UI Touch Controls for Mobiles

This is a documentation for the JNM UI Touch Controls for mobiles in which is explained how to use the contained components.

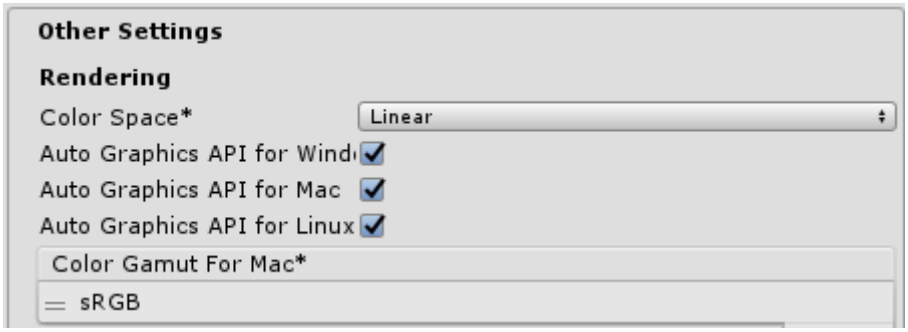
Setup

After downloading and importing the asset from the assetstore you can open the *Demo* scene in the folder *JNMTouchControls/Scenes* to get started.

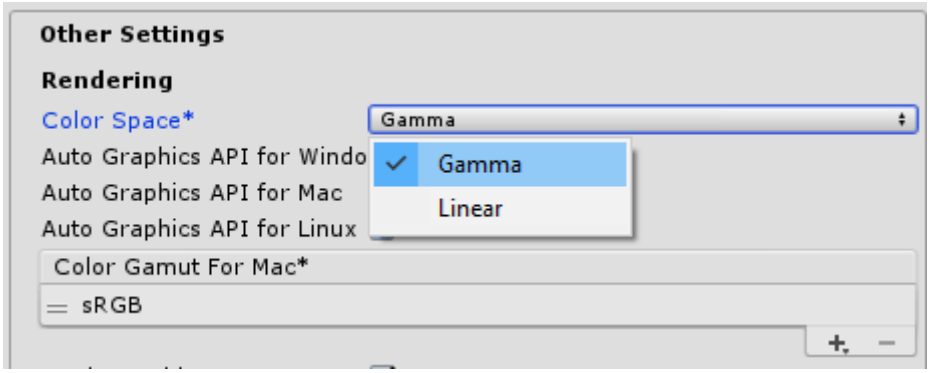
To get the best quality for the graphics it is recommended to download and import the asset *Post Processing Stack* from the unity assetstore which is available here for free.

https://assetstore.unity.com/packages/essentials/post-processing-stack-83912

Also set the Color Space in the Player Settings to *Linear*:

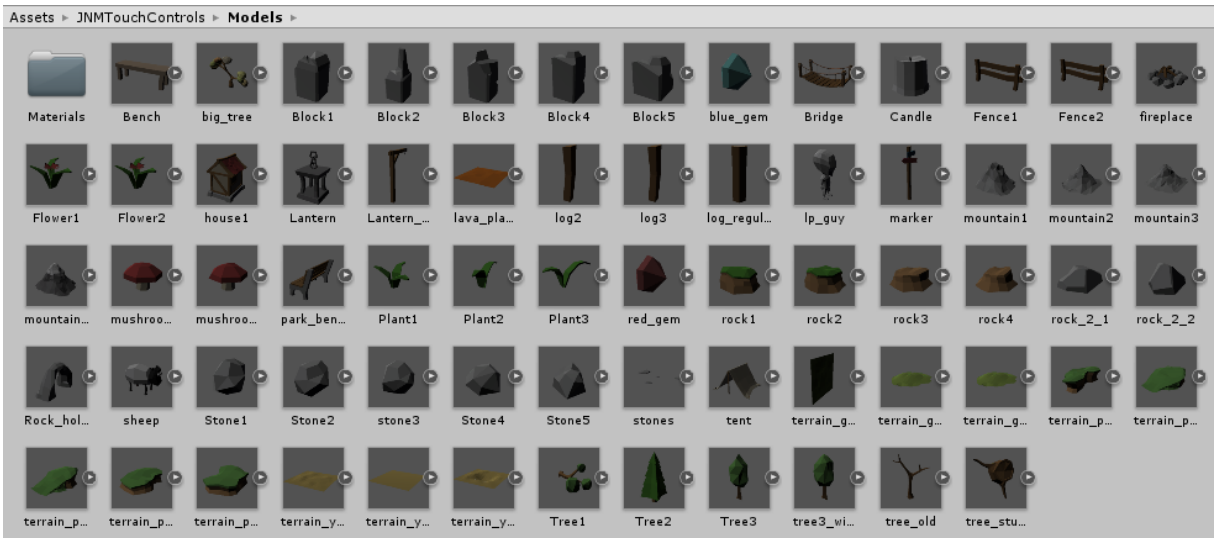


But if you want to deploy your game on a mobile device which only supports Gamma Color Space you have to set it to *Gamma*.

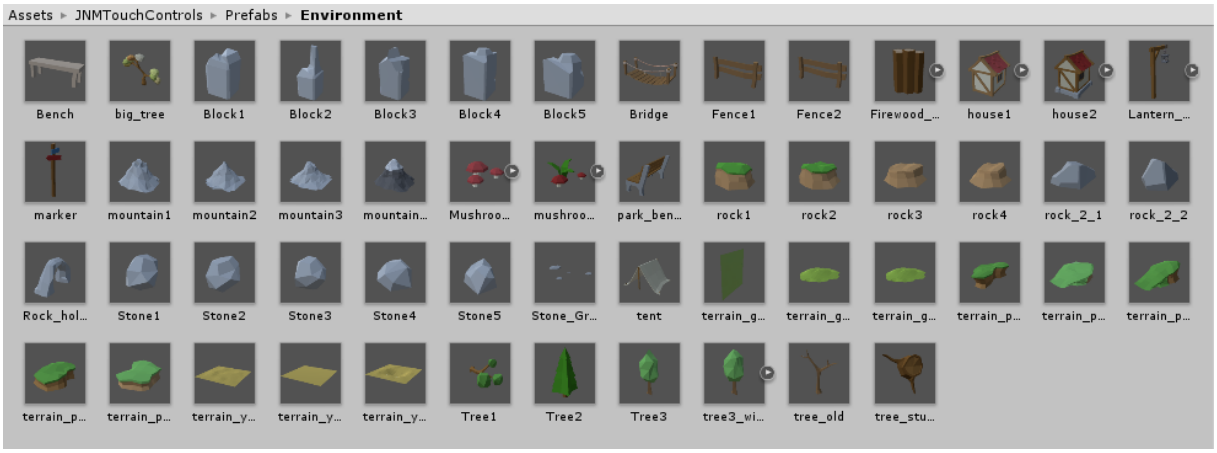


Meshes

As a bonus I added the meshes from my *Low Poly Game Kit* that you can also download from the assetstore to this asset to get you started with a nice demo scene and a third person character.

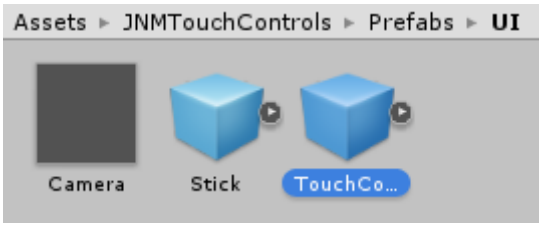


You can also use these meshes as Prefabs, that already have appropriate colliders attached:

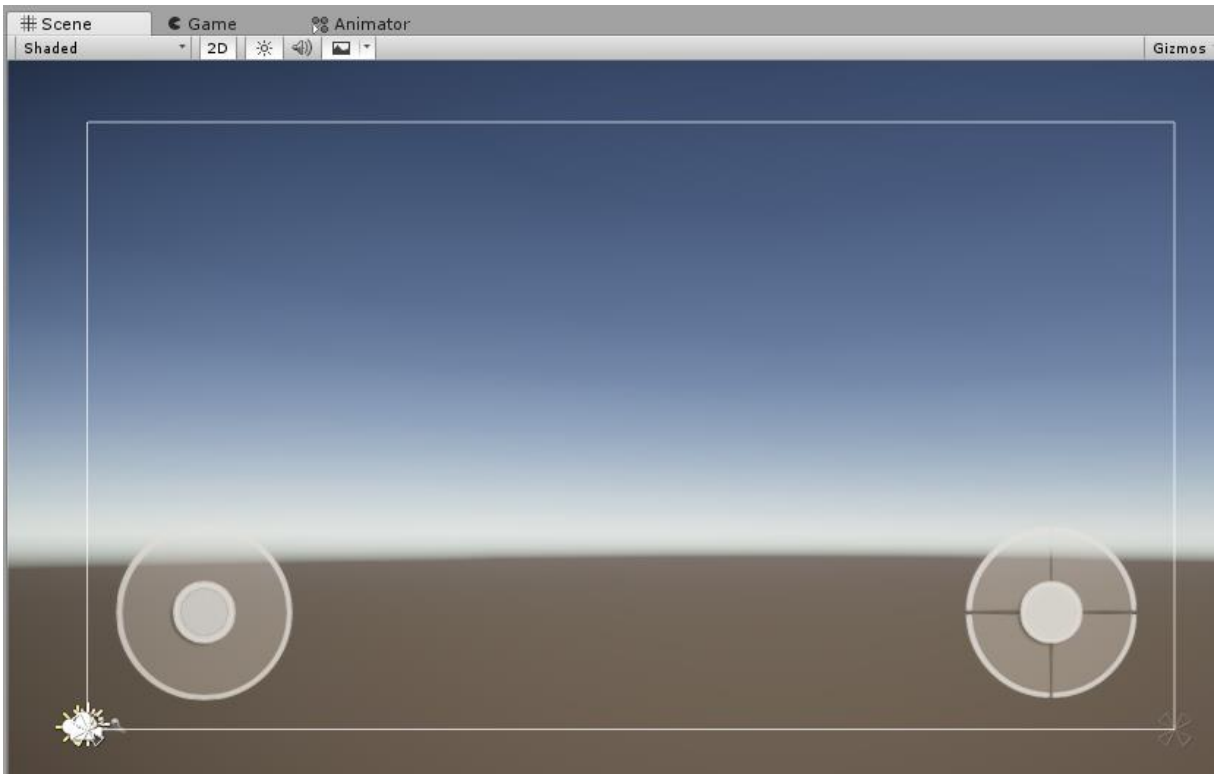


The player character you can find in the Prefabs/Characters folder.

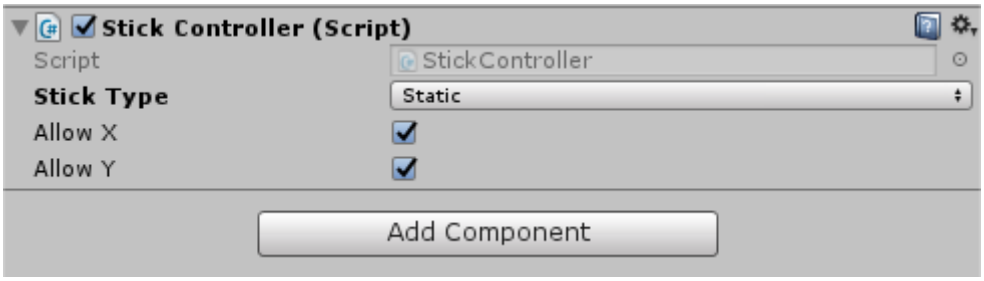
In the Prefabs/UI folder there is a camera prefabs with a script attached to follow the player and the main asset, the TouchControl prefab, that is a UI Canvas with a left and right mobile UI control stick:



When you drag the *TouchControl* into a new scene, or a scene without a Canvas, the result in 2D mode looks like this:



Each stick has a StickController script attached:



You can set the StickType to *Dynamic* to let the stick appear at the position where you touch the screen. You can also limit the allowed axis to X or Y.

The script raises an event that you can use to control the player's movement or the camera's rotation.

Setup player's movement

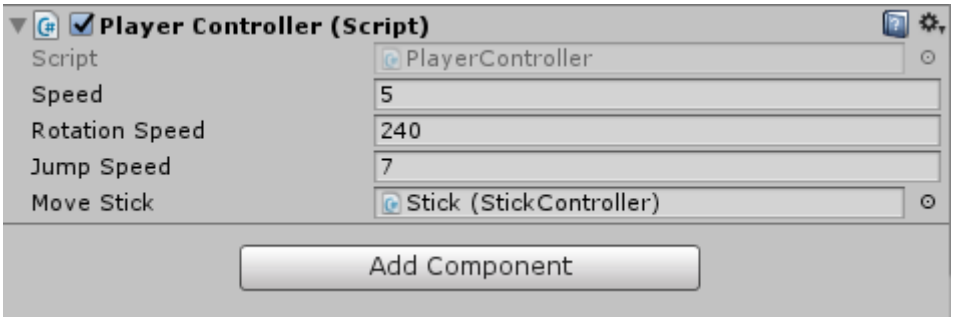
The player and the camera prefabs already have this functionality implemented, this is the code of the *PlayerController* script, that is attached to the player:

```
public StickController MoveStick;

void Awake()
{
    if(MoveStick != null)
    {
        MoveStick.StickChanged += MoveStick_StickChanged;
    }
}

private Vector2 MoveStickPos = Vector2.zero;

private void MoveStick_StickChanged(object sender, StickEventArgs e)
{
    MoveStickPos = e.Position;
}
```

In the *Update* method you can use this MoveTick.Position now instead of the axis values for Horizontal or Vertical movement:

```
// Update is called once per frame
void Update()
{

    // Get Input for axis
    float h = MoveStickPos.x; // Input.GetAxis("Horizontal");
    float v = MoveStickPos.y; // Input.GetAxis("Vertical");
```

In the Unity Editor you just have to assign the stick to the public variable *MoveStick*:



Setup Camera follow

You can use the Camera prefab, for which the setup is already made, but if you want to add the camera rotation functionality to your existing camera you can do it like this.

Add this code to you camera script:

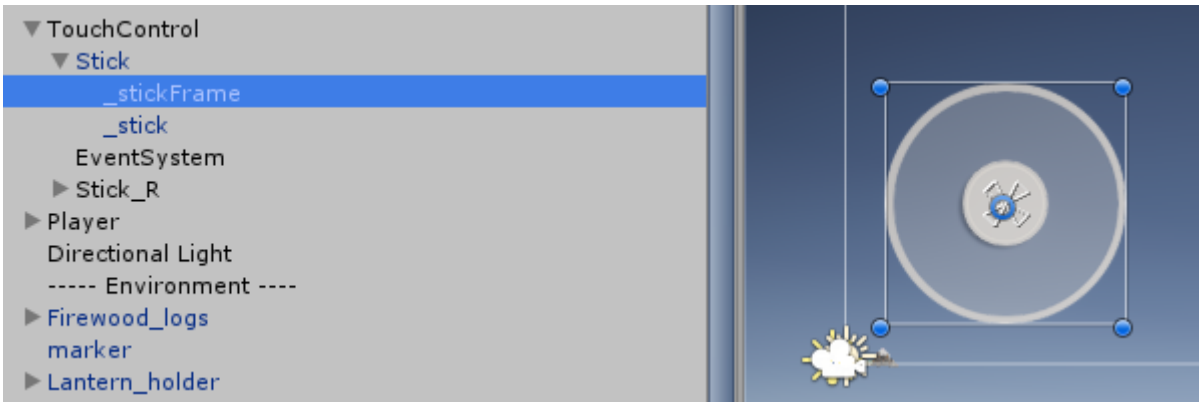
```
public StickController CameraStick;

void Awake()
{
    if (CameraStick != null)
    {
        CameraStick.StickChanged += CameraStick_StickChanged;
    }
}

private Vector2 CameraStickPos = Vector2.zero;

private void CameraStick_StickChanged(object sender, StickEventArgs e)
{
    CameraStickPos = e.Position;
}
```

And in the *LateUpdate* method you can use this code to get the values for camera rotation and pitch:

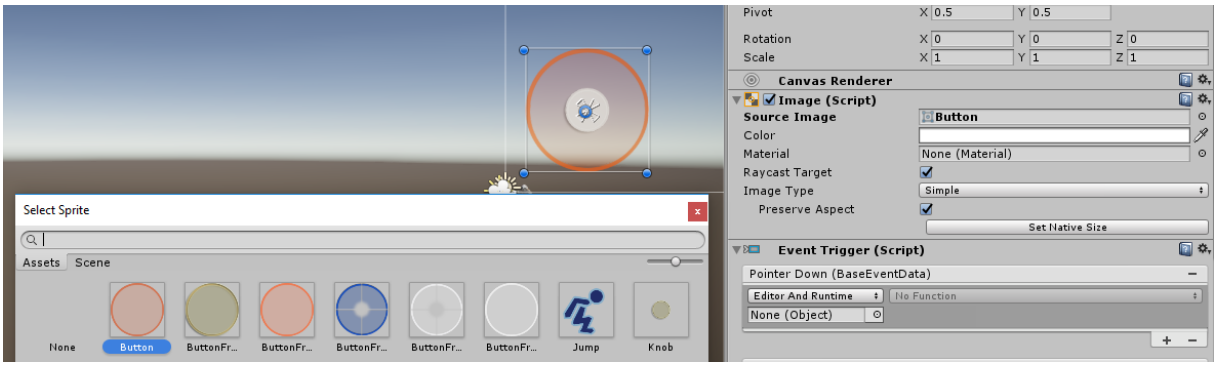
```
// LateUpdate is called after Update methods
void LateUpdate()
{
    float h = CameraStickPos.x * RotationSpeed;
    float v = CameraStickPos.y * PitchSpeed;
```

UI

The Stick control has a _stickFrame and a _stick Sprite:



You can change this easily by assigning a different sprite in the editor:



If you want to scale the control, don’t scale the parent game object, scale the objects _stickFrame and _stick instead.