BUG CONTROLLER DOCUMENTATION

CONTENTS

Bug Controller Editor Values **1**

Leg settings **3**

Leg event 4

How To Use In Code **4**

Bug Controller 4

Look Target 4

Look Position 5

Legs Vertical And Horizontal Positions 5

Set ground alignment Boolean 5

Set body height 6

Nav Mesh Controller 6

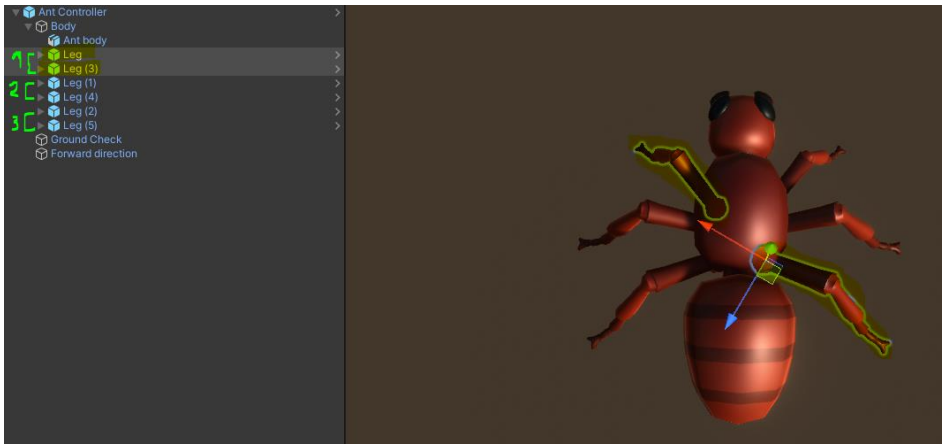
Usage Example: Third Person Controller Template **7**

Video Tutorials **11**

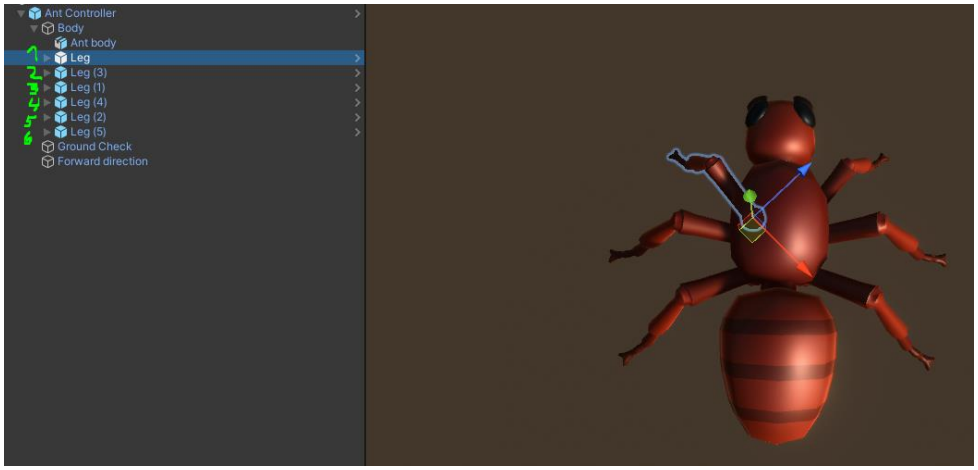
- Overview YouTube link YouTube Link 11

Bug Controller Editor Values

This determines whether the bug's body conforms to the terrain.



If this is true, then every time the bug takes a step, two legs are moved at the same time.



If this is false, every time the bug takes a step, one leg is moved per step. The legs take the steps in the order they are placed in the editor. **Bug must always have an even number of legs**.

Feet only hit objects of this physics level.

This defines how strongly the bug's body adapts to the terrain.

This determines how strongly the legs anticipate which direction the body is going.

This adjustment determines how high the body is off the ground.

This value defines how strongly the height of the body height is fine-tuned when parts of the legs are higher than the body.

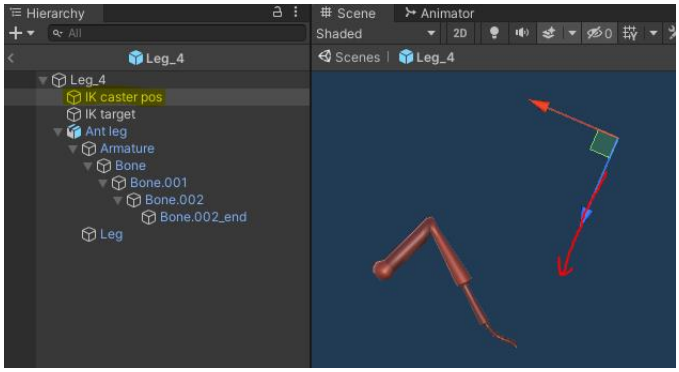
This determines how high the foot rises when taking a step.

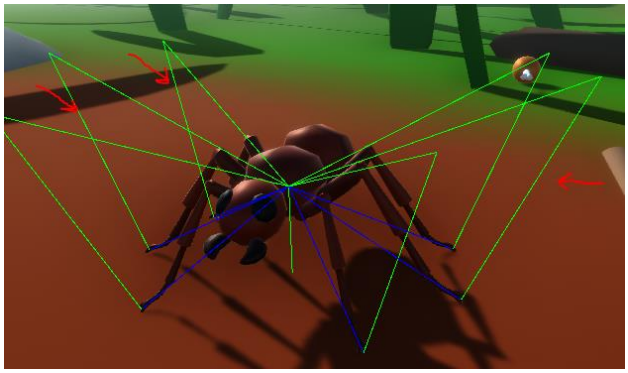
How far the body must move before taking the next step.

How many degrees does the body have to rotate before the leg correction is done.

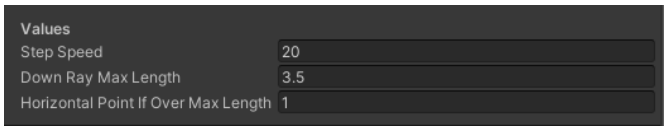
How long does the bug have to be in place before the legs are placed again.

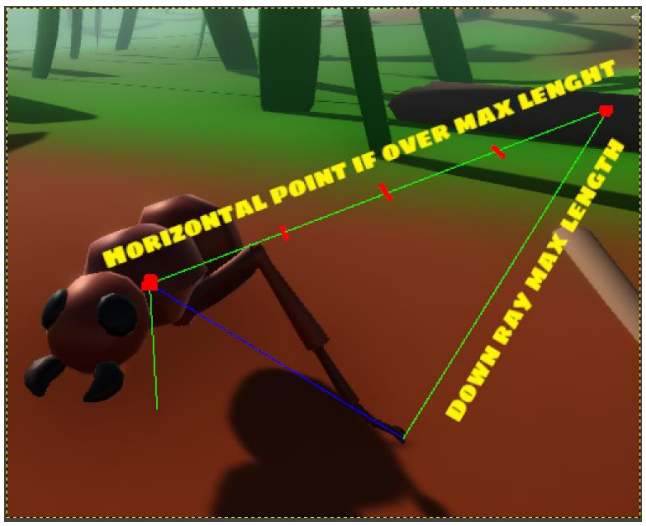
LEG SETTINGS





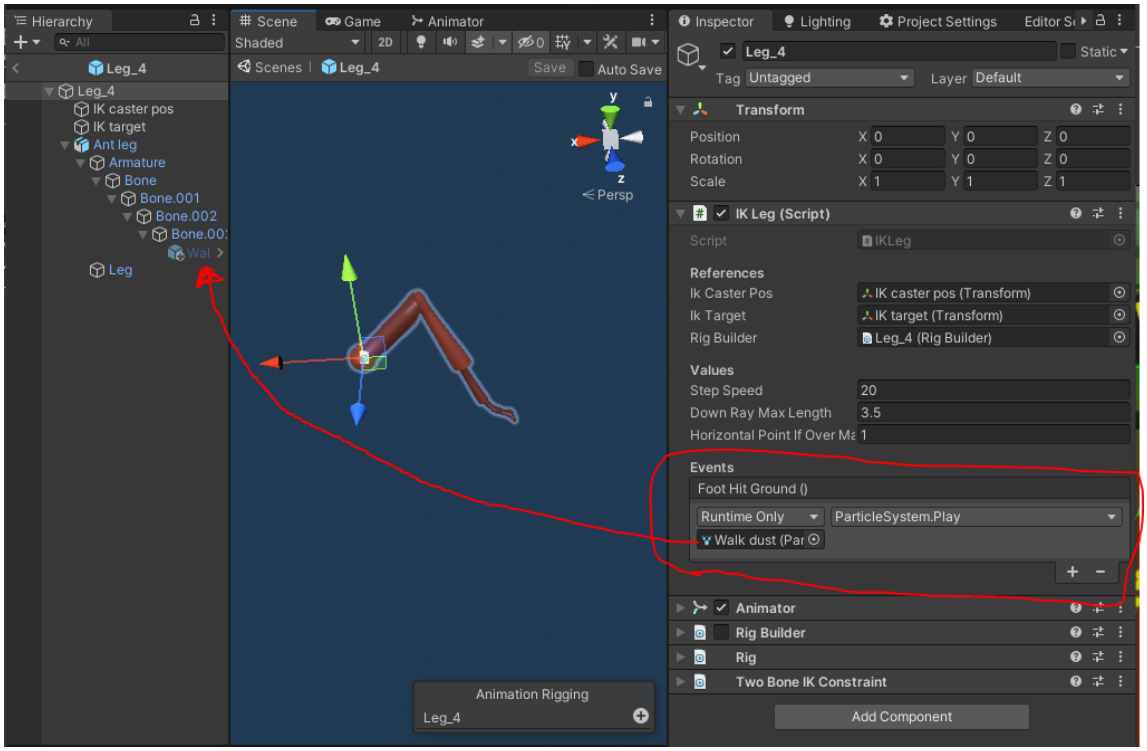
Inside every leg prefab is IK caster pos object, this object defines where the **Leg IK Caster** object is created. The **Leg IK Caster's** job is to shoot a ray down and tell the leg where to place its foot.





If the **Leg IK Caster** ray length is greater than the **Down Ray Max Length** value, **Leg IK Caster** object goes to the point specified in the **Horizontal Point If Over Max Length** value.

LEG EVENT



The legs also have a Unity event that you can use according to your needs. You can find an example in the **Example scenes** folder.

HOW TO USE IN CODE

Remember to use namespace **HeneGames.BugController**

BUG CONTROLLER

LOOK TARGET

```
Unity Script | 0 references
public class BugAimController : MonoBehaviour
{
    [SerializeField] private BugController bugController;
    [SerializeField] private Transform forwardLookPos;
    [SerializeField] private Transform aimLookPos;
    [SerializeField] private float aimSpeed = 10f;

    Unity Message | 0 references
    private void Update()
    {
        if(Input.GetMouseButton(1))
        {
            bugController.LookTarget(aimLookPos, aimSpeed);
        }
        else
        {
            bugController.LookTarget(forwardLookPos, aimSpeed);
        }
    }
}
```

Between the parentheses you can put Transform component, which the bug must look at, and next is the float value that defines the turning speed.

LOOK POSITION

```
Unity Script | 0 references
public class BugAimController : MonoBehaviour
{
    [SerializeField] private BugController bugController;
    [SerializeField] private Vector3 lookPos;
    [SerializeField] private float aimSpeed = 10f;

    Unity Message | 0 references
    private void Update()
    {
        if(Input.GetMouseButton(1))
        {
            bugController.LookPos(lookPos, aimSpeed);
        }
    }
}
```

Almost like the look target function, but instead of the Transform component, it accepts **Vector3** value. With these functions, you can make the bug look where you want (See **Look target example** Scene).

LEGS VERTICAL AND HORIZONTAL POSITIONS

```
Unity Script | 0 references
public class LegsControllerExample : MonoBehaviour
{
    [Header("Bug Controller Reference")]
    [SerializeField] private BugController bugController;

    [Header("Horizontal values")]
    [SerializeField] private bool horizontalOverride;
    [SerializeField] private float horizontalPosition = 1f;

    [Header("Vertical values")]
    [SerializeField] private bool verticalOverride;
    [SerializeField] private float verticalPosition = 1f;

    Unity Message | 0 references
    private void Update()
    {
        bugController.SetLegsHorizontalPosition(horizontalOverride, horizontalPosition);

        bugController.SetLegsVerticalPostition(verticalOverride, verticalPosition);
    }
}
```

These functions override the horizontal and vertical position of all legs. This is convenient to use, for example, when making a jump effect (See **Legs control example** scene).

SET GROUND ALIGNMENT BOOLEAN

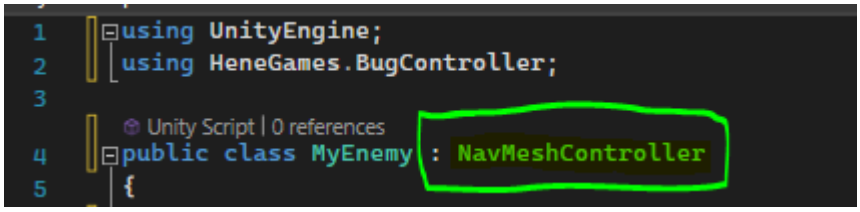
```
[SerializeField] private BugController bugController;

Unity Message | 0 references
private void Update()
{
    bugController.SetGroundAlignment(false);
}
```

SET BODY HEIGHT

```
[SerializeField] private BugController bugController;

Unity Message | 0 references
private void Update()
{
    bugController.SetBodyHeight(myValue);
}
```

NAV MESH CONTROLLER

I made this script to make nav mesh easier to use.



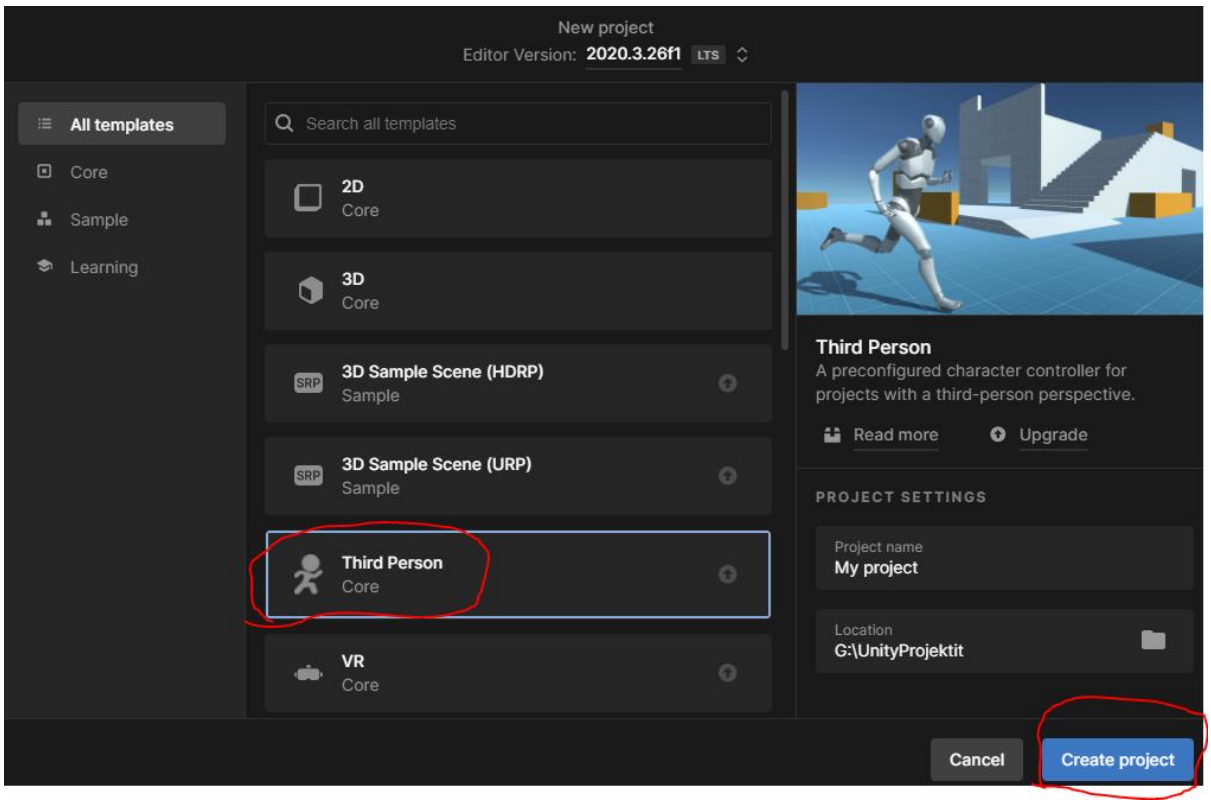
This script is intended to be inherited by one of your own classes. Replace the text **MonoBehaviour** with the text **NavMeshController** to get access to convenient functions.

```
Unity Script | 0 references
public class MyEnemy : NavMeshController
{
    [SerializeField] private Transform runAwayTransform;

    Unity Message | 0 references
    private void Update()
    {
        //Set nav mesh component speed
        NavControllerSetSpeed(1f);

        //If nav mesh component is currently calculating path or moving to destination this return true
        NavControllerCurrentlyNavigate();

        //Returns nav mesh current destination in Vector3
        NavControllerCurrentWalkDestination();

        //Gives an order to the nav mesh component to move to the given point
        //If the offset area value is above zero, the nav mesh component sets a circle size of the offset area at the given point,and searches within it for a point to navigate to
        //If force move boolean is false, the nav mesh component moves to the given point only when it has reached the previous point
        //If force move boolean is true, the nav mesh component moves immediately to the given point
        NavControllerMoveToNextPosition(transform.position, 20f, false);

        //The nav mesh component calculates the route away from the given point
        NavControllerRunAwayFromPos(runAwayTransform.position);

        //This can be used to stop or release the nav mehs component completely
        NavControllerStopNow(true);
    }
}
```

Here are all the functions to use. Look at the **BugAI** script and you will see how this can be used.

USAGE EXAMPLE: THIRD PERSON CONTROLLER TEMPLATE

The core of this asset is the Bug Controller script and ready-made prefabs that can be modified to your liking.

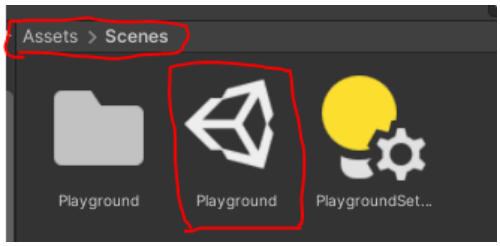


First, create a new project using the Third Person template project and import Bug Controller asset to it.

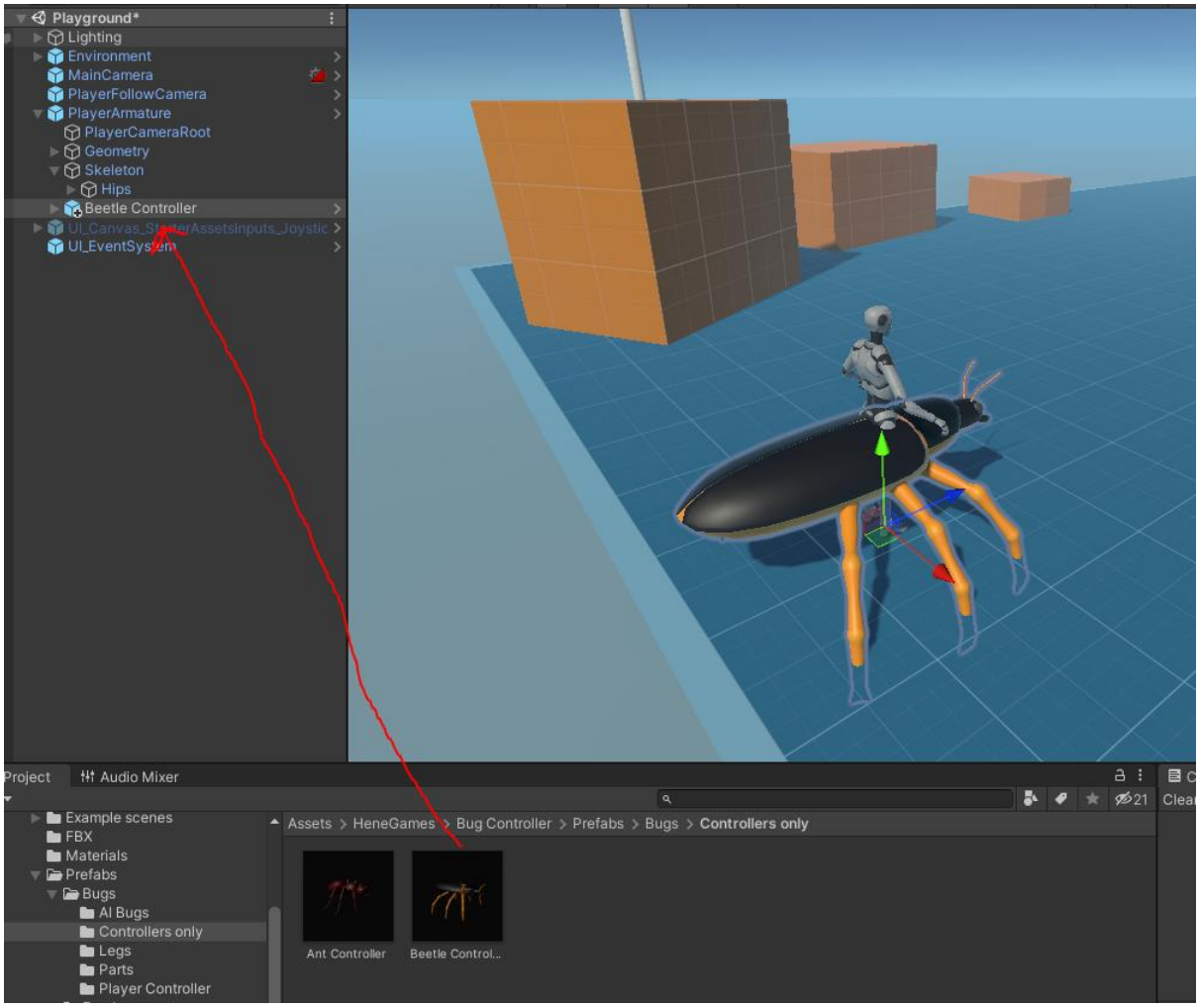
When the project is open, in top left corner click:

Edit > Render Pipeline > Universal Render Pipeline > Upgrade Project Materials to UniversalRP Materials

This action removes all pink materials (If you are using Universal Render Pipeline).

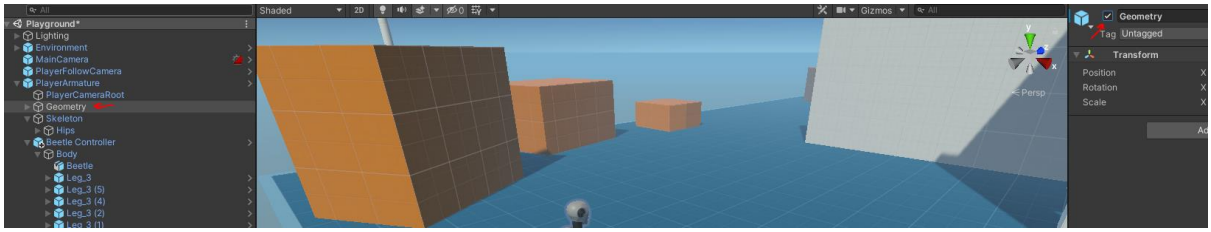


Next, open this scene from the Third Person template.

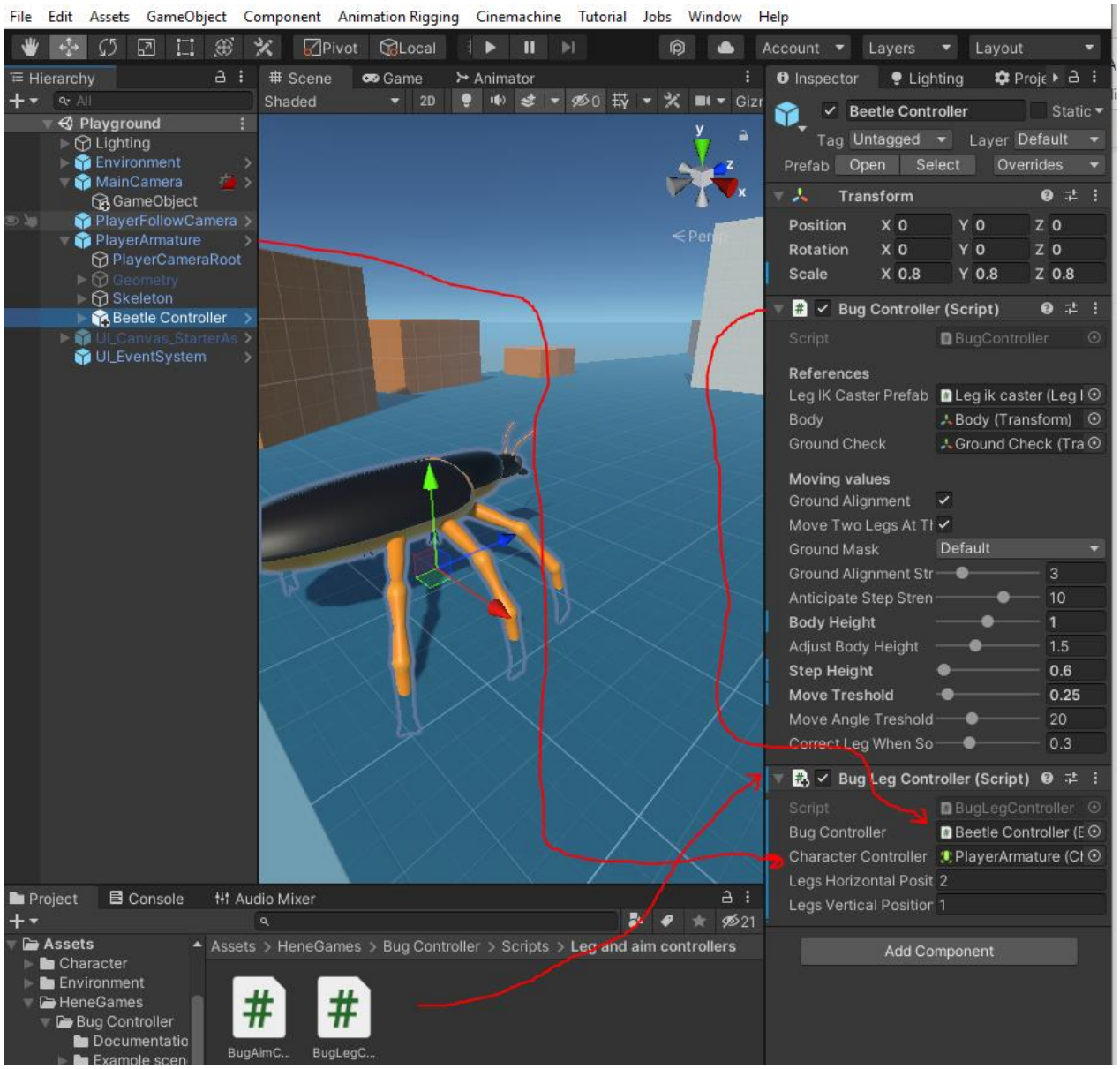


Drag the Beetle Controller prefab inside to Player Armature prefab.

Adjust the values of the bug in the scene according to the image.

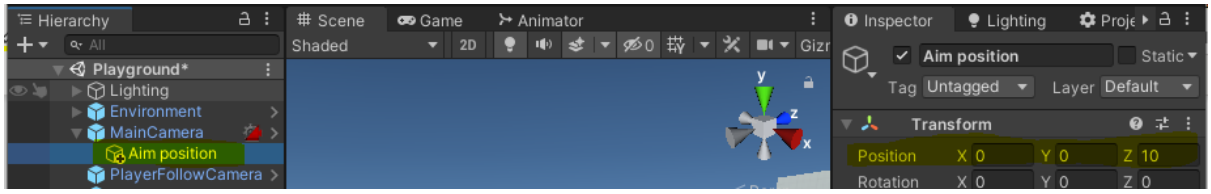


Next, you just hide the character model, and you can move around. If you jump with the character, the bug's legs look silly, so let's fix this problem by adding a **BugLegController** script.

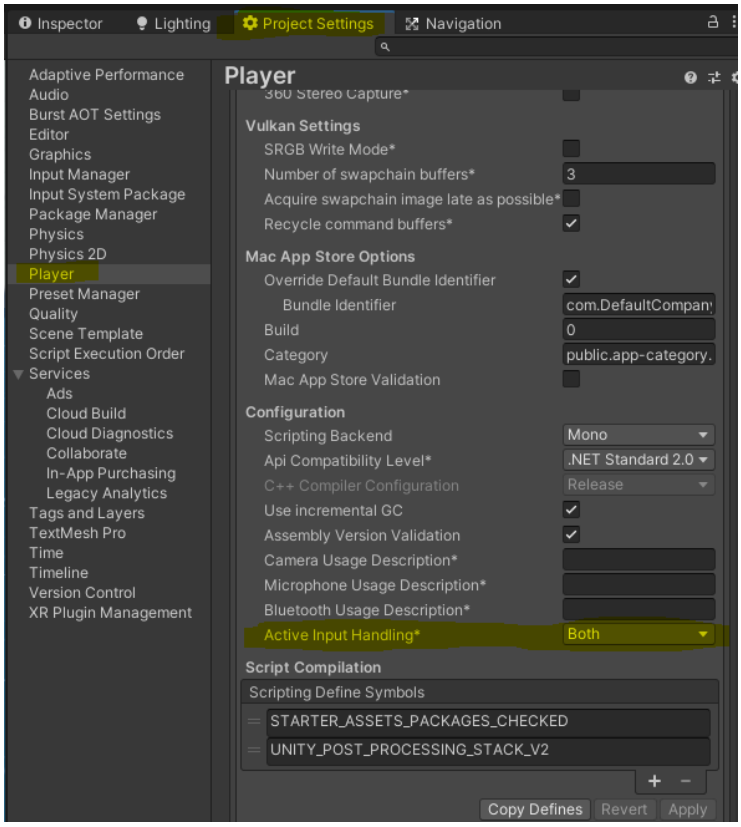


Drag **BugLegController** script to the Beetle Controller prefab and set references. Now the legs don't look stupid when the bug jumps.

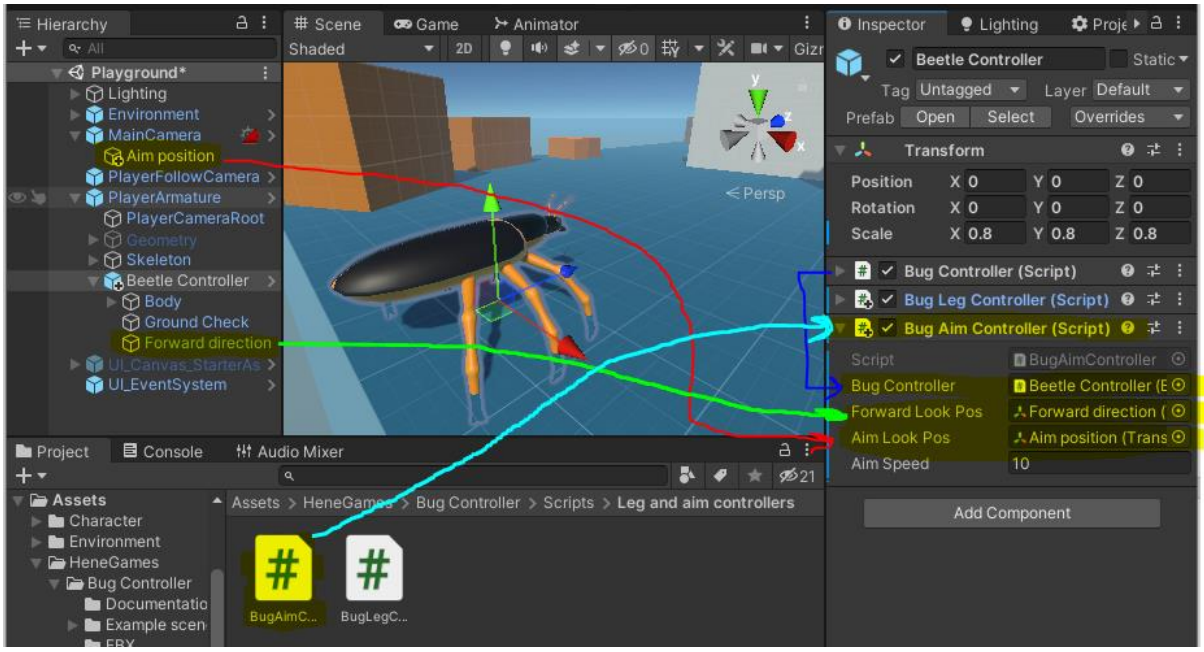
If you want the bug to look in the direction indicated by the camera when you press the right mouse button, do the following.



Set an empty game object as a child of the Main Camera and name it whatever you want. Set the local Z position of this empty game object to 10



Make sure that the Active Input Handling setting in the project settings is set to both.



Drag the **BugAimController** onto the Beetle Controller and set references. Now if you press the right mouse button while playing the game, the bug will look in the direction of the camera.

VIDEO TUTORIALS

- OVERVIEW YOUTUBE LINK YOUTUBE LINK