CEG5205

AI Sensors & Virtual/ Augmented Reality Technologies

Lab 2 – Unity 3D Interface

Location: E4A 06-03

Lecturer: prof. Chengkuo Lee (elelc@nus.edu.sg)

Lab2 GA: Zhongda Sun (e0320823@u.nus.edu)

About Lab2

Preparation

1. Every student are suggested to bring one laptop to the Lab.
2. Every student should **install Unity 3D before the lab**
3. Relative resources used in Lab2 will be provided and can be downloaded through Canvas before the class

Homework (Deadline: **Oct 8th, 12 PM**)

1. Submit a zip file to Canvas named "HW group no. _Lab2.zip", which includes:
   a. One report that includes: cover page (showing Name1-Student ID1_Name2- Student ID2); your codes (screenshot with comments), explanations for the solutions and approaches, a screenshot/video to show the results, and a statement about the contribution of the group member to each problem.
   b. The original code of the scripts.

Unity 3D

Unity 3D is a game development engine compatible with all platforms (pc, android, ios…).

The Interaction with virtual world in this lecture, i.e.,virtual reality (VR), etc., will be realized based on this engine.

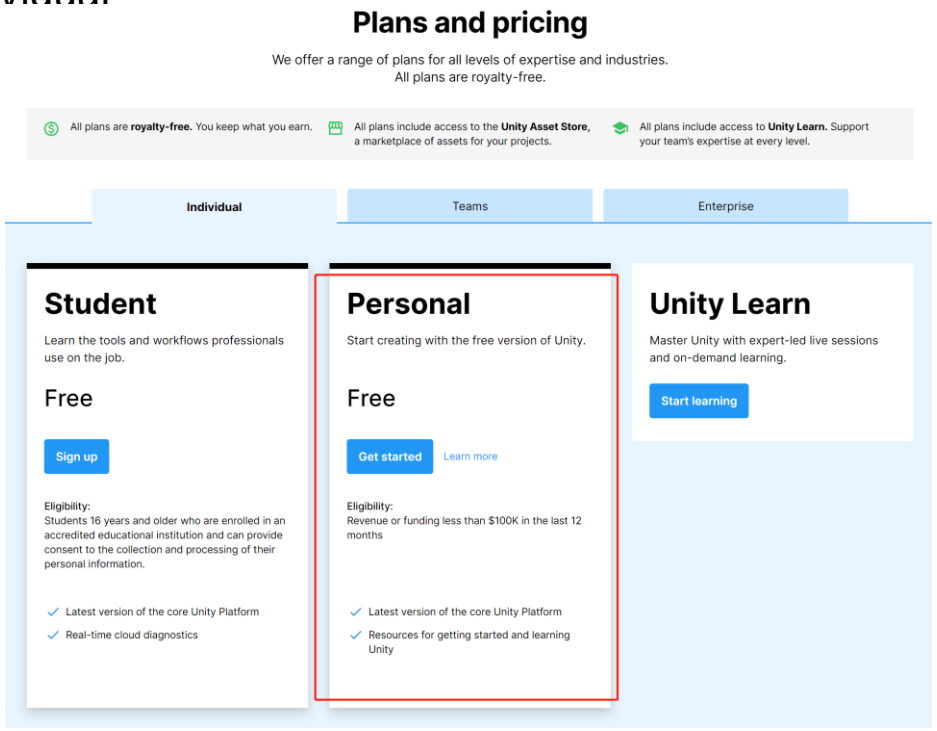
- Other game development platforms, e.g., Unreal Engine, are also supported for the final demonstration project if you have previous experiences, but will not be covered in this lecture.

Unity 3D installation – before the class

1. Download and install the Unity Hub 

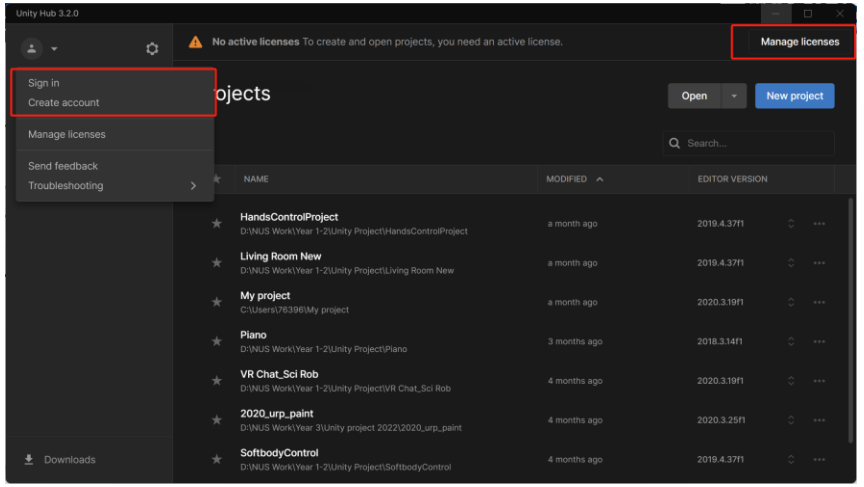
from the official website.

https://store.unity.com/front-page?check_logged_in=1#plans-individual

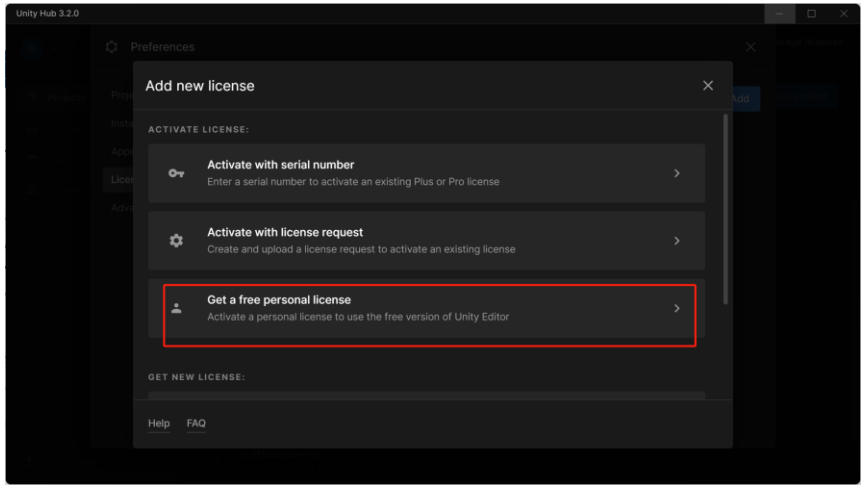


Choose the personal plan for free use.

2. Create an account and sign in the unity hub.

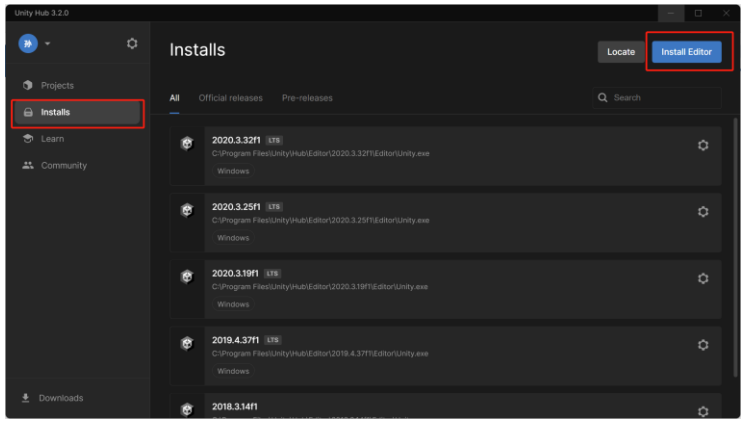


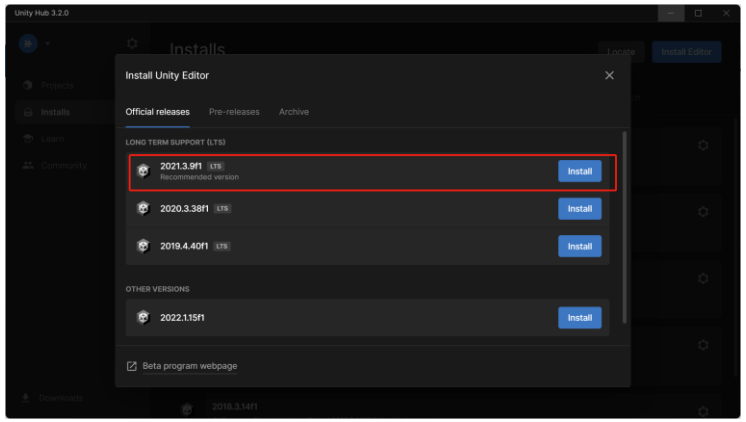
3. Get a free personal license.



Unity 3D installation – before the class

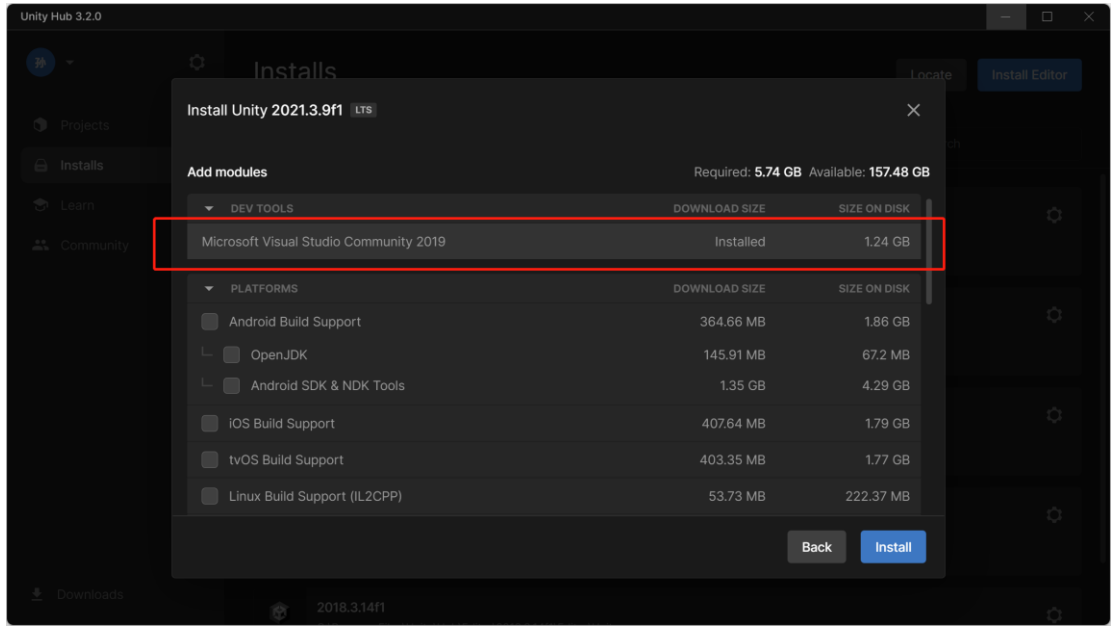
4. Install Editor in the Unity Hub.





Students in the same group are suggested to use the same version of the editor to ensure the compatibility.

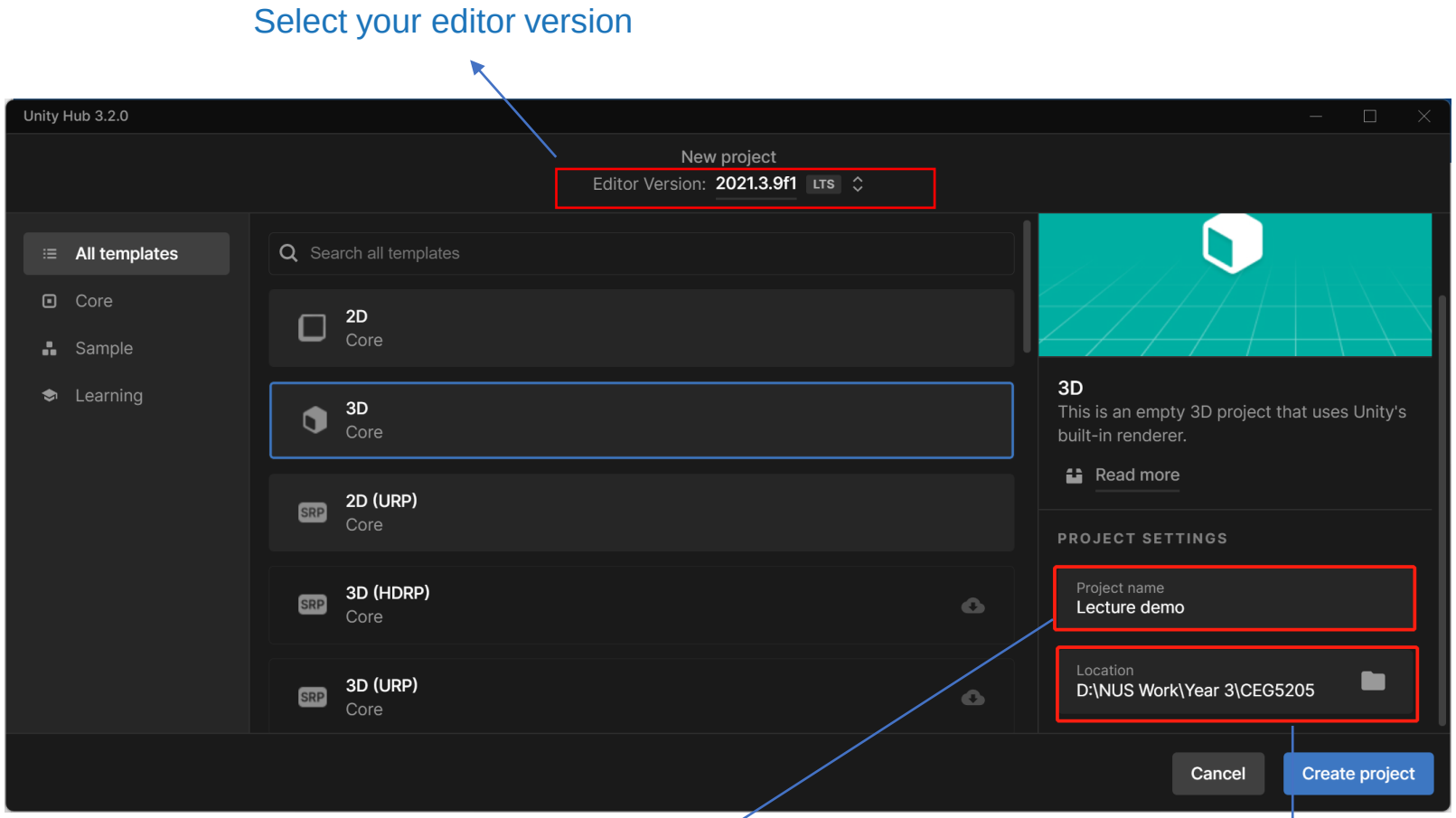
5. The VS dev tool is a necessary module.



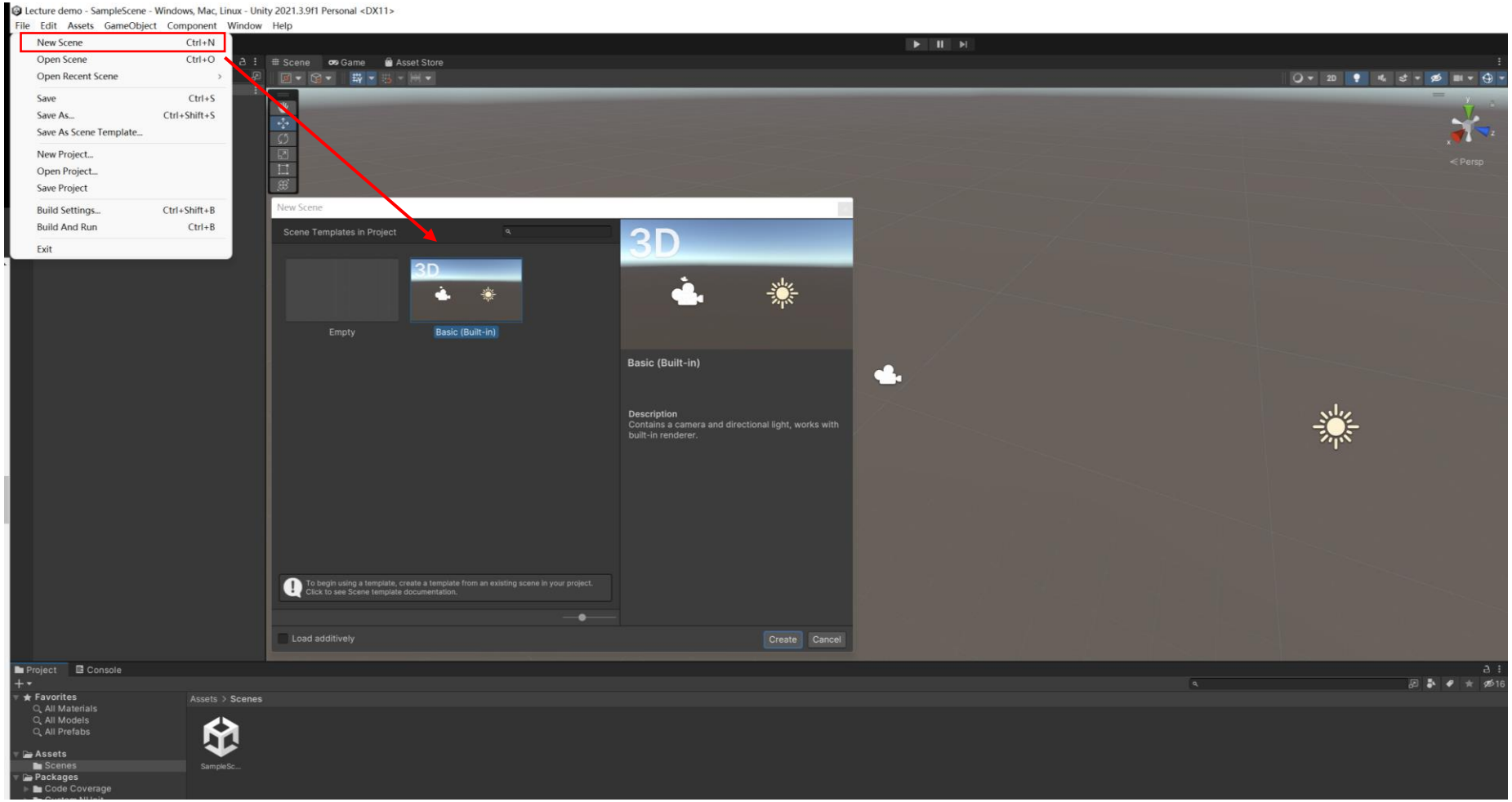
After these steps, the Unity 3D is ready for use.

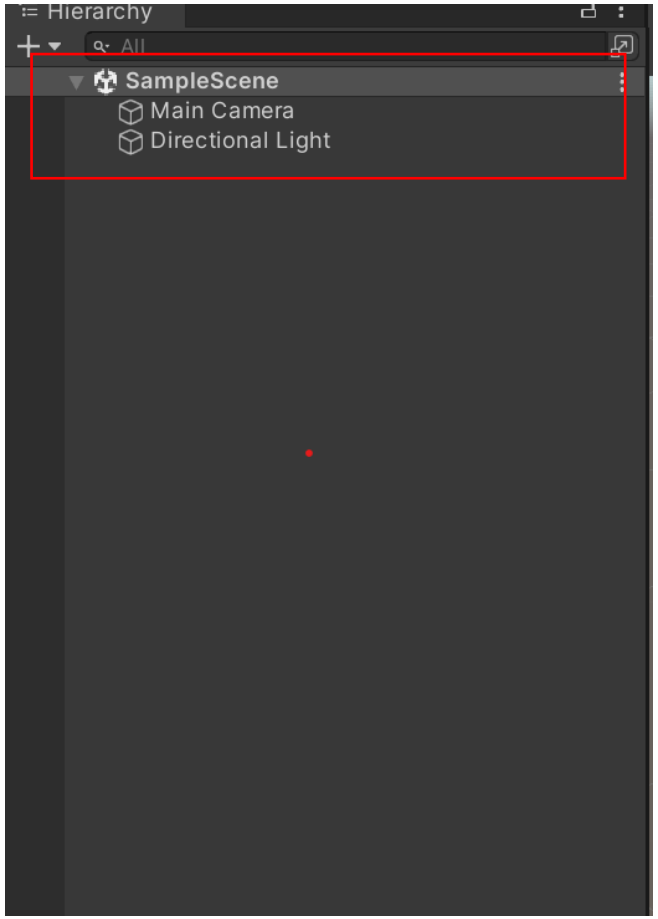
Create a new project

Create a new project based on the basic 3D core.



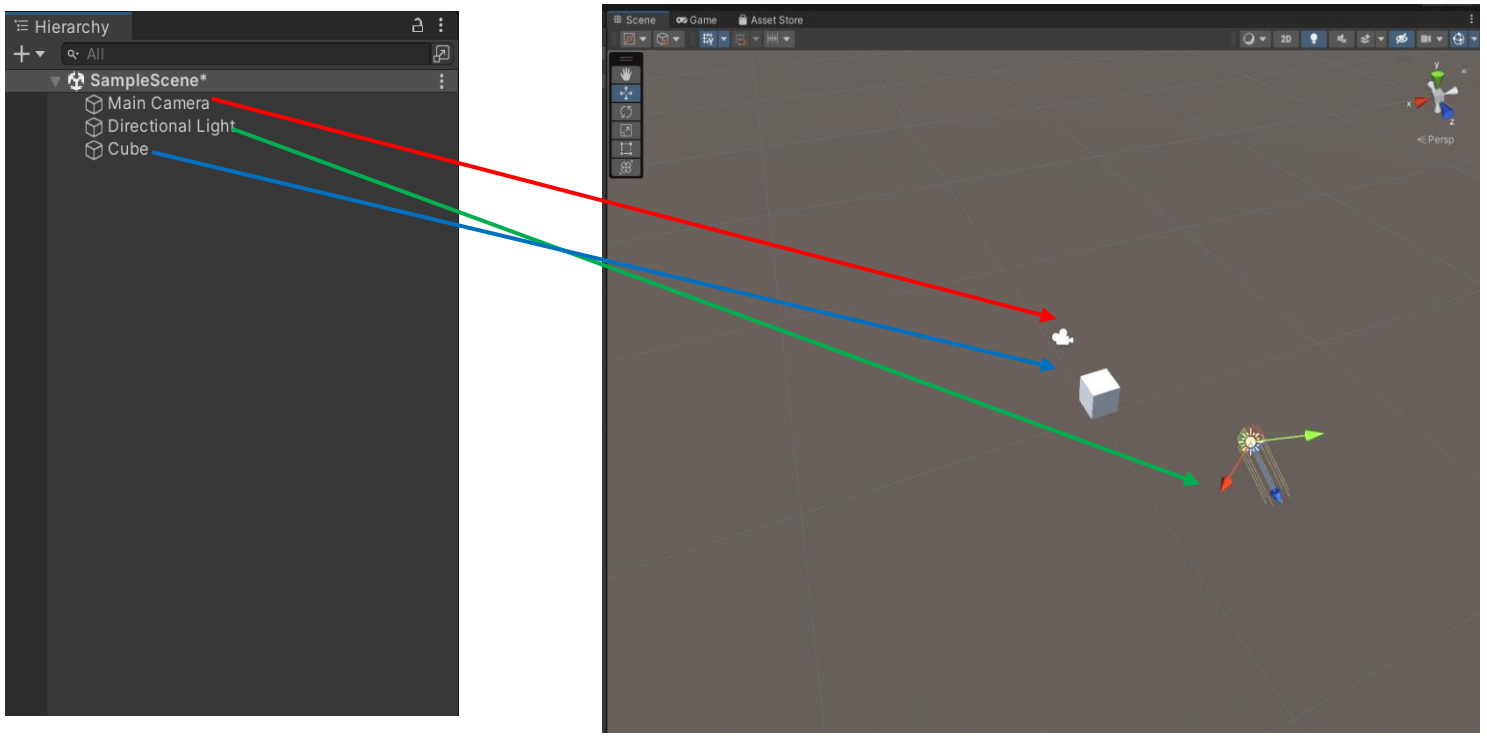
Scene



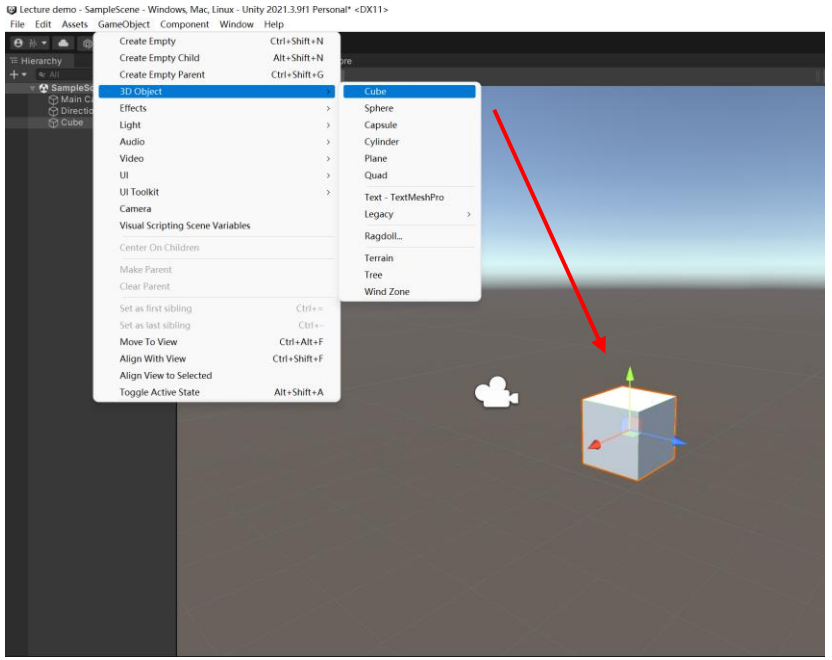


- Multiple scenes can be included in one game.
- All objects shown in the scene need to be contained under the hierarchy of the scene.

GameObject



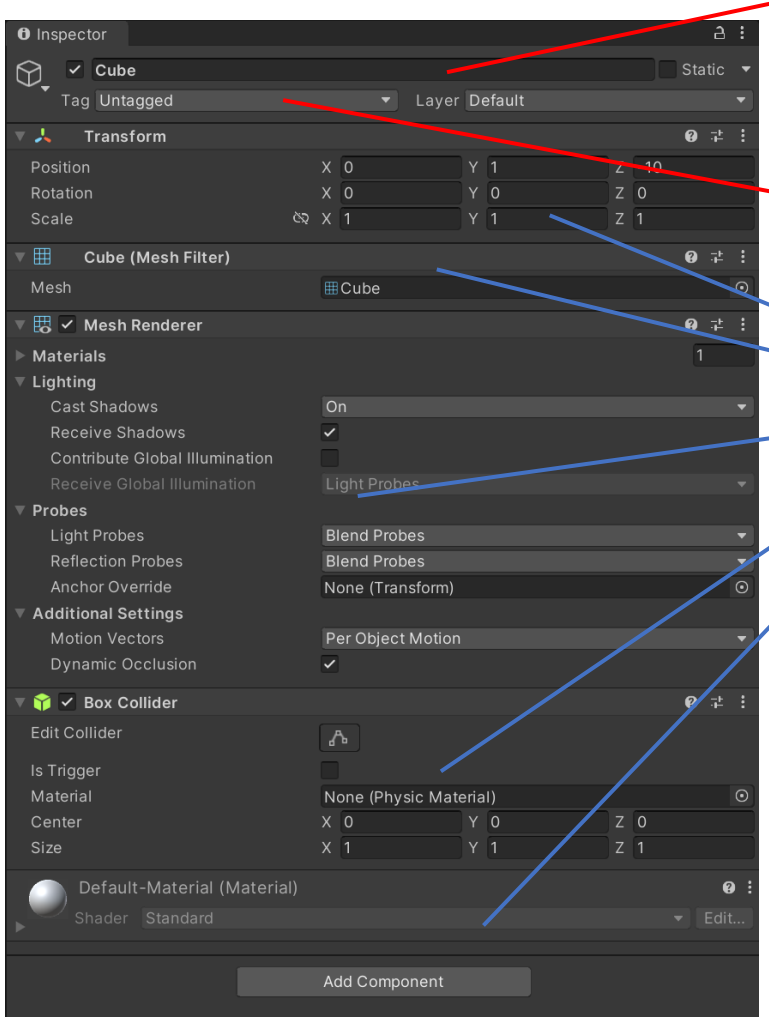
Create GameObject



- All objects in the scene are called GameObject.
- Include objects in the scene, environmental elements, effects…
- Main components of the game.

GameObject

Detailed information of GameObject

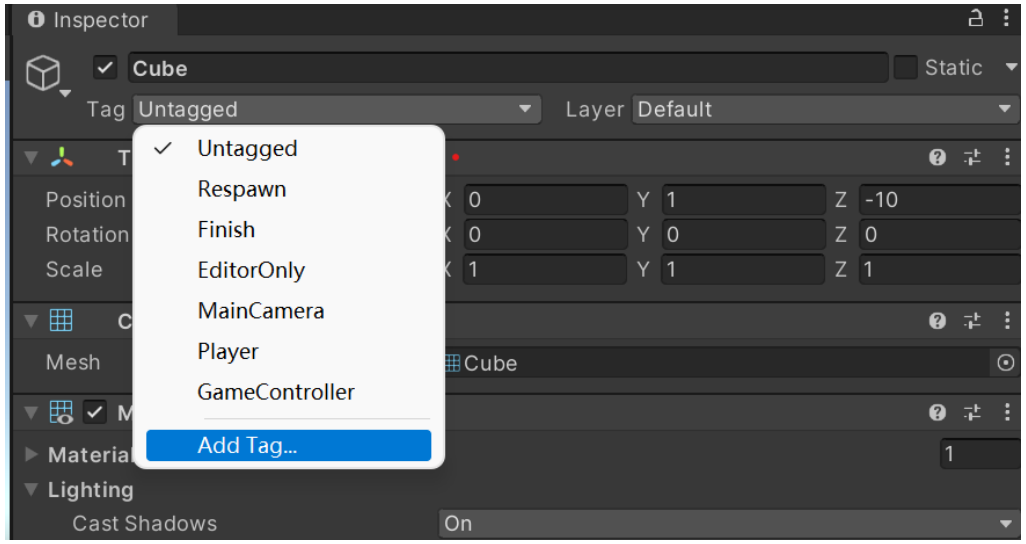


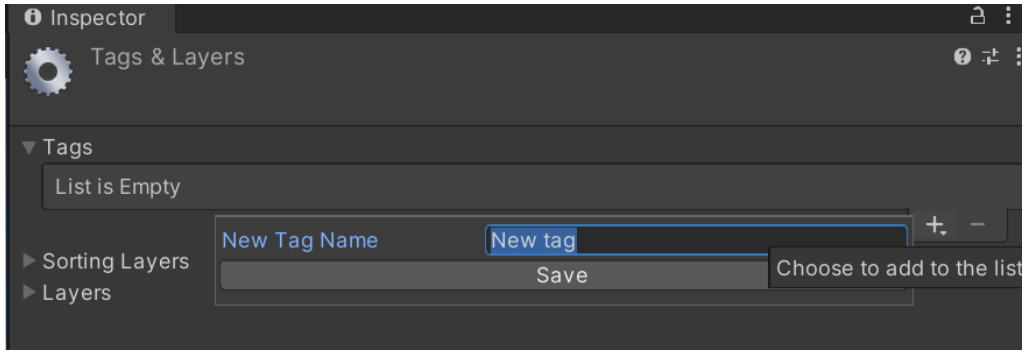
GameObject Name

GameObject Tag

Components

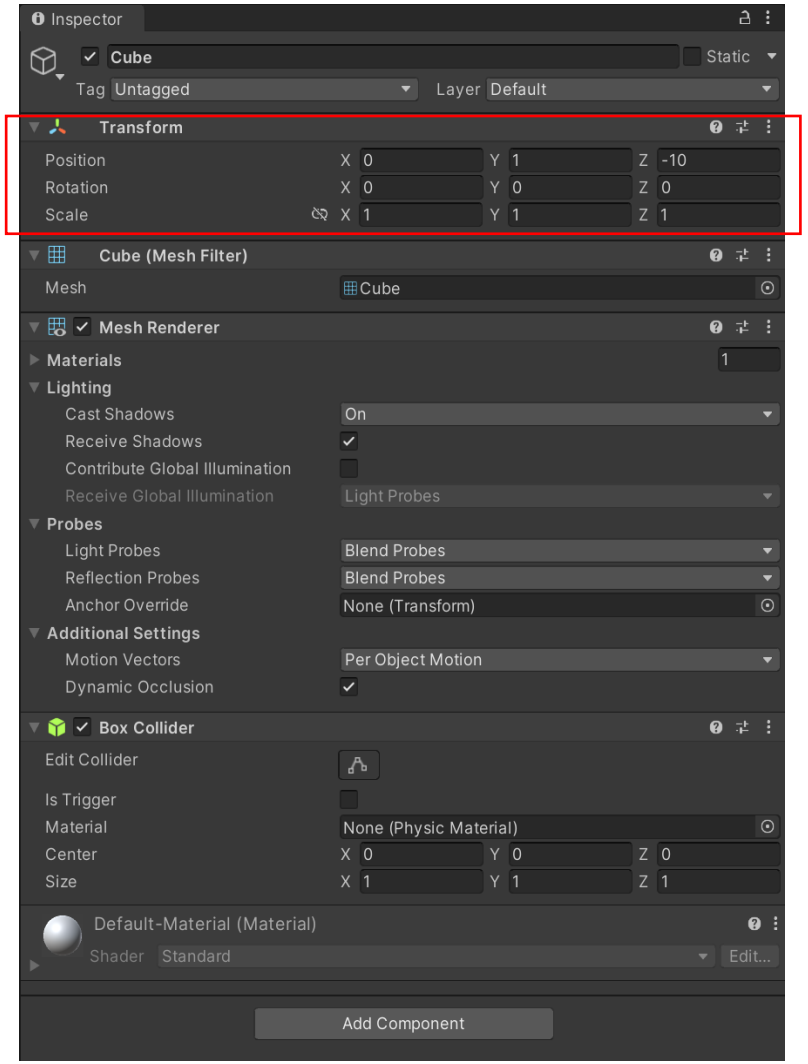
GameObject Tag





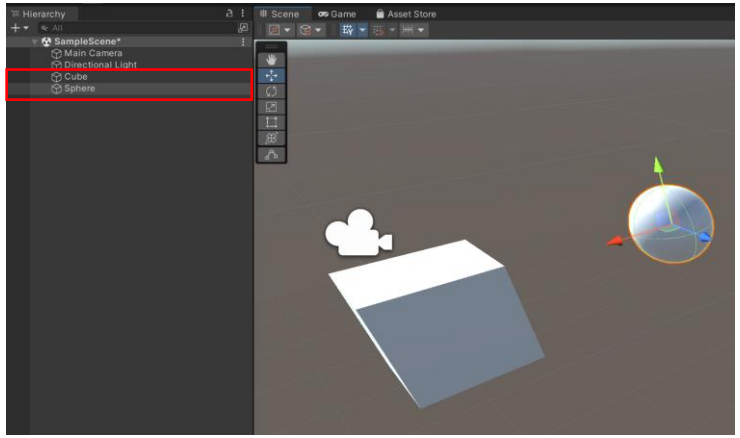
- Add tags to help us classify different objects, e.g., player…
- When writing scripts, it can help us quickly find the corresponding objects.

GameObject

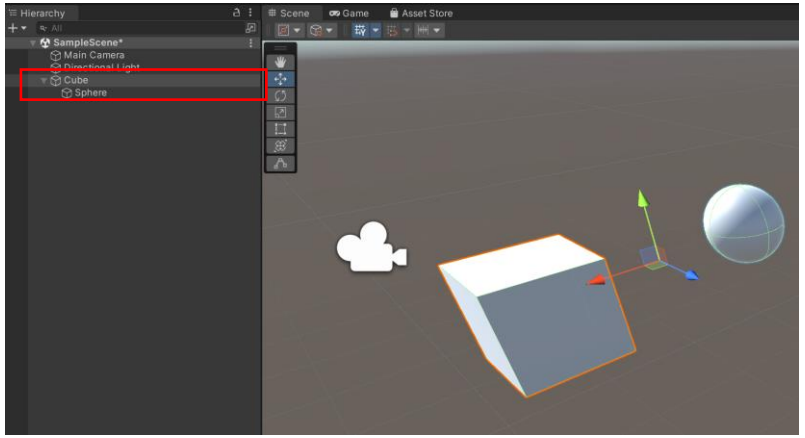


Transform: Relative position, rotation and scale to its parent object.

Relationship: children and parent



Two independent objects.



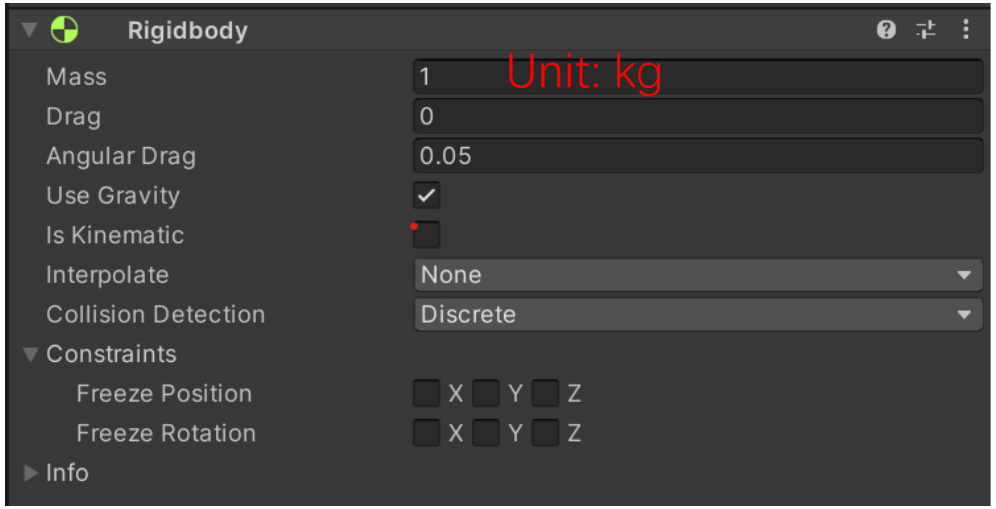
Cube: Parent
Sphere: Child

Sphere will follow Cube's movement in the 3D space.

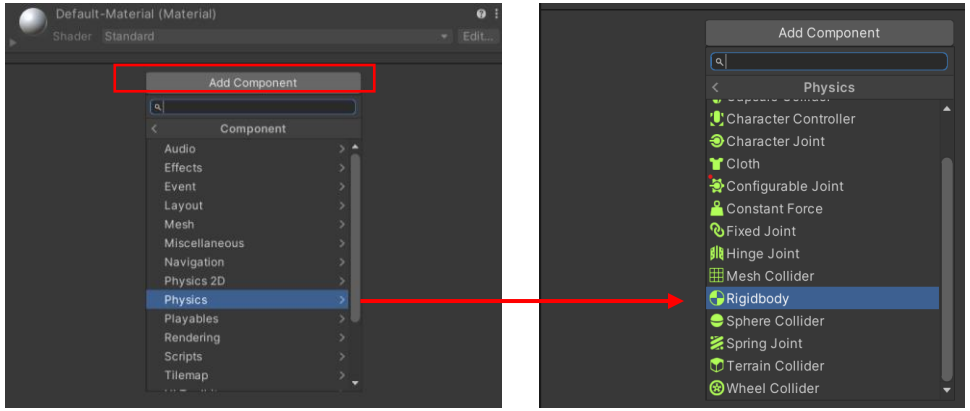
Physics

Physics: To simulate the gravity, collision, friction… in virtual space.

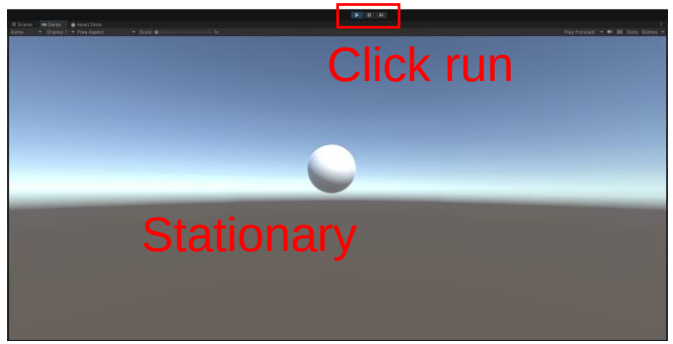
- Physics components
  - Rigidbody
  - Constant Force
  - Collider
- Physics Manager



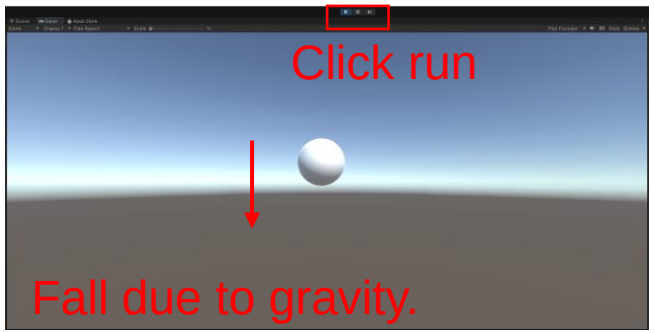
1. Rigidbody: In physics, a rigid body is a solid body in which deformation is zero or so small it can be neglected.



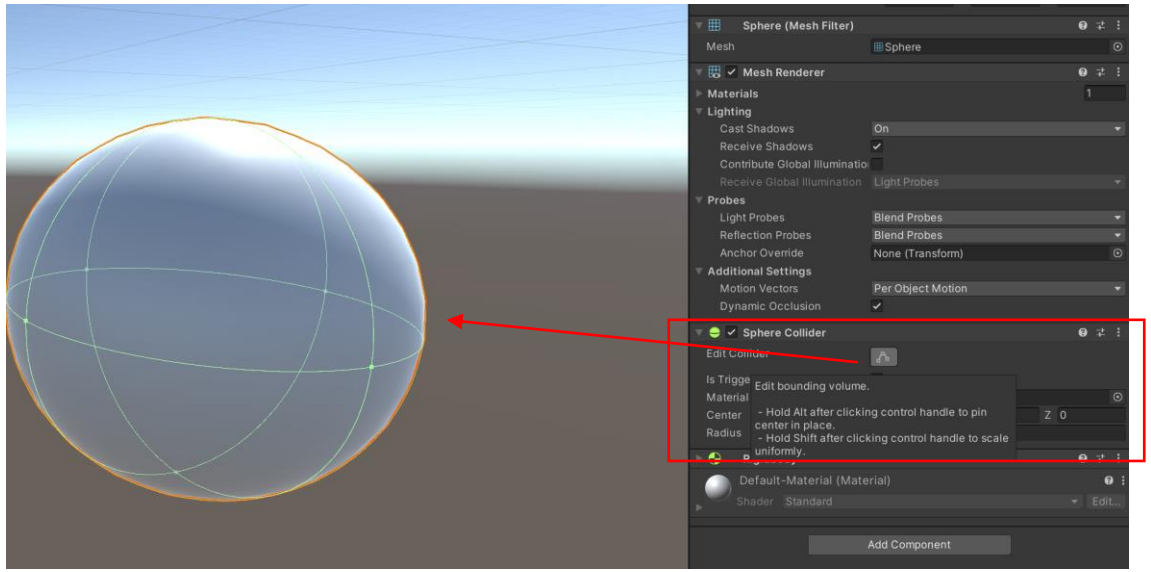
Without rigidbody



With rigidbody & gravity

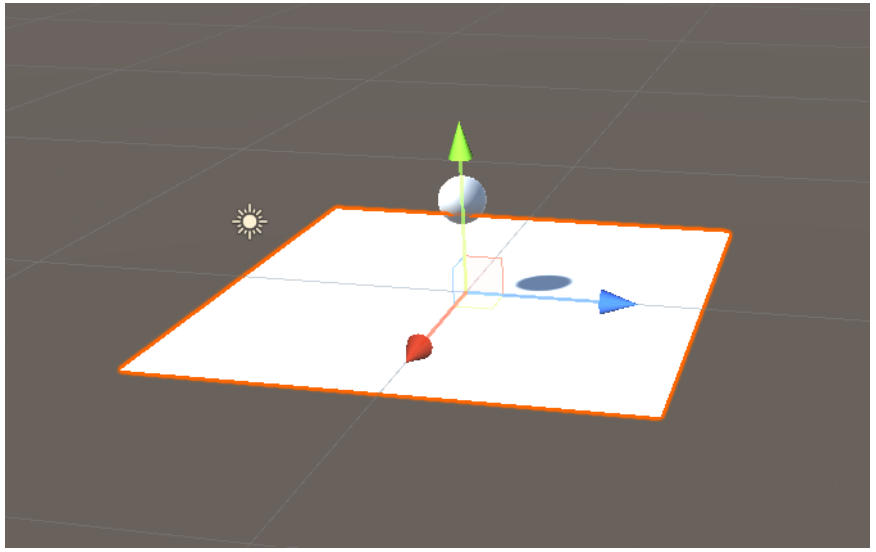


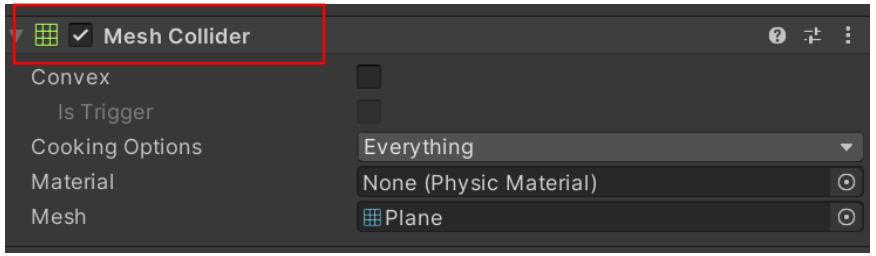
Collider



- Define the shape of the collider for GameObject.
- The physics engine only calculates collisions when both objects have collider components, otherwise the objects will pass through each other.

Create a plane in the scene

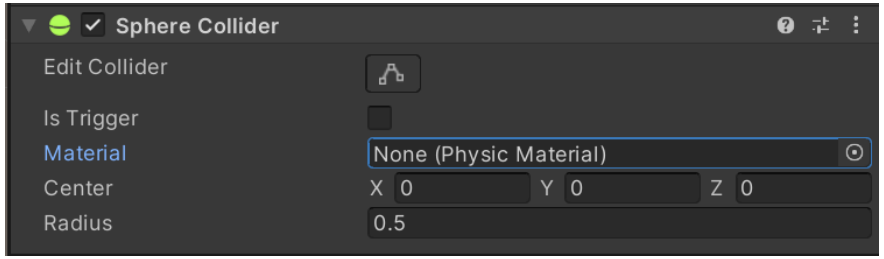




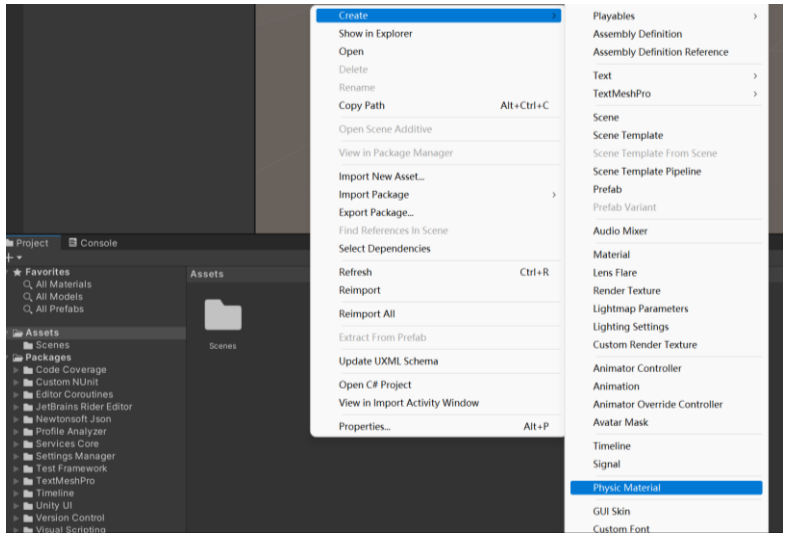
- The sphere will land on the plane surface.

Remove the mesh collider?

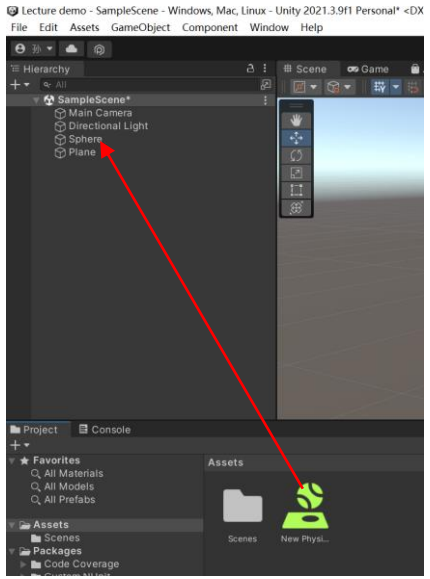
Physical Material



- Affects the collider's response to forces.

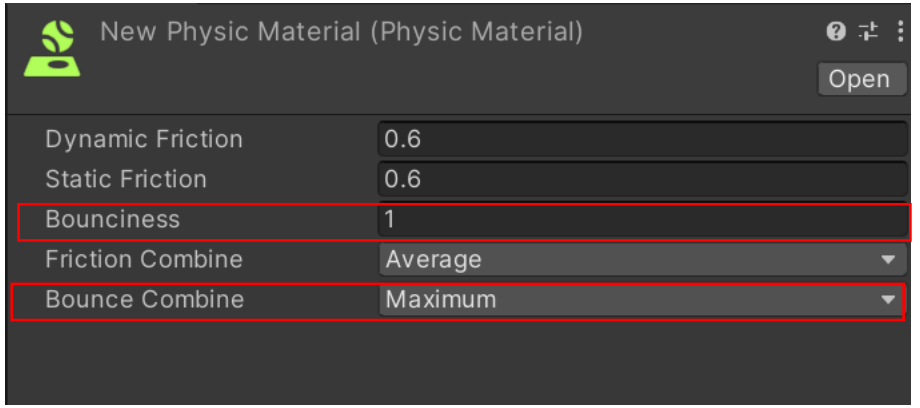


Create a physical material in the Project panel.

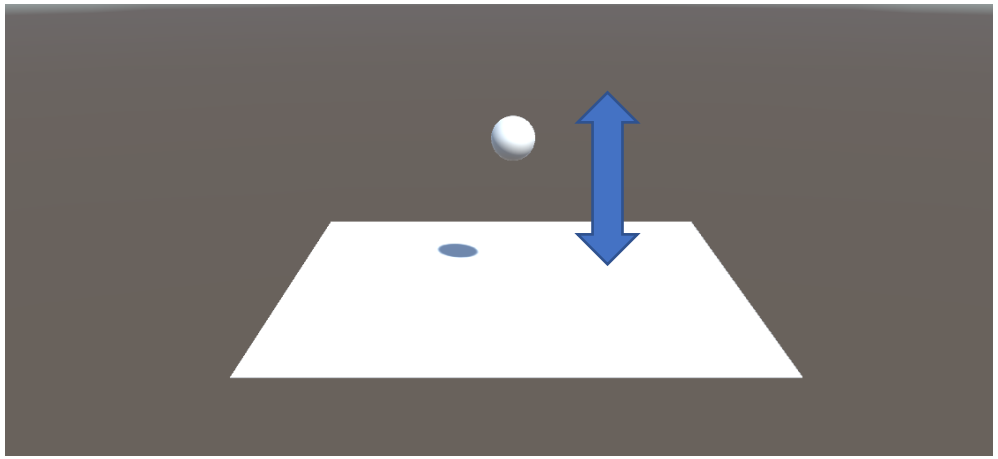


Drag to the object.

Example



Maximum the bounce

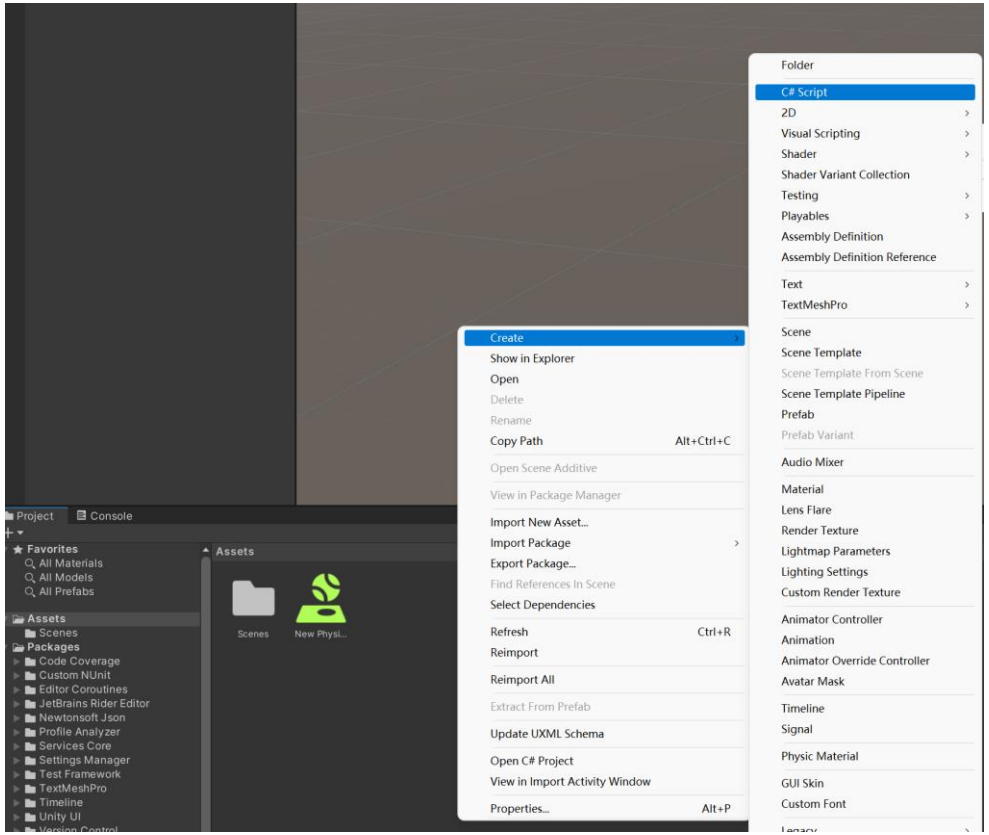


Try different parameters to see the effect.

Scripts

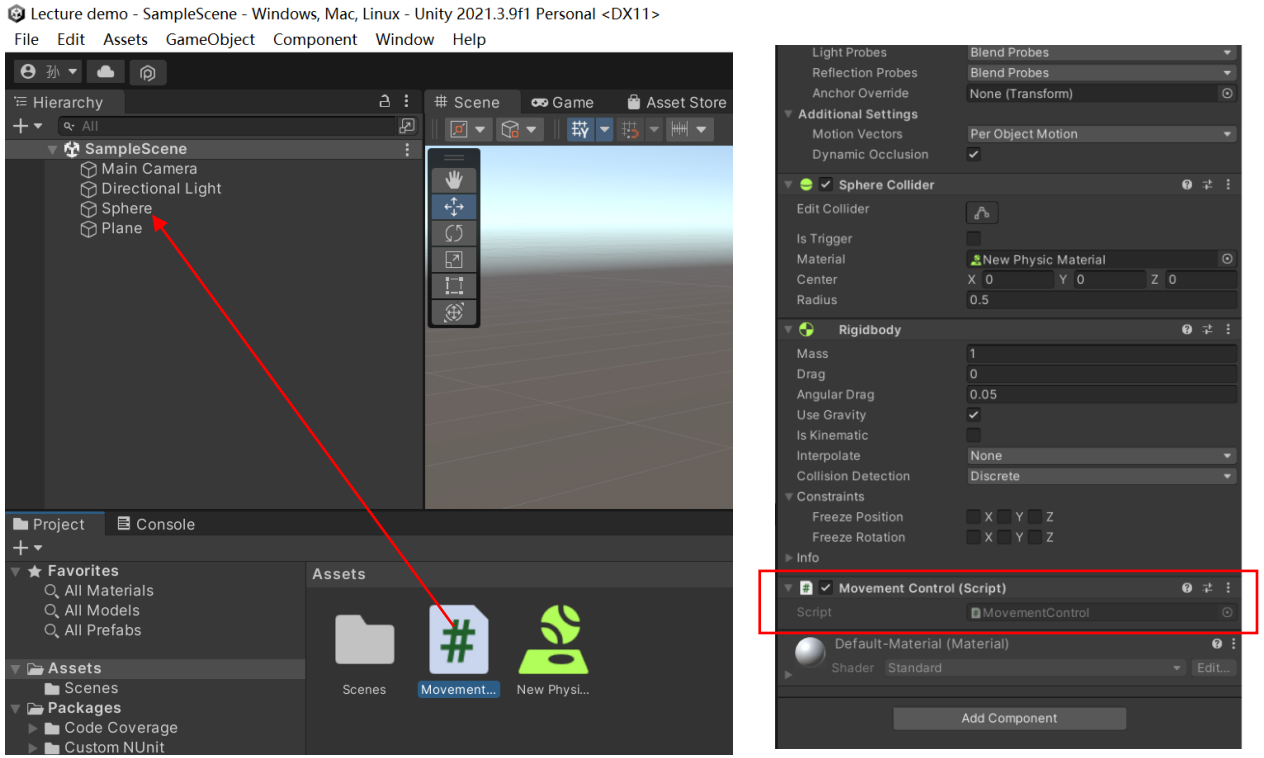
To realize the interactions in the game.

- Player interacts with the game scene through the input system.
- Interactions between GameOjects



Create a script in the Project panel.

We try to create a simple script to control the movement of the small ball.



Drag to the small ball.

Scripts

Script to control the movement of the ball.

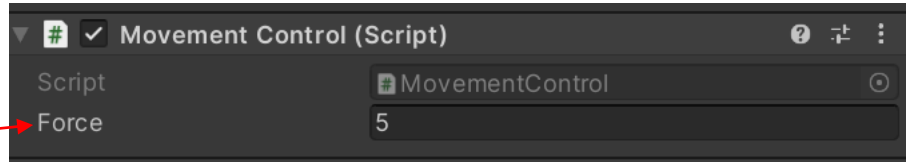
MovementControl.cs

Miscellaneous Files

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class MovementControl : MonoBehaviour
6   {
7
8       private Rigidbody rd;
9       public int force = 5;
10
11      // Start is called before the first frame update
12      void Start()
13      {
14          rd = GetComponent<Rigidbody>();
15      }
16
17      // Update is called once per frame
18      void Update()
19      {
20          float h = Input.GetAxis("Horizontal");
21          float v = Input.GetAxis("Vertical");
22          rd.AddForce(new Vector3(h, 0, v) * force);
23      }
24  }
```

Script name: need to be same

Public parameter can be accessed and modified outside the script



Get the rigidbody of the small ball

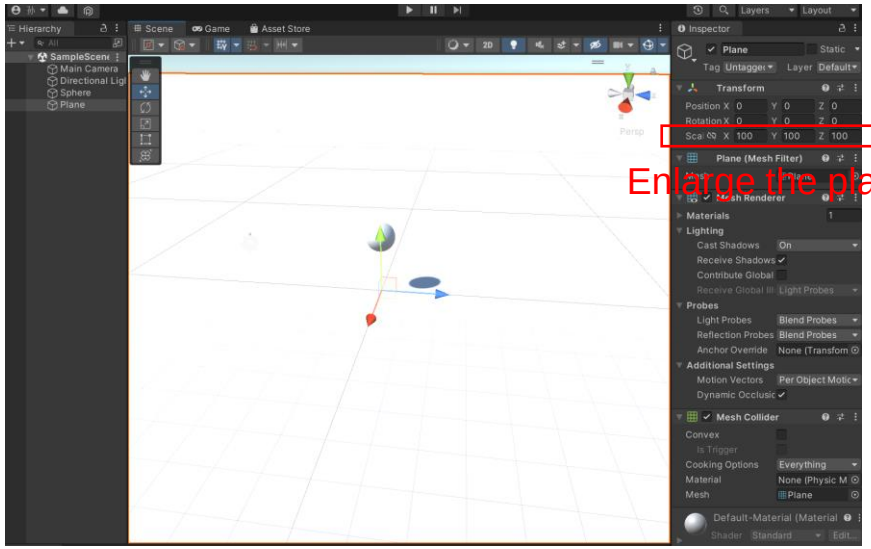
Use the keyboard's arrow keys as control input.

Direction vector * Force

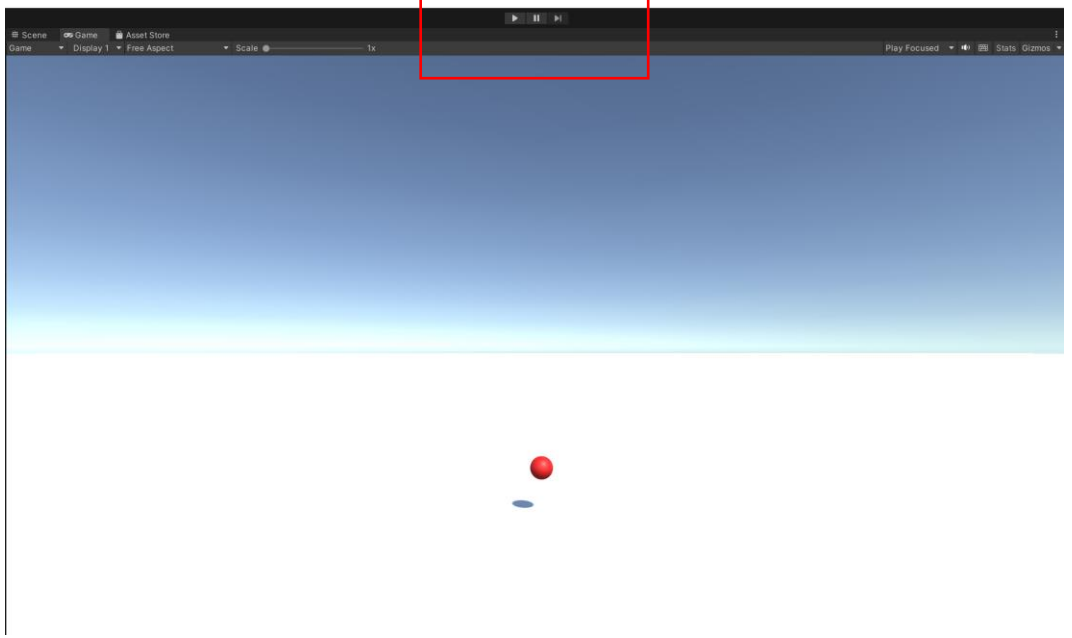
Scripts

Script to control the movement of the ball.

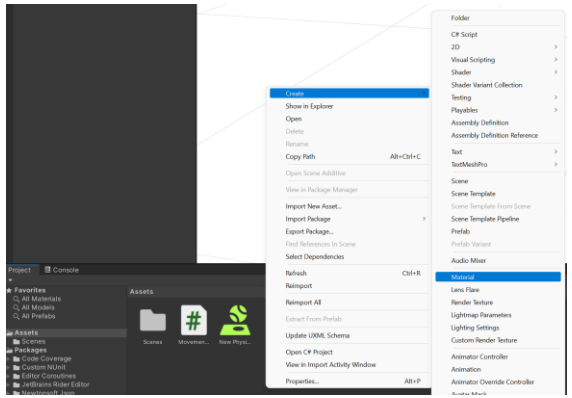
Back to Unity 3D.

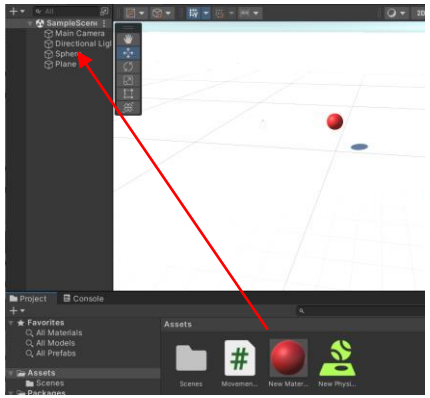


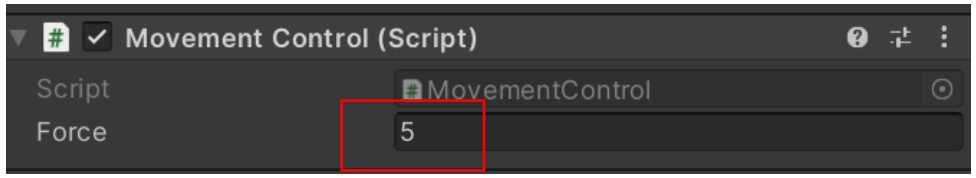
Click Run and try to control the ball



Create a material to change the color of the ball







Change the force value and see effect

Scripts

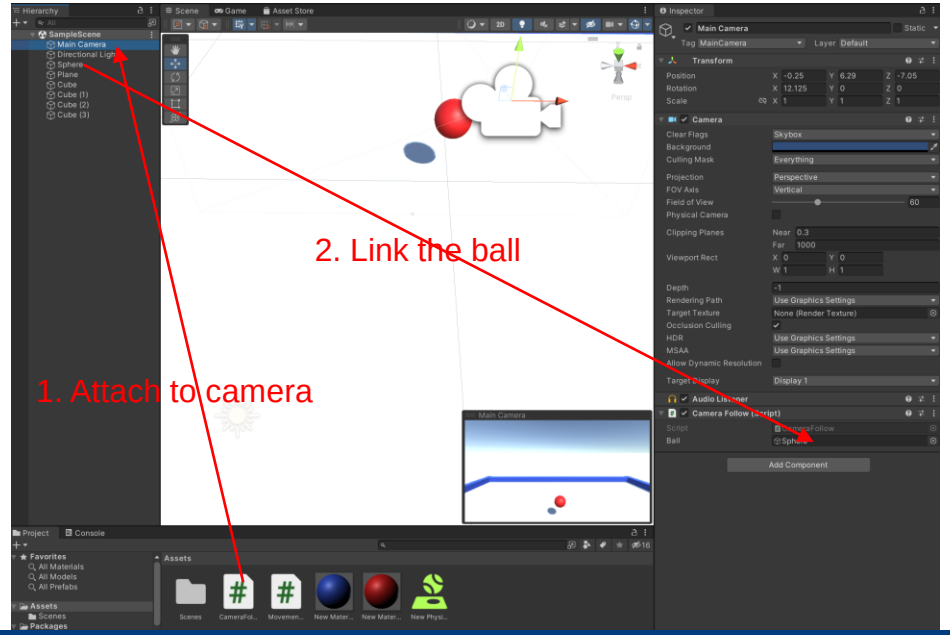
Make the camera follow the ball.

CameraFollow.cs | MovementControl.cs

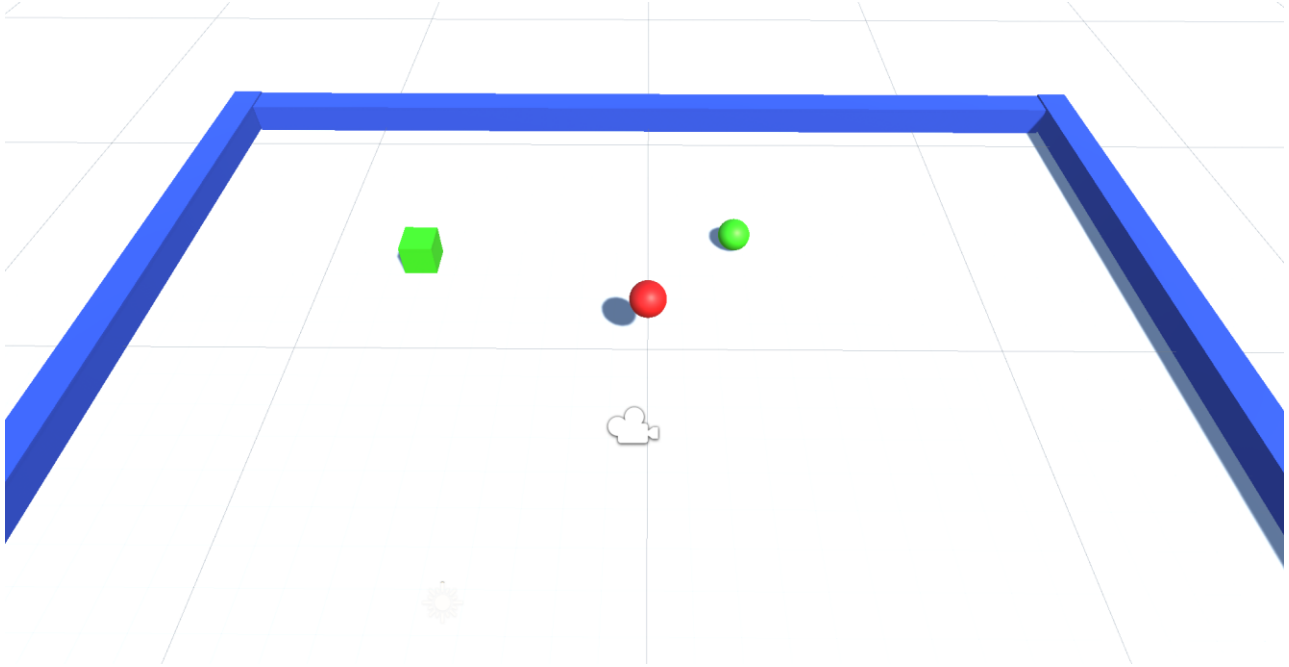
Miscellaneous Files | CameraFollow

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class CameraFollow : MonoBehaviour
6  {
7      public GameObject ball;
8      Vector3 distance;
9      // Start is called before the first frame update
10     void Start()
11     {
12         distance = ball.GetComponent<Transform>().position - gameObject.GetComponent<Transform>().position;
13     }
14
15     // Update is called once per frame
16     void Update()
17     {
18         gameObject.GetComponent<Transform>().position = ball.GetComponent<Transform>().position - distance;
19     }
20 }
21
```

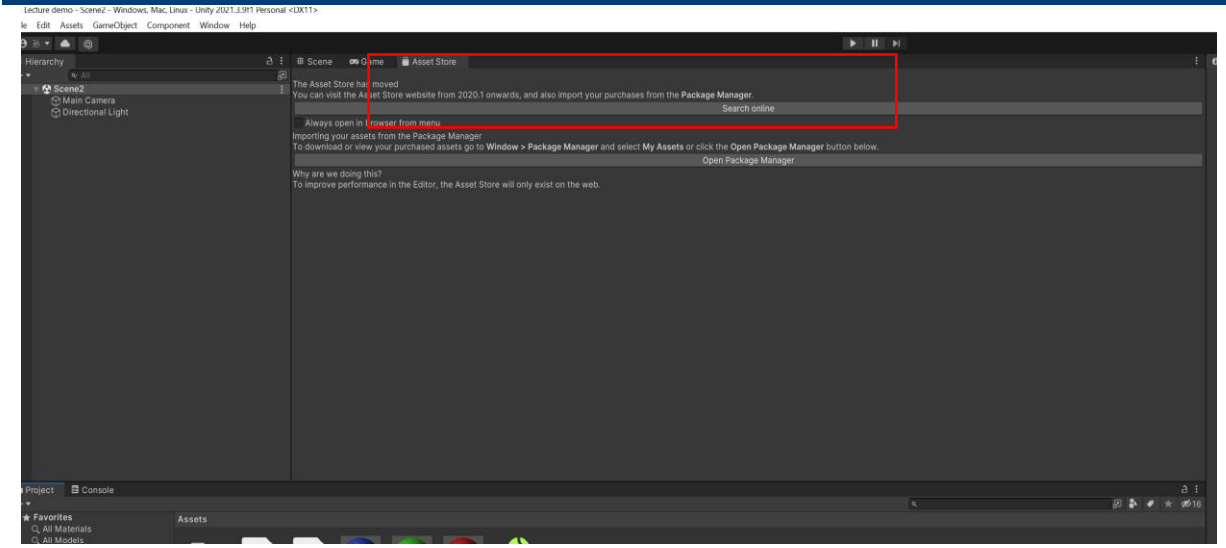
Keep the distance between the ball and camera

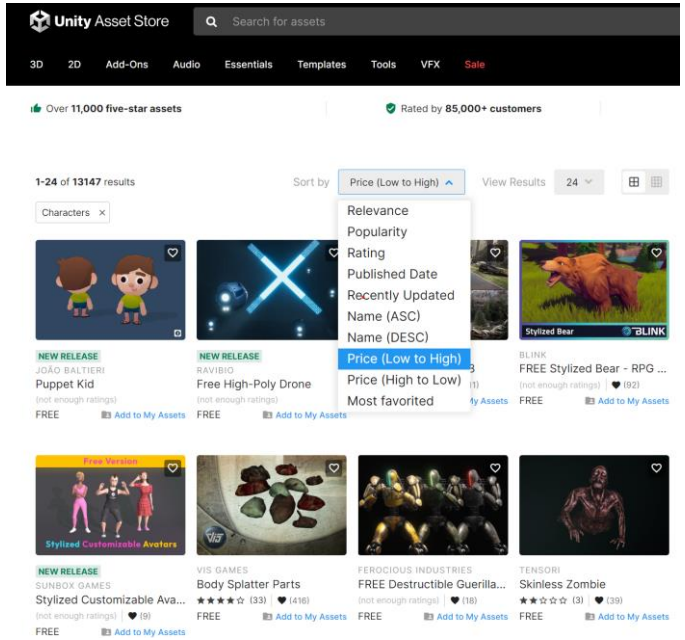


Create some borders (without rigidbody) and objetcs (with rigidbody) in the scene to interect with the ball.

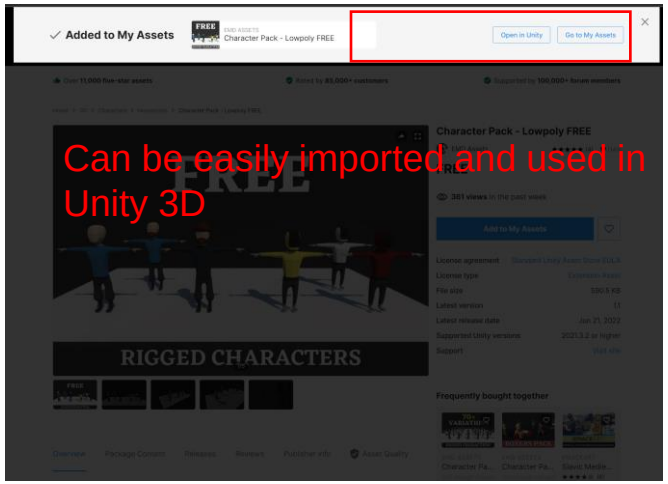


Resources



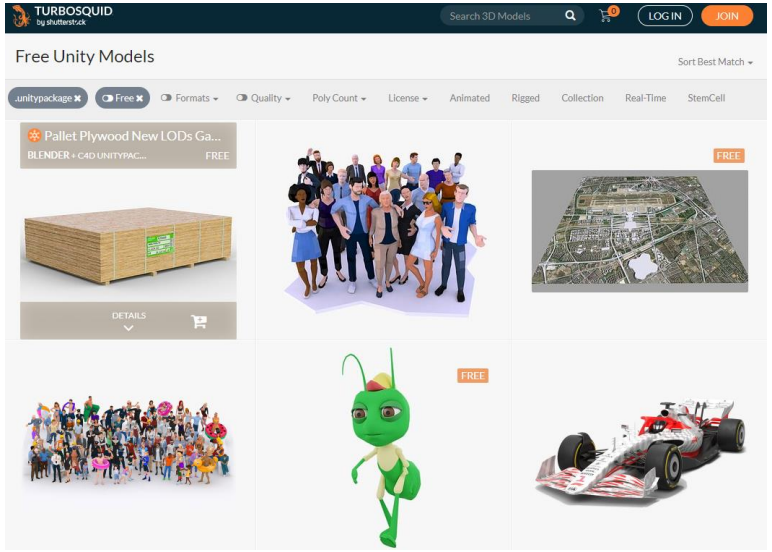


Many free resources are available
In **Unity Asset Store**.



Can be easily imported and used in Unity 3D

Other 3D resources website:
https://www.turbosquid.com/Search/3D-Models/free/unitypackage
……



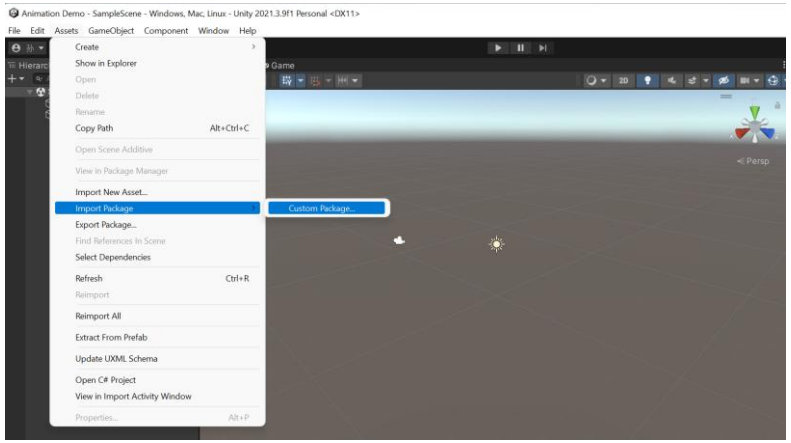
3Dmax

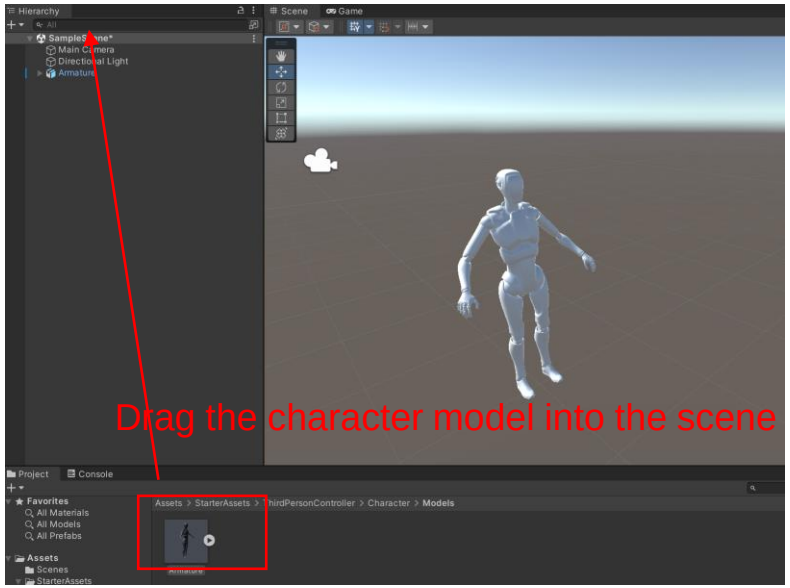
Maya

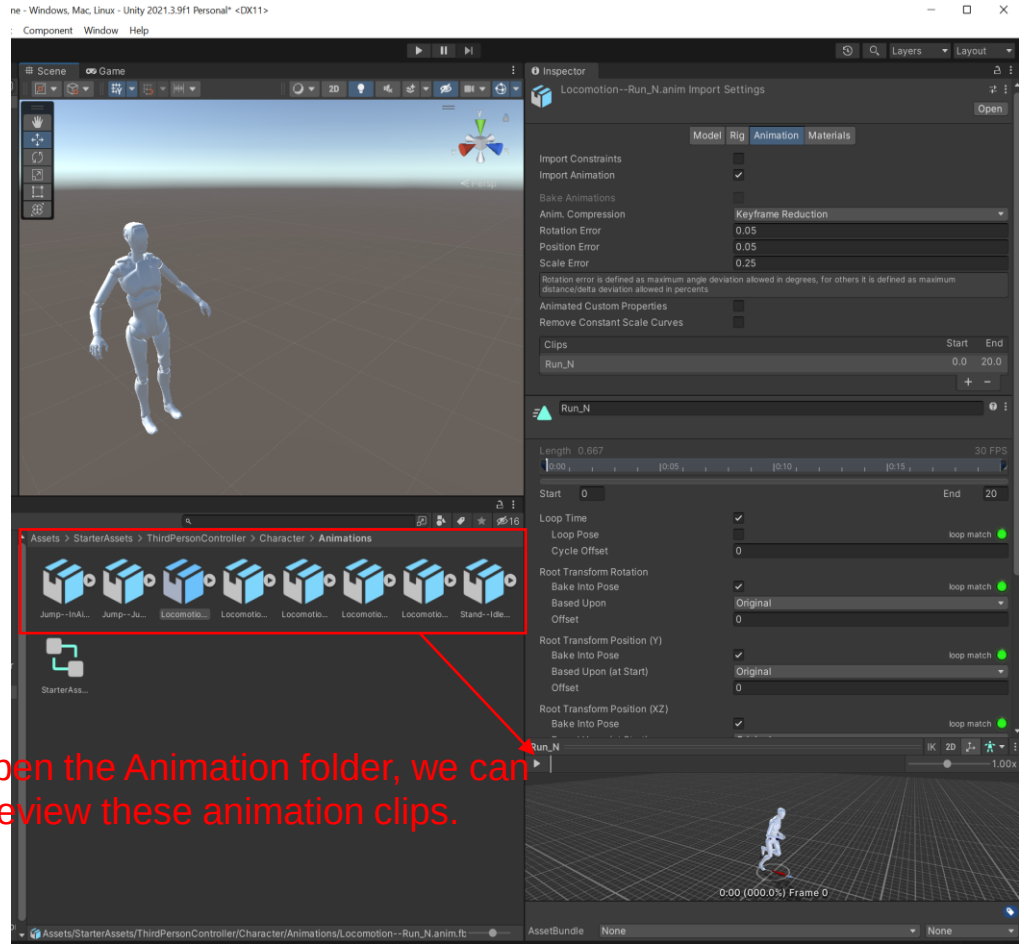
Or designed by yourself.
Not included in this module.

Character and Animation

Import the character package provided in Canvas





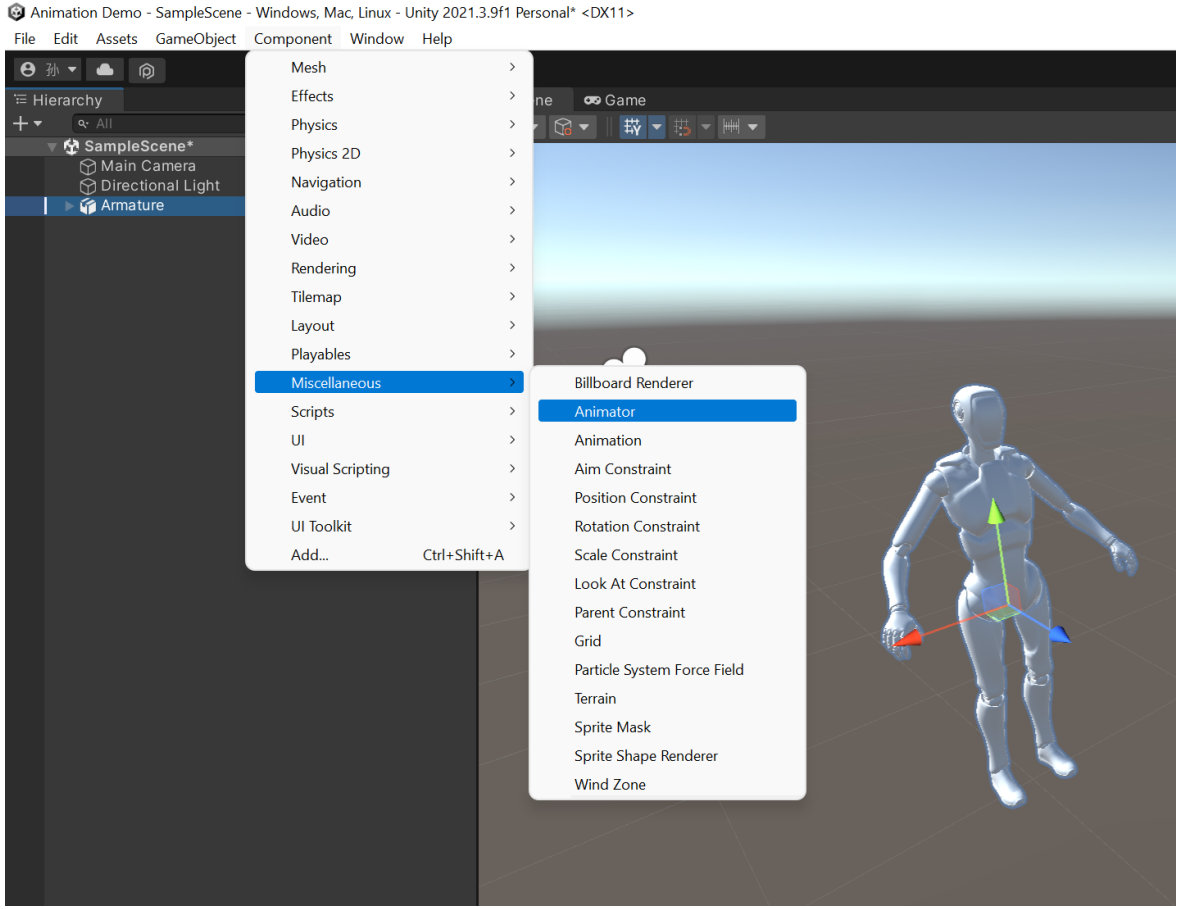


Usually, character model resources in Unity Asset Store will have animations, which are done by other 3D modeling software, e.g., Maya, blender….

Unity 3D also has the animation tool to customized some simple animations.

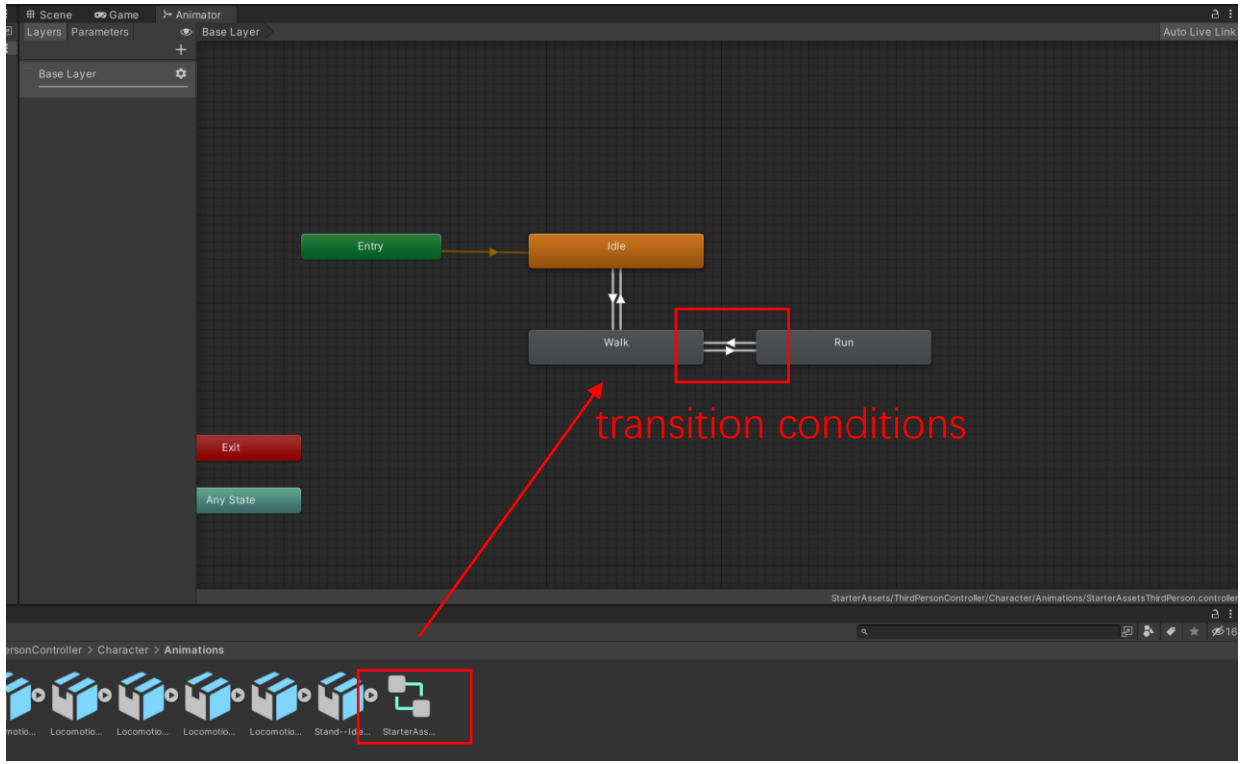
Animator

To control the animations of the character



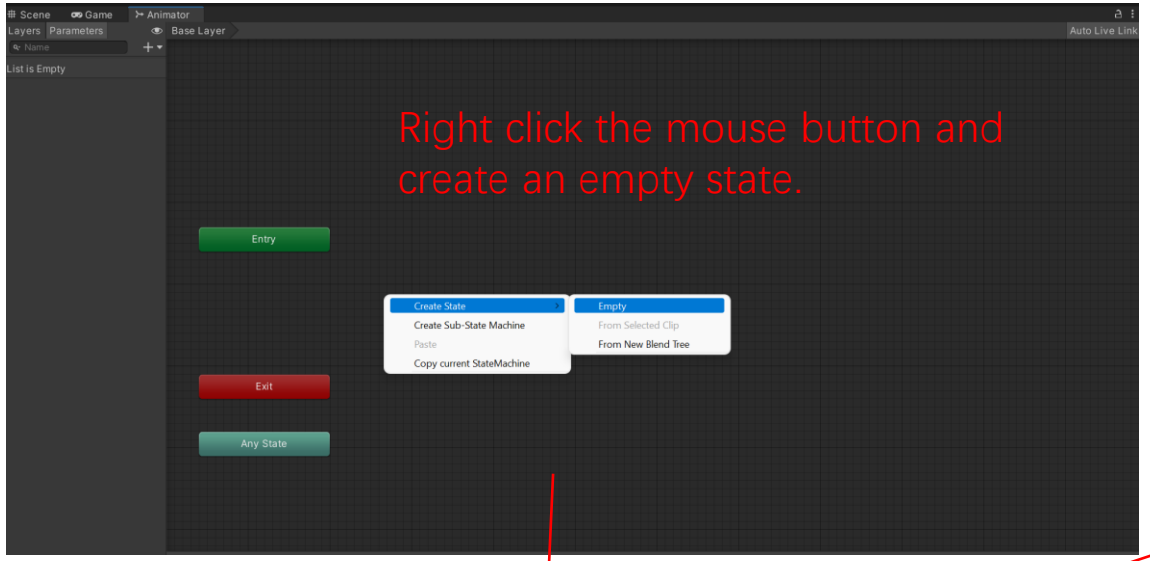
Add an animator to the character.

Animation State Machine

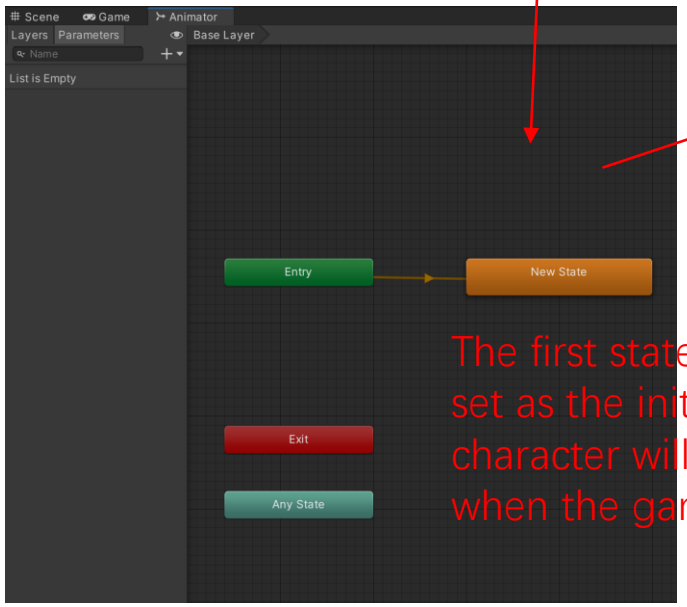


- Each state can be attached with an animation.
- Define the transition conditions to change the state of the character.

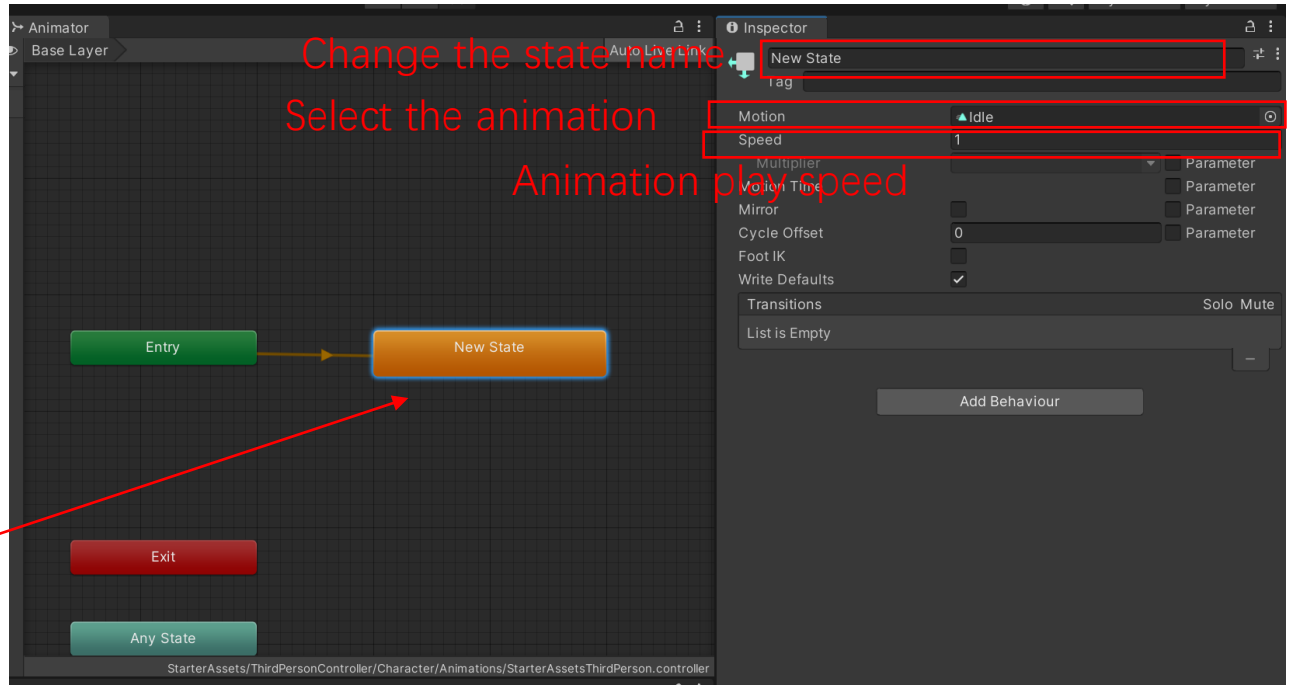
Animator



Right click the mouse button and create an empty state.



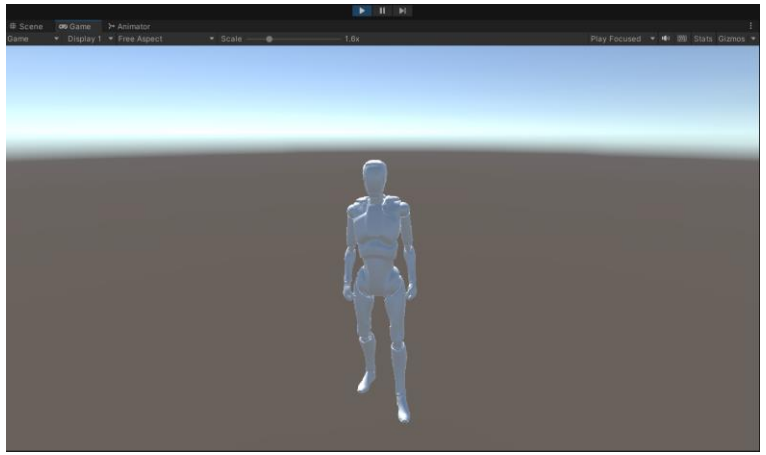
The first state will be automatically set as the initial state, meaning the character will enter into this state when the game starts to run.



Change the state name

Select the animation

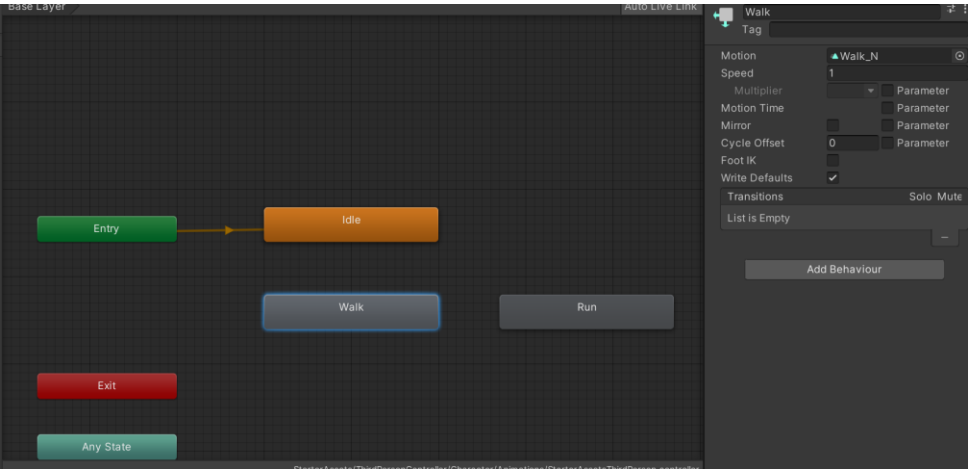
Animation play speed



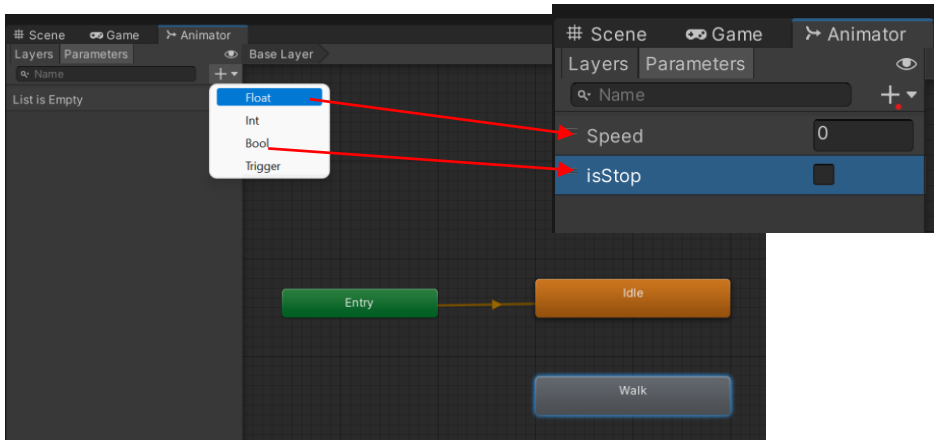
Run game and see the effect.

Animator

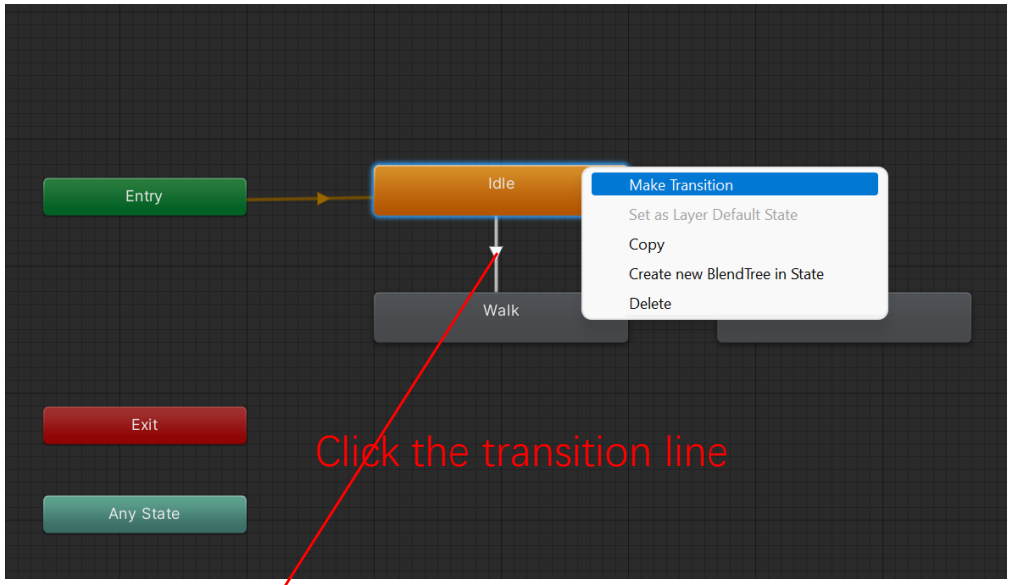
Create other states

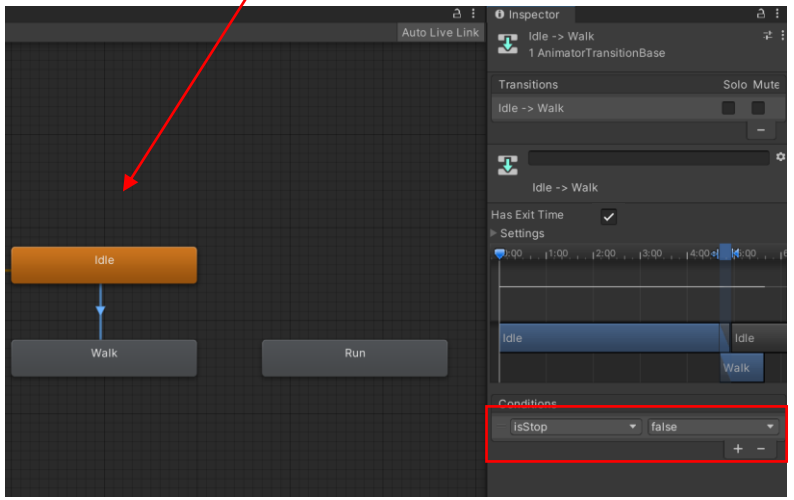


Add transition parameters



Right click the state and select make transition & Link to the next state

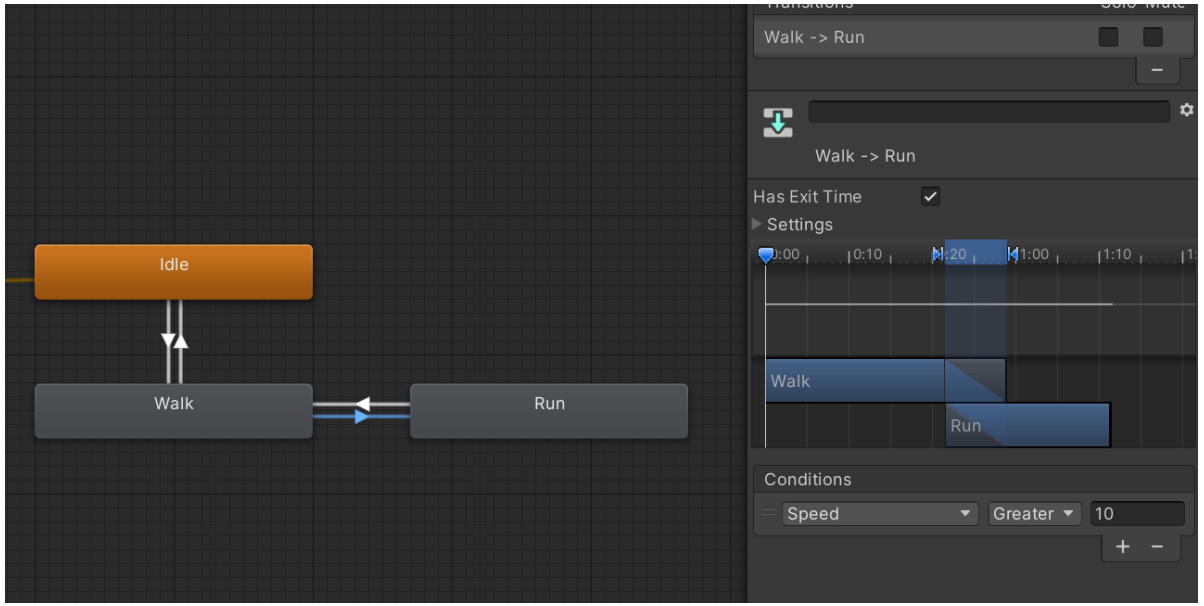




Add the transition parameter.

Animator

Complete other transitions.



Idel —— Walk: isStop false
Walk—— Idel: isStop true
Walk —— Run: Speed > 10f
Run —— Walk: Speed < 10f

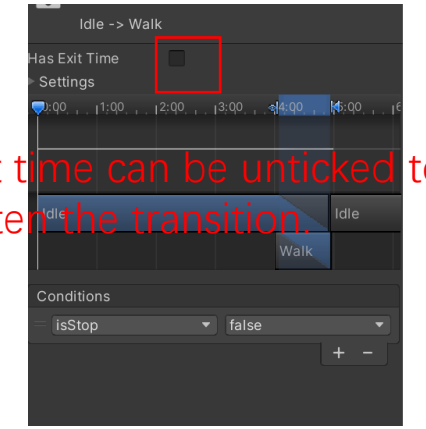
Create a script to control the transition conditions.

AnimationControl.cs

Miscellaneous Files | AnimationControl

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class AnimationControl : MonoBehaviour
6  {
7      private Animator myAnimator;
8      // Start is called before the first frame updat
9      void Start()
10     {
11         myAnimator = this.GetComponent<Animator>();
12     }
13
14     // Update is called once per frame
15     void Update()
16     {
17         if (Input.GetKey(KeyCode.W))
18         {
19             myAnimator.SetBool("isStop", false);
20         }
21         if (Input.GetKey(KeyCode.S))
22         {
23             myAnimator.SetBool("isStop", true);
24         }
25         if (Input.GetKey(KeyCode.Z))
26         {
27             myAnimator.SetFloat("Speed", 15f);
28         }
29         if (Input.GetKey(KeyCode.X))
30         {
31             myAnimator.SetFloat("Speed", 0f);
32         }
33
34     }
35 }
```

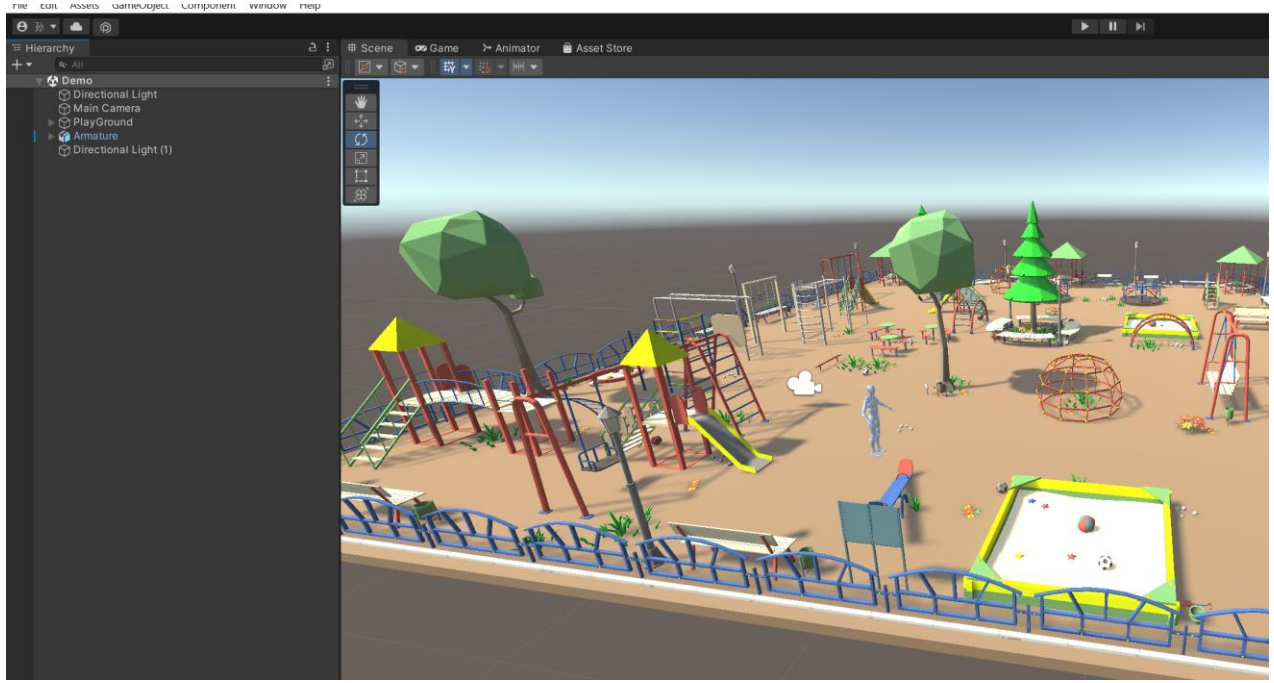
Use keyboard to control



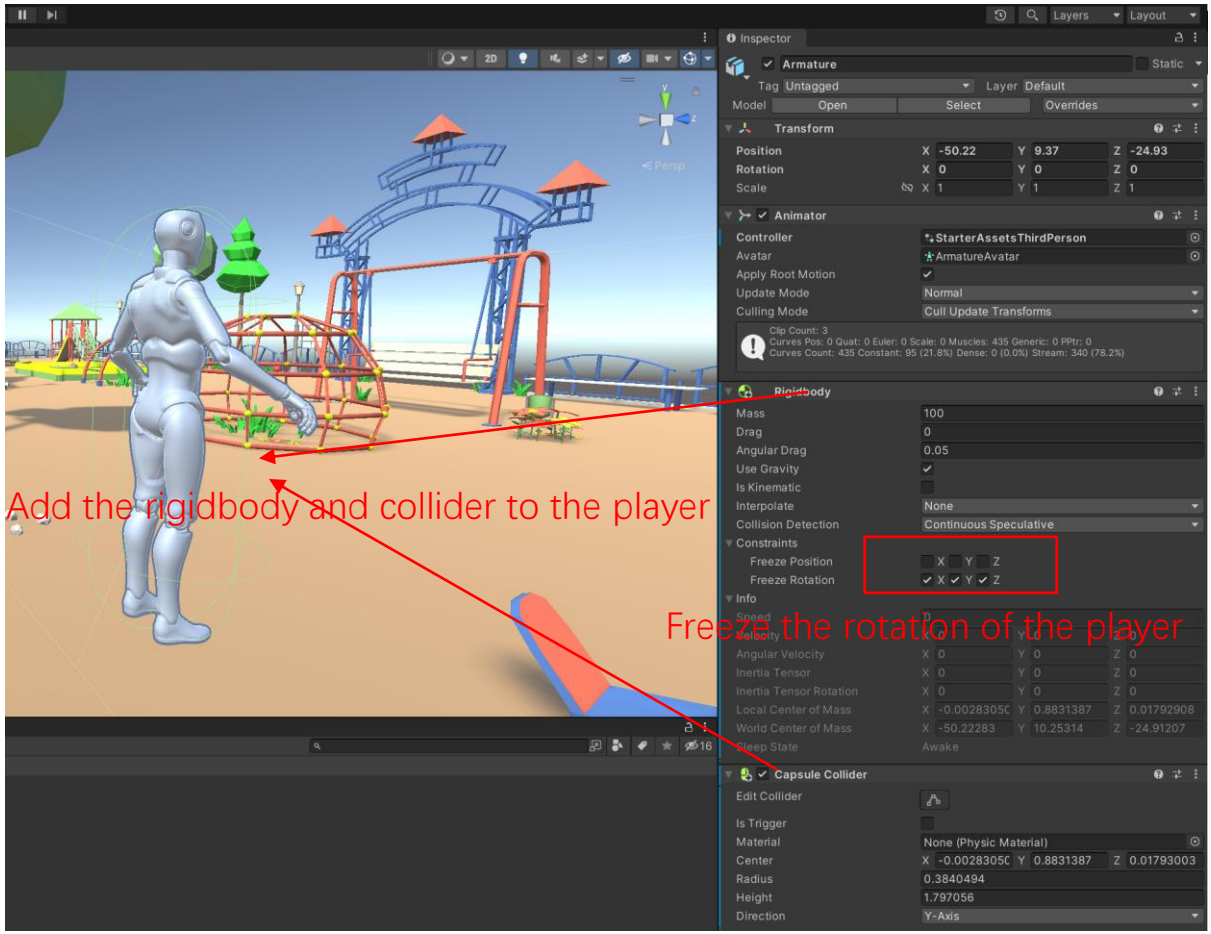
Exit time can be unticked to fasten the transition.

Player movement with animations

https://assetstore.unity.com/packages/3d/environments/playground-low-poly-191533



Import a free environment package from the Unity Asset Store.
Drag the player to an appropriate position of the map.



Player movement with animations

Write a new script to control the movement & animations of the player

MoveControl.cs | AnimationControl.cs

Miscellaneous Files | MoveControl | Start()

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

public class MoveControl : MonoBehaviour
{
    public float moveSpeed = 10.0f;
    public float rotateSpeed = 40.0f;
    public float jumpVelocity = 5f;

    private Animator animator;
    private Rigidbody rigidbody;

    private float hInput;
    private float vInput;

    // Start is called before the first frame update
    void Start()
    {
        animator = GetComponent<Animator>();
        rigidbody = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    void Update()
    {
        hInput = Input.GetAxis("Horizontal");
        vInput = Input.GetAxis("Vertical");
        MoveAndRotate(hInput, vInput);
        Jump();
    }

    void MoveAndRotate(float h, float v)
    {
        if (v > 0) transform.Translate(Vector3.forward * moveSpeed * Time.deltaTime);
        else if (v < 0) transform.Translate(-Vector3.forward * moveSpeed * Time.deltaTime);

        if (v == 0.0f) animator.SetBool("isStop", true);
        else animator.SetBool("isStop", false);

        transform.Rotate(Vector3.up * h * rotateSpeed * Time.deltaTime);
    }

    void Jump()
    {
        AnimatorStateInfo stateinfo = GetComponent<Animator>().GetCurrentAnimatorStateInfo(0);
        if (Input.GetKeyDown(KeyCode.Space))
        {
            animator.SetBool("isJump", true);
            rigidbody.AddForce(Vector3.up * jumpVelocity, ForceMode.VelocityChange);
        }

        if (stateinfo.IsName("Jump") && stateinfo.normalizedTime > 1.0f)
        {
            animator.SetBool("isJump", false);
        }
    }
}
```

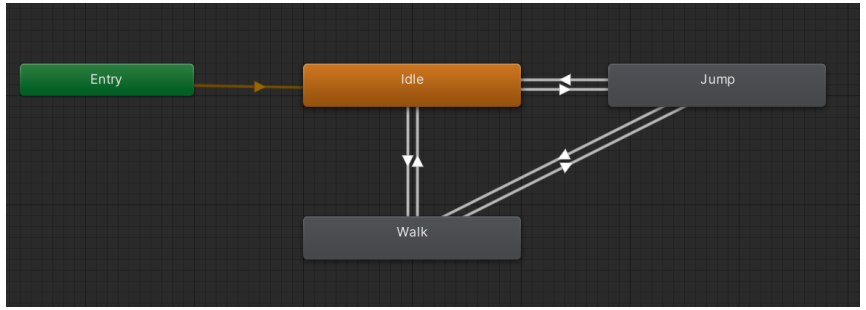
Time.deltaTime: Convert frame-updated displacements to time-updated displacements.

Move and rotate by arrow keys

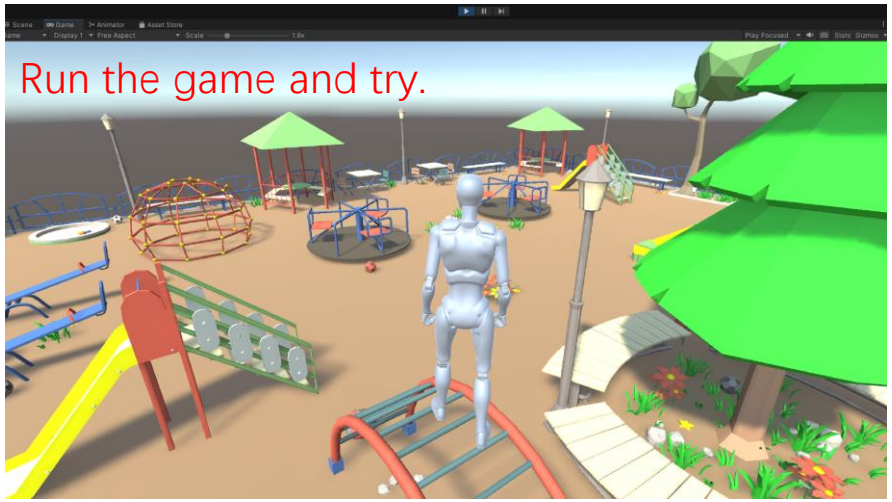
GetKey: Press and trigger continuously.
GetKeyDown: Press to trigger once.

Add a force to enable jump

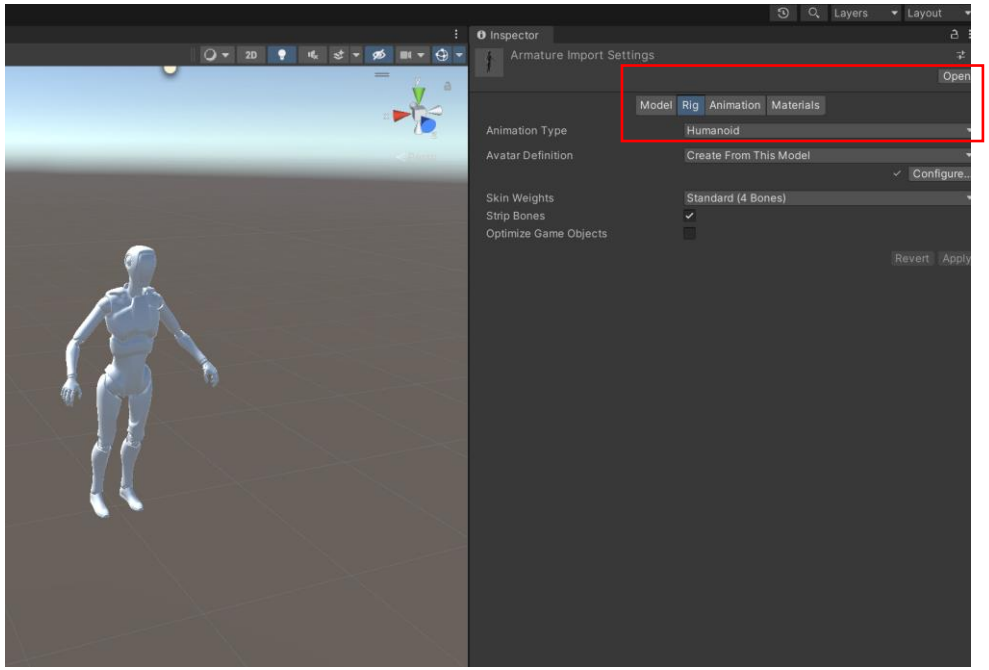
Whether the animation has finished a loop.



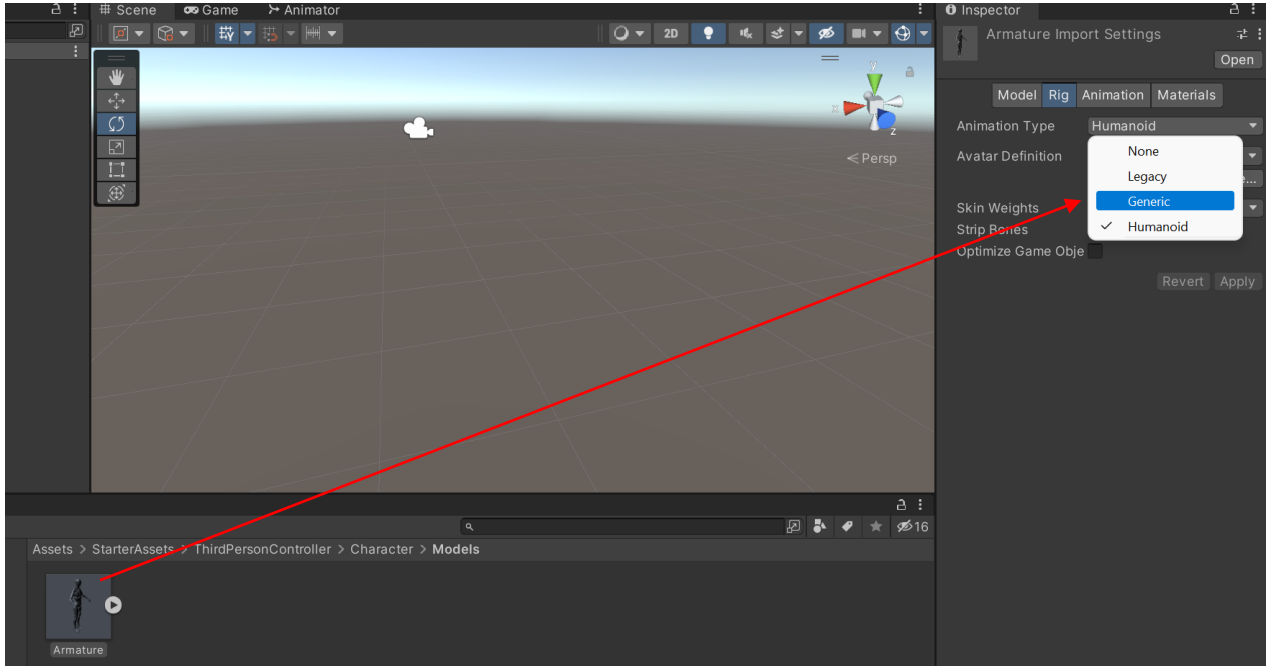
Use “isStop” and “isJump” to switch between three states.



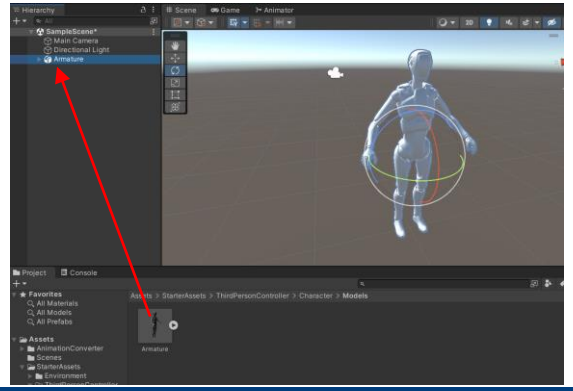
Create new animations with Unity 3D



- Humanoid characters are commonly bonded with humanoid animations.
- Animations created based on Humanoid rig can be applied to other humanoid characters, not limited to this one. (Commonly created by other 3D software: blender, maya….)
- Unity 3D only support creating generic animations, which can only be used for the specific model that used to create the animation.

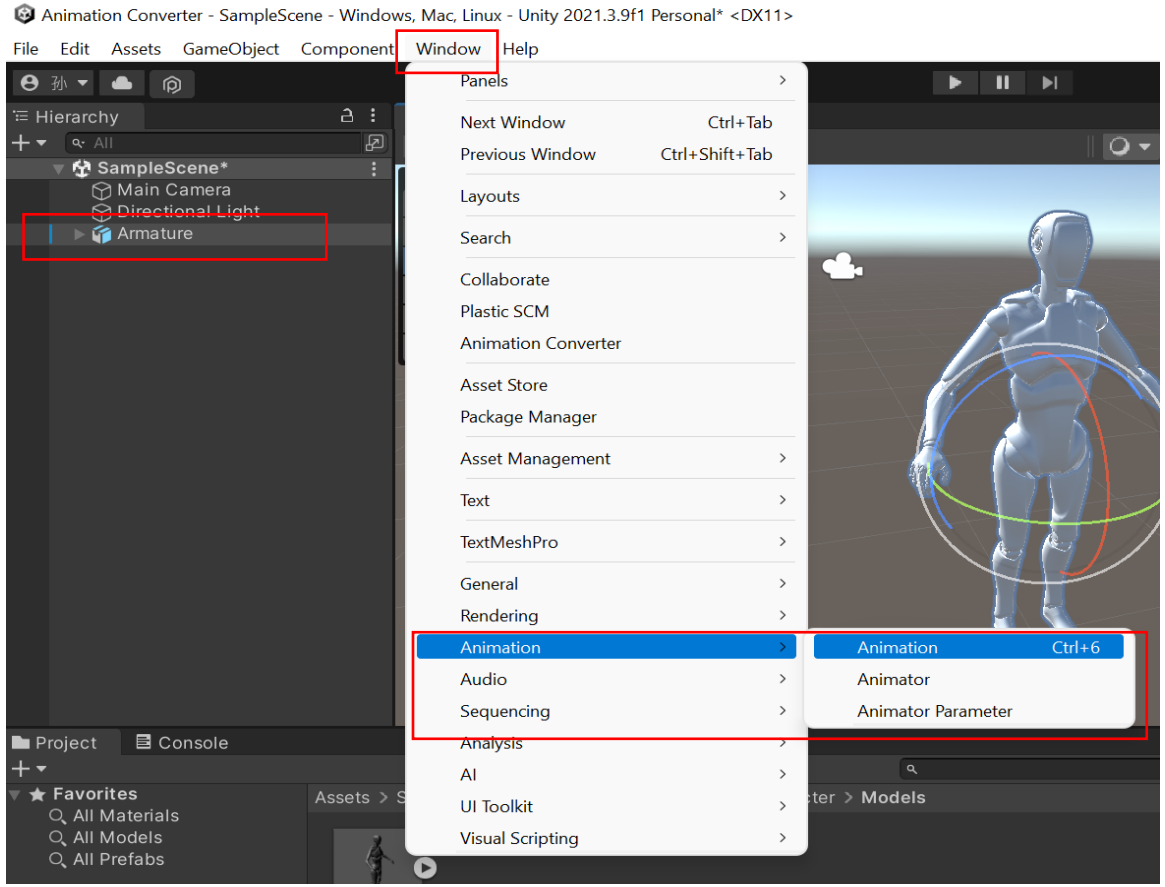


In the resource panel, we click the character model and change the animation type from humanoid to generic.



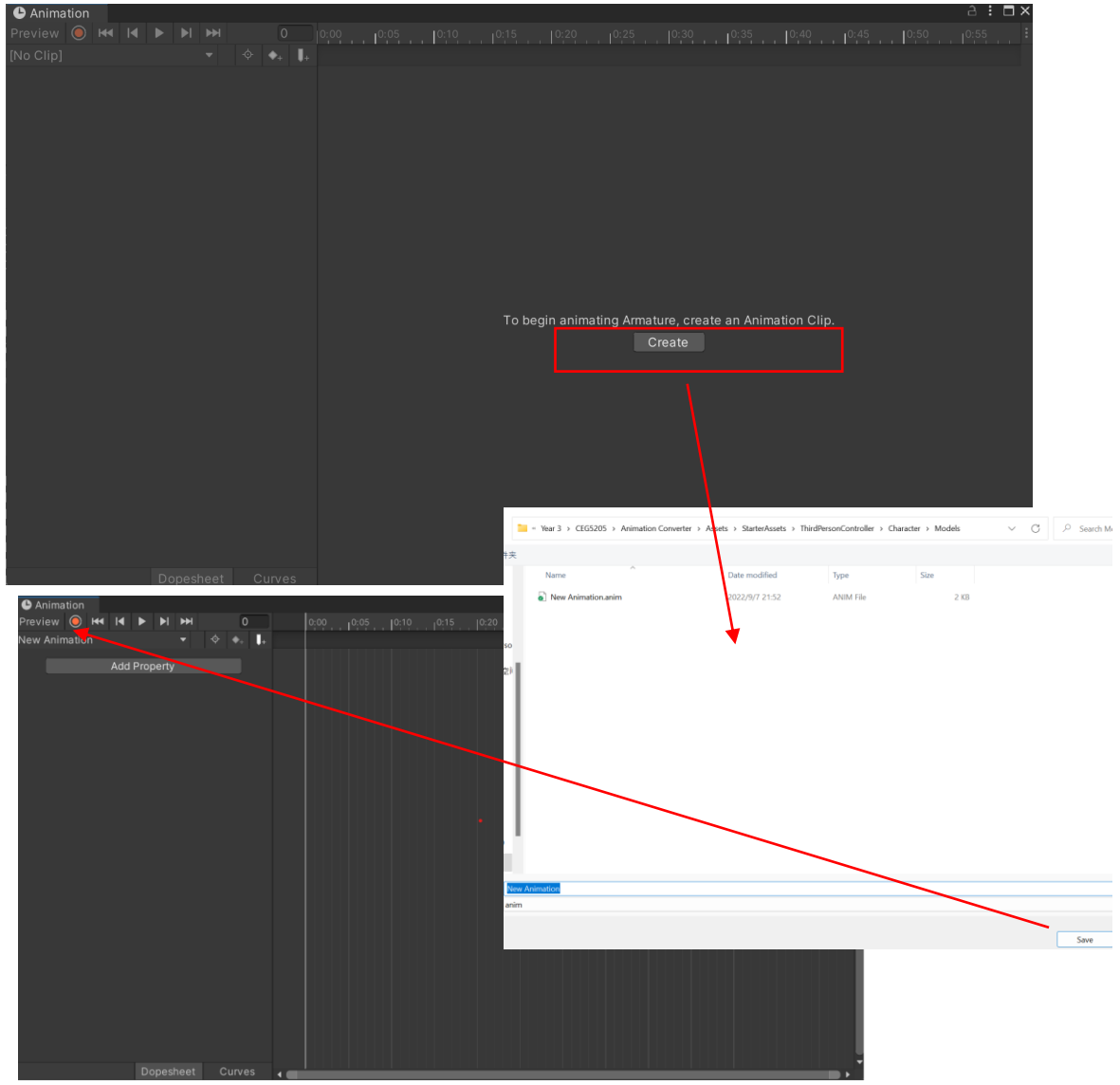
Then drag the generic model into the scene.

Create new animations with Unity 3D

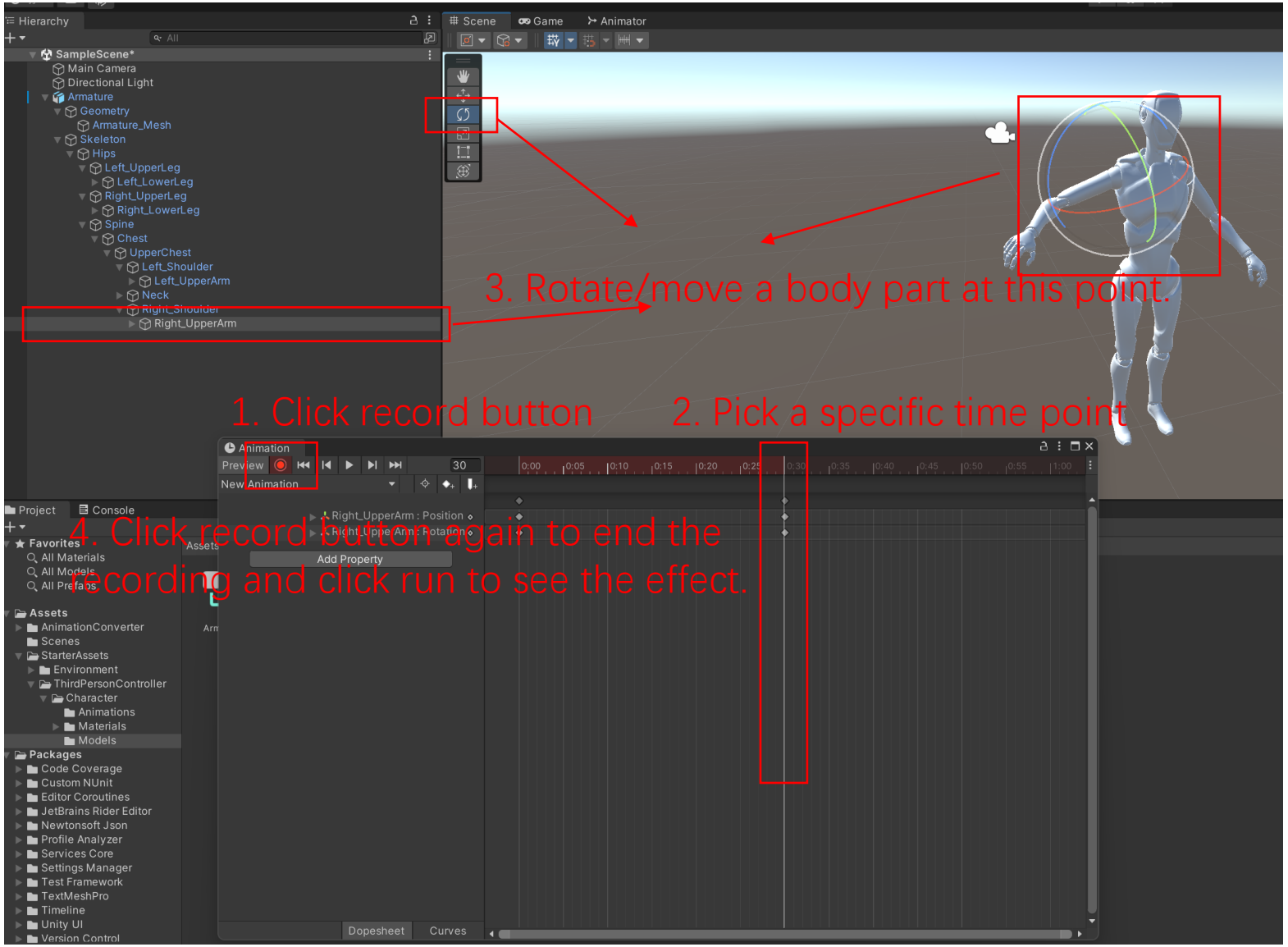


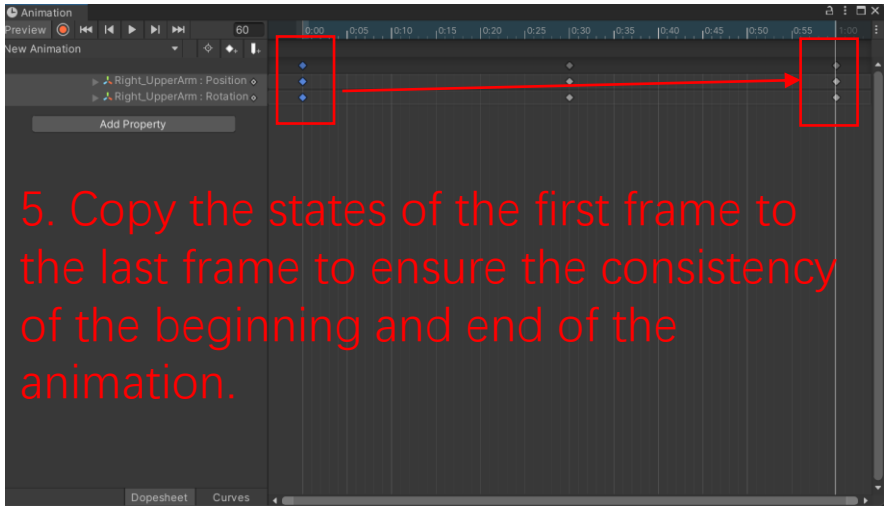
Select the character in Hierarchy, then click window – animation – animation.

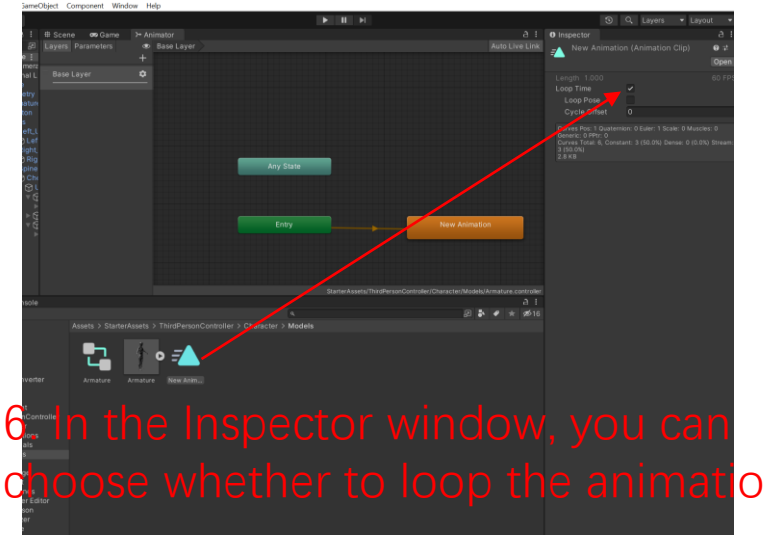
In the pop-up window, we can start to create new animation clips.



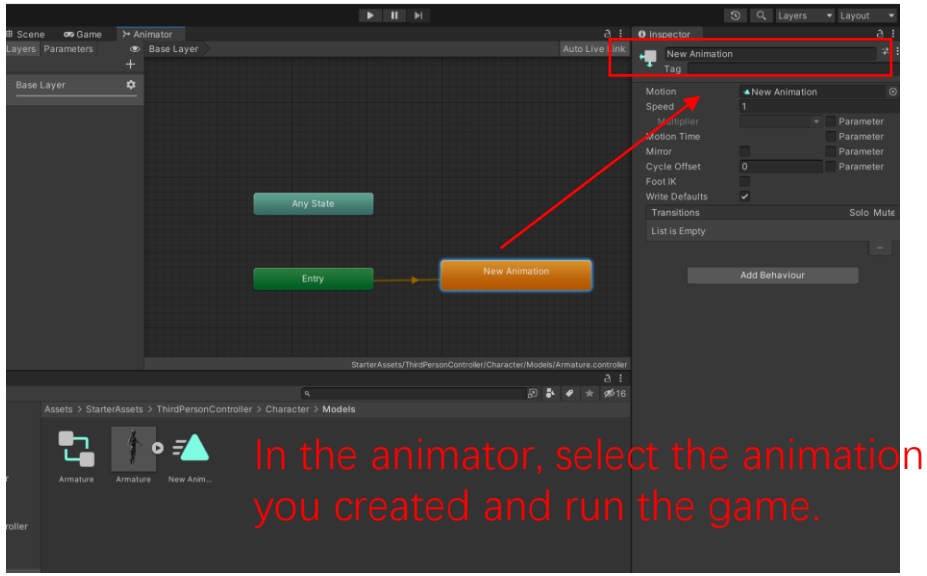
Create new animations with Unity 3D

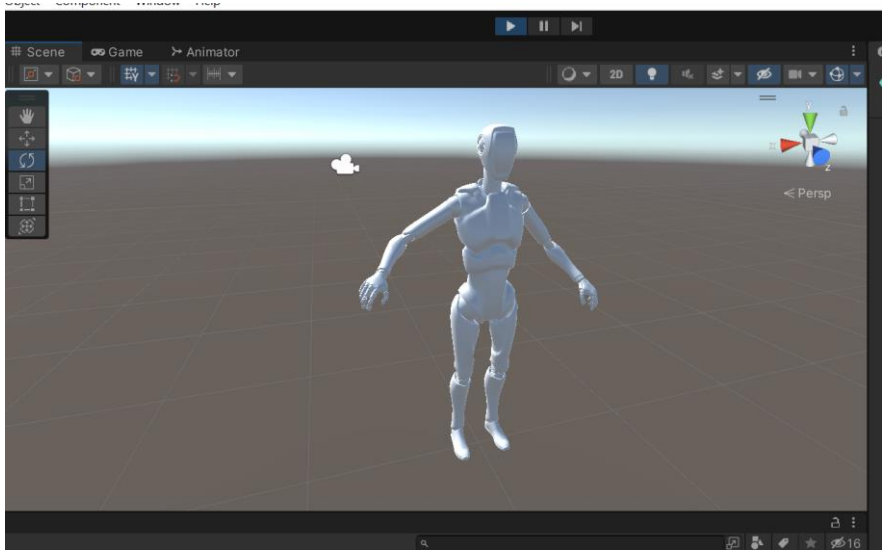


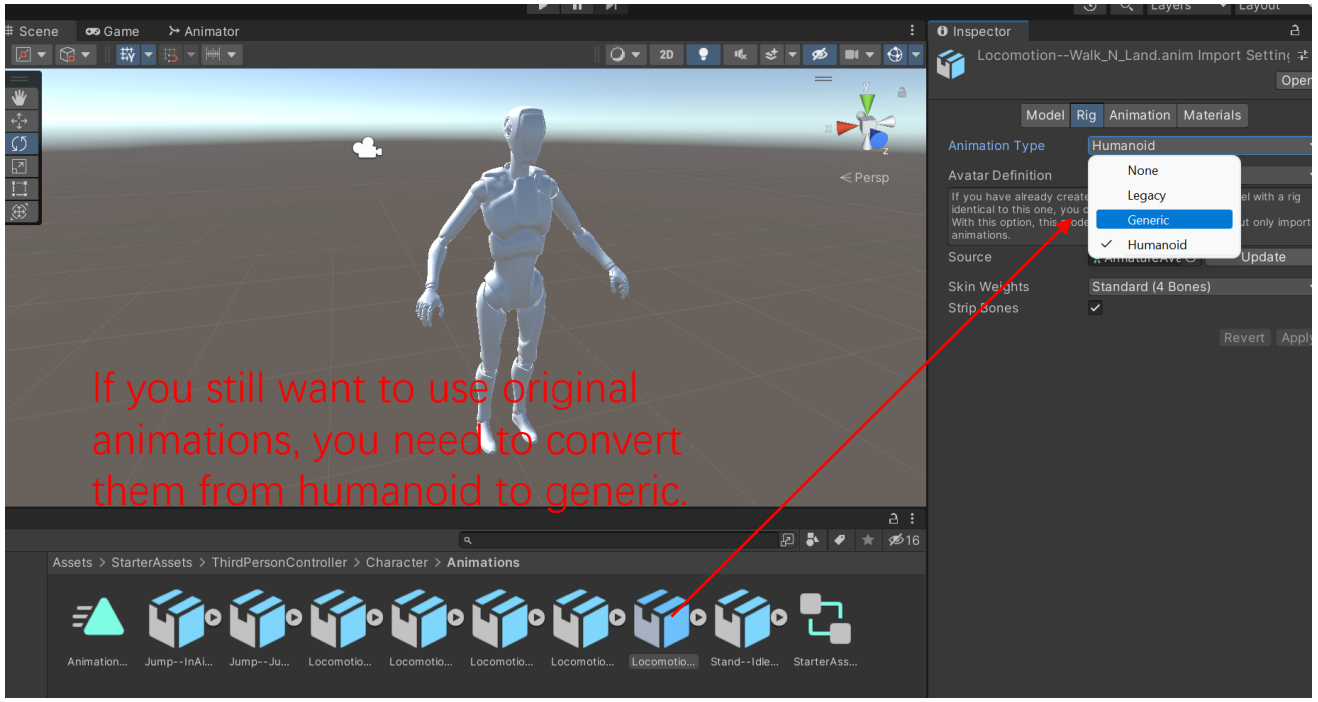


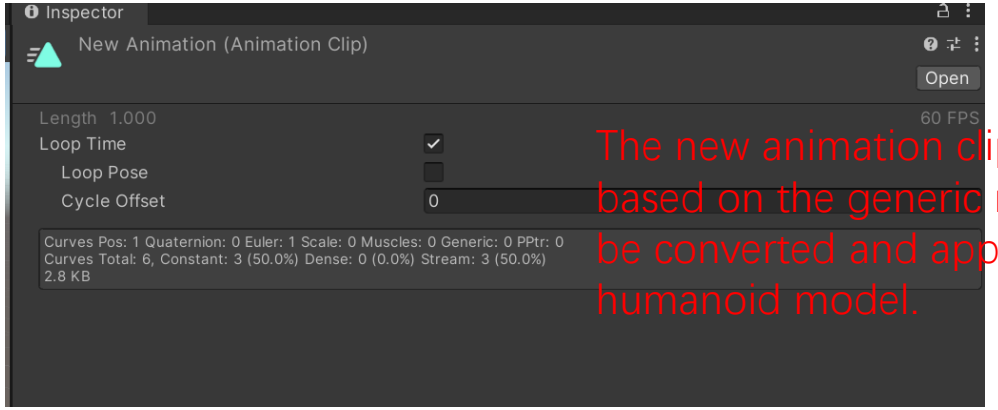


Create new animations with Unity 3D









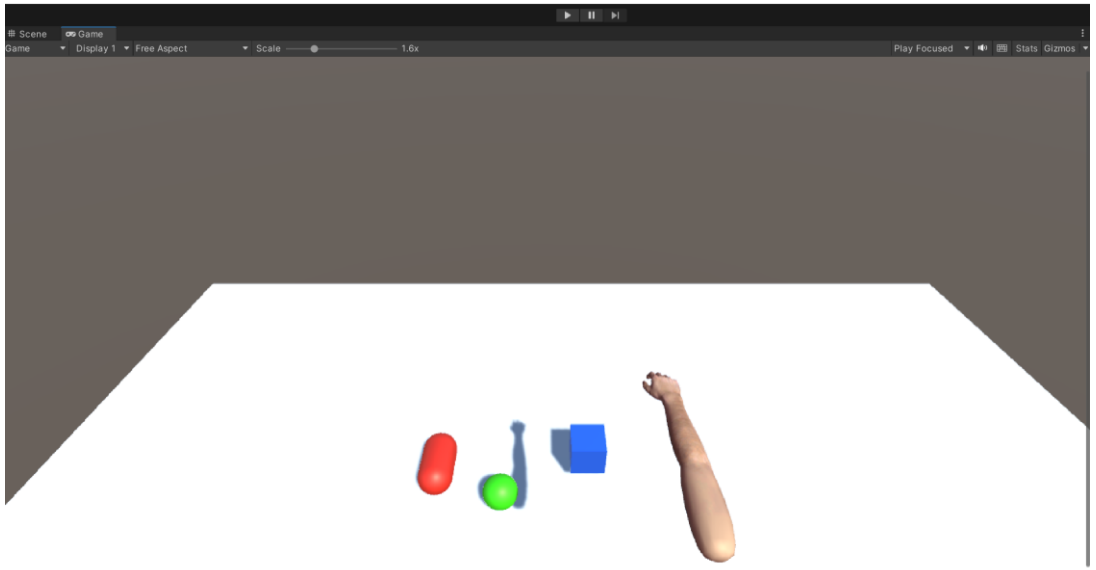
Homework – Lab2

1. Establish data communication and get the sensor signal data in Unity 3D.

In Lab 1, you can read the IMU signal via python interface. Here, you need to write a script and try to establish the communication between the Unity 3D and Python interface. The IMU signal data can be accessed in Unity 3D and used for real-time control. (Tips: TCP-IP, etc., …….Submit the script code.)

2 . Real-time control demo by IMU.

With the IMU data, you can try to control the movement of the VR hand, and interact with some small objects. (A video to show the result.) The VR hand model is available in Canvas.



Task 2.1: Real-time movement control of the hand by IMU. (Can use keyboard if Task 1 is not completed)

Task 2.2: Create the grip animation.

Task 2.3: When the hand is close to the object, use the grasp animation to grab the object. The object can follow the hand's movement, and the object will fall to the plane when the fingers are opened. (Tips: The trigger function of the collider.)

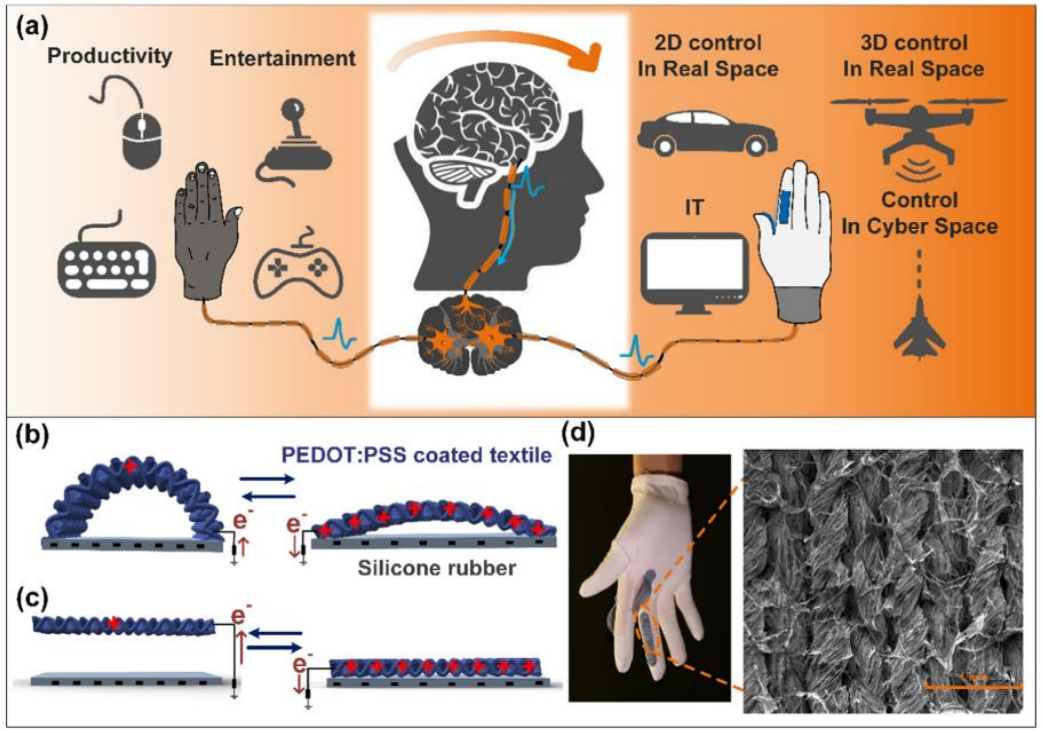
Task 2.4: Define 2 more animations to enable more interactions. E.g., throw away the object in your hand, slap objects on the plane……

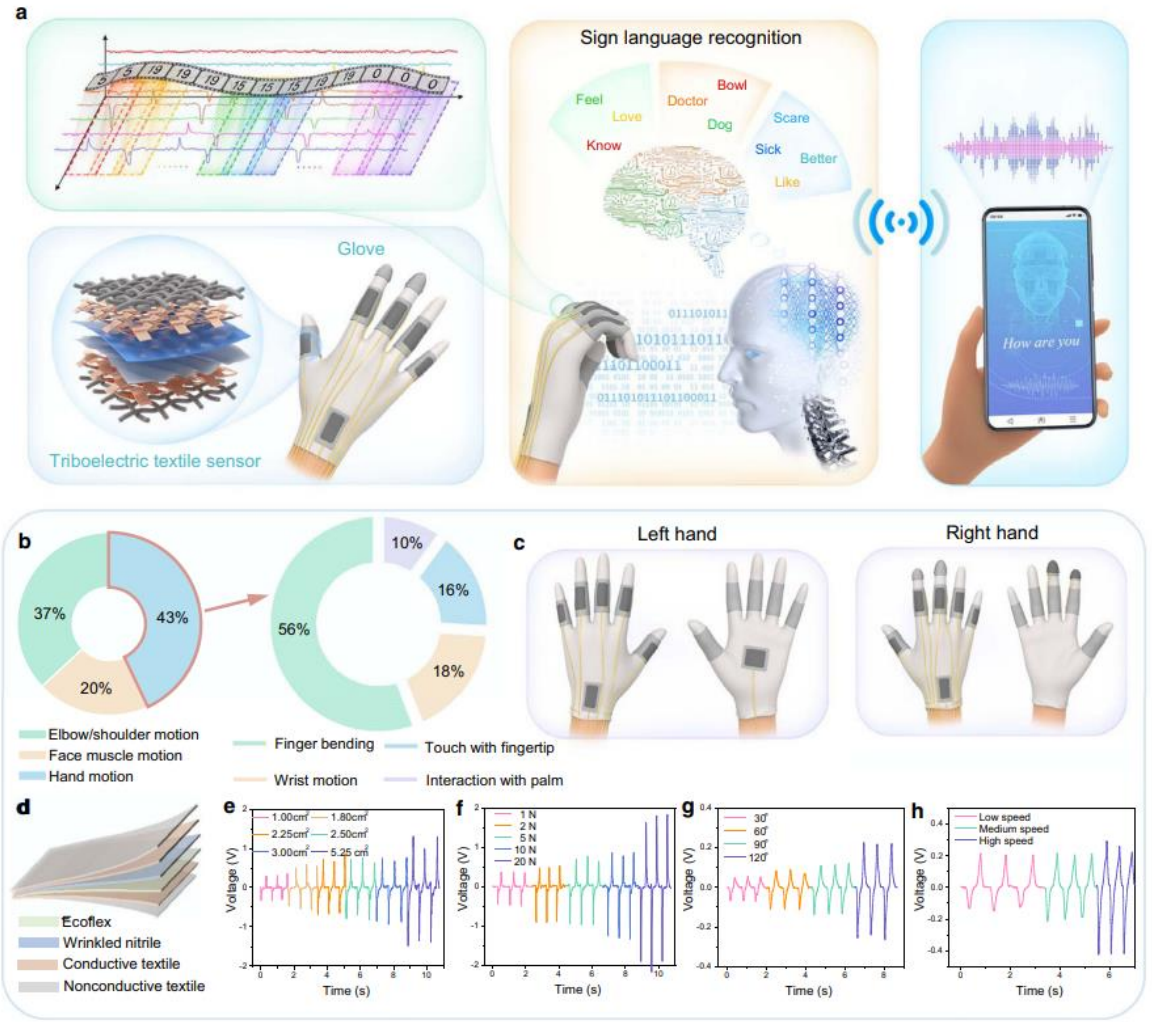
Examples of real-time control for the final project

No.	Tasks	Weightage
Task 1	Sensor system + Wireless transmission and real-time visualization of the sensor signal. ❖ Sensor design, performance, channels. (Arduino-wireless module-Python)	30%
Task 2	Task 2A: Use sensor signal to directly control the motion or movement of the virtual object in Unity 3D. Task 2B: Use sensor signal to trigger the animations defined in Unity 3D. ❖ Comprehensive consideration according to the application scenario and control complexity. (Arduino-wireless module-Python-Unity 3D) Complete one of these two.	40%
Task 3	Achieve real-time gesture/object/… recognition with machine learning analytics. ❖ Number of categories and accuracy. (Arduino-wireless module-Python)	15%
Task 4	Send the ML result to VR space to implement advanced interactions. ❖ Not necessary to include all categories in Task 3. (Arduino-wireless module-Python-Unity 3D) A higher-level Task 2, Completing task 4 can be seen as completing task 2.	15%
Bonus	▪ Improved sensor design to enhance sensing performance. ▪ AR demos are supported in Task 4, but not included in this course. A good AR demo can also give bonus points. ▪ Innovative VR application scenarios not mentioned in this course and this example slides. ▪ Multimodal sensor fusion, e.g., IMU + finger sensor for sign language recognition…	

Example 1: Glove-based system

System overview





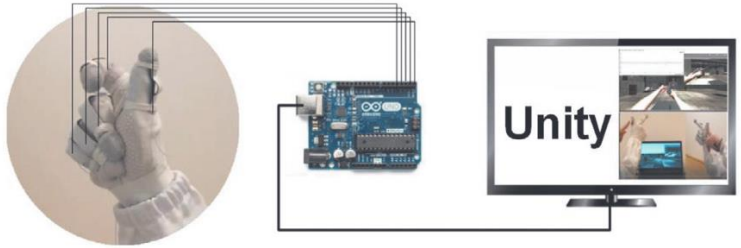
Textile based sensor for finger motion monitoring and gesture recognition.

References:

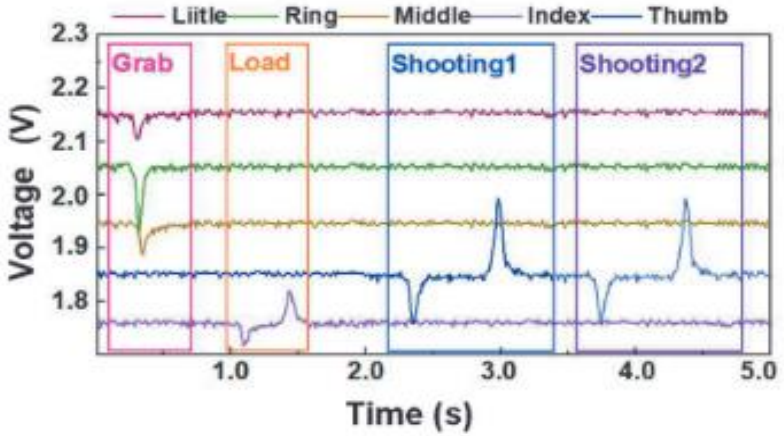
1. Nano Energy, vol. 58, pp.641-651, 2019.
2. Adv. Sci, vol.7, no.14, 2000261, 2020.
3. Nat. Commun., vol.12, 5378, 2021.

Example 1: Glove-based system

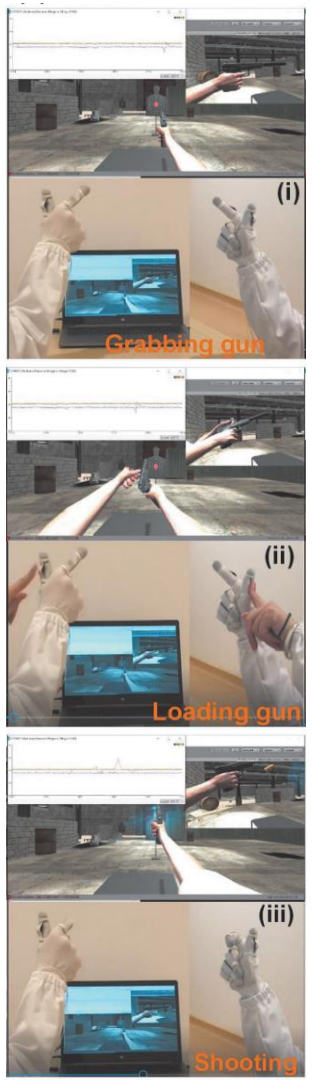
Task 1&2B: Real-time Control - Animation



Wireless transmission



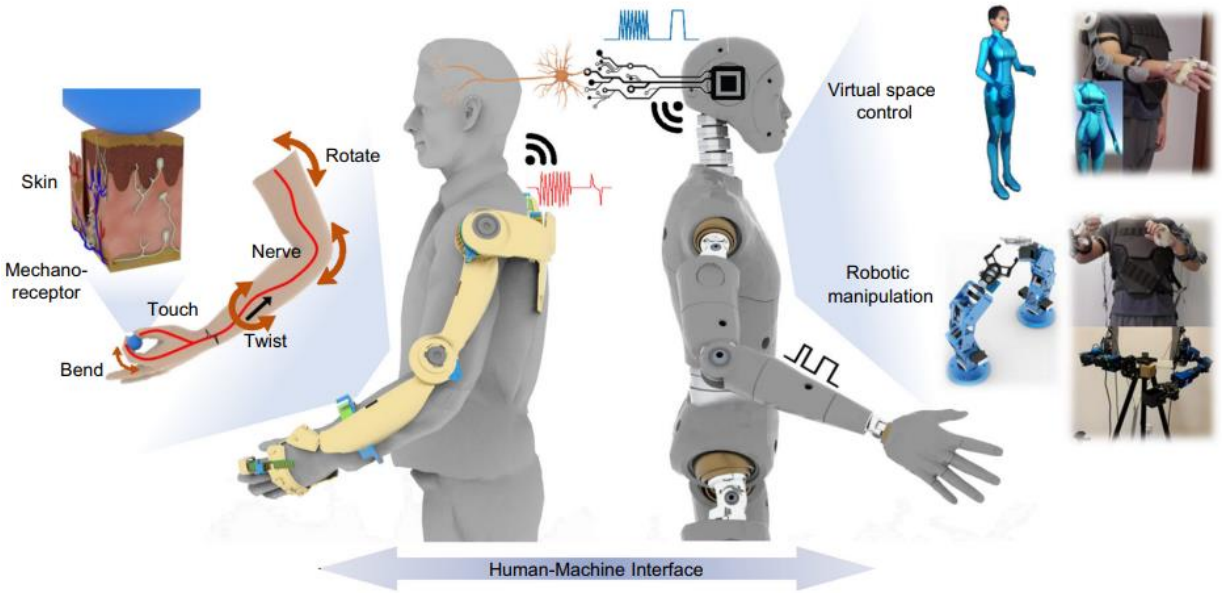
Use the sensor signal directly trigger the animations defined in the virtual space.

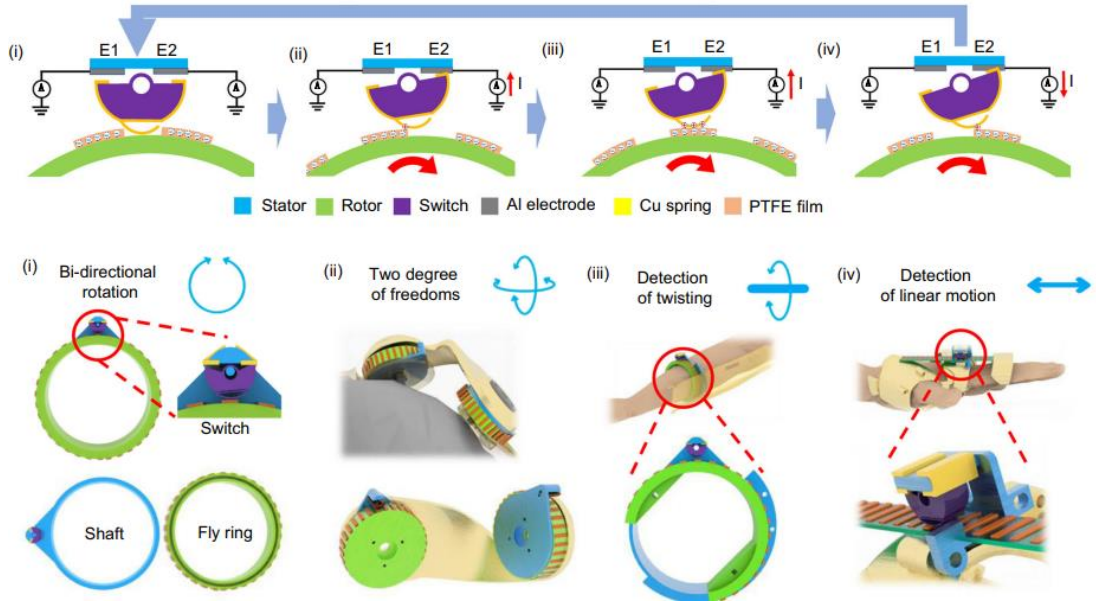


Shooting Game in VR Space

Example 2: Exoskeleton manipulator

System overview



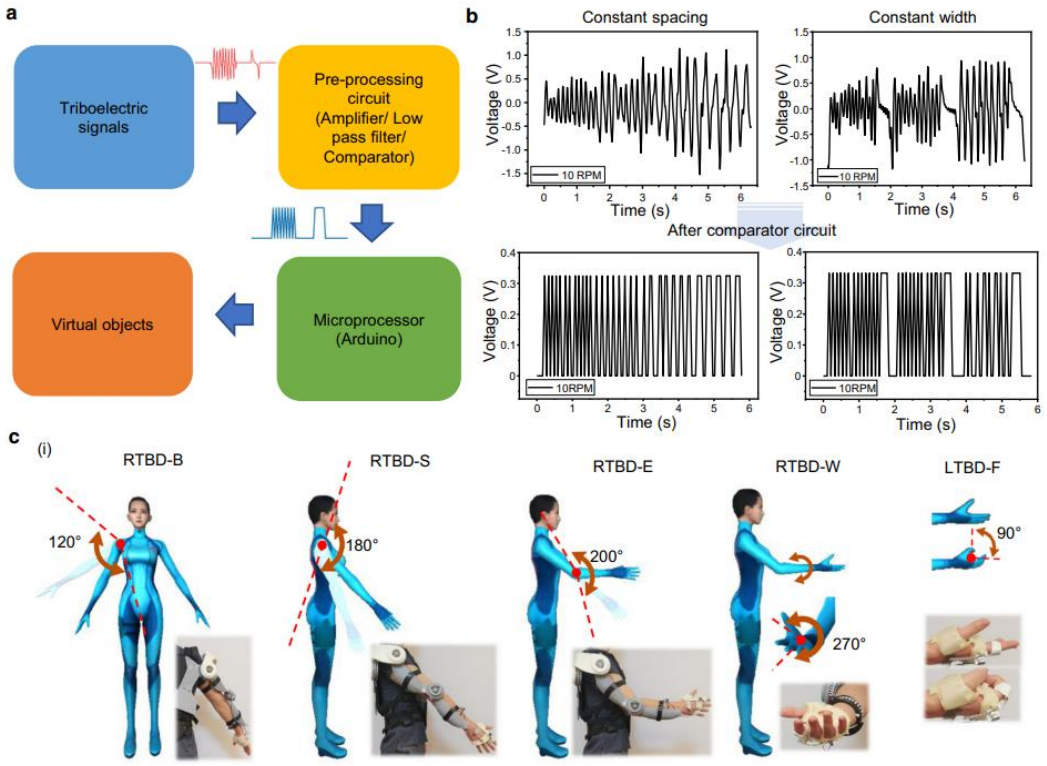


A triboelectric bidirectional sensor as a universal and cost-effective solution to a customized exoskeleton, monitoring movements, including two DOF rotations of the shoulder, twisting of the wrist, and the bending motions for controlling the virtual character and the robotic arm in real-time.

References:
1. Nat. Commun., vol.12, 2692, 2021.

Example 2: Exoskeleton manipulator for VR

Task 1&2A: Parallel control – Direct motion control

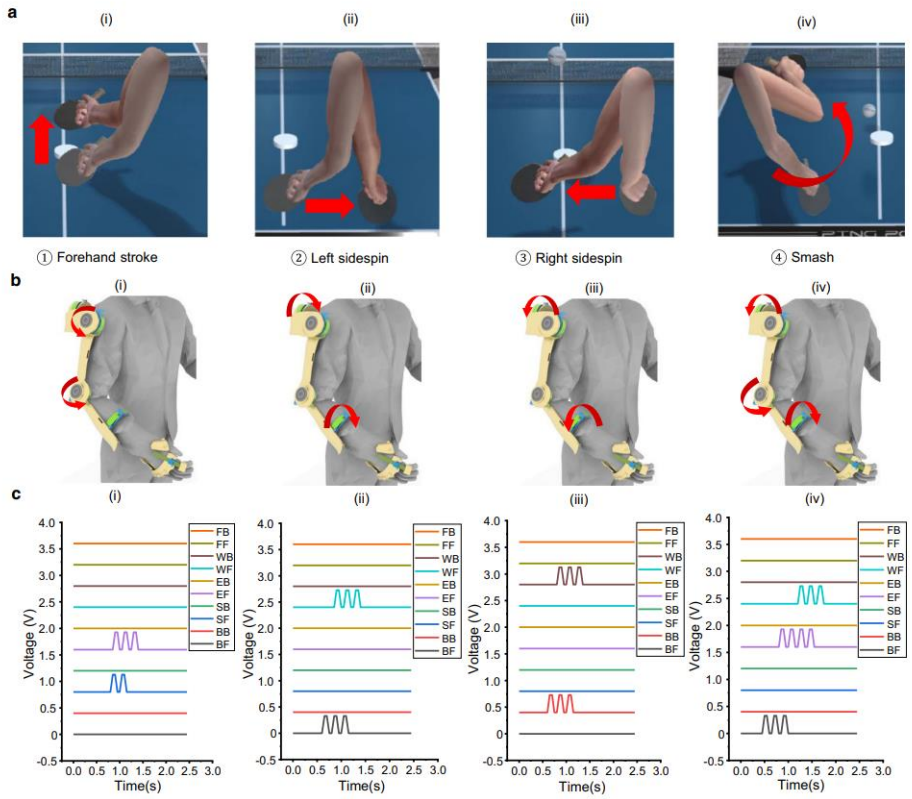


Virtual space demonstrations in Unity

Use sensor signal to realize the parallel control of the virtual avatar.

Example 2: Exoskeleton manipulator for VR

Task 1&2B: Ping-pong demo - Animation



Virtual space demonstrations in Unity

Use sensor signal to trigger the animation of the virtual avatar.

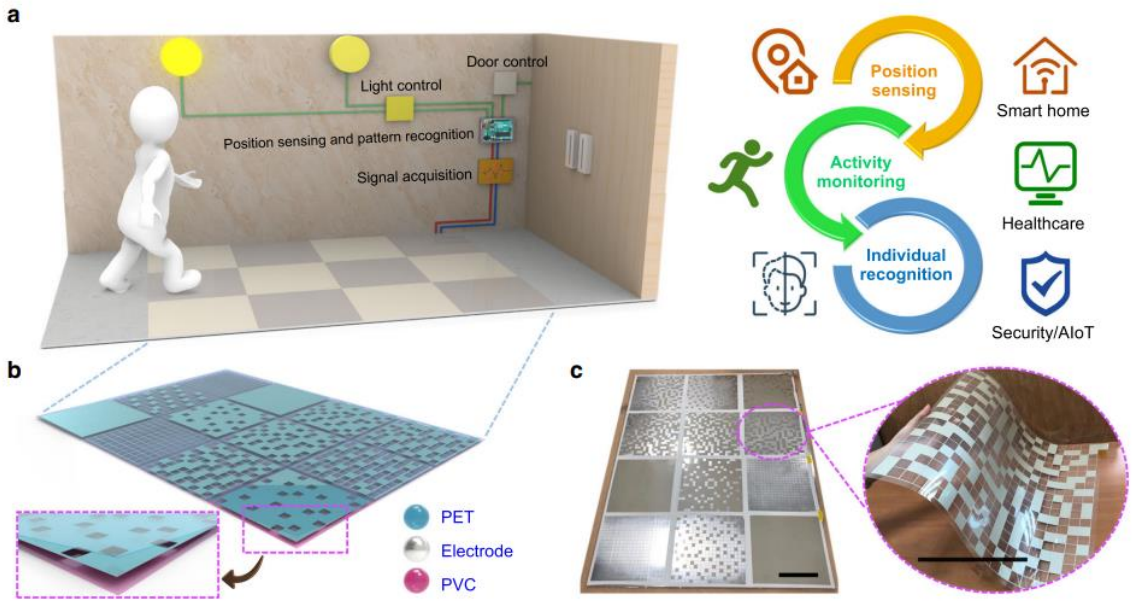
Example 3: Smart mat for smart home applications

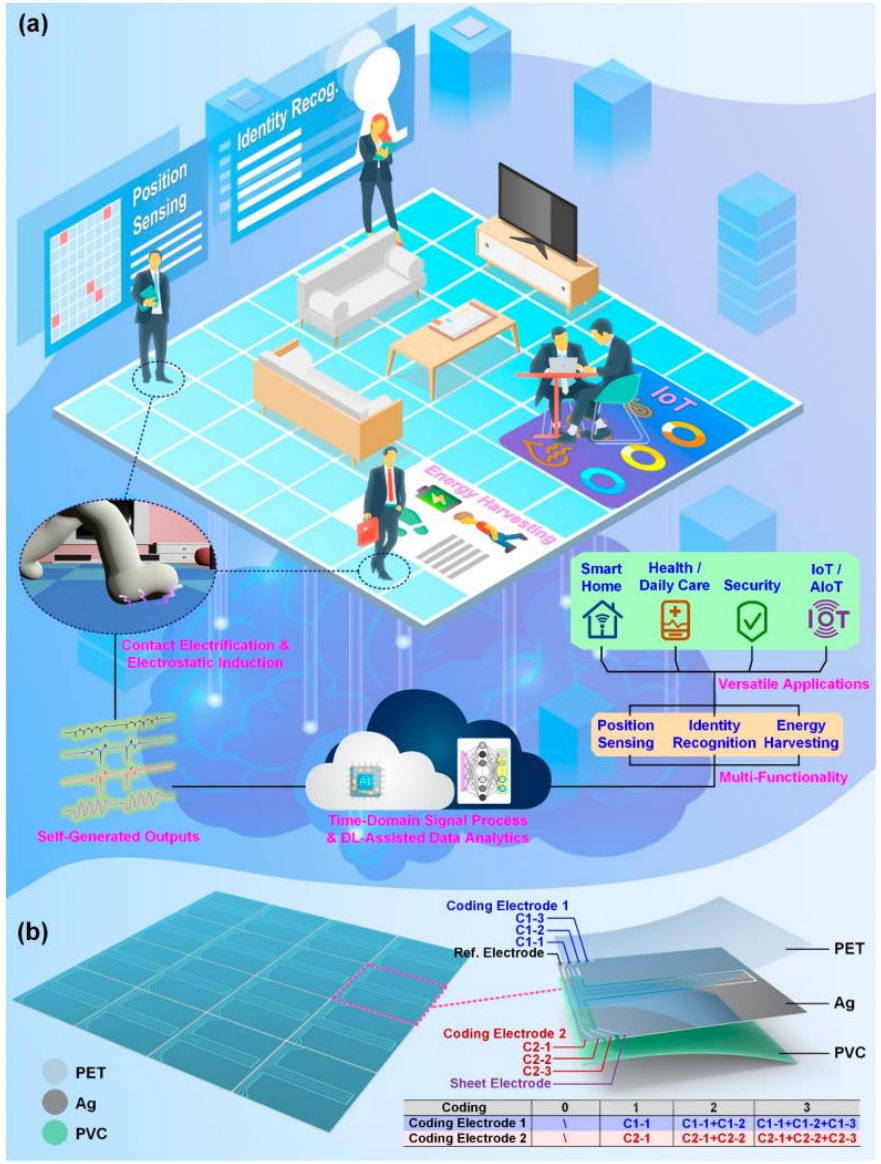
System overview

- A smart floor monitoring system through the integration of self-powered triboelectric floor mats and deep learning-based data analytics.
- The stepping position, activity status, and identity information can be determined according to the instant sensory data analytics.

References:

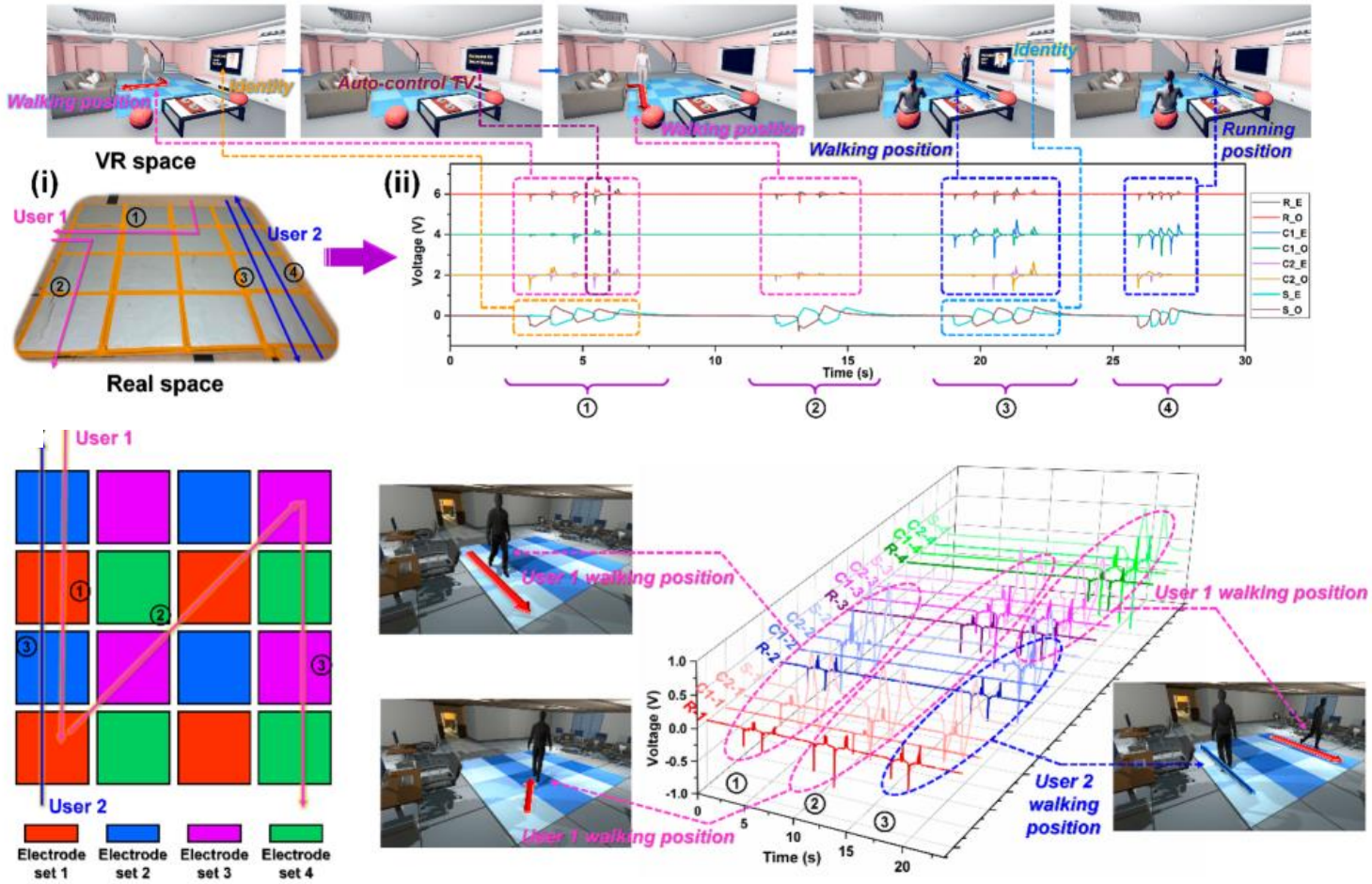
1. Nat. Commun., vol. 11, 4609, 2020.
2. ACS Nano, vol.15, no. 11, 18312–18326, 2021.





Example 3: Smart mat for smart home applications

Task 1&2A: Digital twin based position detection



Demonstration of diagonal walking and multi-user walking detection