Introduction to AR_VR(AT62001)

- Unity is a cross-platform game engine initially released by Unity Technologies.
- 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 deploying, while its most popular platforms are the PC, Android and iOS systems.

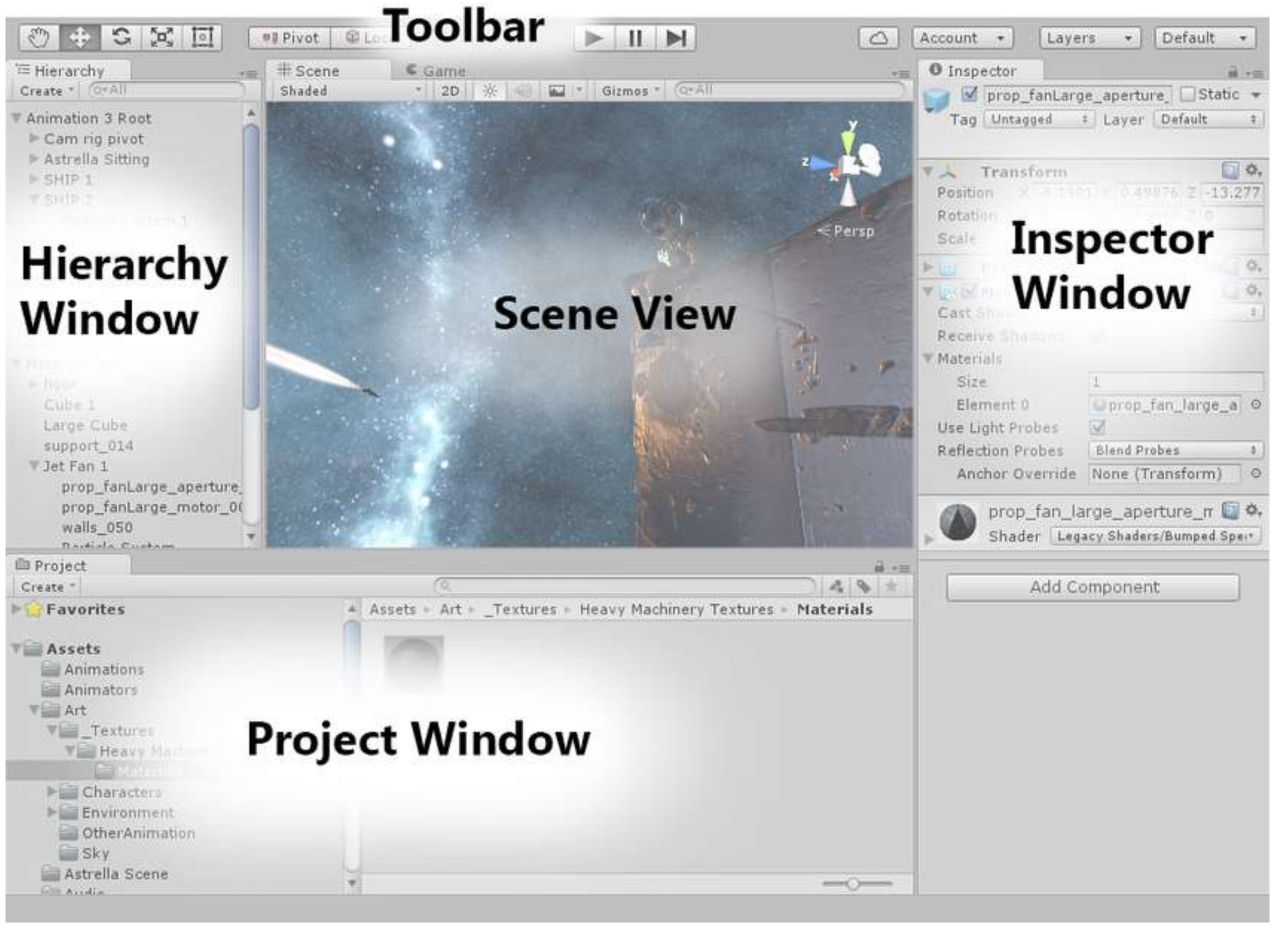
Link: Unity Manual

The Project Window displays your library of assets that are available to use in your project. When you import assets into your project, they appear here.

The Scene View allows you to visually navigate and edit your scene. The scene view can show a 3D or 2D perspective, depending on the type of project you are working on.

The Hierarchy Window is a hierarchical text representation of every object in the scene. Each item in the scene has an entry in the hierarchy, so the two windows are inherently linked.

The Inspector Window allows you to view and edit all the properties of the currently selected object.



GameObject

- GameObjects are the fundamental objects in Unity that represent characters, props and scenery. Camera is a component, and all components have GameObject and transform properties.
- The GameObject is the most important thing in the Unity Editor. Every object in your game is a GameObject. This means that everything in your game has to be a GameObject.
- GameObject can't do anything on its own; you have to give it properties before it can become a character, an environment, or a special effect.

Asset Store

- The Asset Store provides a range of solutions to help you create scene faster and with less complexity.
- Use assets as a placeholder while you build your game, leverage scripts to reduce the burden of coding new logic, and efficiently iterate your prototypes to achieve the exact type of result you want.
- An asset is representation of any item that can be used in your game or project. An asset may come from a file created outside of Unity, such as a 3D model, an audio file, an image, or any of the other types of file that Unity supports.
- It is home to a growing library of free and commercial Assets with a wide variety of Assets is available, covering everything from Textures, Models and animations to whole Project examples, tutorials and Editor extensions.

VUFORIA

- supports AR app development for Android and iOS
- has native engine integration with Unity
- is platform-independent, it can be used with virtually any platform
- Marker-based AR
- Image Recognition

➢Image Target::

- The Engine detects and tracks the image by comparing extracted natural
- features from the camera image against a known target resource database

Terrain

- The Unity Editor includes a built-in set of Terrain features that allow you to add landscapes to your game.
- Create multiple Terrain tiles, adjust the height or appearance of your landscape, and add trees or grass to it.
- At runtime, Unity optimizes built-in Terrain rendering for efficiency.

Trees

- You can paint Trees onto a Terrain in a way that is like painting heightmaps and Textures.
- Trees are solid 3D objects that grow from the surface.
- Unity uses optimizations like billboarding for distant Trees to maintain good rendering performance. This means that you can have dense forests with thousands of Trees and keep an acceptable frame rate.

Material

- This class exposes all properties from a material, allowing you to animate them. You can also use it to set custom shader properties that can't be accessed through the inspector.
- To get the material used by an object, use the Renderer.material property.

PhysicMaterial

- Physics material describes how to handle colliding objects (friction, bounciness).

Physics in Unity

- Physics in Unity ensures that objects accelerate and respond to collisions, gravity, and various other forces.
- Unity provides us with different physics engine implementations which we can use according to our Project needs.
- Unity differentiates its physics engines under two categories.

-Built-in physics engine for object-oriented projects (2D/3D).

-Physics engine packages for data-oriented projects (DOTS).

Introduction to Scripting

- Scripting tells our GameObjects how to behave.
- Unity uses C# language.

➢ Variables:
- hold values and references to objects.
- starts with a lowercase letter.

➢ Functions:
- collections of code that compare and manipulate the variables.
- starts with an uppercase letter.

➢ Classes:
- it's a way to structure code to wrap collections of variables and functions.
- together to create a template that defines the properties of an object.
- public or private.

Mono Behavior class

- Base class from which every Unity script derives
- Unity calls and executes various event methods in a pre-determined order.

Types:

- Start() -- when a script is enabled just before any of the Update methods
- Update() -- called once per frame
- Awake() -- only once during the lifetime

  -- executed even if the script is not enabled

  --Is called in a random order

  --Awake can be usable in another GameObject's Awake

  --called before any Start method

- OnEnable() -- executes after Awake method

  -- whenever script is enabled

  -- SetActive(true)

- OnClossionEnter() -- whenever two colliders hit each other

- OnDestroy() -- when an object is destroyed

  -- Destroy(gameObject)

- OnDisable() -- whenever a script is disabled

  -- SetActive(false)

Rigidbodies:

- components that allow a GameObject to react to real-time physics.
- Control of an object's position through physics simulation.
- Adding a Rigidbody component to an object will put its motion under the control of Unity's physics engine.
- Even without adding any code, a Rigidbody object will be pulled downward by gravity and will react to collisions with incoming objects if the right component is also present.

COLLIDERS:

- components which define a region in which collision interaction between GameObjects occur

USES:

1.determine physics interactions

2. get code callbacks when gameobjects strike or intersect each other

Types:

Box collider, Mesh collider, Sphere collider, Terrain collider

Collider functions:

- 1. OnCollisionEnter-- called when this collider/rigidbody has begun touching another rigidbody/collider.

- 2.OnCollisionExit-- when this collider/rigidbody has stopped touching another rigidbody/collider.

- 3. OnCollisionStay-- once per frame for every collider/rigidbody that is touching rigidbody/collider.

Collision Detection Modes:

1.Discrete :
- continuous collision detection is off for this rigid body
- will not work very well for fast-moving objects such as a bullet
- collisions for this Collider will only be checked at the contents time fix Delta time

2.Continuous:
- used when we need our object to collide with static mesh geometry
- collision should occur between two fixed updates steps

3.Continuous Dynamic :
- continuous collision detection is on for colliding with both static and dynamic geometry.
- should only be used for fast moving objects

4.Continuous Speculative
- used on both dynamic and kinematic objects
- handles angular motion much better

Triggers

- a special setup of the Colliders that give them the ability to trigger events when they touch each other or overlap.
- 2 objects won't collide anymore (they will simply pass through each other) if one of the Colliders is setup as a Trigger as it will use the event system.

➢Example:

EVENT TRIGGER:

- Events are specialized delegates that are useful for when you want to alert other

Trigger	Collision
OnTrigger is called when a game object with a collider passes through a game object with “Is Trigger” checked collider.	OnCollision is called when a game object's collider or rigidbody has begun touching another game object's collider or rigidbody.
Physics is disabled for trigger colliders	enable physics to the game object.
In games, you trigger events when some event happens. For instance, you may want to open a door when a player comes to a certain point. In these cases, we use triggers to execute some pieces of code.	OnCollisionEnter,OnCollisionExit,OnCollisionStay
OnTriggerEnter,OnTriggerExit,OnTriggerStay	

User Interface(UI) in Unity

- Unity provides three UI systems that you can use to create UI for the Unity Editor and applications made in the Unity Editor:

1. UI Toolkit
2. The Unity UI package (uGUI)
3. IMGUI(Immediate Mode Graphical User Interface)

- UI toolkit is a GameObject-based UI system that uses Components and the Game View to arrange, position, and style user interfaces.
- uGUI is an older, GameObject-based UI system that you can use to develop runtime UI for games and applications.
- IMGUI is a code-driven UI Toolkit that uses the OnGUI function, and scripts that implement it, to draw and manage user interfaces. Used to create custom Inspectors for script components, extensions for the Unity Editor, and in-game debugging displays. It is not recommended for building runtime UI.