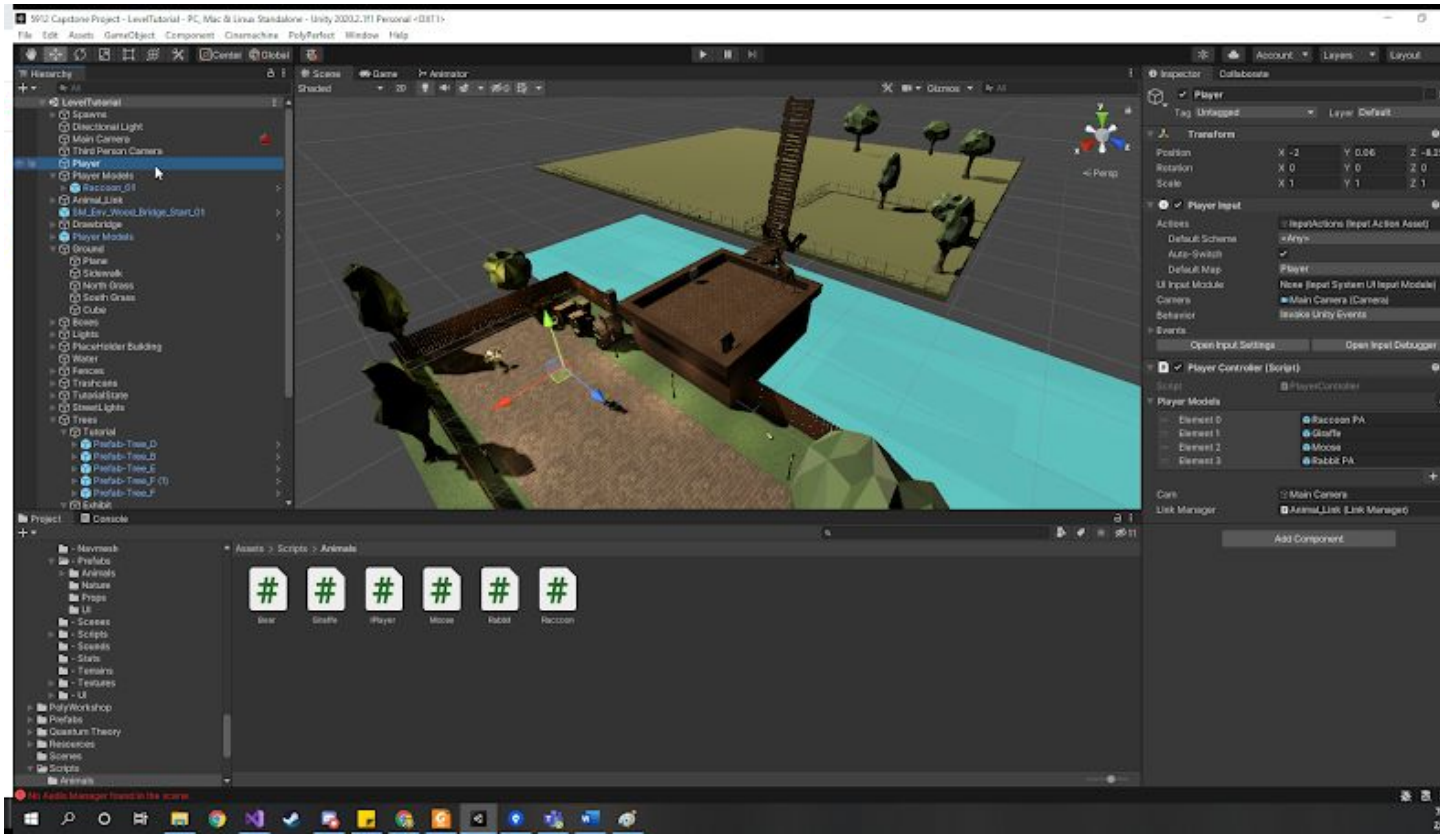
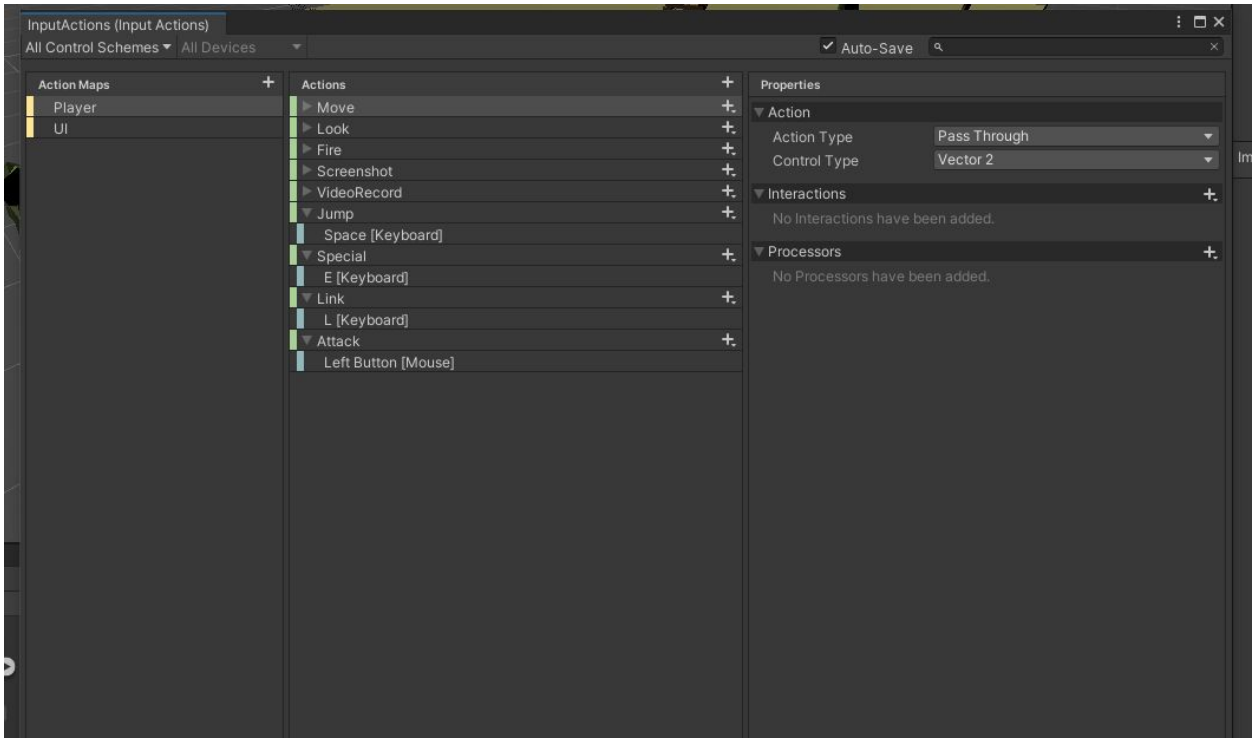
Controller (Player)

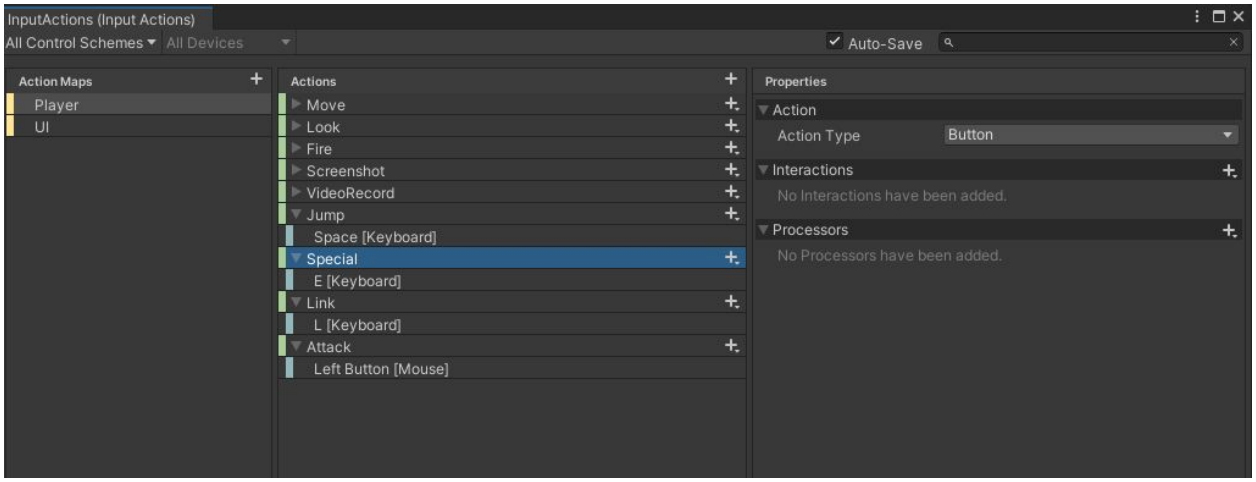


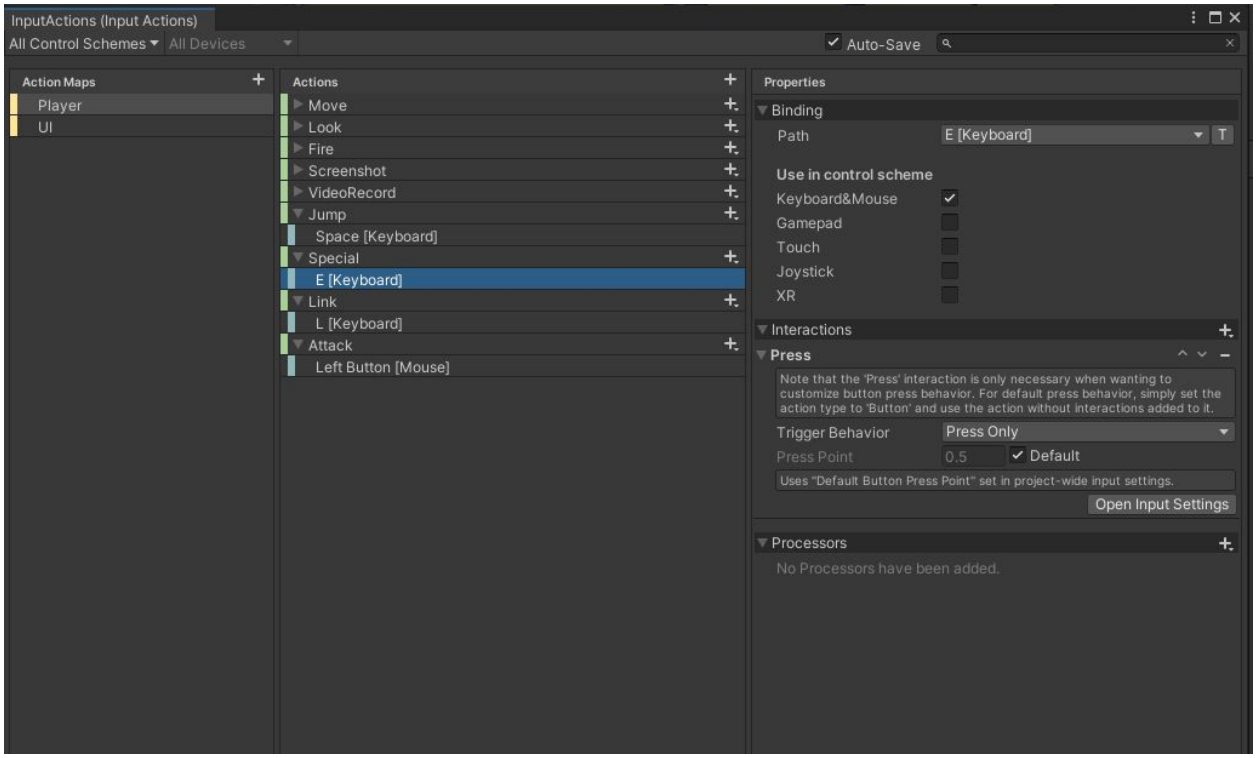
Player Input.
This is Unity's new Input manager.

Here, all the actions will take place



I create an example here on “Special”





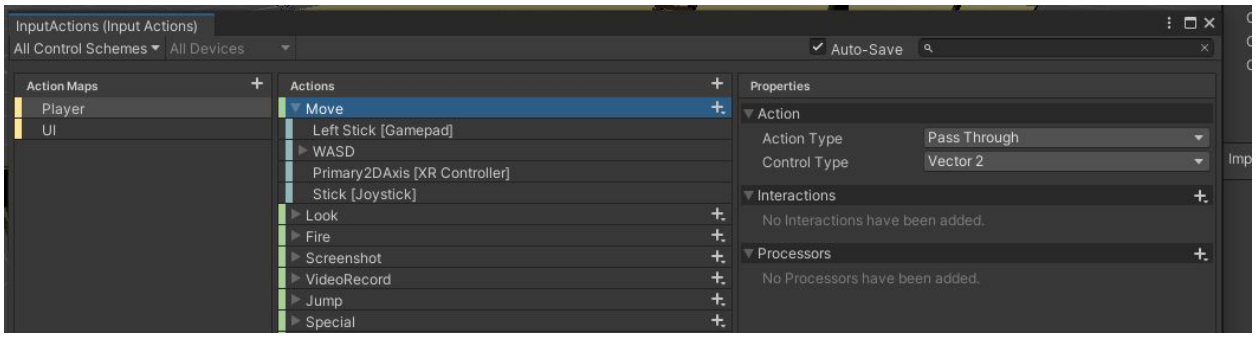
It is important you have press only on Trigger Behavior if you only want it to trigger when you press a key down.

Now inside the PlayerController script I added a couple of functions

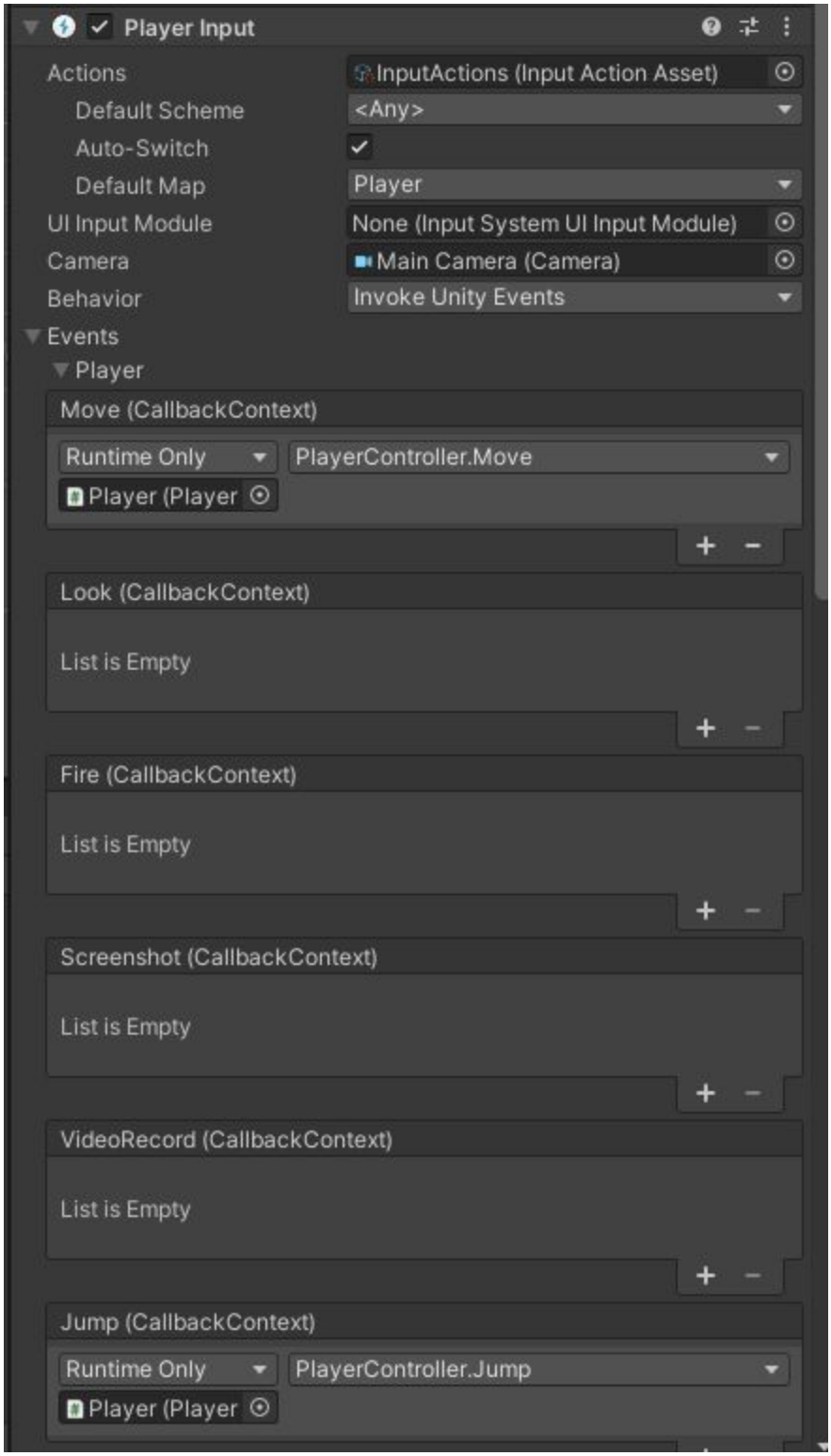
```
0 references | Alex Chan, 1 day ago | 1 author, 1 change
public void Move(InputAction.CallbackContext context)
{
    if (context.performed)
        inputDirection = context.ReadValue<Vector2>();
}

0 references | Alex Chan, 1 day ago | 1 author, 1 change
public void Jump(InputAction.CallbackContext context)
{
    if(context.performed)
        animal.Jump();
}

0 references | Alex Chan, 1 day ago | 1 author, 1 change
public void Attack(InputAction.CallbackContext context)
{
    if (context.performed)
        animal.Attack();
}

0 references | Alex Chan, 1 day ago | 1 author, 1 change
public void Link(InputAction.CallbackContext context)
{
    if (context.performed)
        animal.Link(LinkManager);
}

0 references | Alex Chan, 1 day ago | 1 author, 1 change
public void Special_One(InputAction.CallbackContext context)
{
    if (context.performed)
        animal.Special_One();
}
```

Context.performed is needed as if you don't have that, the event will trigger 3 times (why???)

You notice Move has a Vector2 read value,

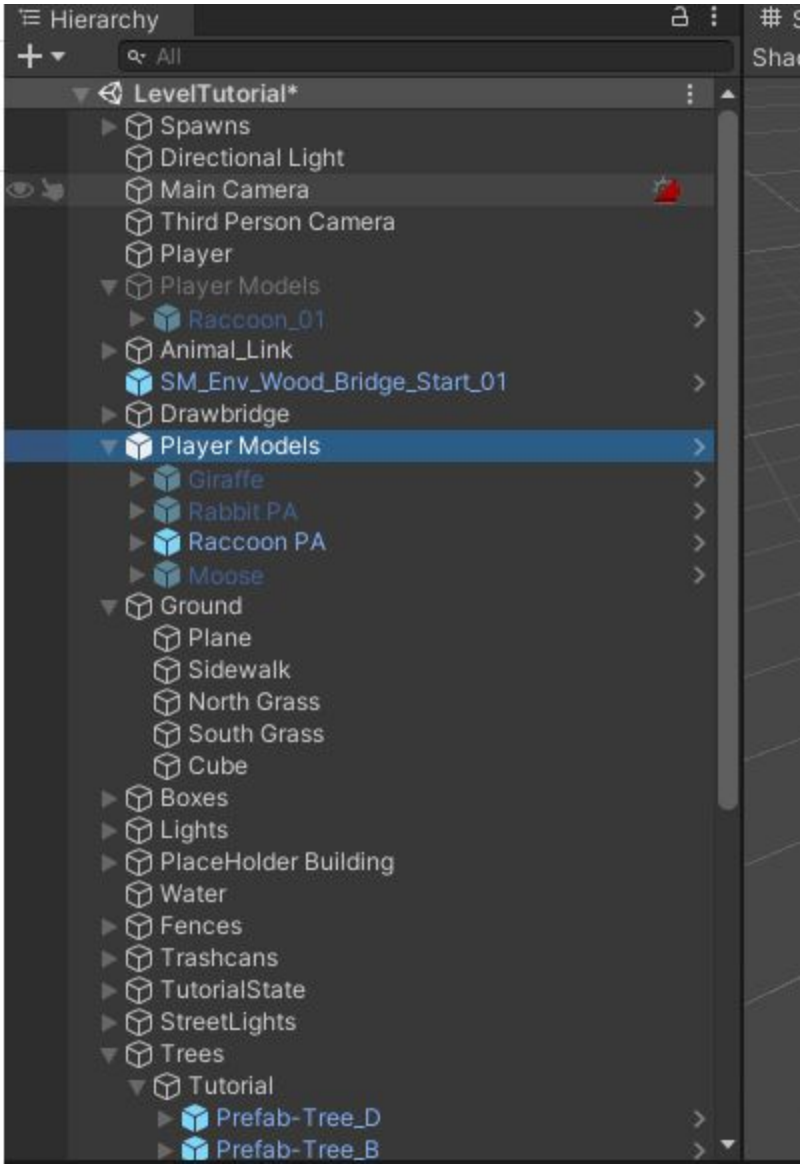


Next, go back to unity, and expand on Events and expand player



Here, you can call the methods you have made in the PlayerController Script.

IPlayer Documentation



The “Player Models” prefab is where the magic takes place.

IPlayer is an interface

```
public interface IPlayer
{
    10 references | Alex Chan, 1 day ago | 1 author, 1 change
    PlayerController AnimalPlayerController { get; }
    28 references | Alex Chan, 3 days ago | 1 author, 1 change
    Animator playerAnimator { get; set; }
    35 references | Alex Chan, 1 day ago | 1 author, 1 change
    CharacterController characterController { get; set; }
    13 references | Alex Chan, 3 days ago | 1 author, 1 change
    float movementVelocity { get; set; }
    24 references | Alex Chan, 3 days ago | 1 author, 1 change
    float jumpVelocity { get; set; }
    15 references | Alex Chan, 3 days ago | 1 author, 1 change
    float gravity { get; set; }
    13 references | Alex Chan, 3 days ago | 1 author, 1 change
    float maxJumpVelocity { get; set; }

    22 references | Alex Chan, 3 days ago | 1 author, 1 change
    Vector3 movement { get; set; }

    10 references | Alex Chan, 3 days ago | 1 author, 1 change
    Vector3 position { get; set; }

    9 references | Alex Chan, 3 days ago | 1 author, 1 change
    Vector3 rotation { get; set; }

    6 references | Alex Chan, 3 days ago | 1 author, 1 change
    void Jump();

    6 references | Alex Chan, 3 days ago | 1 author, 1 change
    void Update();

    6 references | Alex Chan, 3 days ago | 1 author, 1 change
    void Attack();

    6 references | Alex Chan, 3 days ago | 1 author, 1 change
    void Special_One();

    5 references | Alex Chan, 3 days ago | 1 author, 1 change
    void Interact();

    6 references | Alex Chan, 2 days ago | 1 author, 1 change
    void Link(LinkManager LinkManager);
    6 references | Alex Chan, 2 days ago | 1 author, 1 change
    GameObject LinkHitBox { get; }

    6 references | Alex Chan, 3 days ago | 1 author, 1 change
    void Move(Vector3 movementDirection);
```

The PlayerController is the PlayerController from above
The Animator is specific to the animal and animates it
The CharacterController is also specific to the animal which is basically a collider
Then there are stats, and methods.
These methods are called by pressing a key on teh keyboard as seen above.

The LinkHitBox basically is a hitbox that makes it so if you are close enough to an animal, it will link it.

Each animal that is implemented to IPlayer can have unique implementations of these functions and stats.

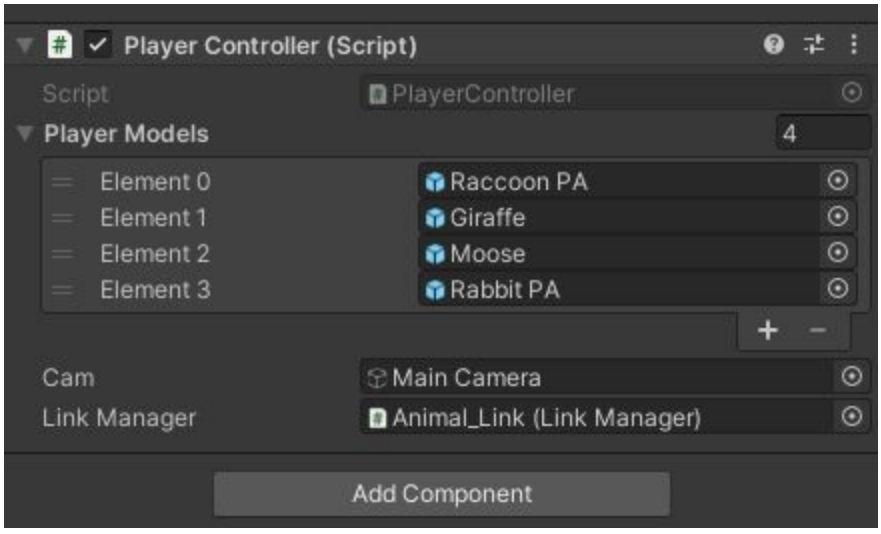
AnimalModelIndex

This class creates new animals based on their index number
This class will be revamped once we put the Iplayer scripts into the specific animals

But for now, there is a strict order to the animals

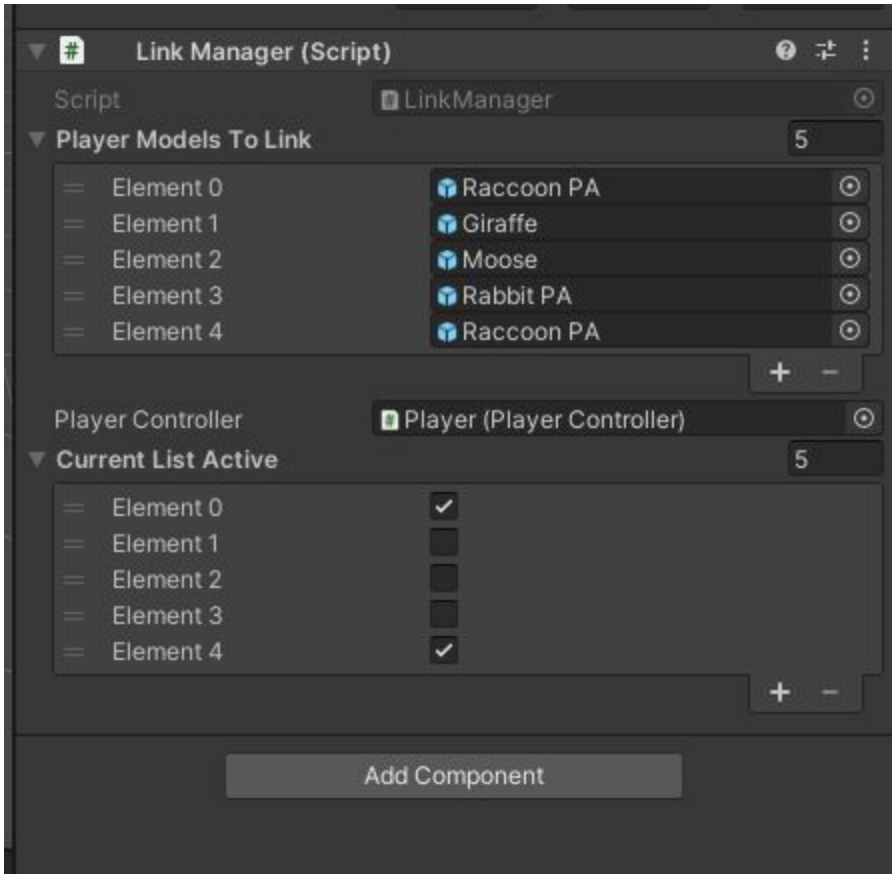
```
case 0:
   return new Raccoon(CharacterController, playerAnimator, playerController);
case 1:
   return new Giraffe(CharacterController, playerAnimator, playerController);
case 2:
   return new Moose(CharacterController, playerAnimator, playerController);
case 3:
   return new Rabbit(CharacterController, playerAnimator, playerController);
```

This order is also important for the PlayerController



Animal_Link

This keeps track of which animals you can link in that specific stage. And if the animal is linked (and thus can switch to that animal)



Here, the ordering is important again. Since you start out as a raccoon, if there are any placeholder animal indexes, just put the Raccoon, and make sure to check that the link to raccoon is true (element 0 and 4)

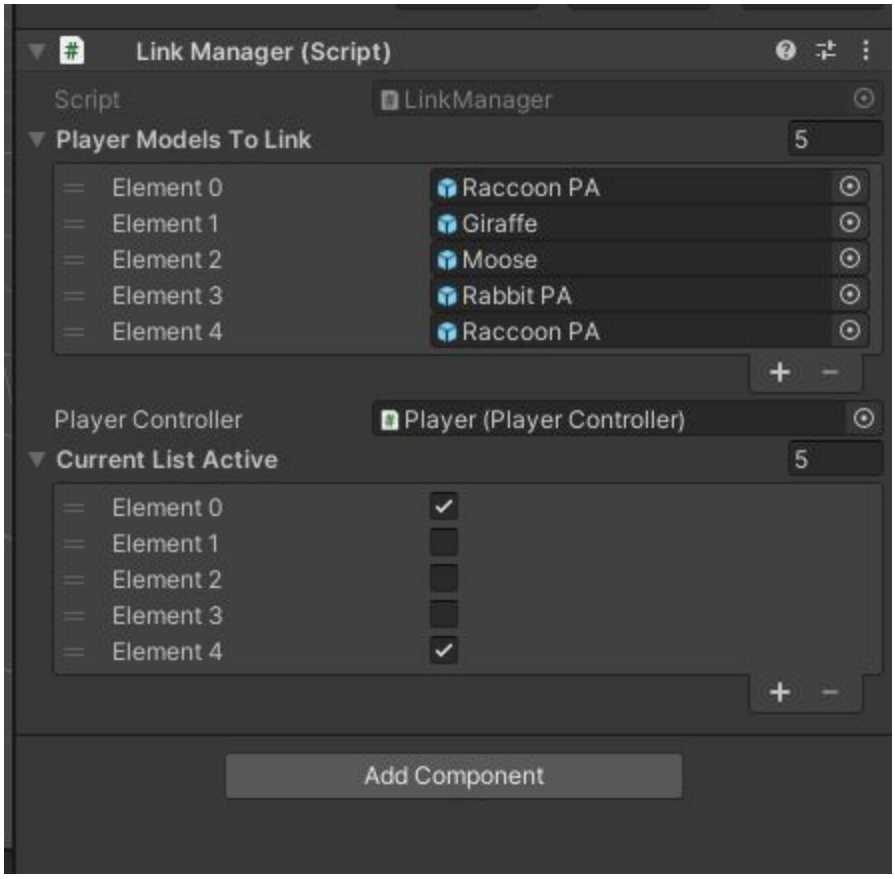
Once you link, the current list active will be true for that animal

```
public class LinkManager : MonoBehaviour
{
    public GameObject[] playerModelsToLink = new GameObject[5];
    public PlayerController playerController;
    public List<IPlayer> AnimalLinkList = new List<IPlayer>();
    public IPlayer[] CurrentList = new IPlayer[5];
    public bool[] CurrentListActive = new bool[5];

    0 references | Alex Chan, 2 days ago | 1 author, 1 change
    public LinkManager()
    {
        CurrentListActive[0] = true; //raccoon
        CurrentListActive[1] = false;
        CurrentListActive[2] = false;
        CurrentListActive[3] = false;
        CurrentListActive[4] = true; //placeholder
    }

    5 references | Alex Chan, 2 days ago | 1 author, 1 change
    public void LinkAnimal()
    {
        for (int i = 0; i < playerModelsToLink.Length; i++)
        {
            if (!CurrentListActive[i])
            {
                Vector3 playerLoc = playerController.animal.position;
                Transform hitbox = playerModelsToLink[i].transform.Find("LinkHitBox"
                Debug.Log(i);
                Vector3 LinkLocation = hitbox.position;
                if (Vector3.Distance(playerLoc, LinkLocation) <= 5f)
                {
                    //Link the animal
                    CurrentListActive[i] = true;
                    playerModelsToLink[i].SetActive(false);
                }
            }

        }
    }


}
```

This is where the link check is

(The start() method does nothing, ignore it, you can check the boxes outside the script)

This class could be written a bit better, but if there is only like a max of 5-6 animals per level, it is not that hard to swap out the animals in this part



But THe AnimalModelIndex will be wrong

But after we put the scripts of the animal to the gameobject, the animaModelIndex will be revamped and it won't be hard to set different animals per level.