Idea research & design

Idea 1:

Farm game

Example games: *Stardew Valley*, *Frostpunk*, *Little Witch in the Woods*

Special points: GET AS MUCH MONEY AS YOU CAN, fun of design & building player's own place

Advantages: easy to build the basic mechanics and player's growth system, easy to add advanced mechanics (fishing, finding treasure, etc.), art assets friendly...?

Disadvantages: needs a unified art asset (art style), too easy/boring…?

(*Startdew valley*. Yep that's my farm!)

Idea 2:

Characters collection + Strategy game

Example games: *Hearthstone*, *Clash Royale*
(Japanese anime style) *Arknights*, *Blue Archive*

Special points: Collecting, growing your characters and put them into the fights; strategy thinking & success

Advantages: Mechanic is fun enough, challenges enough. Could get a lot of experiences while building it. Might be good for future works.

Disadvantages: hard to build the basic mechanics, don't think about arts at this point but still need high quality of arts. More like demo.

(*Blue Archive*. The characters fighting and moving by themselves, but still needs player to decide & release the skills.)

(*Clash Royal.* RTS game.)

(*Arknights*. Tower defense game.)

Farm game developing steps:

- Plants growing and harvesting
- Buying, sale and gold system
- Mobile functions & Player control
- Inventory system
- Data storage
- Cooking, brewing & potioning
- Experiences system
- Farm design, building & natural views
- NPC
- Other additional mechanics (fishing, exploring, etc.)

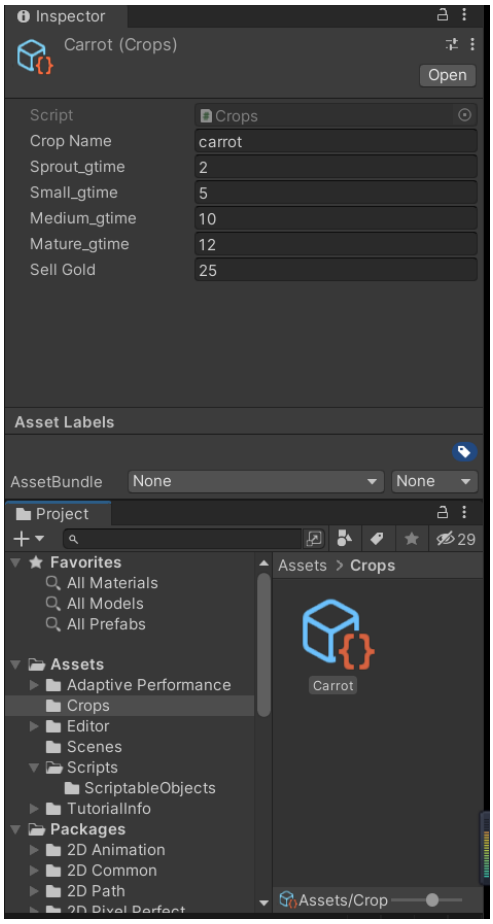
Project GitHub repo

https://github.com/PANZ5/Pan_moblie-game-dev

1. Plants growing and harvesting

Basic Plants growing tutorial:
https://www.youtube.com/watch?v=gEnogKcRf-g

Plants scriptable objects:



In Slot Manager & Crop Manager script, set up status updating & harvesting function according to the parameters of Crops ScriptableObject.

PlotsManager.cs | CropsManager.cs

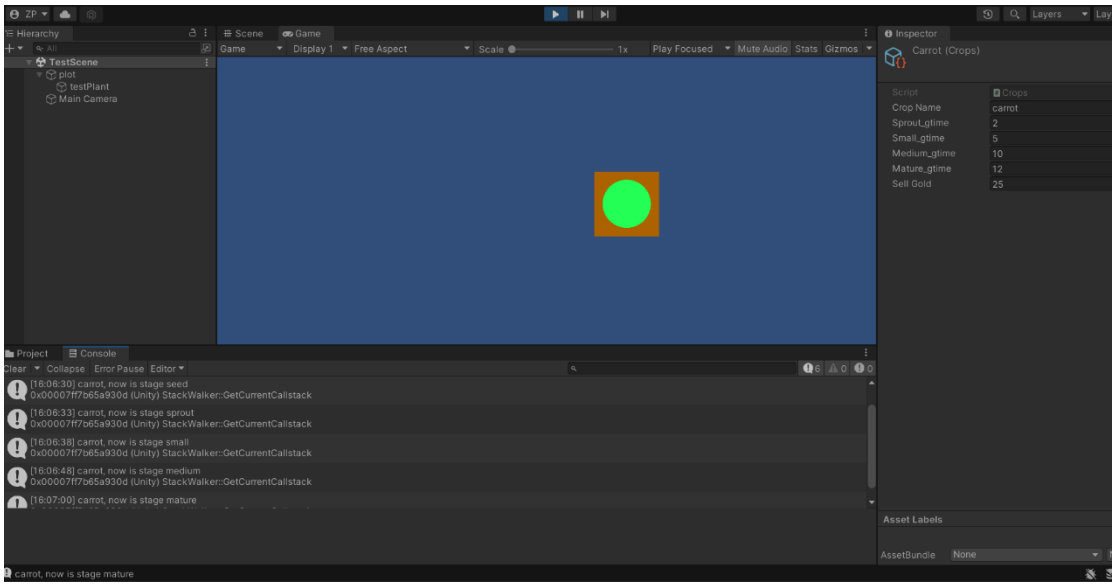
PlotsManager.cs

```
    }

    private void OnMouseDown()
    {
        if (isPlanting)
        {
            Harvesting();

            Debug.Log("Growing");
        }
        else
        {
            isPlanting = true;
            // reset plant/crop data before planted again
            plant.GetComponent<CropsManager>().Reset();
            plant.SetActive(true);

            Debug.Log("Planting");
        }
    }

    public void Harvesting()
    {
        plant.GetComponent<CropsManager>().CheckHarvest();

        // if the crop is mature, harvest it
        if (plant.GetComponent<CropsManager>().isMature)
        {
            plant.SetActive(false);
            isPlanting = false;

            // add gold after harvest
            goldSystem.Gold += plant.GetComponent<CropsManager>().harvestGold;
        }
    }

}
```

PlotsManager.cs | CropsManager.cs

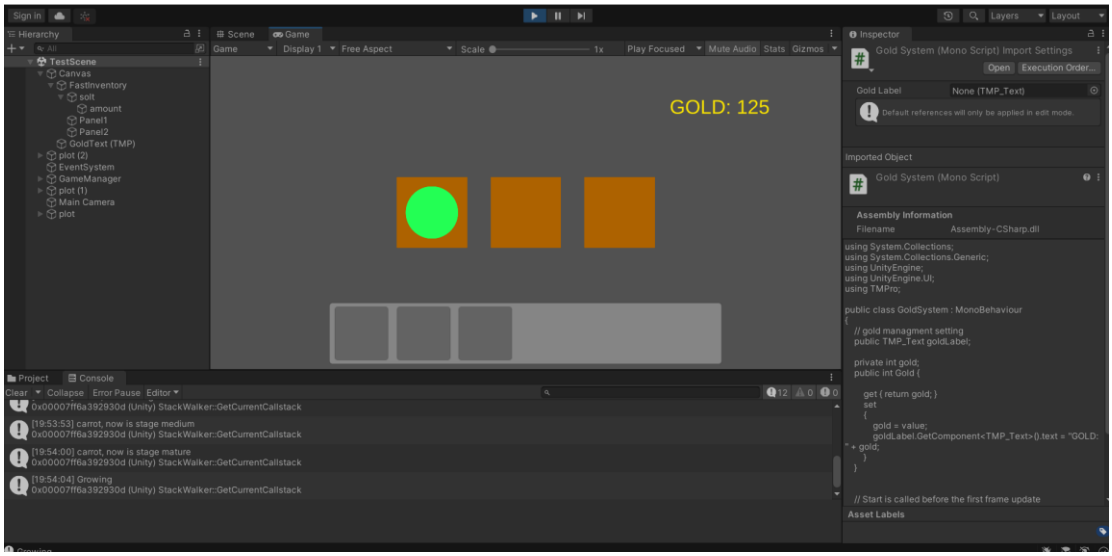
CropsManager.cs

```
    }

    public void Growing()
    {
        growStage++;
        SetUpGrowGap(growStage);
    }

    public void SetUpGrowGap(int growStage)
    {
        // convert enum to int solution reference:
        // https://forum.unity.com/threads/switching-enum-int.321451/
        switch (growStage)
        {
            case (int) Grow.seed:
                currentStage = "seed";
                growGap = crop.sprout_gtime;
                break;
            case (int) Grow.sprout:
                currentStage = "sprout";
                growGap = crop.small_gtime;
                break;
            case (int) Grow.small:
                currentStage = "small";
                growGap = crop.medium_gtime;
                break;
            case (int) Grow.medium:
                currentStage = "medium";
                growGap = crop.mature_gtime;
                break;
            case (int) Grow.mature:
                currentStage = "mature";
                break;
            default:
                break;
```



Basic gold system.

While plant harvested, adding money, update textPro.



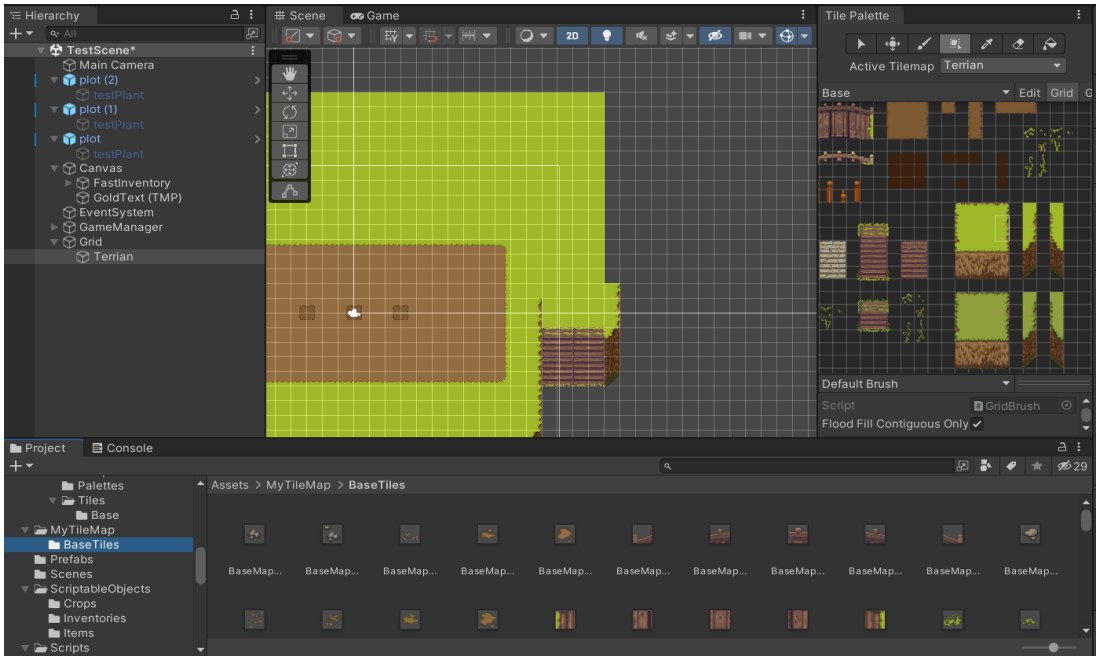
2. Environment, Map and Tilemap

Tilemap tutorial:
https://www.youtube.com/watch?v=QkbGr1rAya8&t=1470s

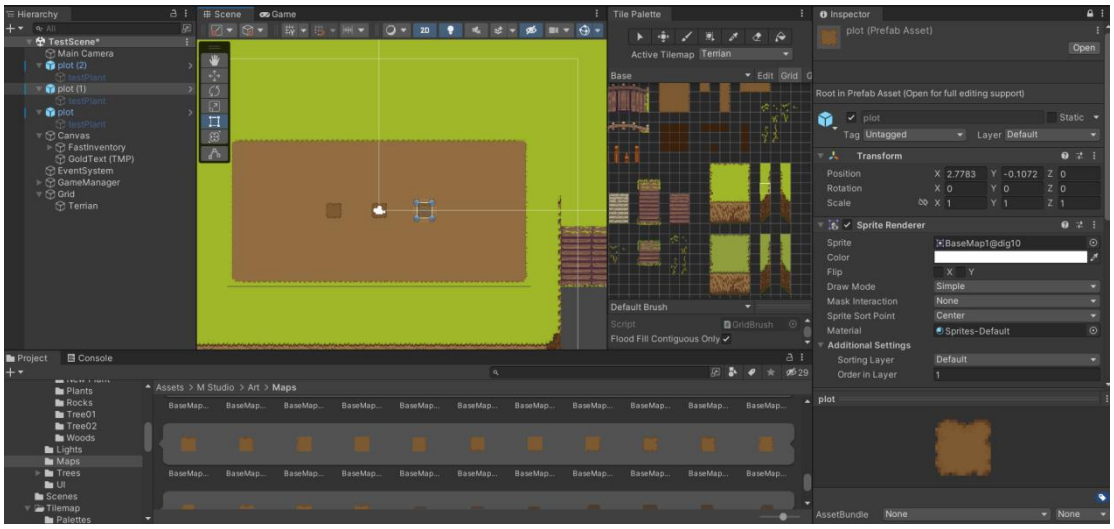
Getting new asset for farm game from lecturer!

Because the sprite image of baseMap has already been chopped pretty well, I can just drag those images into Tile Palette to generate the tilemap.

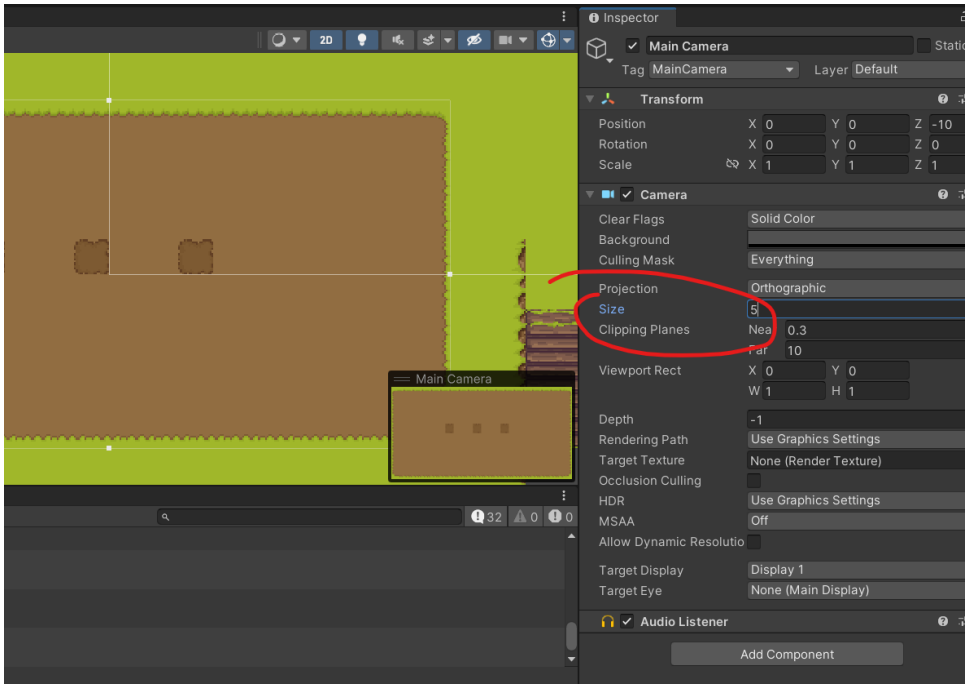
Create a grid, using Tile Palette to paint the units in grid, which makes our 1st level map!



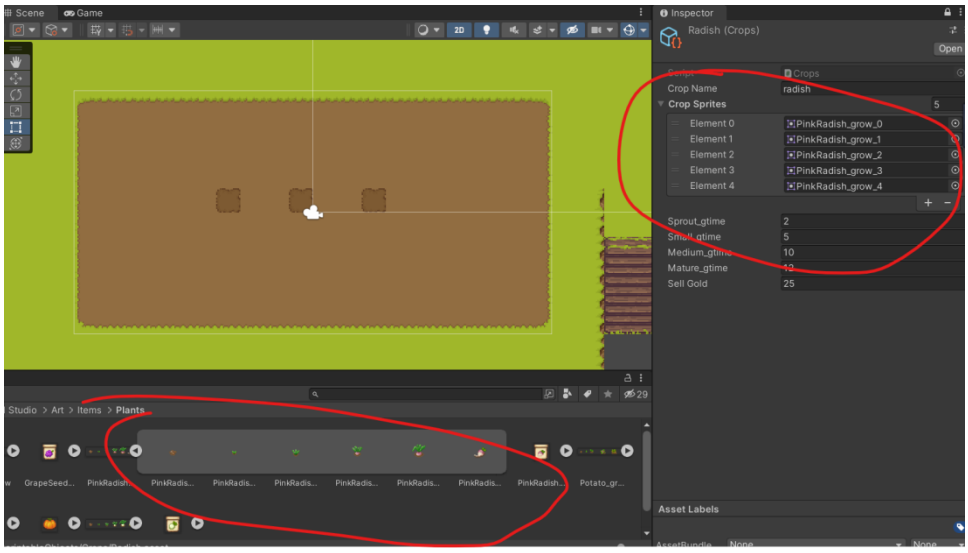
I also updated the sprite (image) of plots as well, which is no longer a simple brown square.



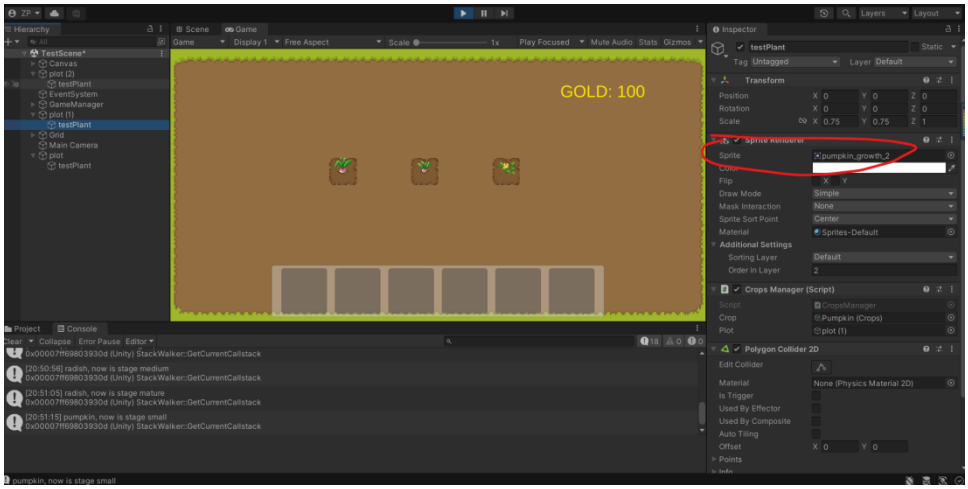
Manage the camera size, so that it fits mobile screen.



In Crops ScriptableObject, import plant images to update crop image list.



Now the plants can render images according to their growing stage:

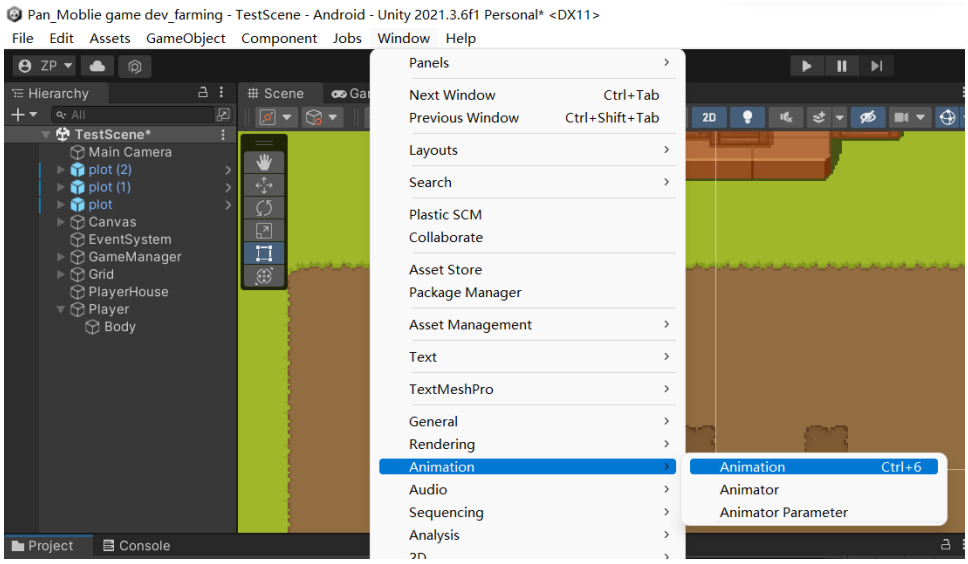


3. Player Animation

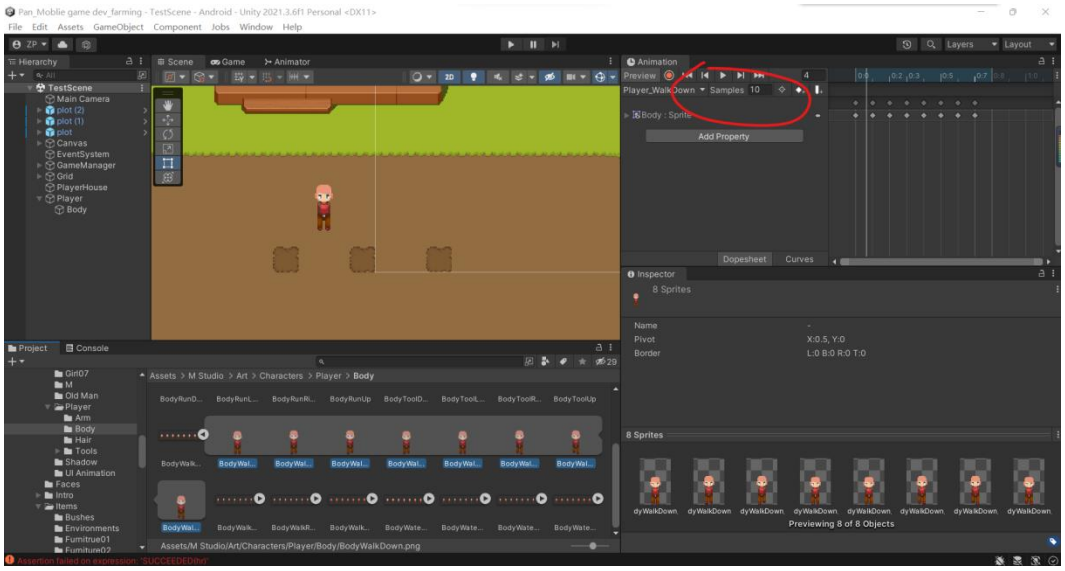
Animation tutorial:
https://www.youtube.com/watch?v=whzomFgjT50

After creating player object with 2d rigid body & simple testing control script, we can add animation for it.

Click window>animation to create animation by yourself. It will automatically generate a new animation controller for player object as well.



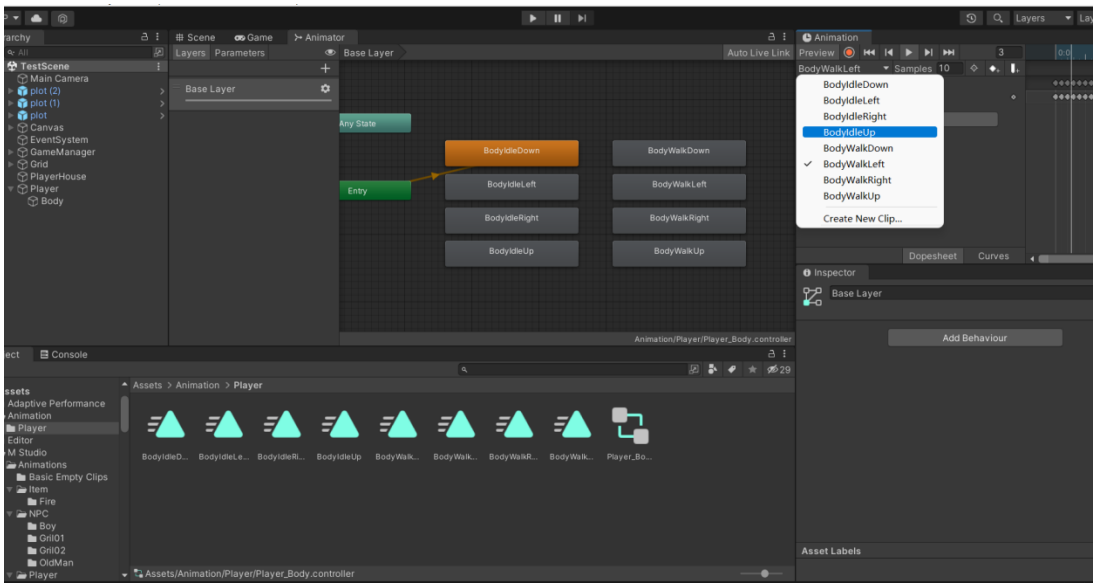
Select suitable sprites from asset. Drag them into the animation we just created. Adjust samples value to make animations run a bit slower.



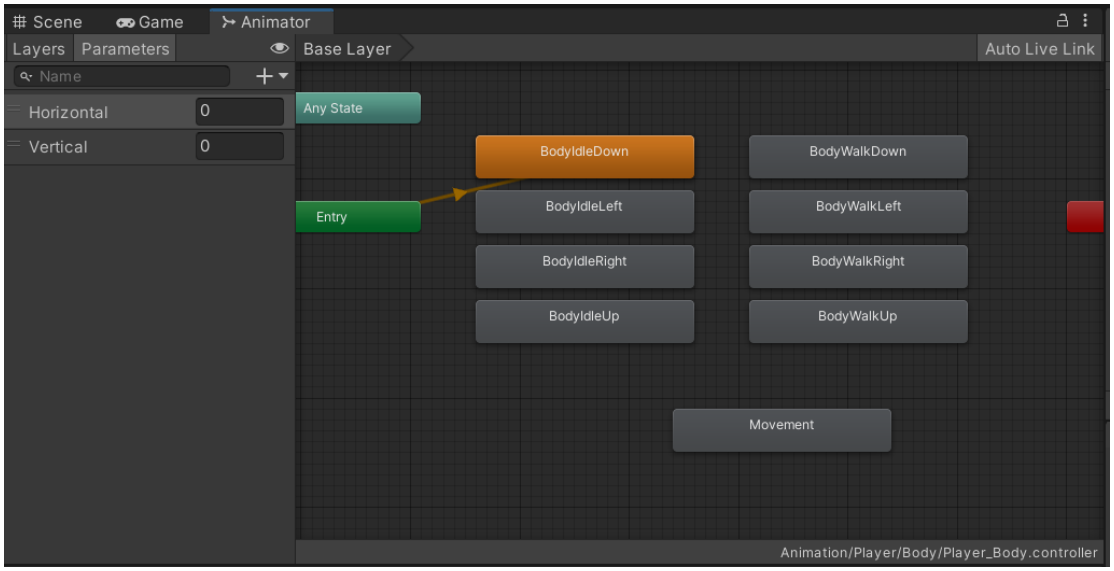
That’s the way to create animations manually.

The assets I’ve got contain the animation files which have already been set well by assets’ author. In this case we could just drag those animation files into animator controller to apply animations for player character.

Open animator panel, drag animations file into animator panel. The animation has imported & played fine.

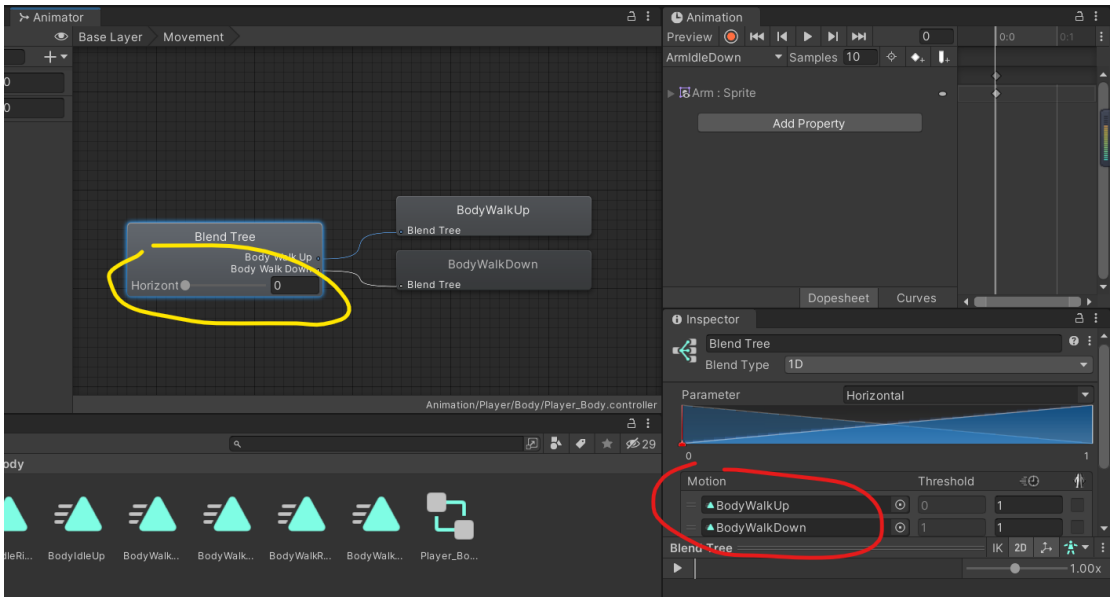


Create 2 parameters called Horizontal and Vertical, then create a new blend tree in animator controller called Movement.



Double click to enter Movement's blend tree layer.
In motion list, append 2 motions for WalkUp and WalkDown.

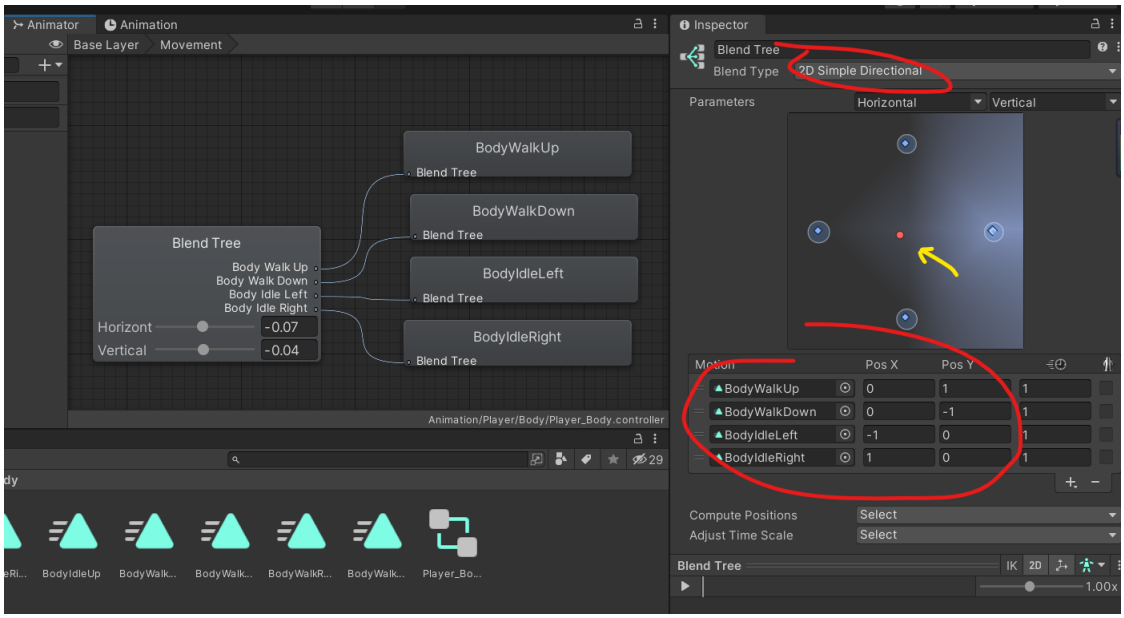
In yellow area, you can see if the Horizontal value change, the animation of motions will change as well (if h<0.5 then displaying WalkUp, otherwise displaying WalkDown).



Changing blend type from 1D into 2D simple so that we could add WalkLeft and WalkRight.

Append WalkLeft and WalkRight into motion list. Building up motion list value so that the X & Y axis value should fits the Horizontal & Vertical movement.

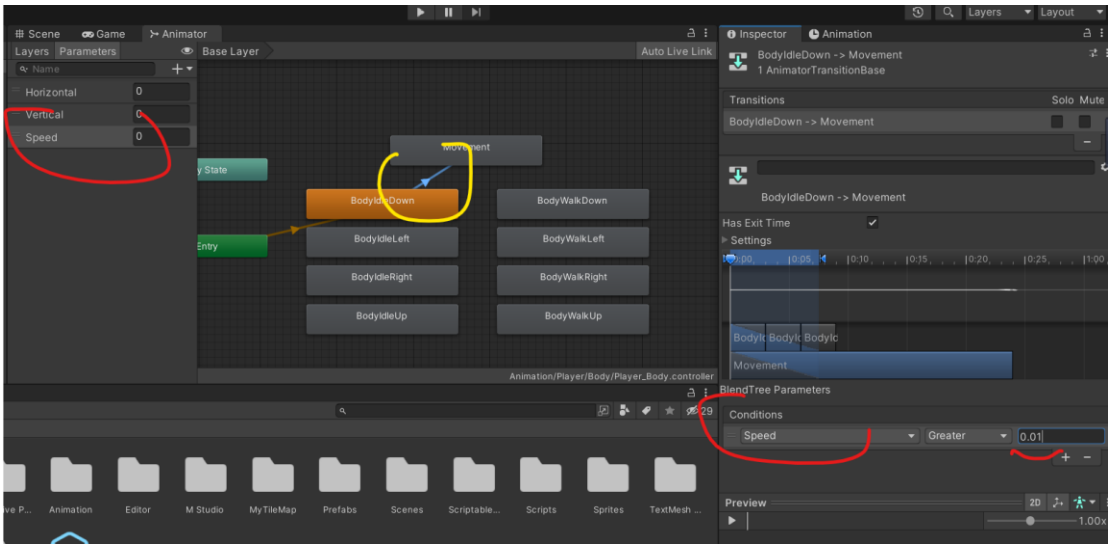
In yellow area, the red dot represents player position. Drag this red dot so we could know when should display the correct animation while player's movement tending to go into a direction.



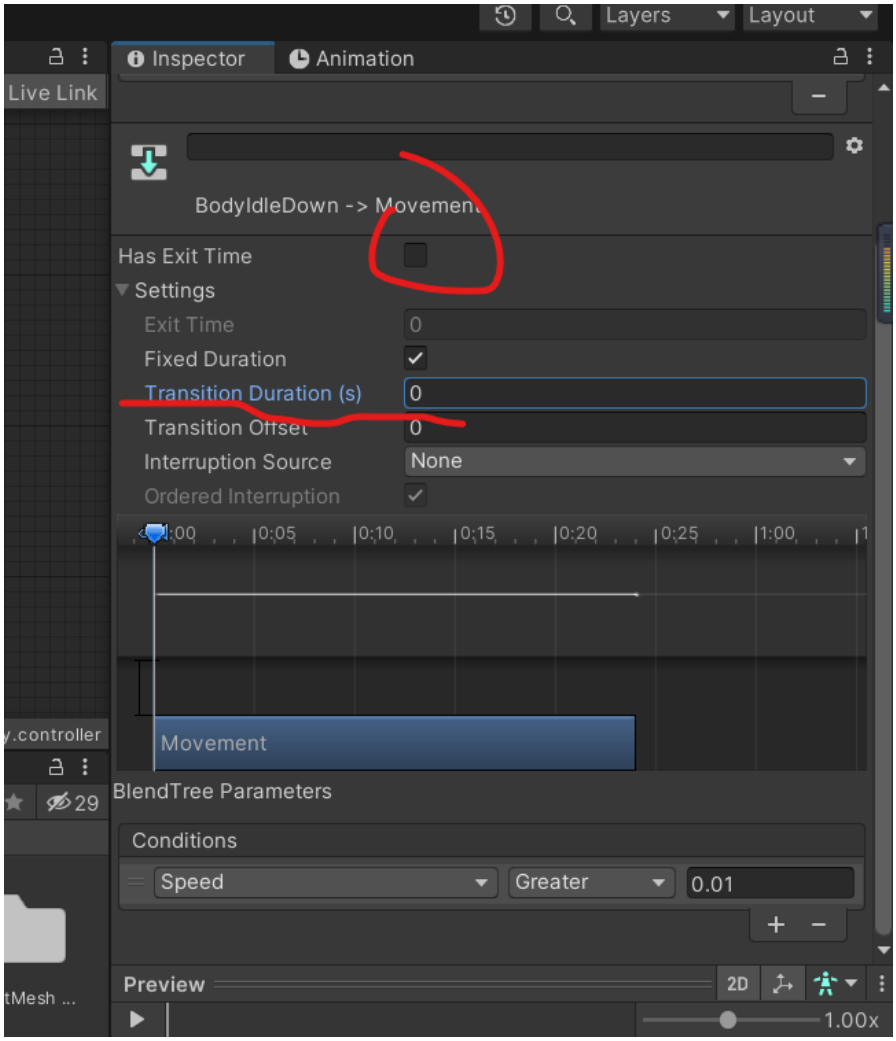
Link default animation (Idel animation) to Movement blend tree.

Create another parameter called Speed. So now, we could assign Speed into transaction base, and use the moving speed of player to judge when the Movement clips should be presented.

(For example, if player moves, the speed will definitely greater than 0.001, then, play Movement animation)

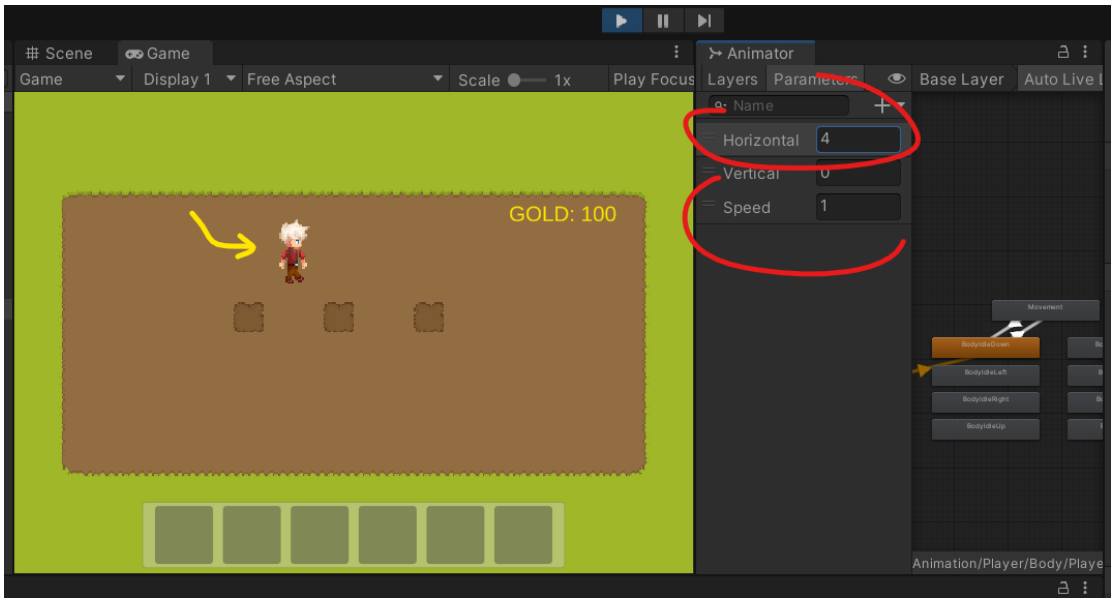


Turn off Exit time, reduce Transaction Duration to 0.



Run the game, now you can see the **player's body** does movement with animation!

If you leave the Animator window opened, you can monitor the parameters change while body moves.



Applied all the settings to not only player's body, but also player's hair & arm, so that you can get real movements.

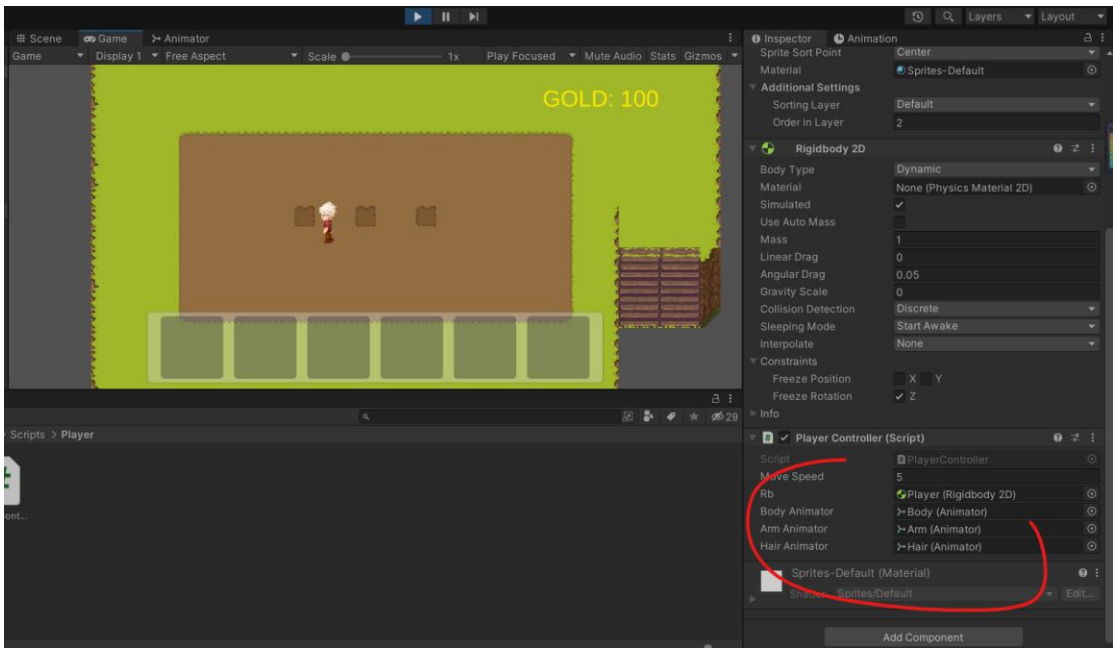
Don't forget to assign all 3 animators in PlayerController script.

PlayerController.cs M

D: > Studying > BIT Y3S1 > Pan_moblie-game-dev > Pan_Moblie game dev_farming > Assets > Scripts > Player > PlayerCo

```
5   public class PlayerController : MonoBehaviour
6   {
7       public float moveSpeed = 5.0f;
8
9       public Rigidbody2D rb;
10
11      // Animators
12      public Animator bodyAnimator;
13      public Animator armAnimator;
14      public Animator hairAnimator;
15
16      Vector2 movement;
17
18      // Update is called once per frame
19      void Update()
20      {
21          movement.x = Input.GetAxis("Horizontal");
22          movement.y = Input.GetAxis("Vertical");
23
24          // Animations
25          // Body
26          bodyAnimator.SetFloat("Horizontal", movement.x);
27          bodyAnimator.SetFloat("Vertical", movement.y);
28          bodyAnimator.SetFloat("Speed", movement.sqrMagnitude);
29          // Arm
30          armAnimator.SetFloat("Horizontal", movement.x);
31          armAnimator.SetFloat("Vertical", movement.y);
32          armAnimator.SetFloat("Speed", movement.sqrMagnitude);
33          // Hair
34          hairAnimator.SetFloat("Horizontal", movement.x);
35          hairAnimator.SetFloat("Vertical", movement.y);
36          hairAnimator.SetFloat("Speed", movement.sqrMagnitude);
37      }
```

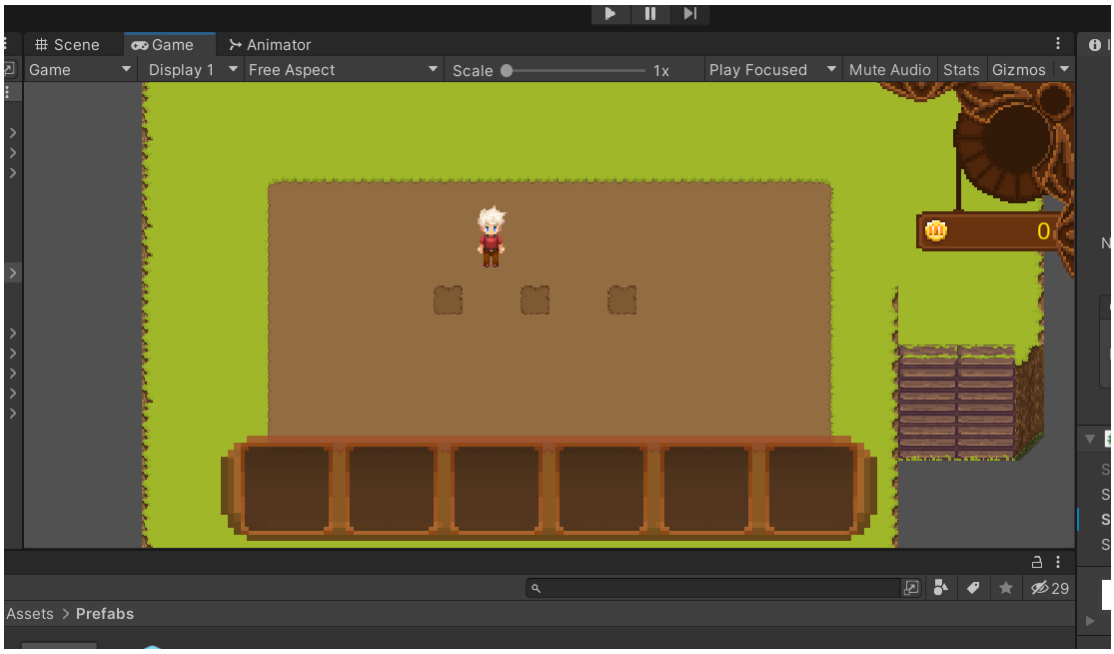
Once you set up everything you will see real movements of Player!



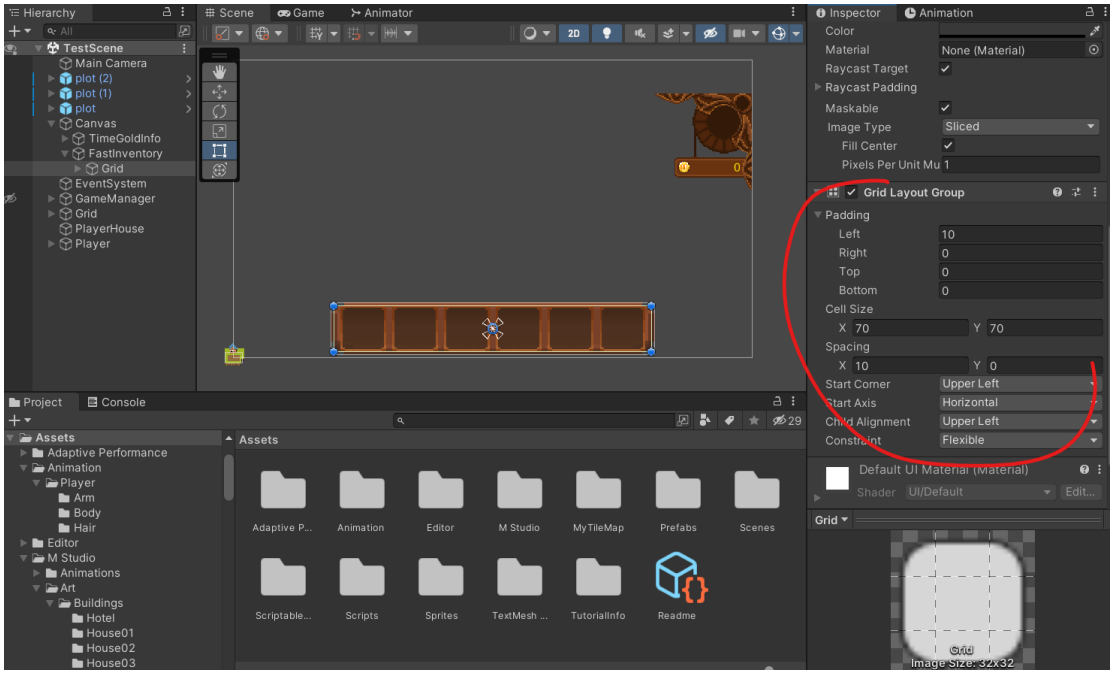
4. Inventory System

Inventory tutorial (in Mandarin ver):
https://www.bilibili.com/video/BV1cJ411X7hN/?spm_id_from=333.337.search-card.all.click&vd_source=45942140cf4b8dc6a8e96510b6fb89b3

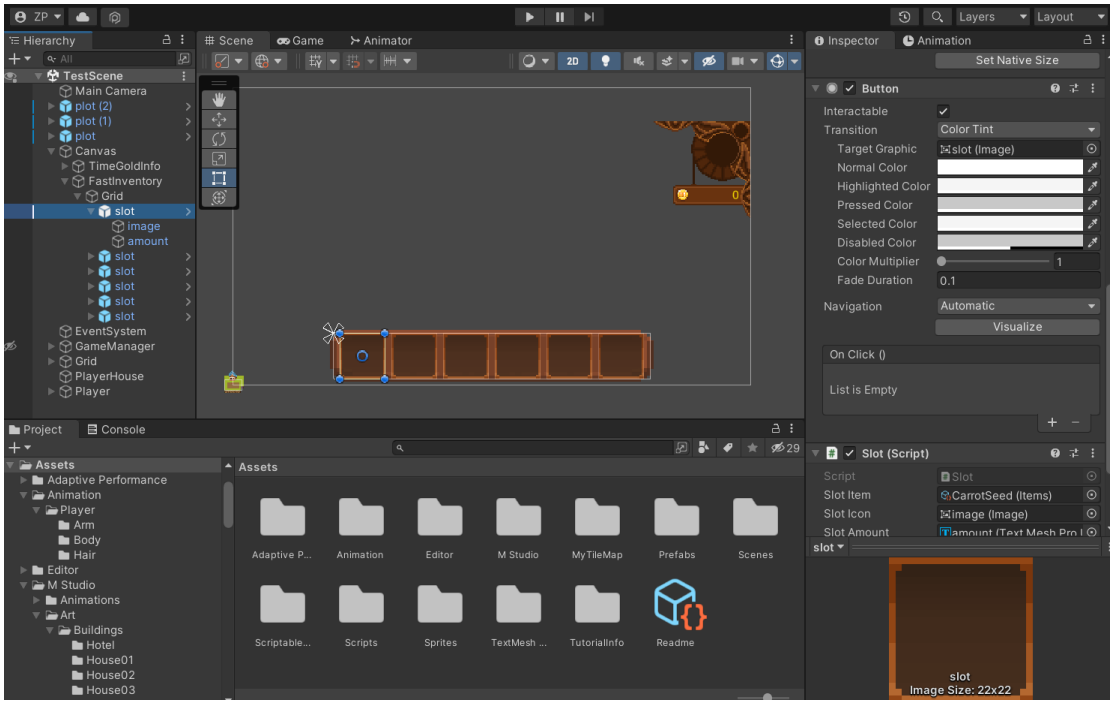
In UI canvas, import art asset, create a new game object called FastInventory.



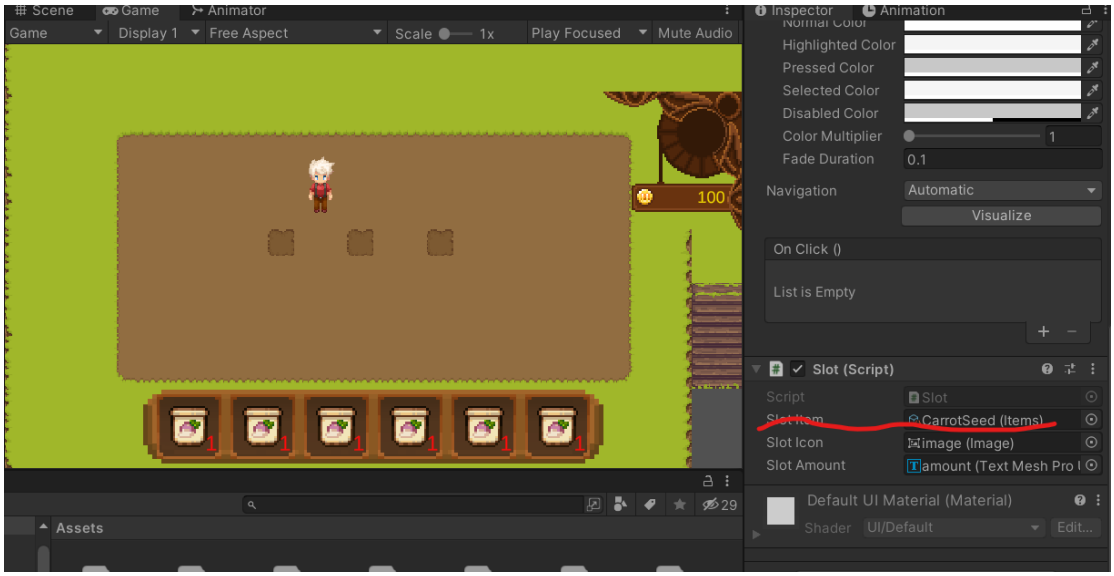
Create a new grid in FastInventory, adjust the ranges between cells.



Set all of the grid cells as Slots. Each Slot should have an image, an amount text (at this stage). Each Slot should also been set as a clickable button.



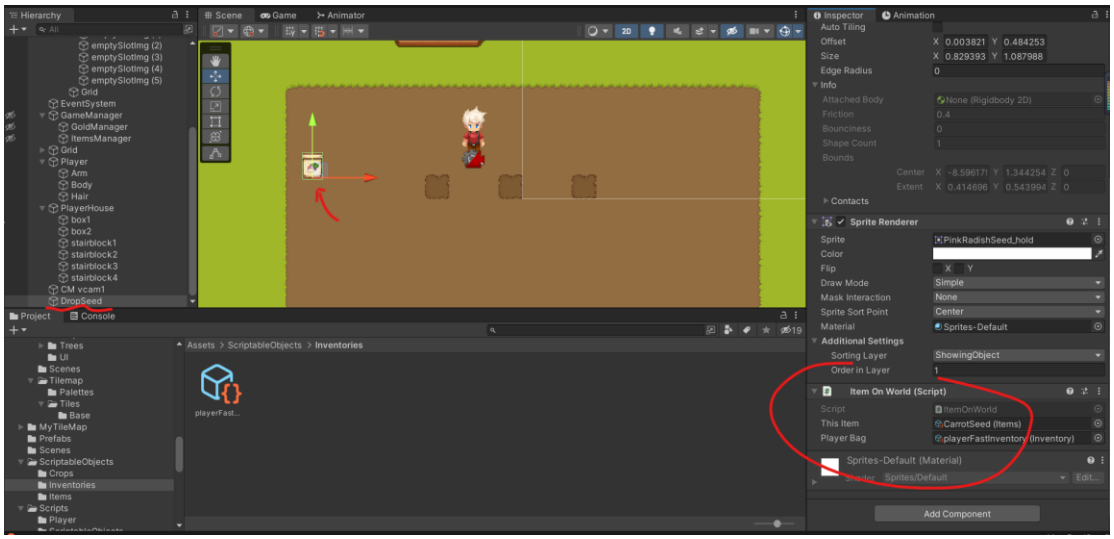
Import an Item ScriptableObject (CarrotSeed item) for testing purpose.



Empty the testing Slot item before start upgrading inventory. Now we will set some items which dropped on the ground, while player getting close items, it will be “picked up” and appended in fast inventory.

Set up new script called Item On World. It will assign an Item ScriptableObject, and called Inventory ScriptableObject (player’s bag) to store this item.

Attach this script to DropSeed game object. Add a **triggered** 2D box collider for DropSeed, and modify the references of script.

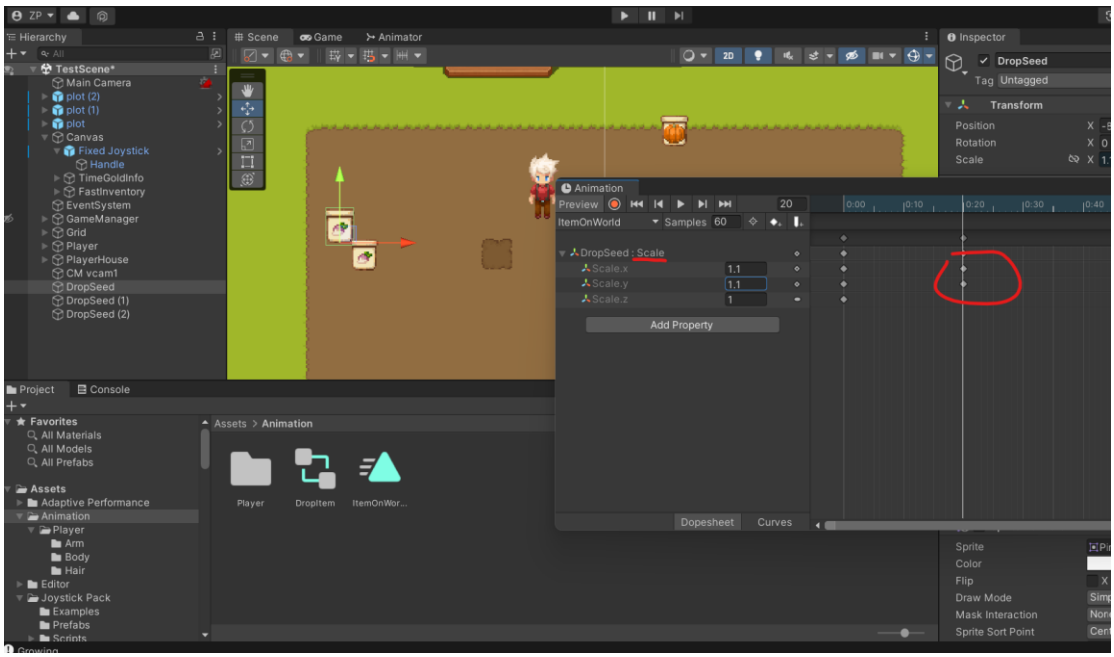


Run the game, let player walking near to the DropSeed, it should be “picking up” and

appears in fast inventory.

For directing player to pick up items on the ground, we could add some animation for it. Create a new animation.

In the new animation, add **scale** of DropSeed on timeline.

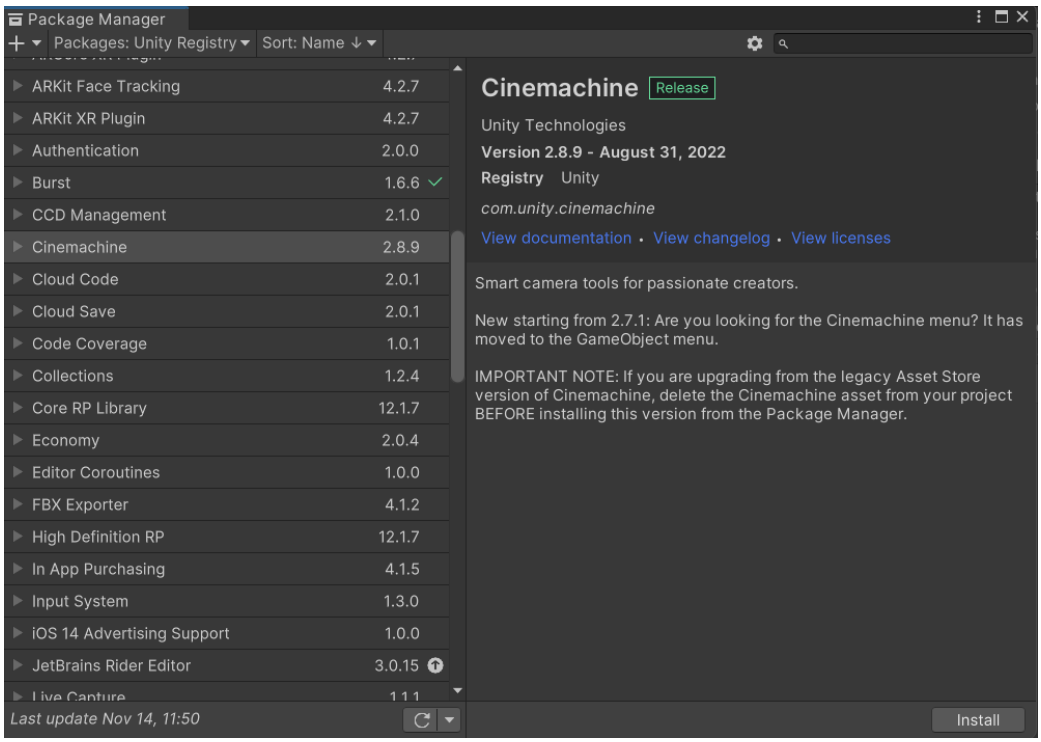


Run the game, now you should see the DropSeed is zooming itself like floating.

5. Player Control Modify

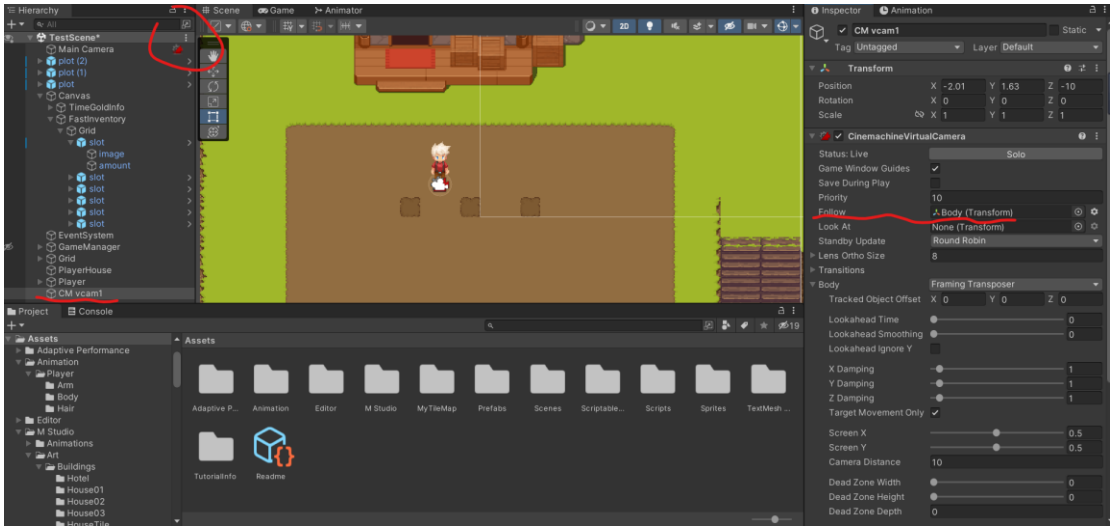
Camera Following tutorial (in Mandarin ver):
https://indienova.com/u/mcatin/blogread/27795

To let camera following the player, install Cinemachine. Import Cinemachine into project.



In hierarchy, create a new Cinemachine 2D camera. Let the camera follow Player's body by setting "Follow" parameter.

Notice that there's a special icon near the Main Camera. That mean's Cinemachine's camera has automatically synchronized the settings with Main Camera.



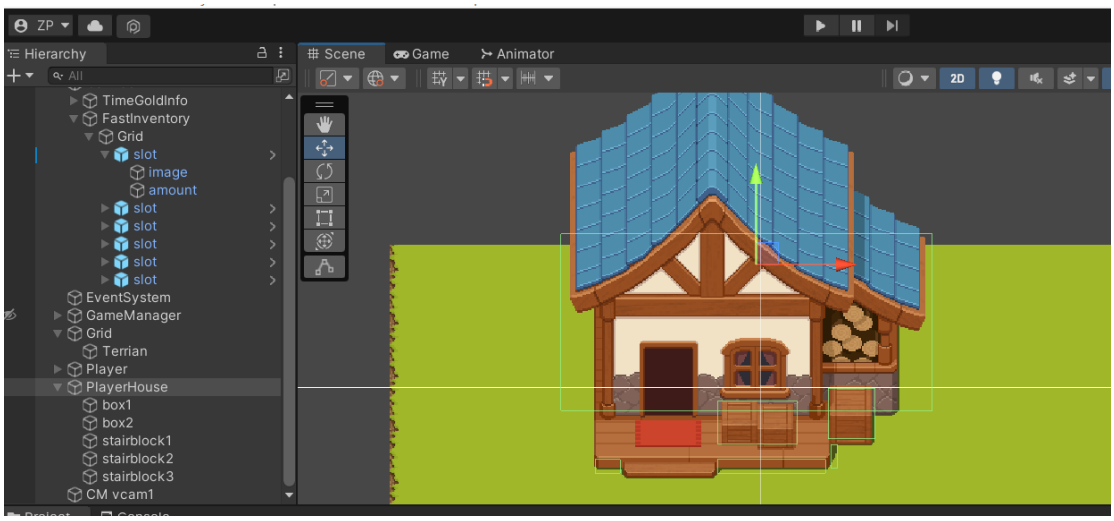
Next step is going to modify the Collider Blocks, which will block player to walking over the scenery objects/scene.

Block function tutorial:
https://www.youtube.com/watch?v=05eWA0TP3AA

Upgrade Player Control script, add a new ContactFilter 2D called movement filter. In layer mask set it as a new layer tag "Block".

By those setting now, all the 2D Colliders which are in the layer "Block" (like PlayerHouse) will block player's movement.

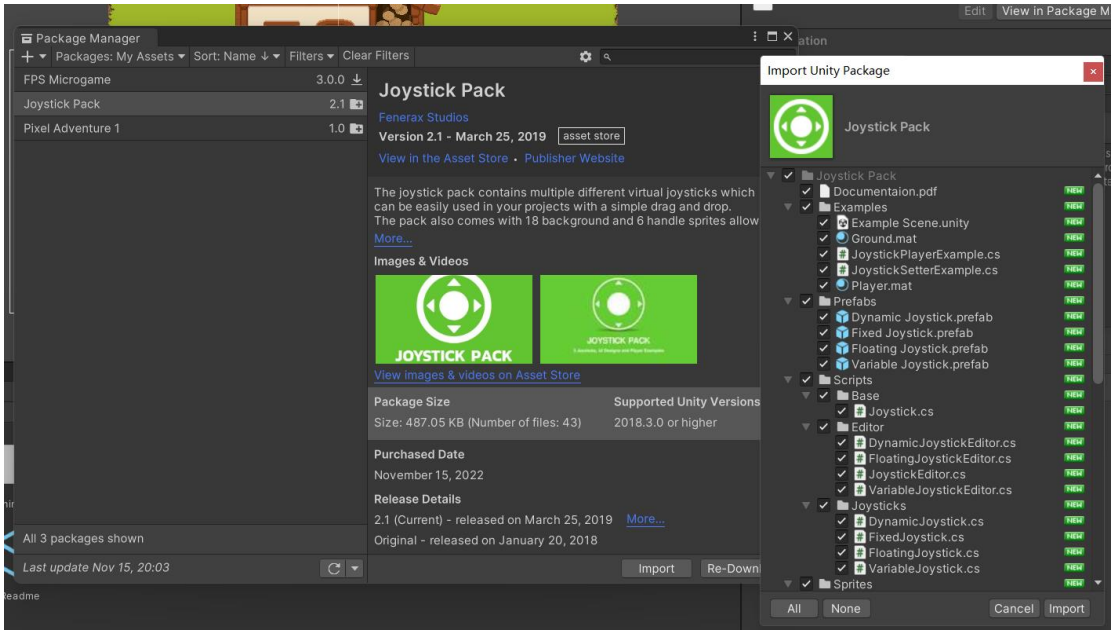
Add more 2D Box Collider for player’s house.



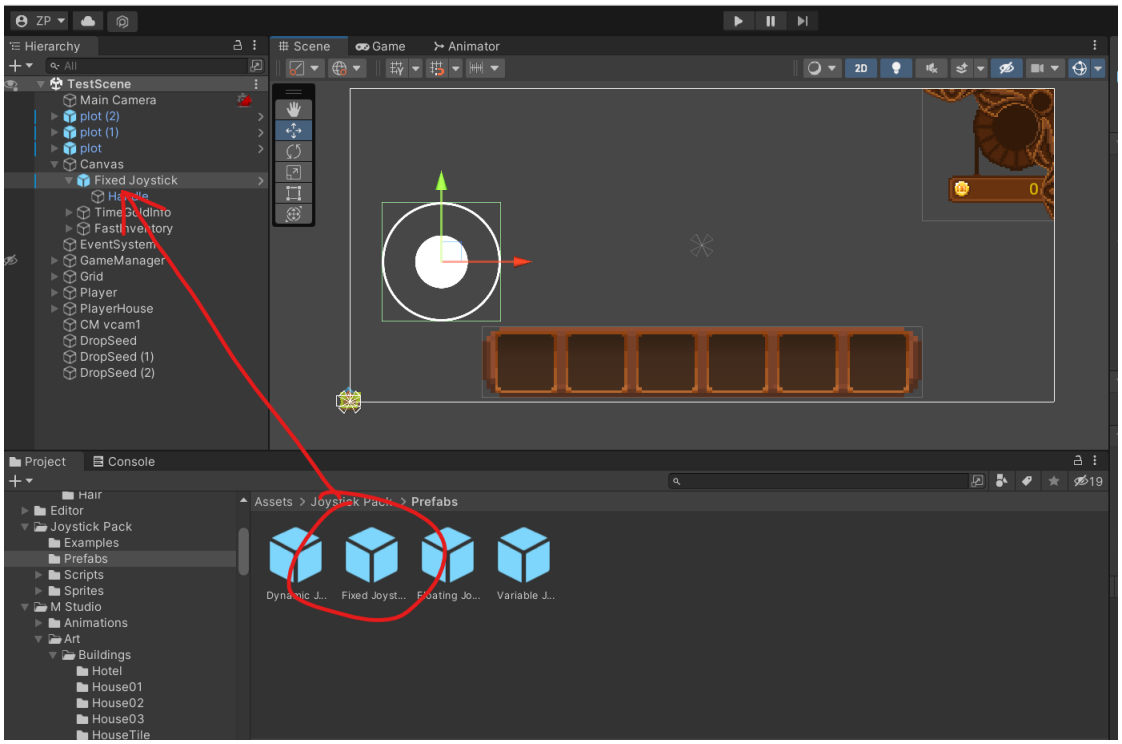
Next step is convert player’s control from PC developing env into mobile env.

Mobile control & Joystick tutorial:
https://www.youtube.com/watch?v=bp2PiFC9sSs&t=697s

Download & import Joystick package in package manager.



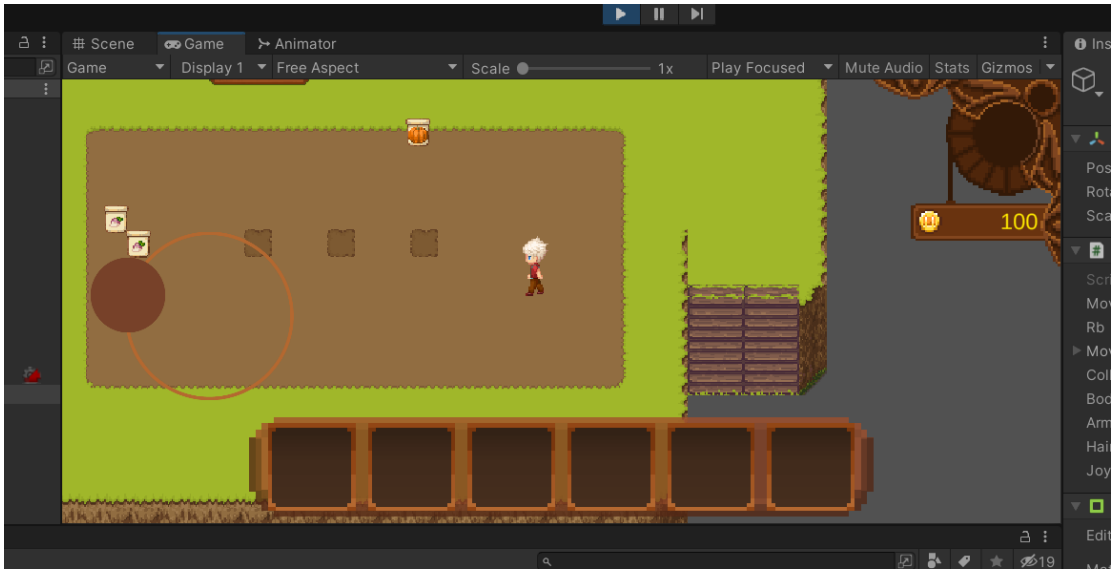
Drag fixed joystick into project hierarchy and modify its color & scale. (Fixed joystick means: no matter how player drag this joystick during control characters, the joystick will still stay in a fixed position.)



In script PlayerController, change movement.x and movement.y parameters into joystick parameters.

```
17
18      // Animators
19      public Animator bodyAnimator;
20      public Animator armAnimator;
21      public Animator hairAnimator;
22
23      // Joystick
24      public Joystick joystick;
25
26      Vector2 movement;
27
28      // Update is called once per frame
29      void Update()
30      {
31          movement.x = joystick.Horizontal;
32          movement.y = joystick.Vertical;
33
34          // Animations
35          // Body
```

Open Unity Remote on your phone, connect your phone to computer, you can now control player character with joystick on phone screen, rather than computer keyboard.



Project review

- What is your mobile game application?

A Unity game project, which is related to farming simulation.

- What considerations did you make when planning your mobile game application?

  For theme of the mobile game, I compared the advantages and disadvantages of each idea, from developing difficulty to player's user experience.

  For development plan, I started to develop core function firstly, then building optional expandable function later, so the entire dev-plan could be time flexible but still keeps functional.

- How did you effectively use the game engine?

  I was using the package manager of Unity to import assets into my project, and using it to modify phone related function, such as mobile phone input event (Touch, Joystick) and mobile testing env (Unity Remote).

  Also, during development process, I found that Scriptable Object is a powerful datatype which could store similar dataset, saving RAM calculation (by get rid of calling just one Prefab for hundreds of times) and referencing/transfer friendly. For this reason, I always store my gaming data as Scriptable Object, like items, inventory and crops.

- What are some areas for improvement?

  Data-sharing of Scriptable Object.
  Sometimes 2 independent Crops/Slots will share the data since they're both referencing data from same Scriptable Object.

  Game screen will become vague/unclear when I testing & run the game demo on my phone by using Unity Remote. Can't figure out what reason cause it happened.

- If you were to continue with the mobile game application, what are the next steps?

  Linking crops system and inventory system together.
  Painting more Tilemap.
  Adding store to let player buying seeds or selling crops.