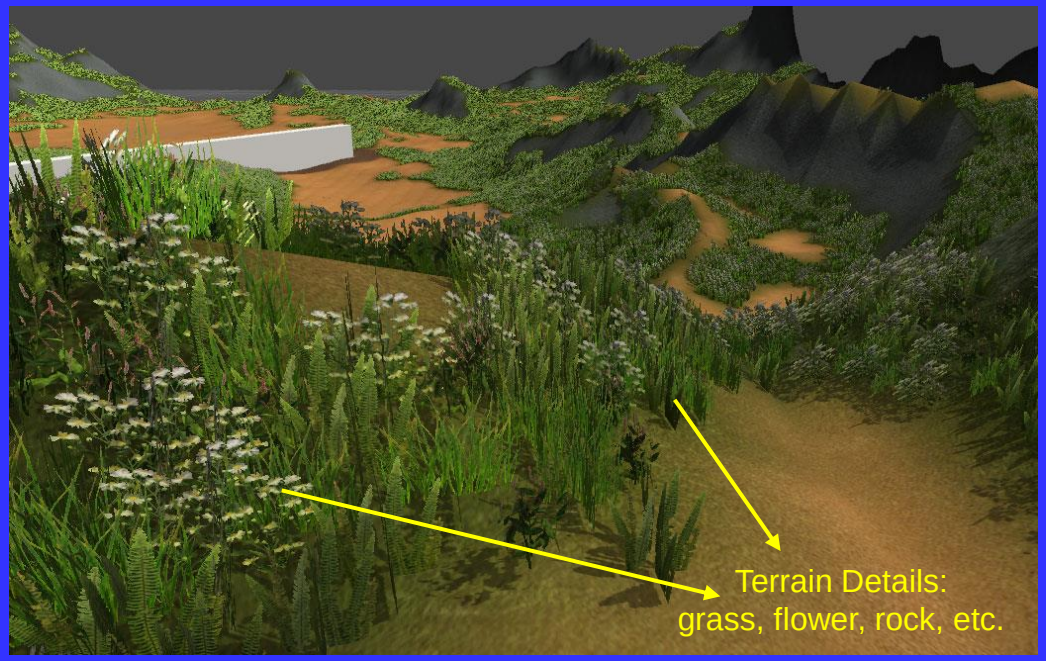


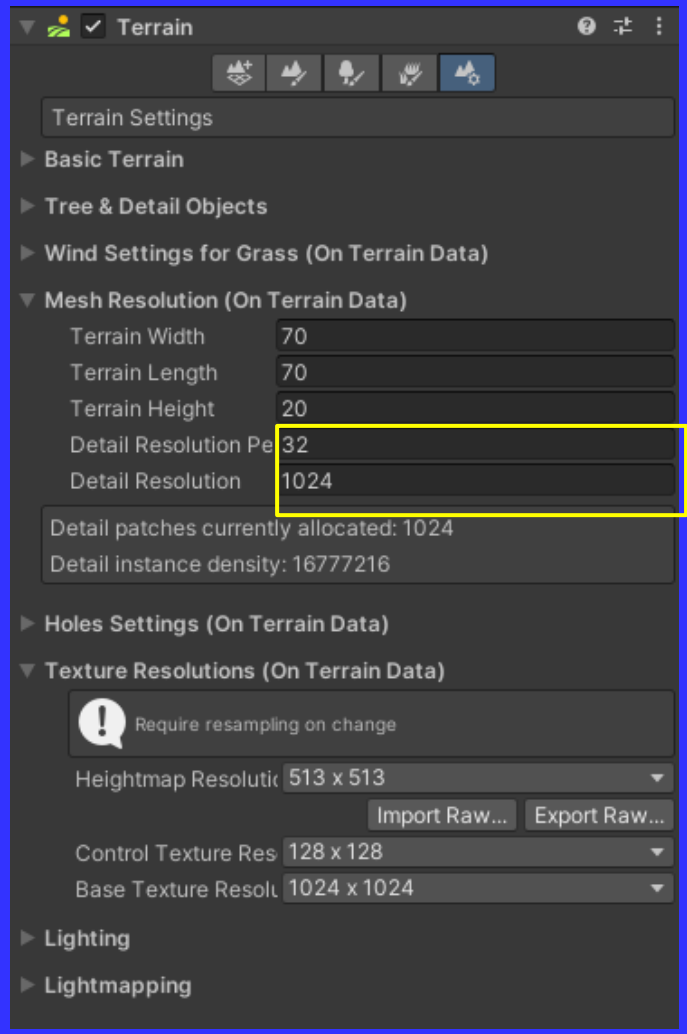
Tiled Splat Textures



Resolution of the “splatmap” that controls the blending of the different terrain textures

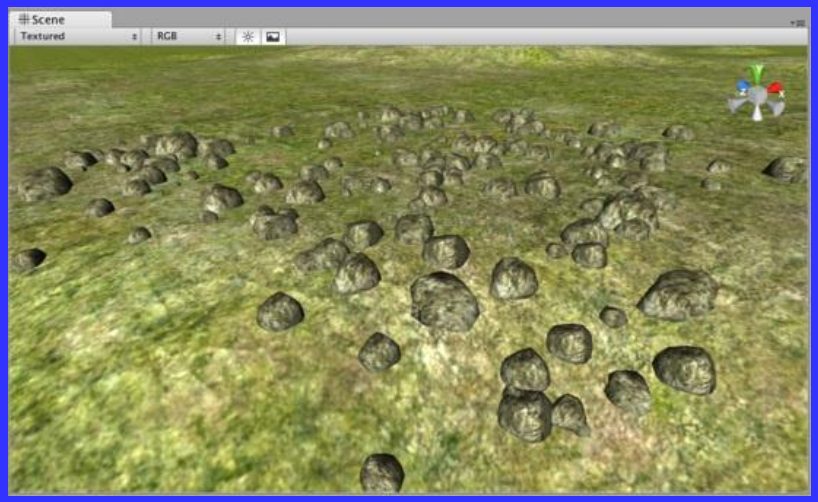
Terrain Details





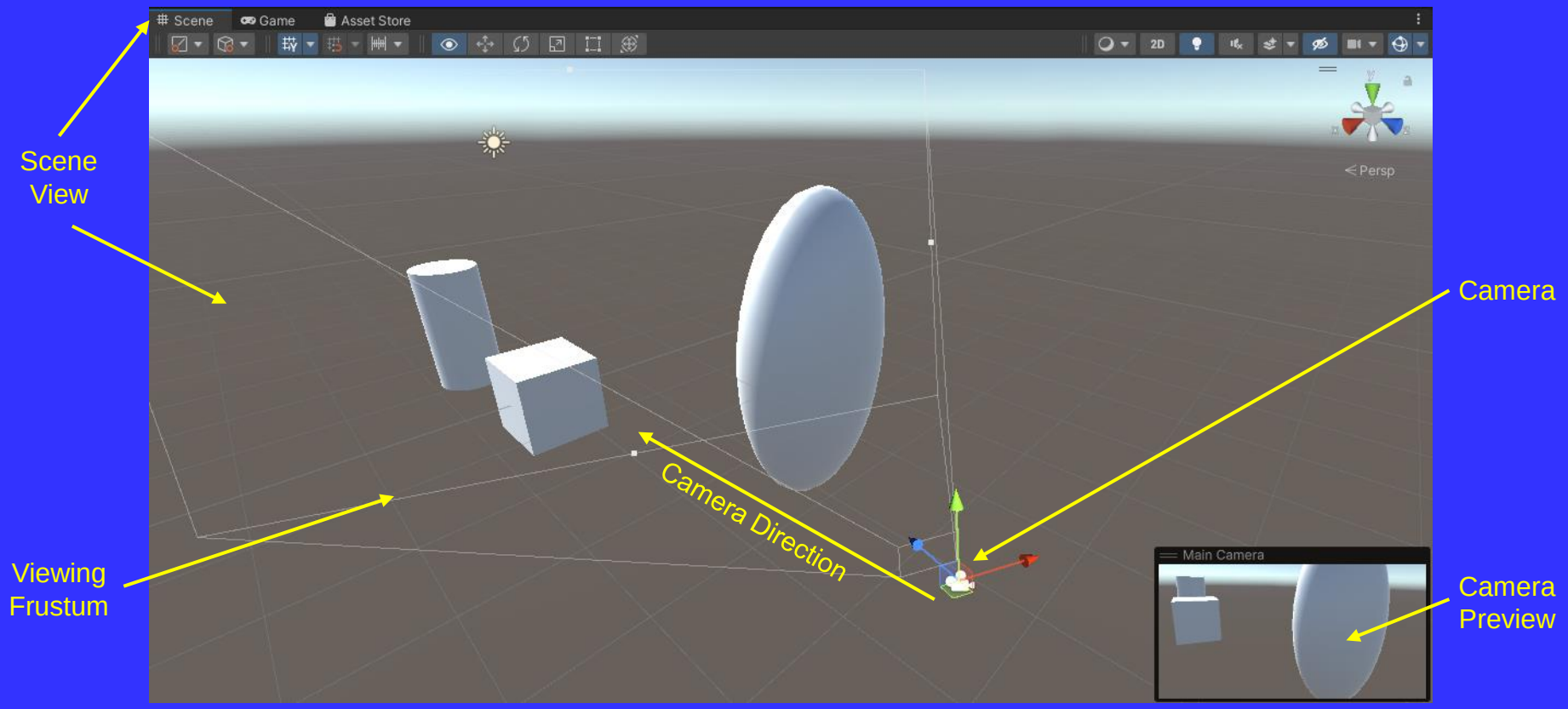
Length/width of the square of patches renderered with a single draw call

Resolution of the map that determines the separate patches of details/grass. Higher resolution gives smaller and more detailed patches



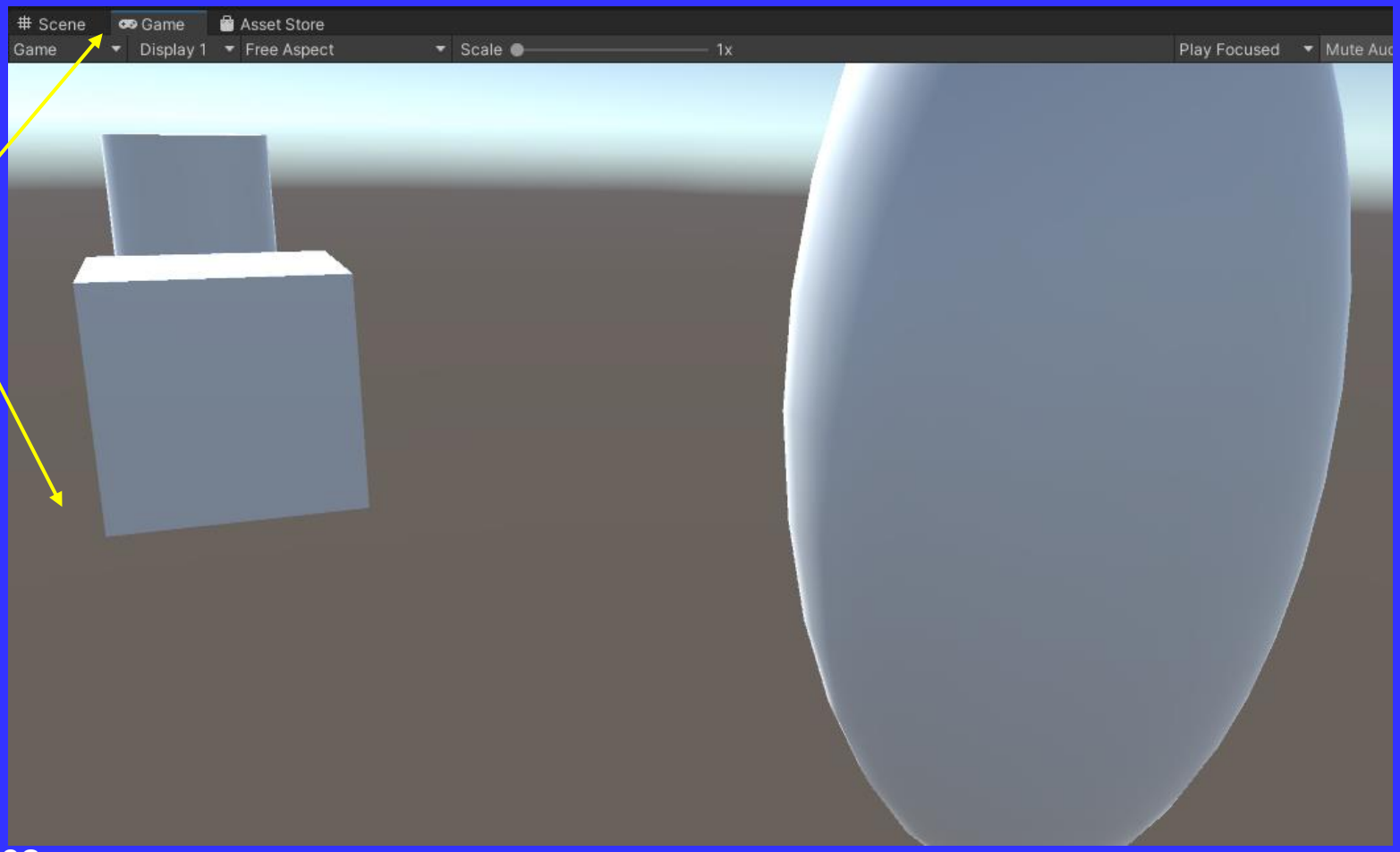
Camera

- A Camera is used to render/visualize a 3D scene on the window



Camera

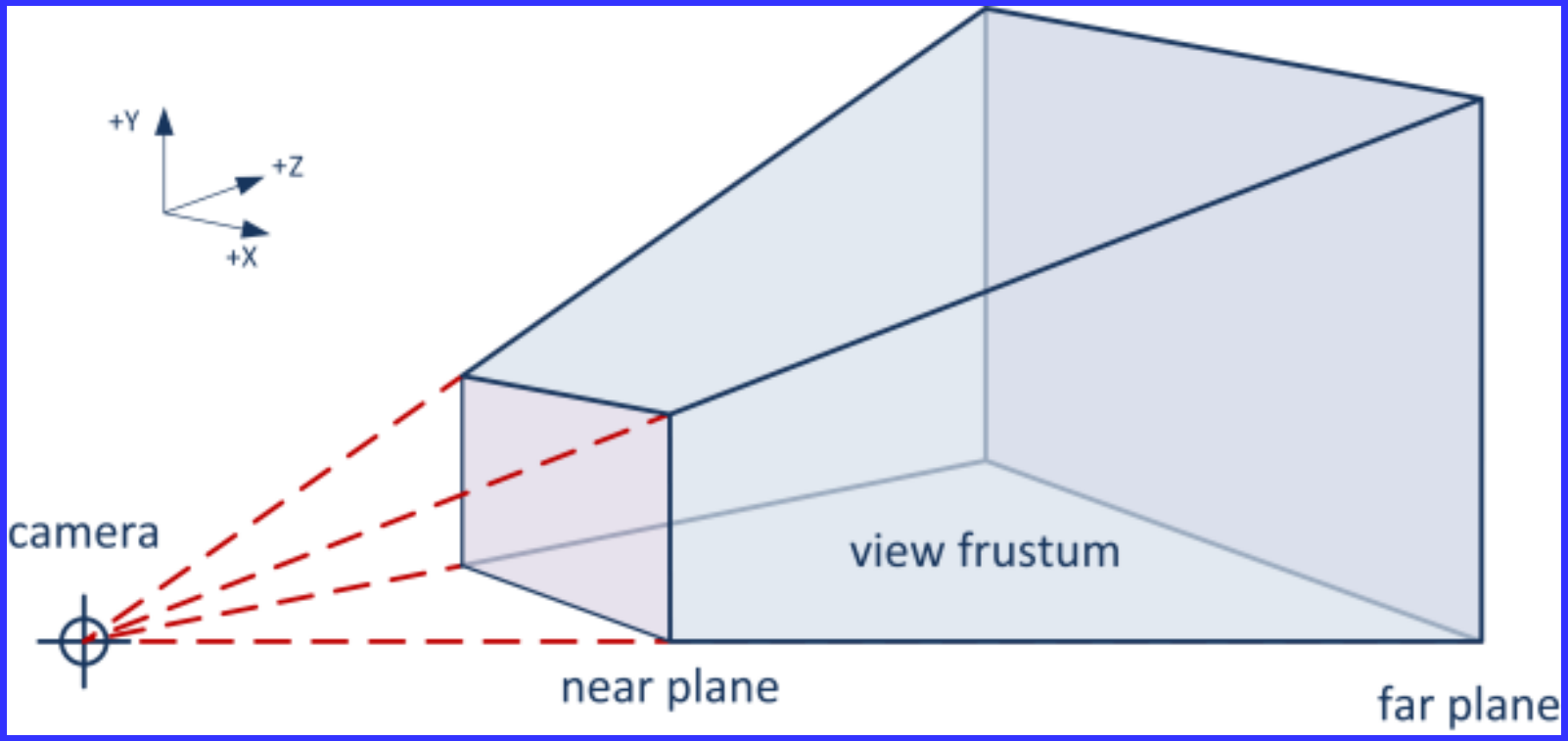
- Inside the game you look into the scene from the camera position and to camera direction



Game View

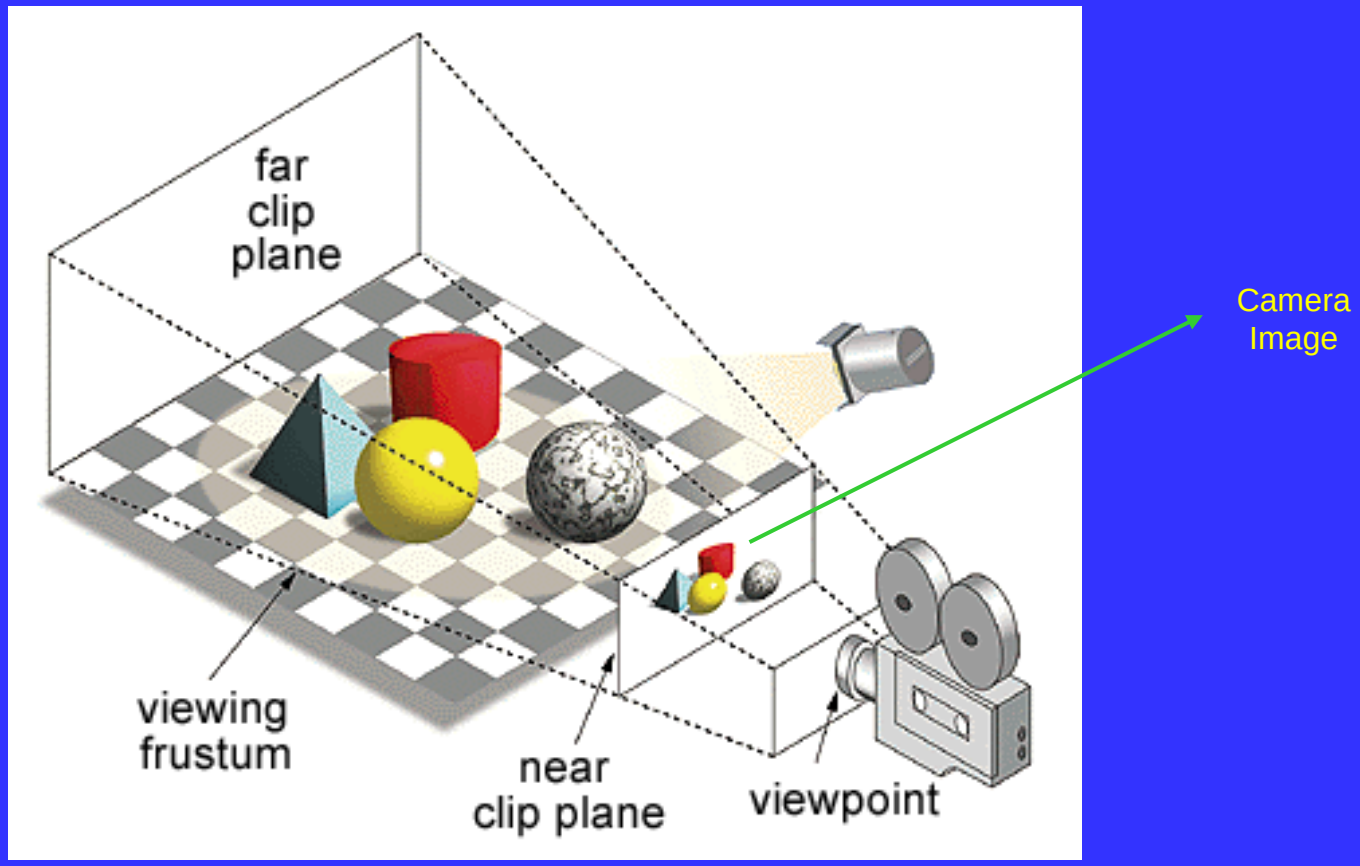
Camera Viewing Frustum

- Viewing frustum is a cropped piramid like shape showing the maximum volume a camera can potantially show on screen



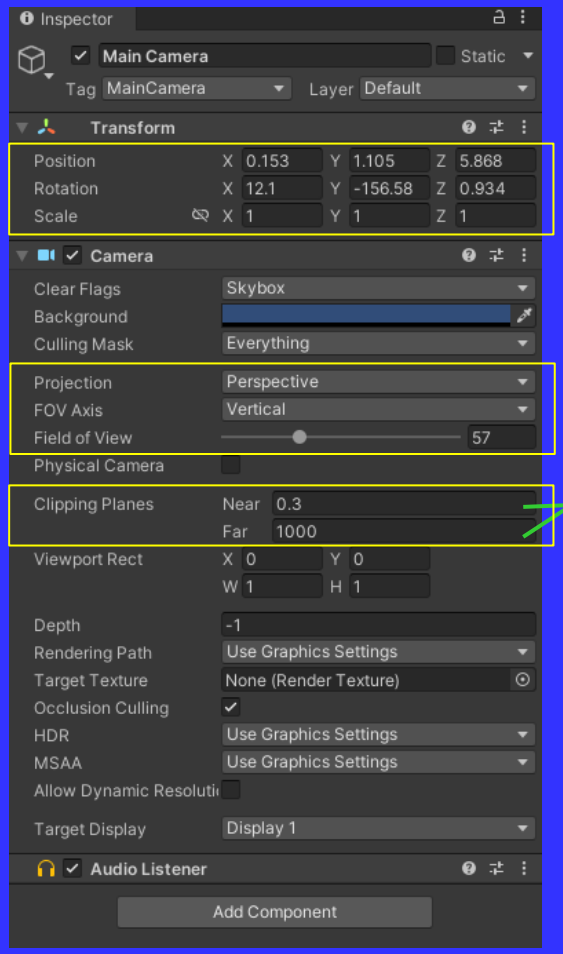
Camera Viewing Frustum

- The volume inside the viewing frustum is projected on the camera image and displayed using perspective projection techniques

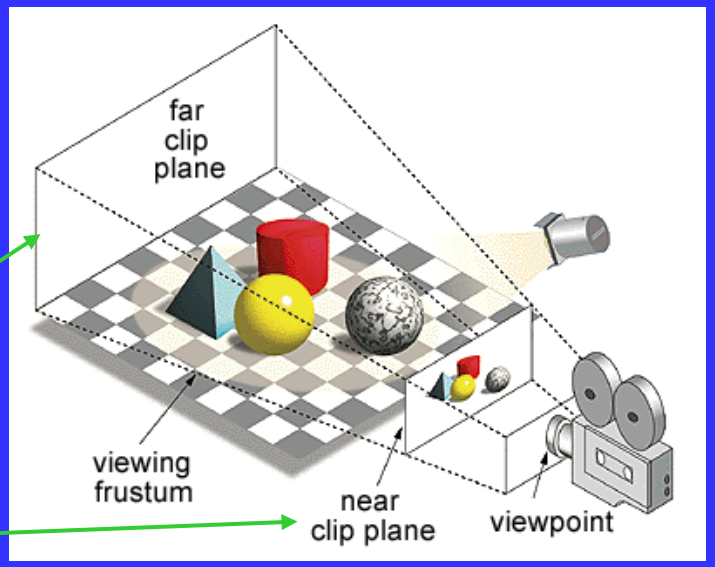


Camera Parameters

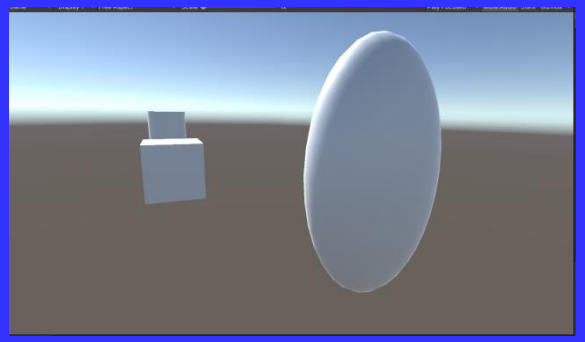
- Camera is adjusted using camera parameters

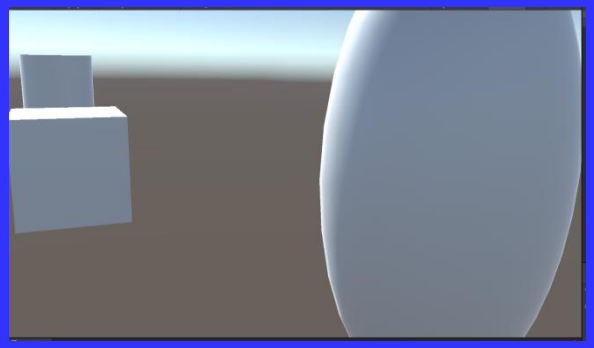


Position and direction of camera



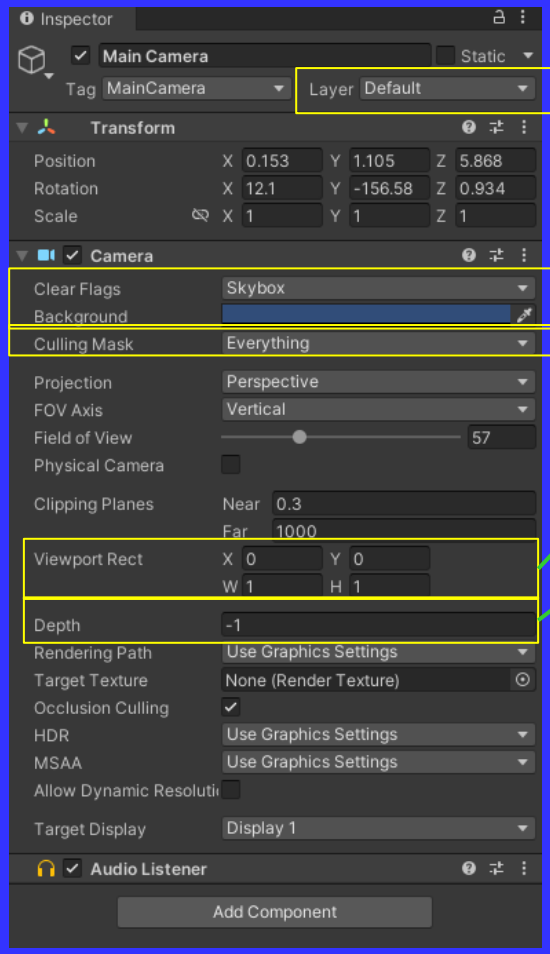
57 degree fov → Zoom in → 32 degree fov





Camera Parameters

- Camera is adjusted using camera parameters

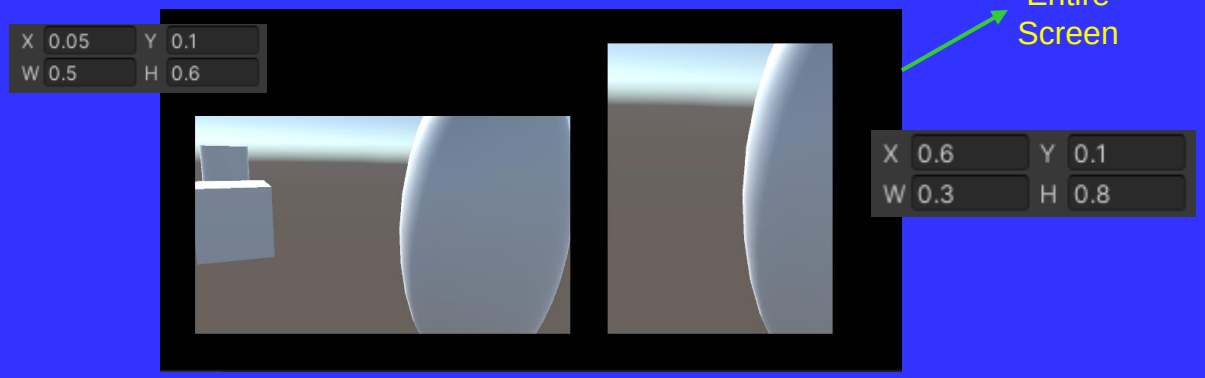


Objects can be assigned to a layer (assigning a category to an object)

Background style of camera (skybox, solid color, depth only, don't clear)
The layers of objects, which camera will see

The region where camera will be drawn on the screen
Values are given as ratio on screen between 0 and 1

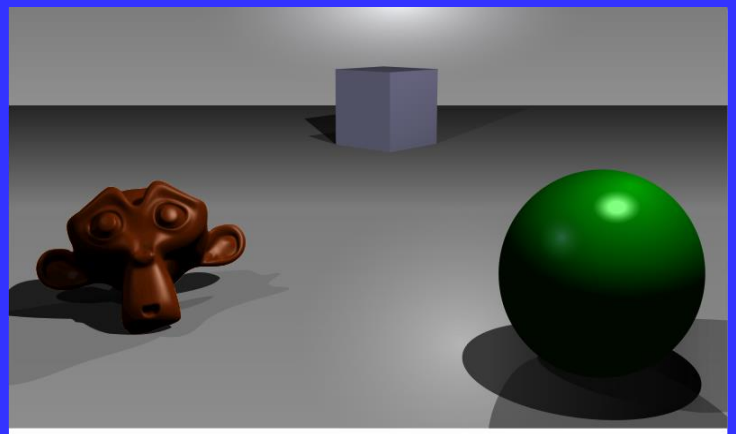
Drawing order of camera (higher is more front, lower is more back)



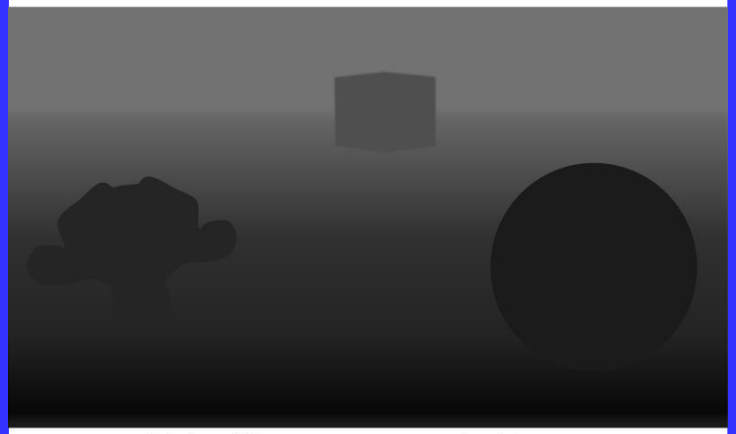
Entire Screen

Camera Depth-Buffer

- A depth buffer, also known as a z-buffer,
- Is a type of data buffer used in computer graphics
- To represent depth information of objects in 3D space from a particular perspective.
- Depth buffers are used for rendering a scene with correct order of polygons.



A simple three-dimensional scene

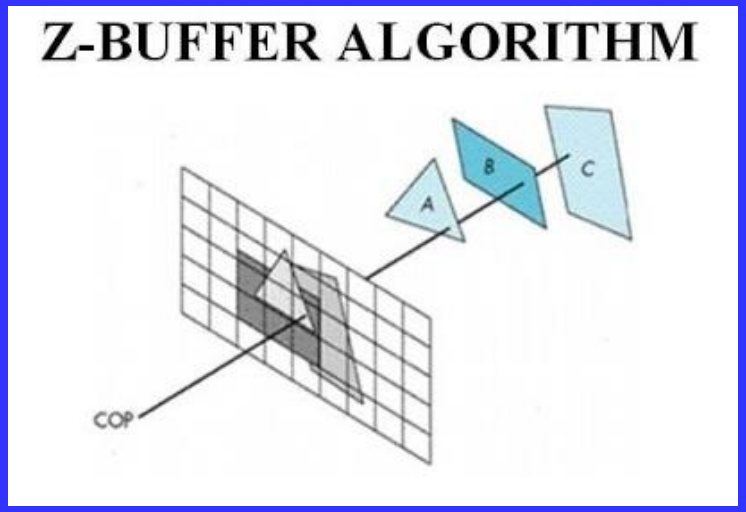


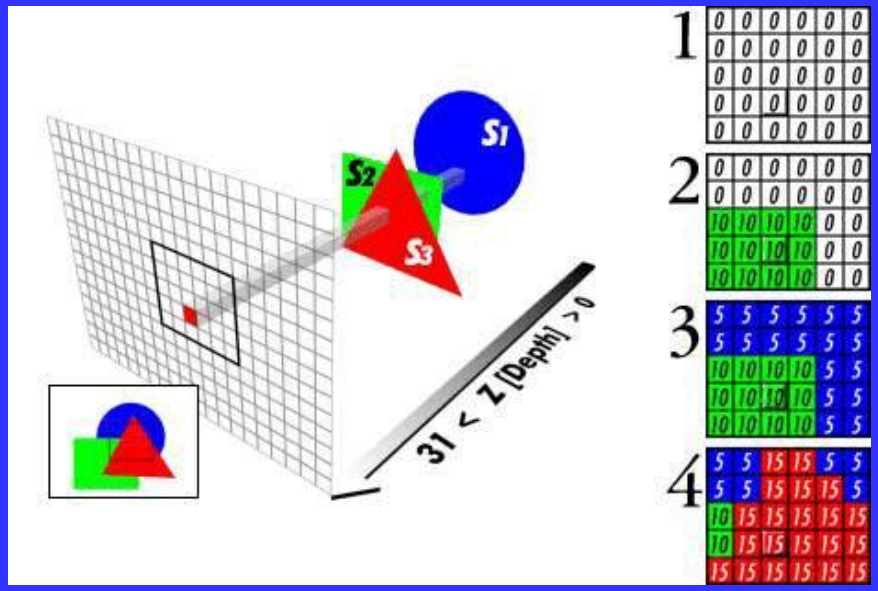
Z-buffer representation

- Clear flag “Depth Only”?
- Override the background, ignore depth

Camera Depth-Buffer

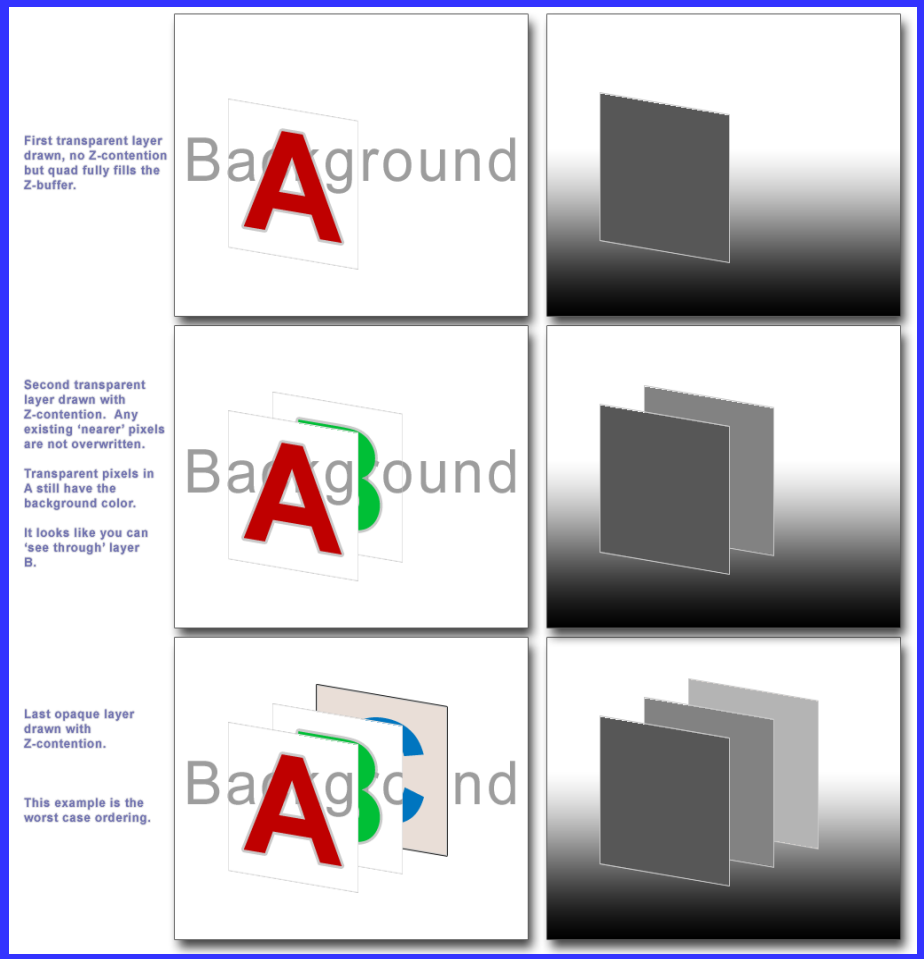
- In a 3D-rendering pipeline,
- When an object is projected on the screen,
- The depth (z-value) of a generated fragment in projected screen image is compared
- To the value already stored in the buffer (**depth test**),
- And replaces it if the new value is closer.

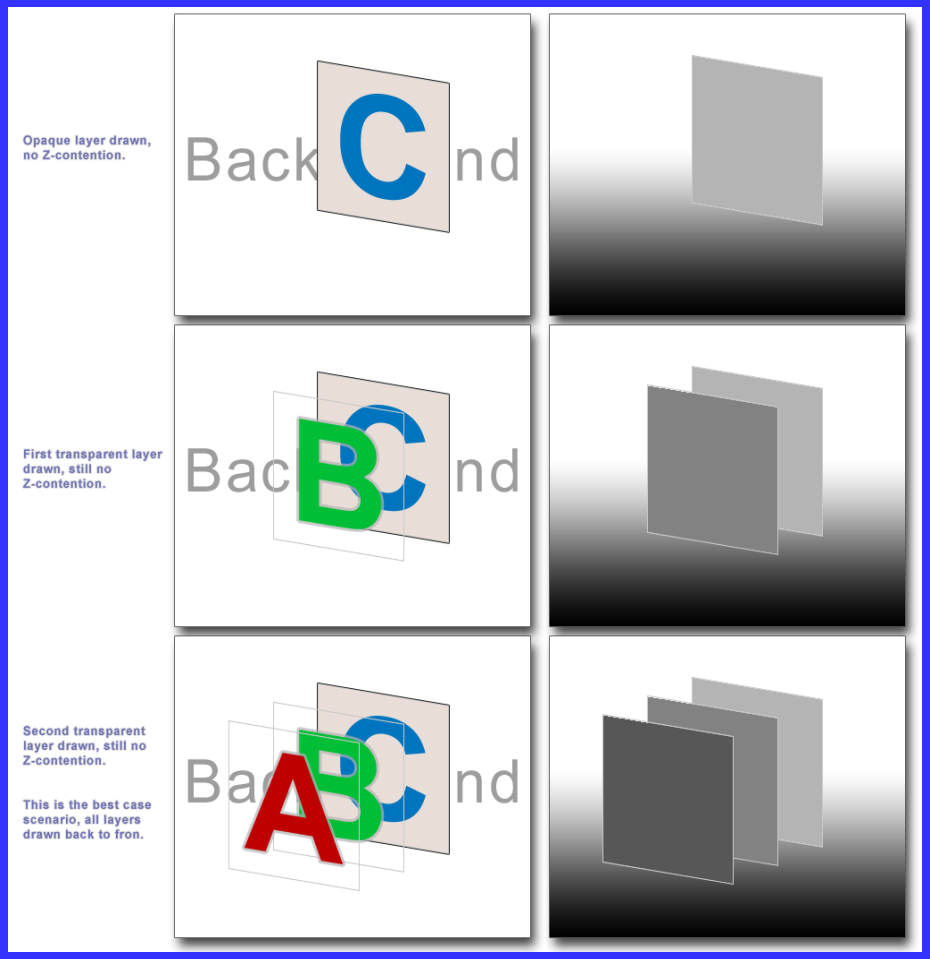




Transparent Object Sorting

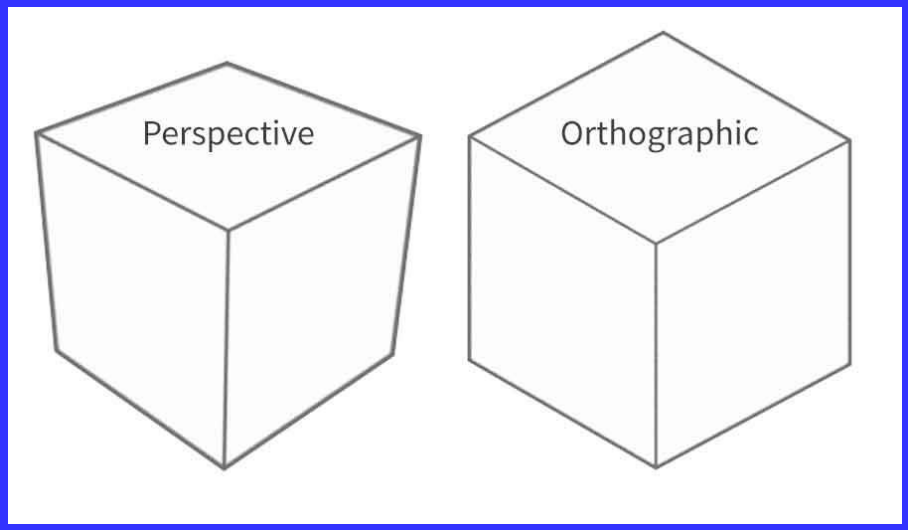
- Transparent objects shall be sorted before rendering

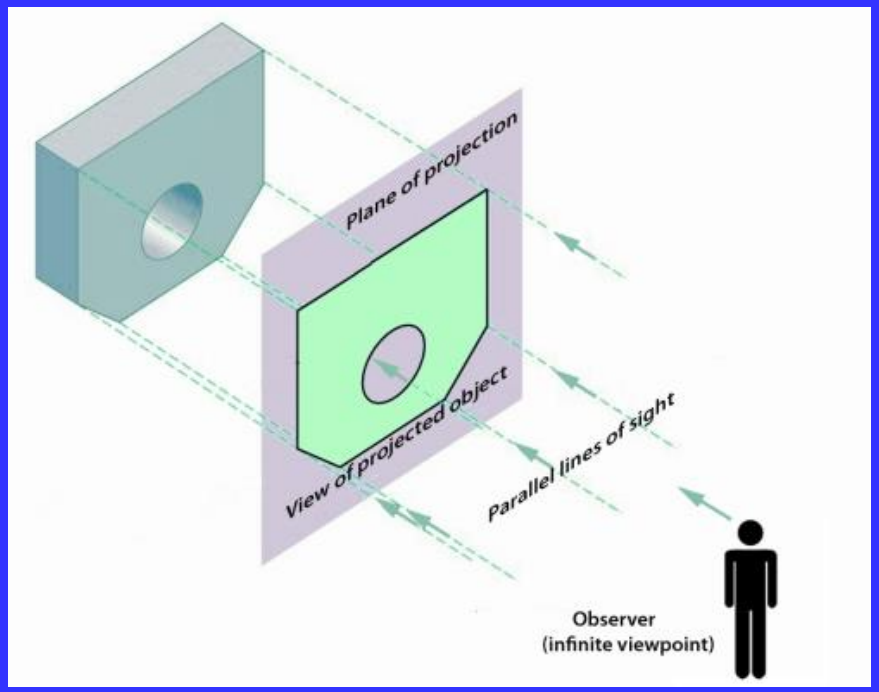




Camera Orthografic Projection

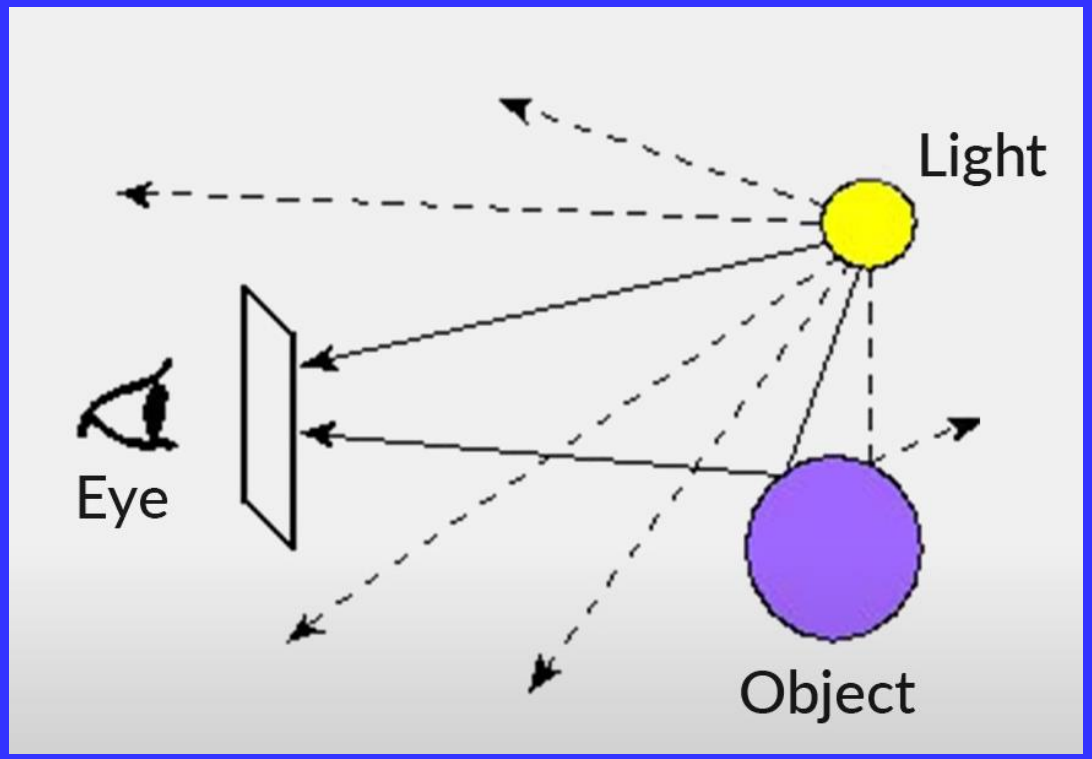
- Orthographic projection is a form of parallel projection in which all the projection lines are orthogonal to the projection plane





Lighting

- Eye see objects by sensing rays of light directly comming from lights or reflected from objects

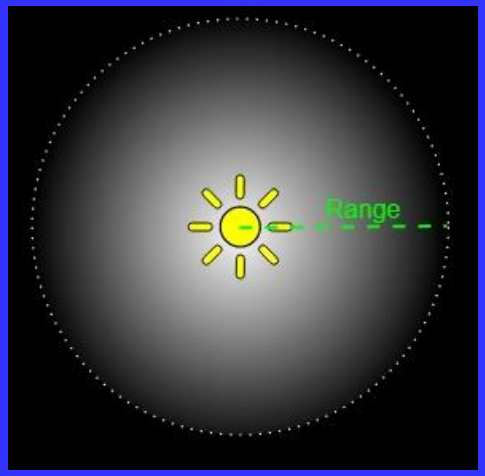


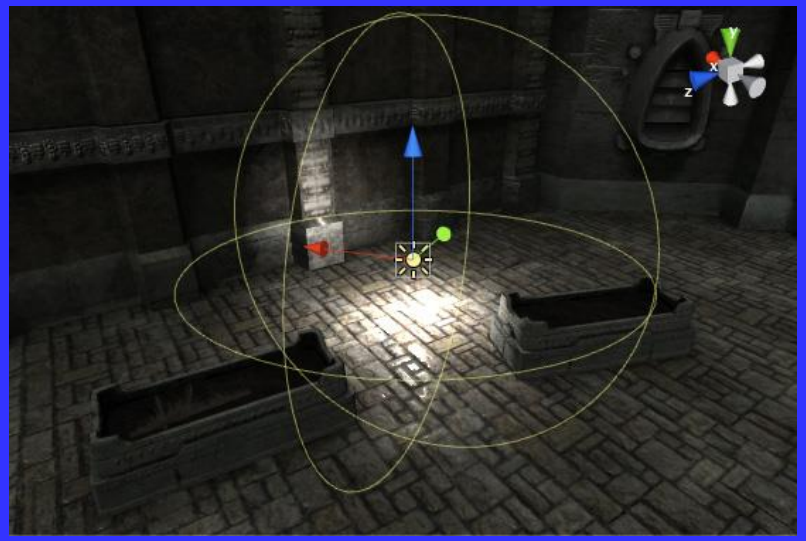
Light Types

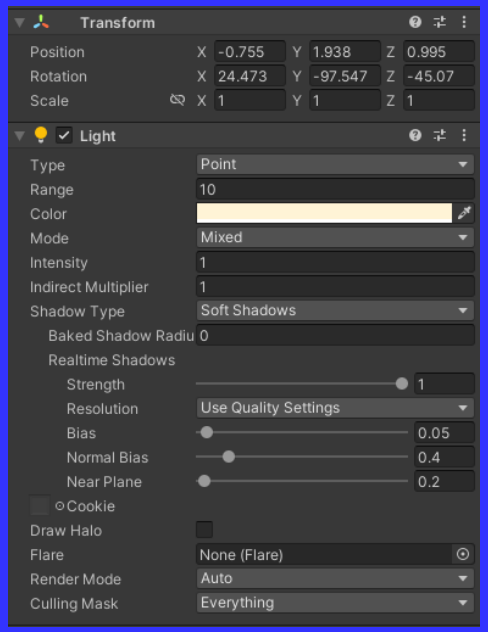
- Point
- Directional
- Spot
- Area (For baked lightmaps only)

Point Lights

- A Point Light is located at a point in space
- Sends light out in all directions equally.
- The direction of light hitting a surface is the line from the point of contact back to the center of the light object.
- The intensity diminishes with distance from the light, reaching zero at a specified range.
- Light intensity is inversely proportional to the square of the distance from the source.

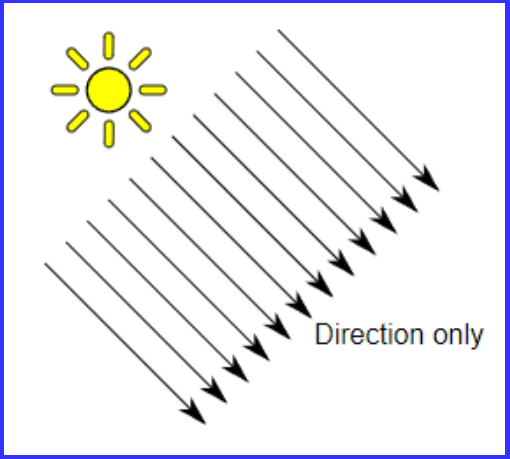


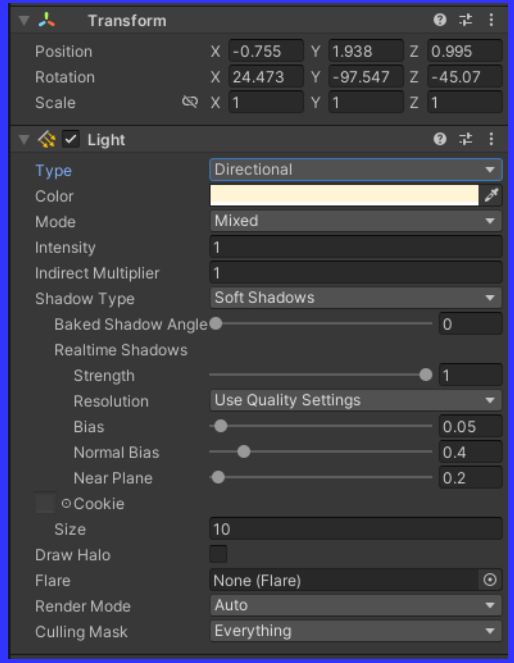




Directional Lights

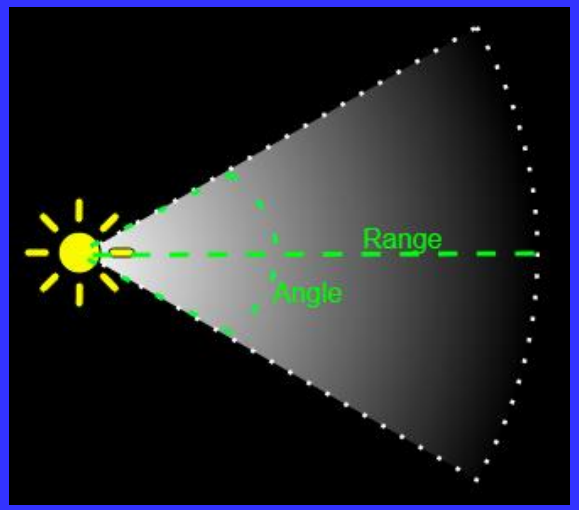
- Usefull for creating effects such as sunlight in your scenes.
- Behaving in many ways like the sun
- Directional lights can be thought of as distant light sources which exist infinitely far away.
- A Directional Light doesn't have any identifiable source position
  - The light object can be placed anywhere in the scene.
- All objects in the scene are illuminated as if the light is always from the same direction.
- The distance of the light from the target object isn't defined and so the light doesn't diminish.

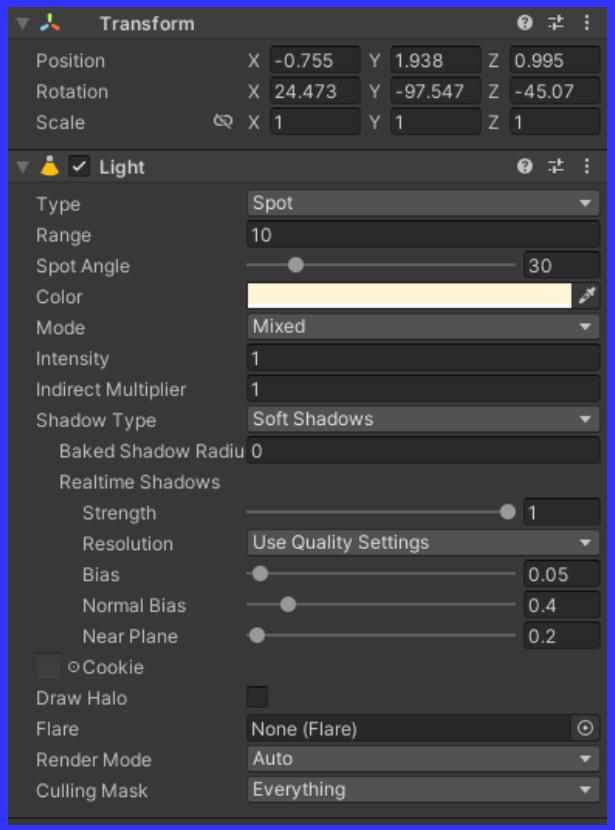




Spot Lights

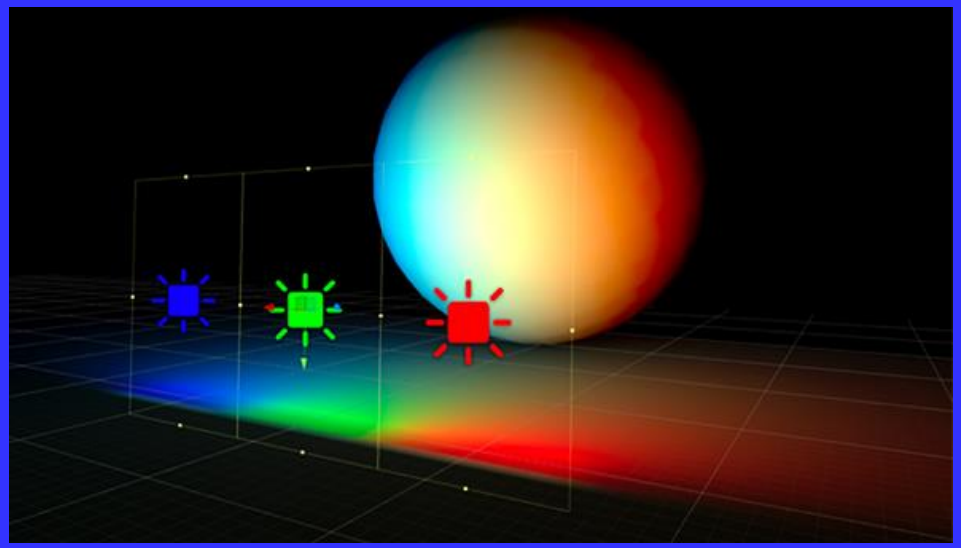
- Like a Point Light, a Spot Light has a specified location and range over which the light falls off.
- A Spot Light is constrained to an angle, resulting in a cone-shaped region of illumination.
- The center of the cone points in the forward (Z) direction of the light object.
- Light also diminishes at the edges of a Spot Light's cone.
- Widening the angle increases the width of the cone and with it increases the size of this fade.

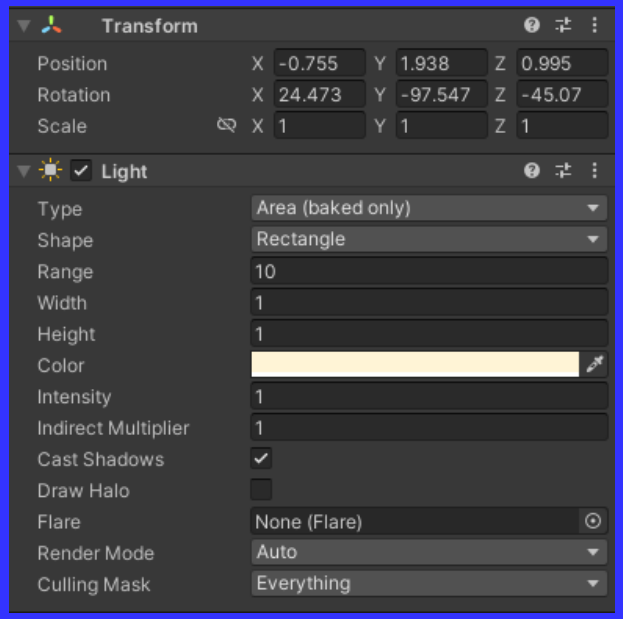




Area Lights

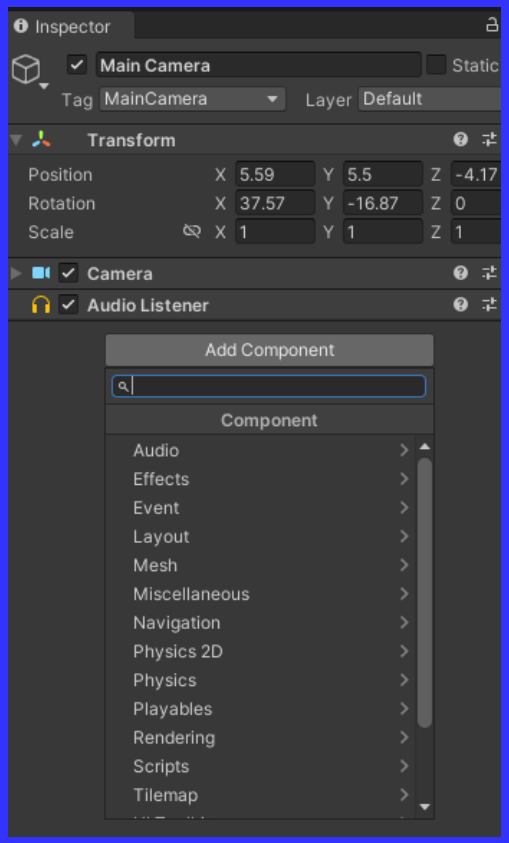
- Define an Area Light by one of two shapes in space: a rectangle or a disc.
- An Area Light emits light from one side of that shape.
- Emitted light spreads uniformly in all directions across that shape's surface area.
- The **Range** property determines the size of that shape.
- The intensity of the illumination provided by an Area Light diminishes at a rate determined by the inverse square of the distance from the light source
- Only of baked lighting

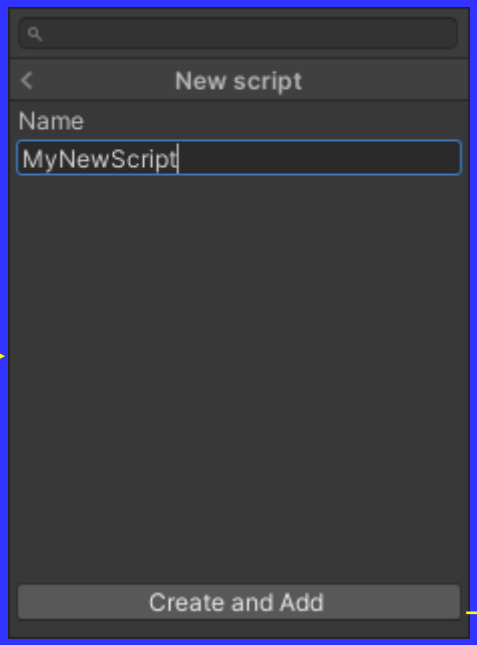


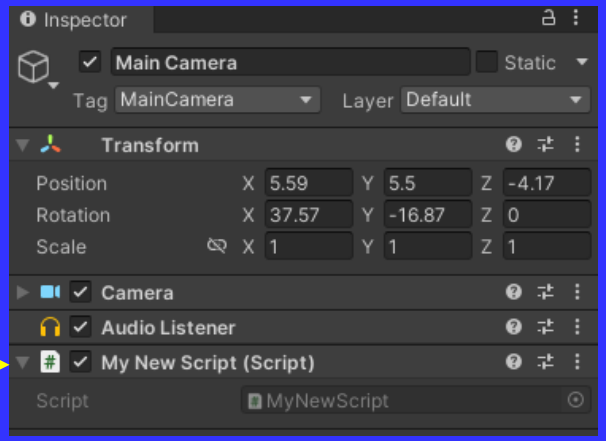


Unity Scripting

- Add C# scripts on any game object via ‘’Add Component’’ button
- Add an existing script in the project
- Or add a new script by clicking ‘’New Script’’ item



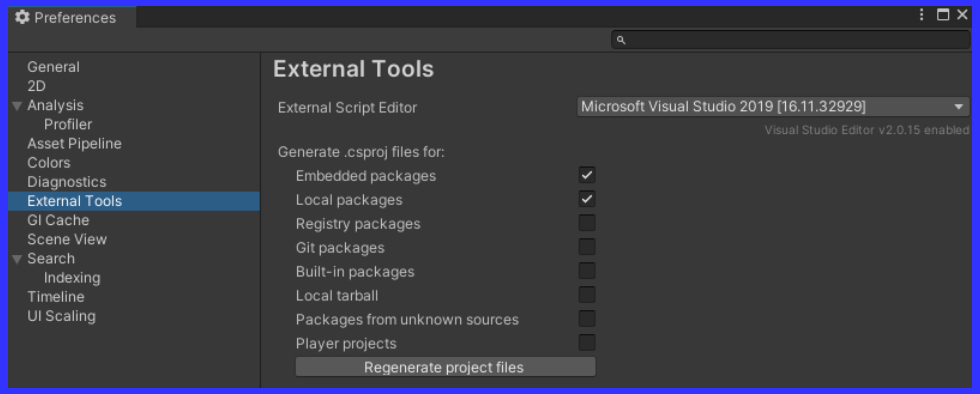


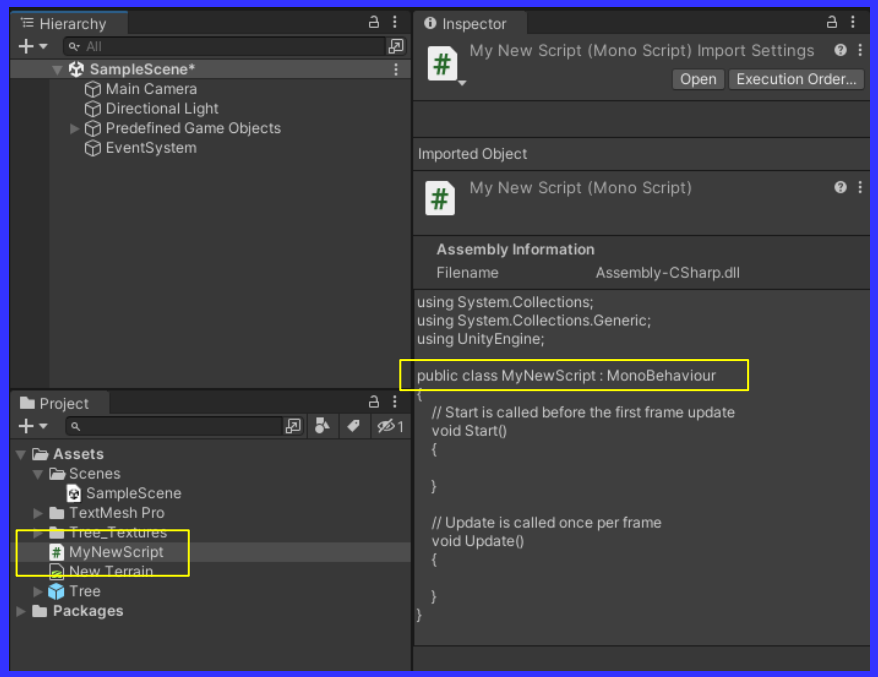


Unity Scripting

- Unity scripts that can be put on game objects are inherited from the build-in class called ‘’MonoBehaviour’’.
- The your class name and the file name shall be the same.
- You can add and use classes that are not inherited from ‘’MonoBehaviour’’, but these will be backgound code, and will not be directly put on objects.

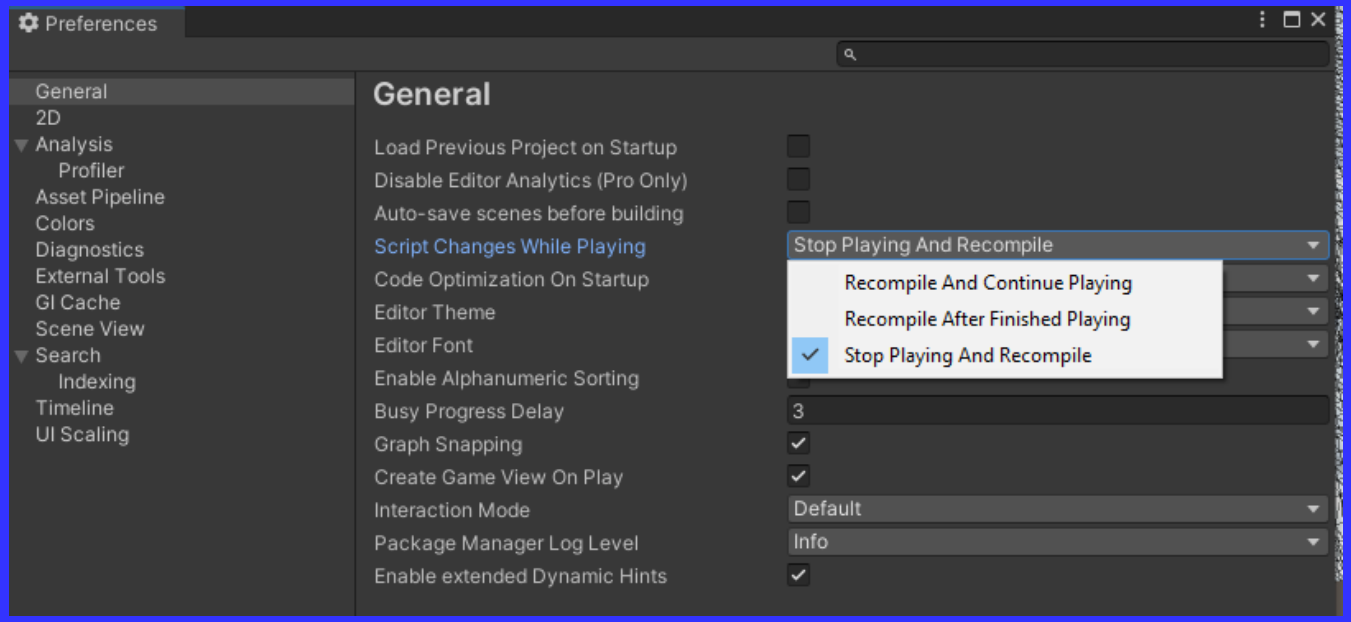
- Click to open in Visual Studio





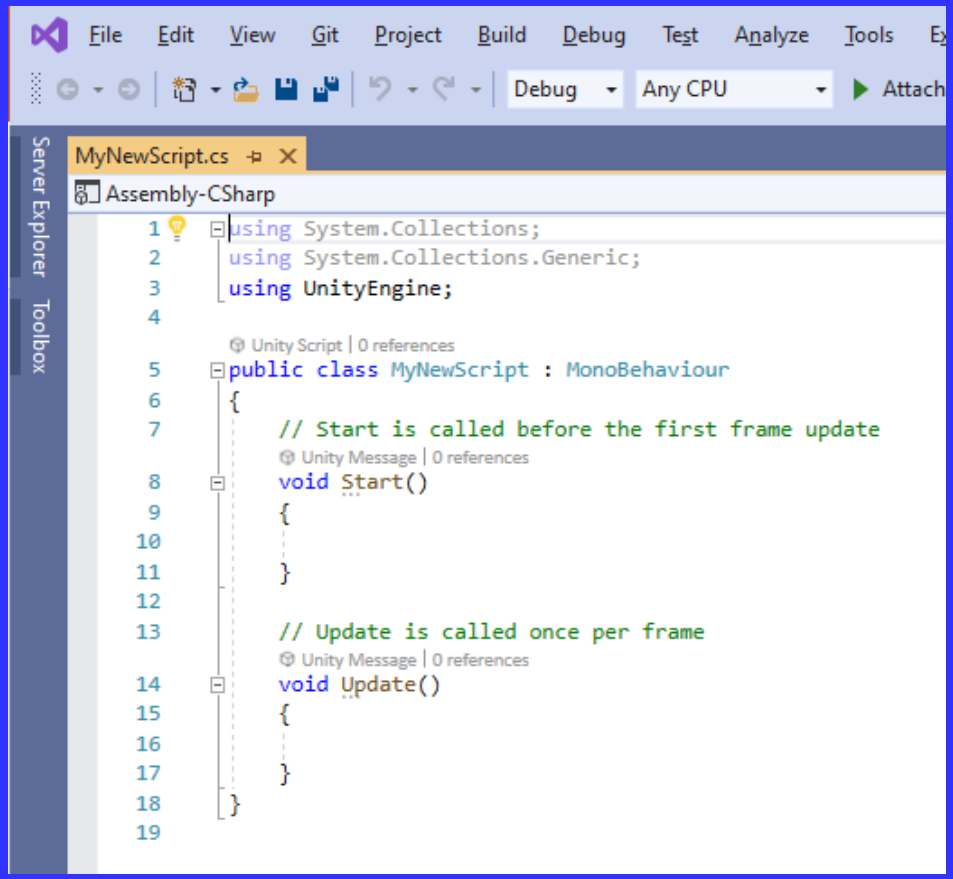
Unity Scripting

- After a script is updated from an external tool
  - When you focus to Unity Editor
    - Unity Editor checks for modifications and automatically recompile the scripts even in run-time.
    - Compiling in run-time sometimes makes unity hang
    - So for big Projects, I advice to disable this.



Unity Scripting

- When a new script is created
  - 2 common methods are automatically added for you
  - MonoBehaviour has many more predefined methods



Unity Scripting

- MonoBehaviour has many more predefined methods

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	Frame-rate independent MonoBehaviour.FixedUpdate message for physics calculations.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all GameObjects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	OnCollisionStay is called once per frame for every Collider or Rigidbody that touches another Collider or Rigidbody.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.

Unity Scripting

- MonoBehaviour has many more predefined methods

OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the Collider.
OnMouseExit	Called when the mouse is not any longer over the Collider.
OnMouseOver	Called every frame while the mouse is over the Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New partic property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a Particle System meet the conditions in the trigger module.
OnParticleUpdateJobScheduled	OnParticleUpdateJobScheduled is called when a Particle System's built-in update job has been scheduled.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.

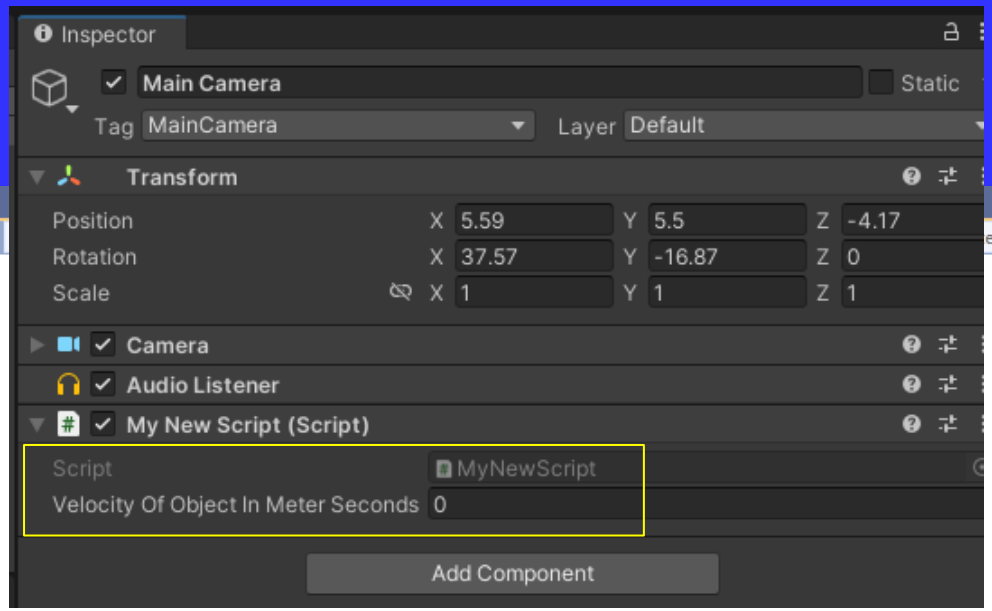
Unity Scripting

- MonoBehaviour has many more predefined methods

OnPostRender	Event function that Unity calls after a Camera renders the scene.
OnPreCull	Event function that Unity calls before a Camera culls the scene.
OnPreRender	Event function that Unity calls before a Camera renders the scene.
OnRenderImage	Event function that Unity calls after a Camera has finished rendering, that allows you to modify the Camera's final image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when a direct or indirect parent of the transform of the GameObject has changed.
OnTriggerEnter	When a GameObject collides with another GameObject, Unity calls OnTriggerEnter.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	Editor-only function that Unity calls when the script is loaded or a value changes in the Inspector.
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Unity Scripting

- When a public primitive property is added to your class
  - It is automatically added to the Inspector
  - You can change its value in design and run-time.



MyNewScript.cs

Assembly-CSharp

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class MyNewScript : MonoBehaviour
{
    public float VelocityOfObjectInMeterSeconds = 0;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        transform.position = new Vector3(transform.position.x, transform.position.y, transform.position.z + VelocityOfObjectInMeterSeconds * Time.deltaTime);
    }
}
```

Unity Scripting

- When a public serialized class property is added
  - It is also automatically added to the Inspector
  - You can change its properties in design and run-time.

MyNewScript.cs

Assembly-CSharp | VelocityInformation | RotationalVelocityInMSec

```
using System;

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[Serializable]
2 references
public class VelocityInformation
{
    public float LinearVelocityInMSec = 0;
    public float RotationalVelocityInMSec = 0;
}

Unity Script (1 asset reference) | 0 references
public class MyNewScript : MonoBehaviour
{
    public VelocityInformation Velocity = new VelocityInformation();

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        transform.position = new Vector3(transform.position.x, transform.position.y, transform.position.z + Velocity.LinearVelocityInMSec * Time.deltaTime);

        transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles.x,
                                              transform.rotation.eulerAngles.y + Velocity.RotationalVelocityInMSec * Time.deltaTime,
                                              transform.rotation.eulerAngles.z);
    }
}
```