3D Interactive Knobs, Sliders and Buttons

By float_oat

This Unity asset package allows you to add knobs, sliders and buttons to 3D scenes. The player can interact with these controls to change different variables in the game, such as moving a door, changing the sound volume or turning off lights. If you have any questions or need help, send me an email at floatoat@gmail.com

Usage quick guide **1**

Basic use tutorials **1**

Adding controls to a scene 2

Opening a door with a knob 2

Changing audio volume with a slider 6

Turning off lights with a button 11

All settings **13**

Knobs and sliders 13

Knobs 14

Sliders 14

Buttons 14

Extending functionality **15**

Changing the models 15

Adding functionality in the code 15

Interacting with other devices than the mouse 16

Endnotes **16**

Usage quick guide

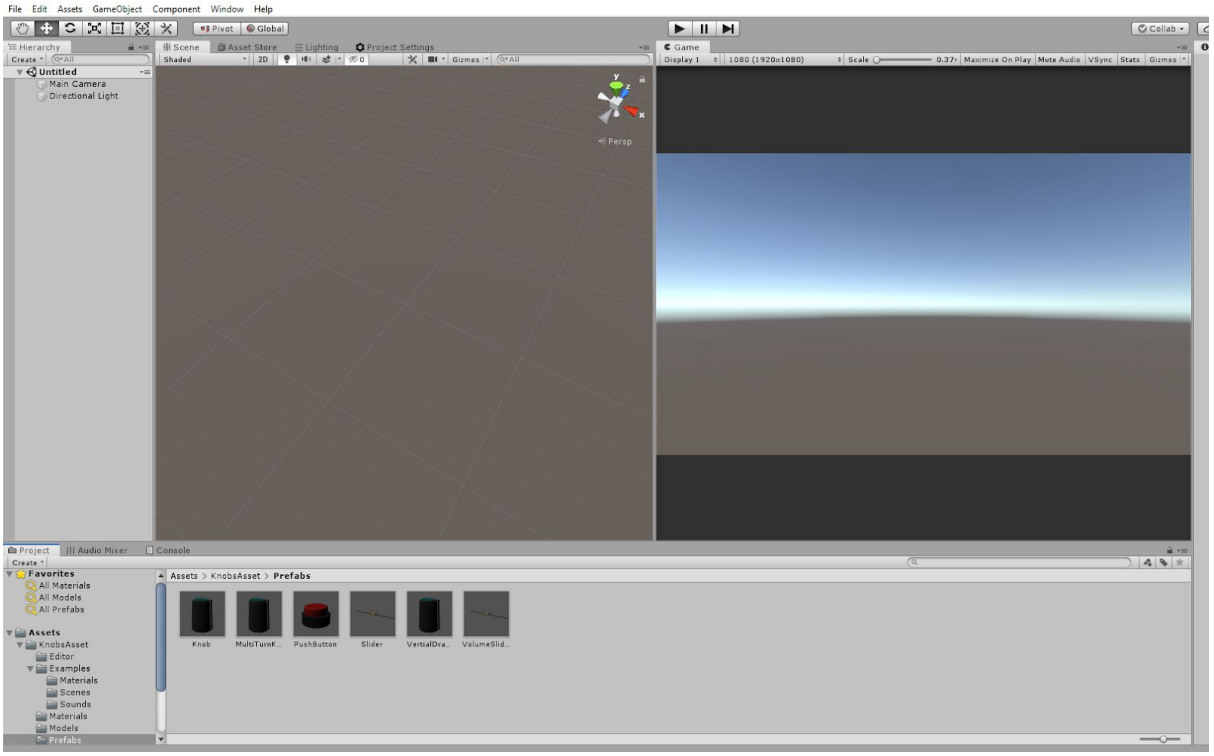
Drag prefabs from the prefabs folder into your scene. For buttons, define Unity actions for what happens when the button is pressed. For knobs and sliders, attach different “KnobListener” scripts (TransformKnobListener, LightKnobListener, MixerParamKnobListener, or your own implementation of KnobListener) to define what the knobs/sliders control. Knobs and sliders can also use Unity actions if you want.

Basic use tutorials

This section contains tutorials for using this asset package. These tutorials require that you import the package, including the “Examples” folder. Completed versions of these tutorials can be found under Examples/Scenes/TutorialScenes.

Adding controls to a scene

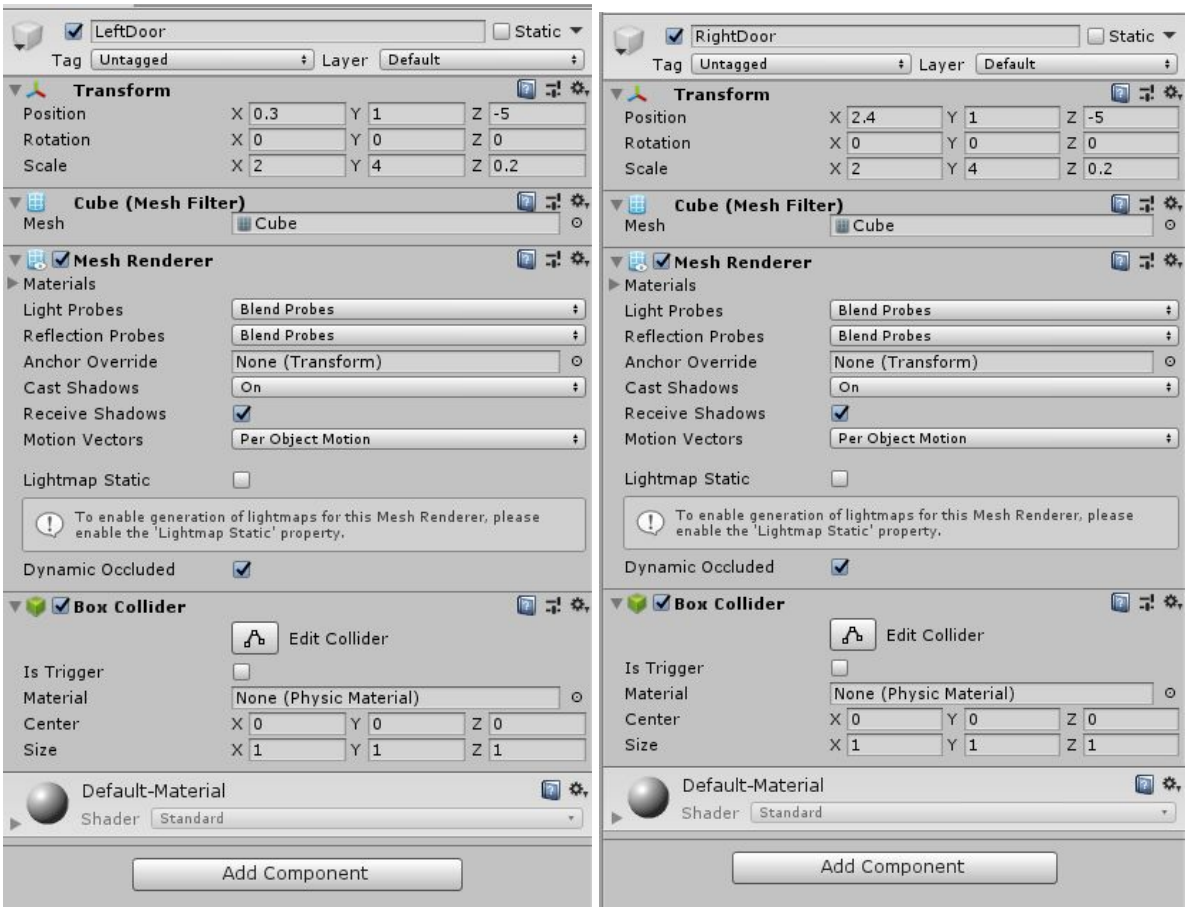
To start using the controls, navigate to the prefabs folder of the asset package.



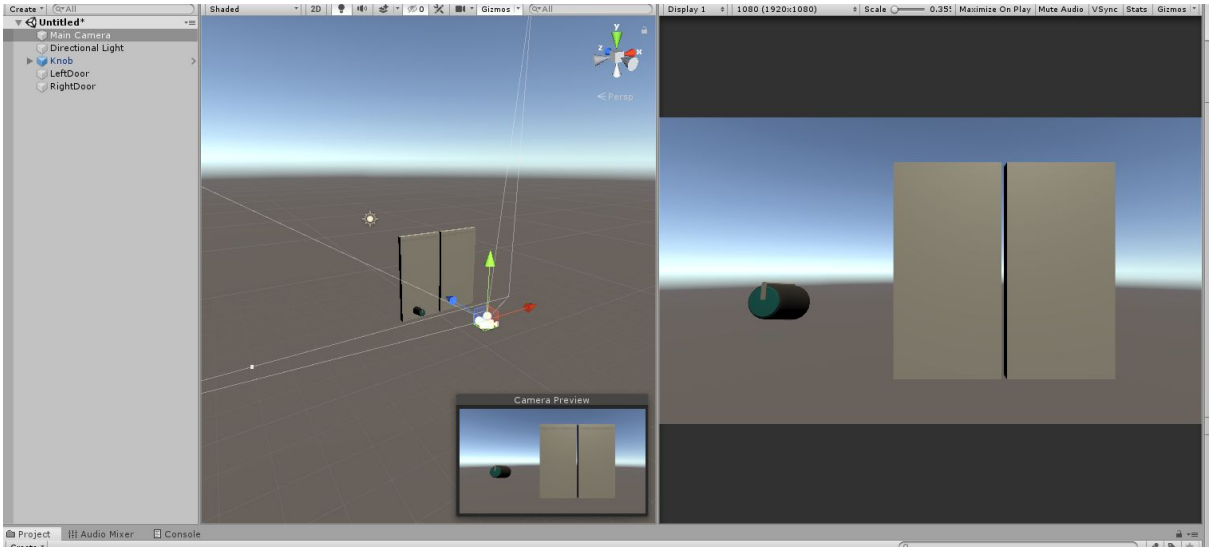
The basic versions of the prefabs with standard settings are “Knob”, “PushButton” and “Slider”. Drag the prefabs into your scene, then position and scale them to where you want them to be.

Opening a door with a knob

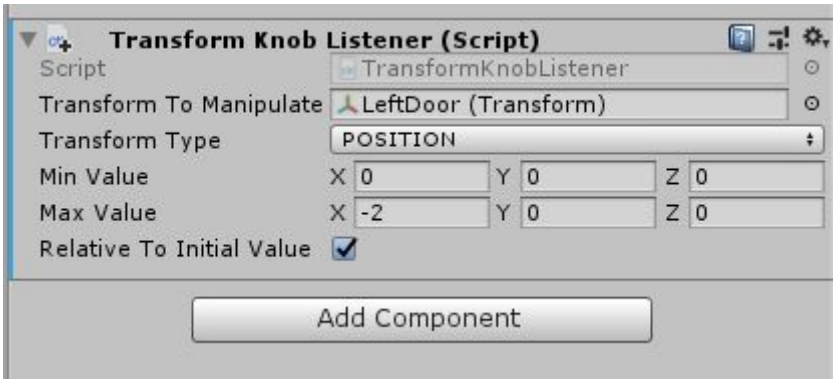
We’re going to be making a pair of sliding doors, like on subway trains. Your player will be able to open and close these doors by turning a knob. Add a knob (found in the prefabs folder) and position it to be facing the camera. Add a cube to your scene, renamed it “LeftDoor” and scale it to be door shaped. Duplicate it, rename the duplicate “RightDoor” and position it to be next to the first door. Here are my left and right door objects, although it’s ok if yours have different transform values.



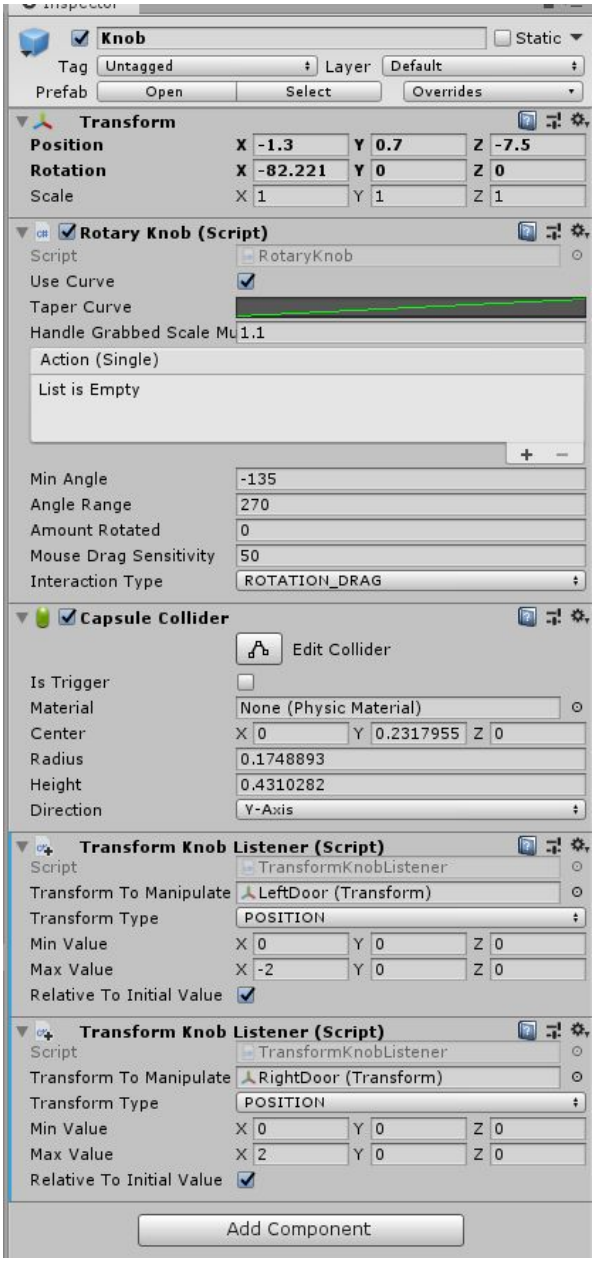
Your scene should look something like this:



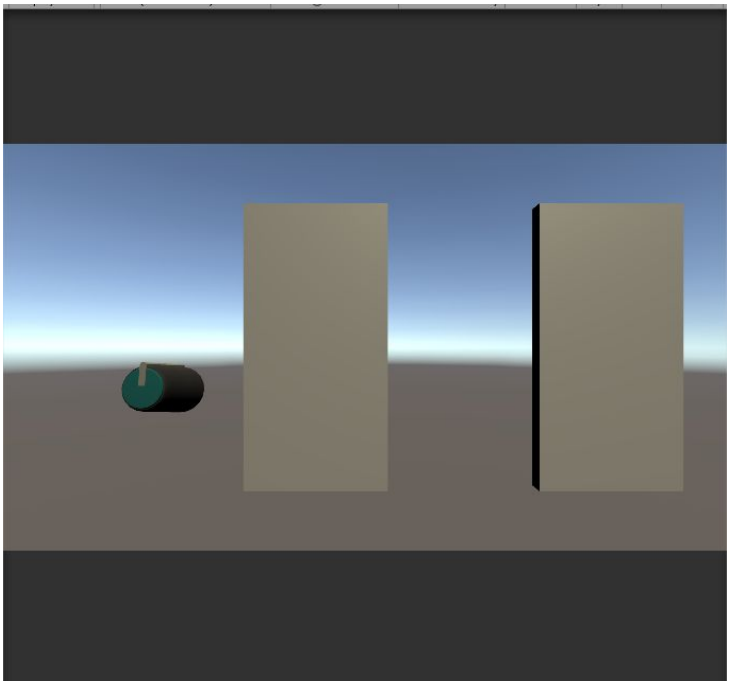
If you run the scene now, you should be able to click and drag on the knob to turn it, but the doors will not be opening yet. To add functionality, we will be adding knob listeners to the knob. Select the knob object, click “Add Component” and type in “Transform Knob Listener” and add it to the game object. Drag the LeftDoor Into the “Transform To Manipulate” field, and change the “Max Value” to be [-2, 0, 0]



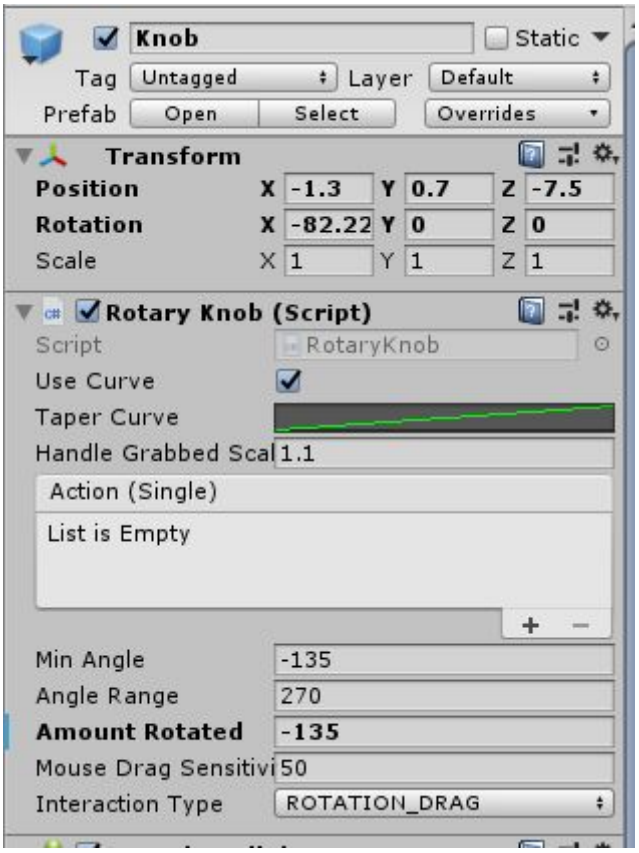
If you run the scene now, turning the knob should move the left door back and forth. Now, let's make the right door move too. Add another Transform Knob Listener to the knob, and this time drag in the right door as the "Transform to manipulate", and set the Max Value to [2, 0, 0]. The knob object should look something like this now:

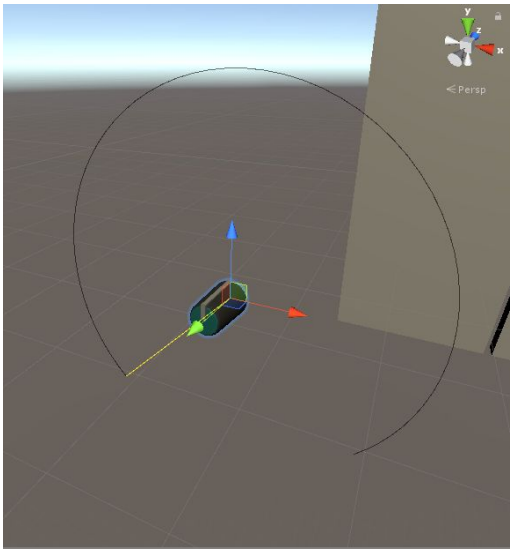


When we run the scene now, both doors are moved by the knob. However, they start off halfway open.



We want the doors to start closed, so to fix this we need to set the knob's initial rotation.

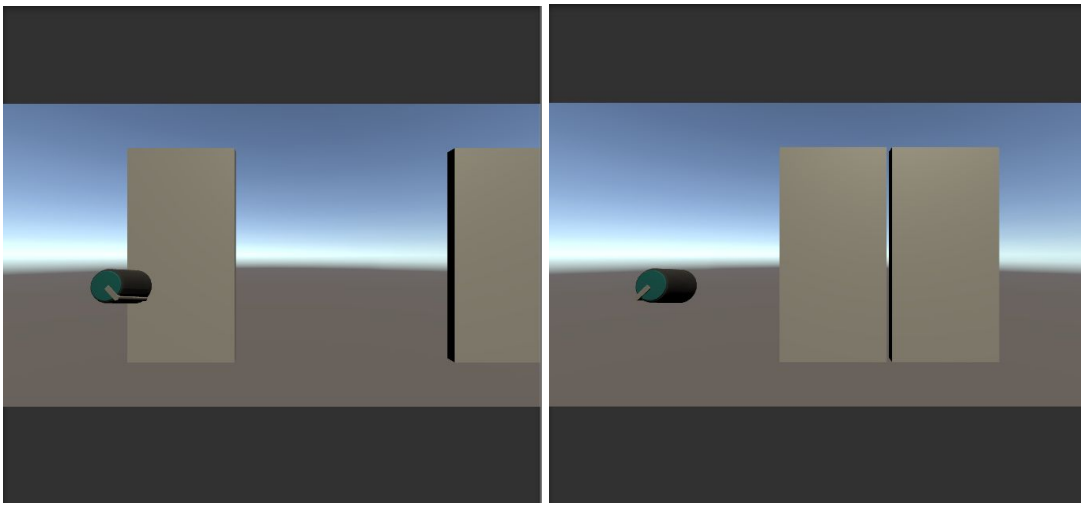




If we look at the parameters of the Rotary Knob component, we can specify the minimum angle of rotation, the angle range and amount rotated. The "Amount Rotated" variable contains how far past 0 the forward vector the knob is currently rotated. Let's change it from 0 to -135 so that the knob starts at its minimum position. With the knob object selected, we

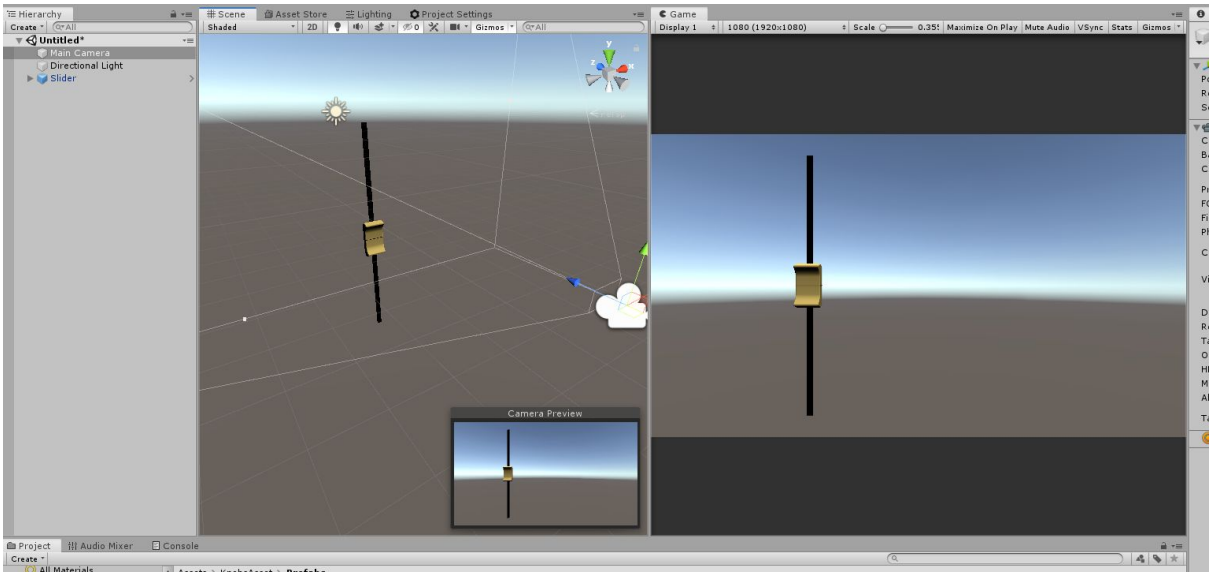
should see in the Scene view that the filled red angle to not showing, meaning the knob will start rotated all the way to the left.

Play the scene now, and the doors will start off closed, and can be opened by turning the knob.



Changing audio volume with a slider

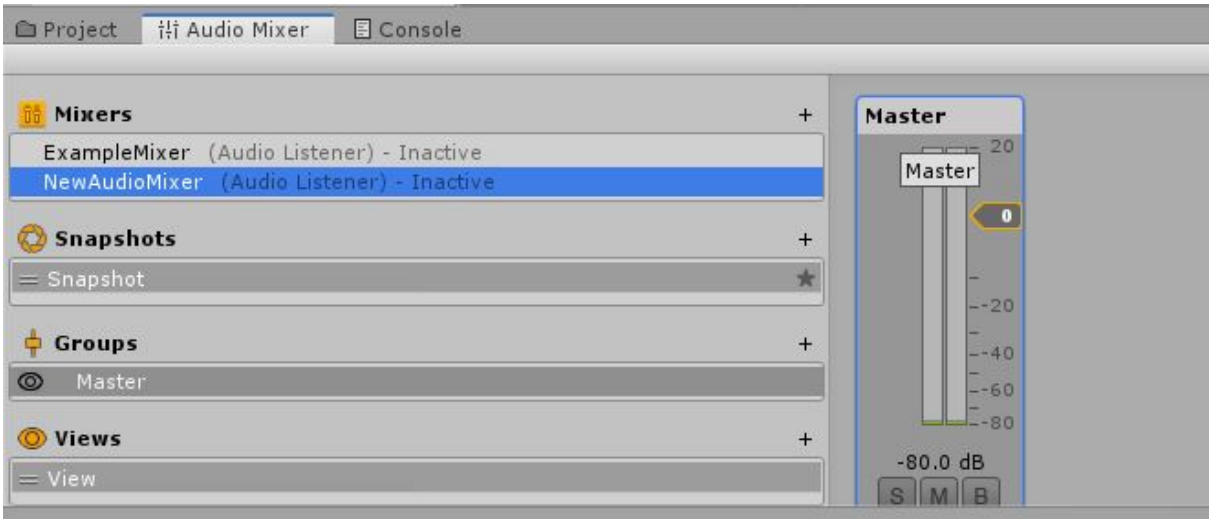
Let's make a slider for controlling the game's music volume. Add a slider (from the prefabs folder) to the scene. Position and rotate it so that it's facing the camera.



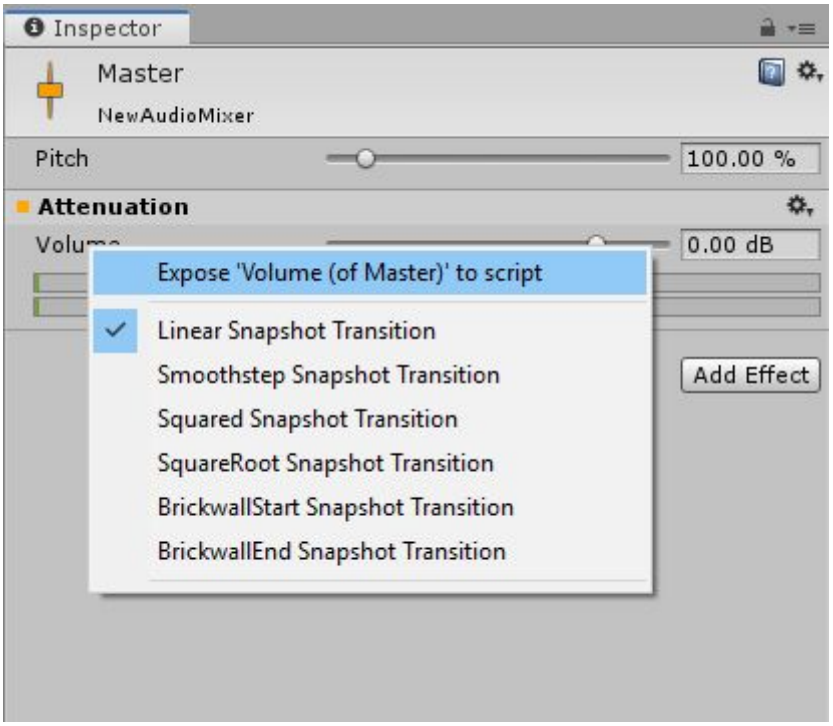
If you run the scene now, you should be able to click and drag on the slider to move it up and down.

Now, to make the slider into a volume control, we'll need some sound and an Audio Mixer.

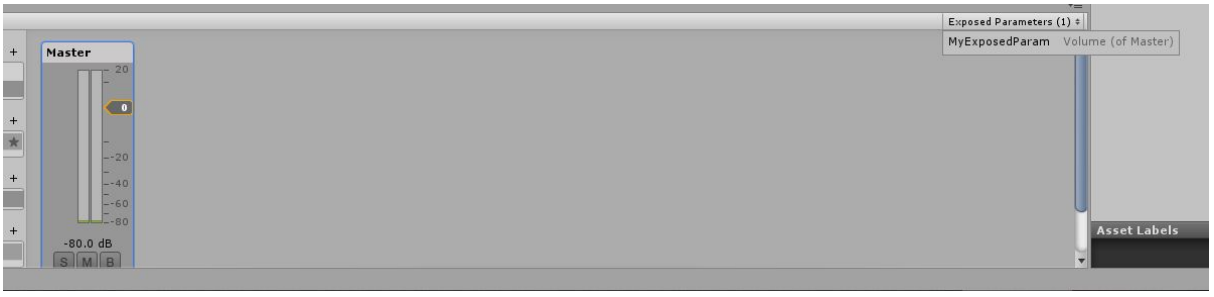
First, let's set up the mixer. Create an AudioMixer and select the master AudioMixerGroup.



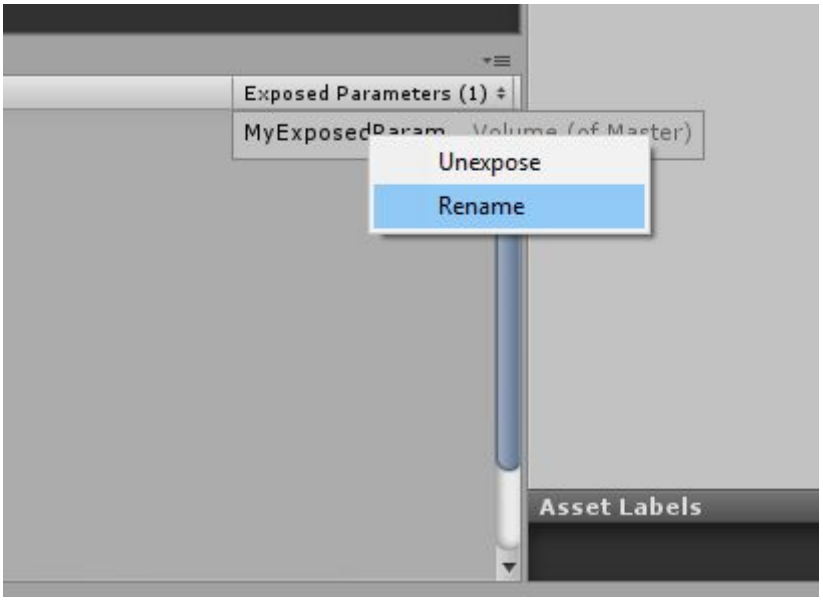
Then, with master selected, go to the inspector and right click on the Volume parameter of the Attenuation effect. Select “Expose ‘Volume (of Master)’ to script”.

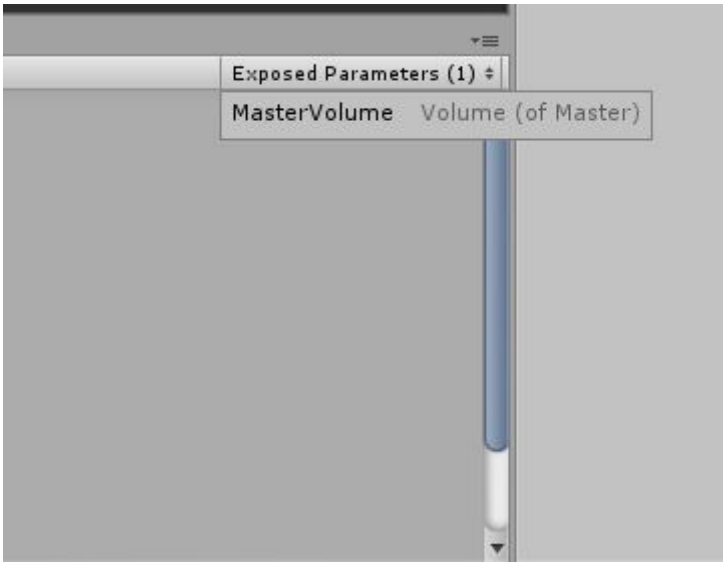


Now, find the “Exposed Parameters” dropdown in the mixer and click it.



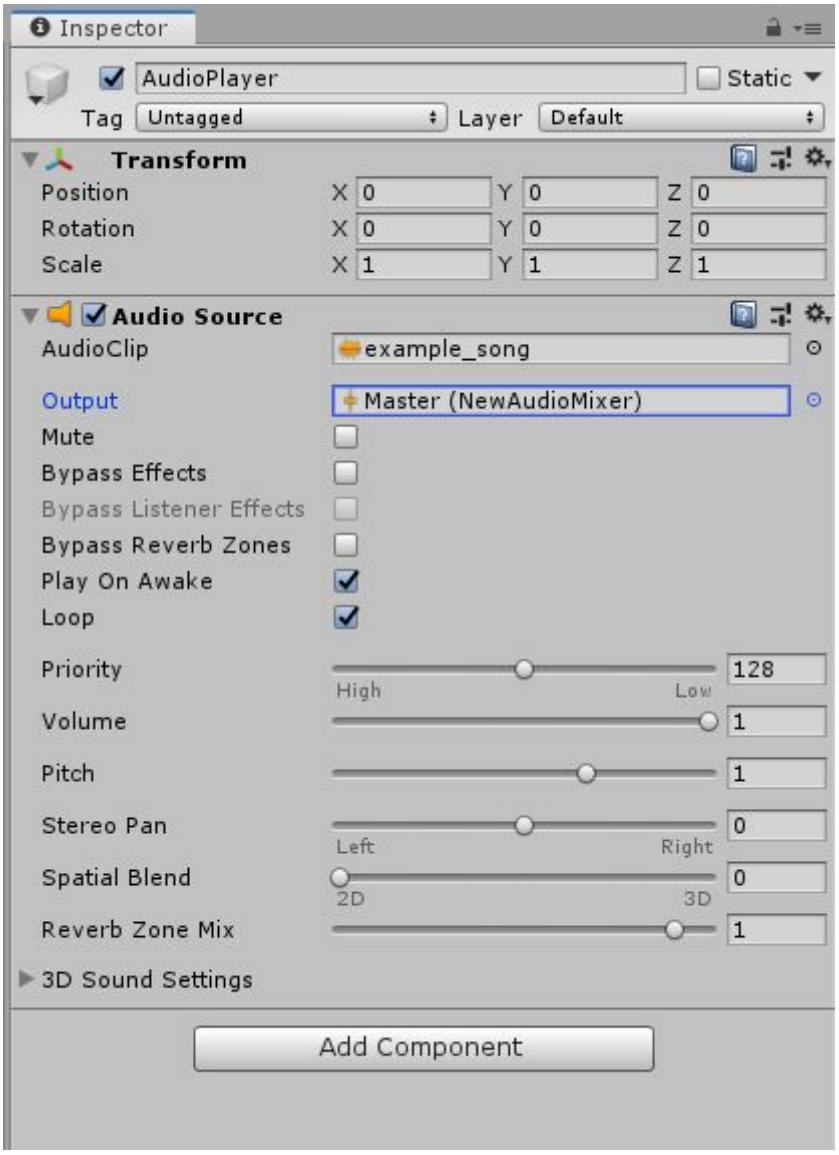
Then, right click on “MyExposedParam” and rename it to “MasterVolume”



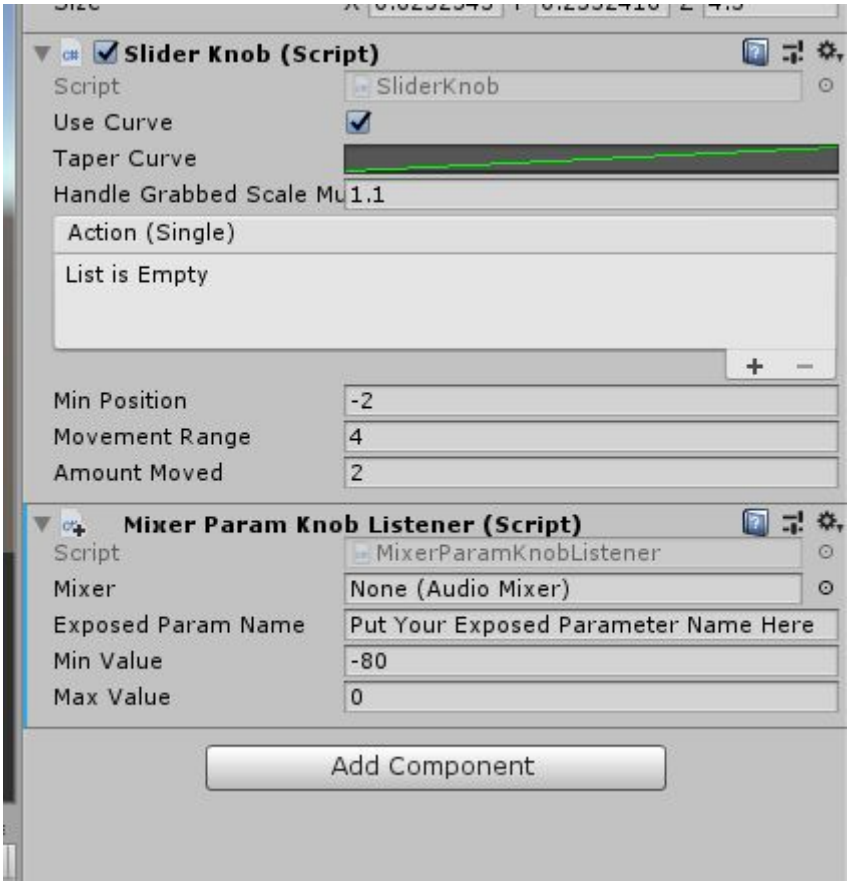


Our AudioMixer is now setup to have it's volume controlled by the slider.

We need our scene to play some music. Add a gameobject to the scene, attach an AudioSource, set the clip to "example_song", check the "loop" checkbox, **and set the output to be the Master channel of the mixer you just made**. Your audio object should look like this:


Inspector
AudioPlayer
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Audio Source
AudioClip
example_song
Output
Master (NewAudioMixer)
Mute
Bypass Effects
Bypass Listener Effects
Bypass Reverb Zones
Play On Awake
Loop
Priority
128
High
Low
Volume
1
Pitch
1
Stereo Pan
0
Left
Right
Spatial Blend
0
2D
3D
Reverb Zone Mix
1
3D Sound Settings
Add Component

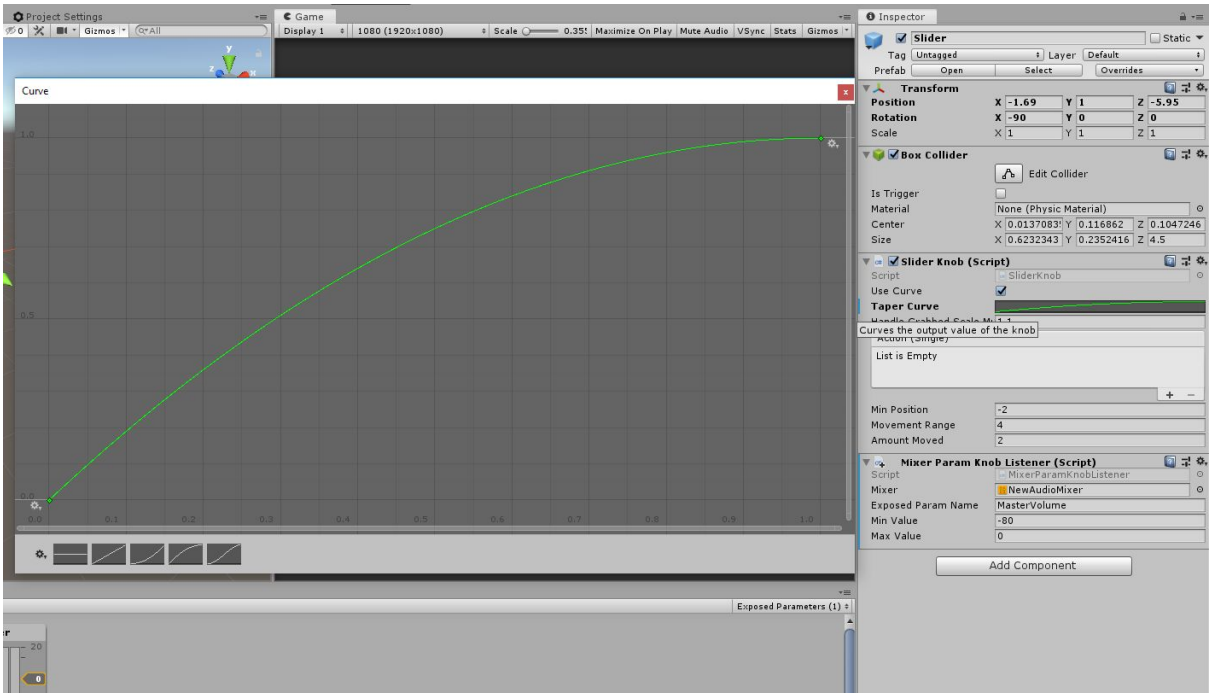
Now the last part, lets setup the slider to control the volume. Select the slider object in the scene and add a “Mixer Param Knob Listener” component.



Set the Mixer field to be the mixer you created earlier, and set the Exposed Param Name to be “MasterVolume”.

The Min Value and Max Value are already set to the correct values for a volume control.

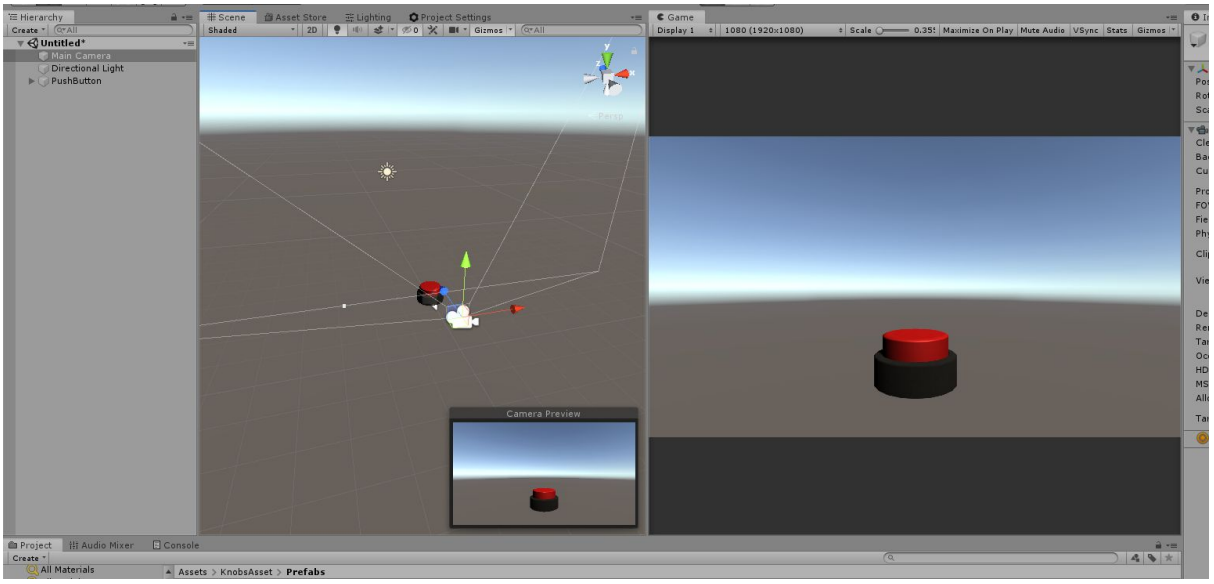
Run the scene and drag the slider up and down. The slider should be working as a volume control, but it’s not quite right because the music seems silent for most of the slider positions except the very top. This is because audio volume is perceived logarithmically, but the slider is controlling the volume linearly. To fix this, select the slider object and click on the “Taper Curve” field. Change the curve to this shape (select it from the presets at the bottom).



Now, the volume control should work as expected. This preset curve is not a true logarithmic curve, but it's close enough for most uses. Try adjusting it to be steeper at the beginning to be closer to a logarithmic curve.

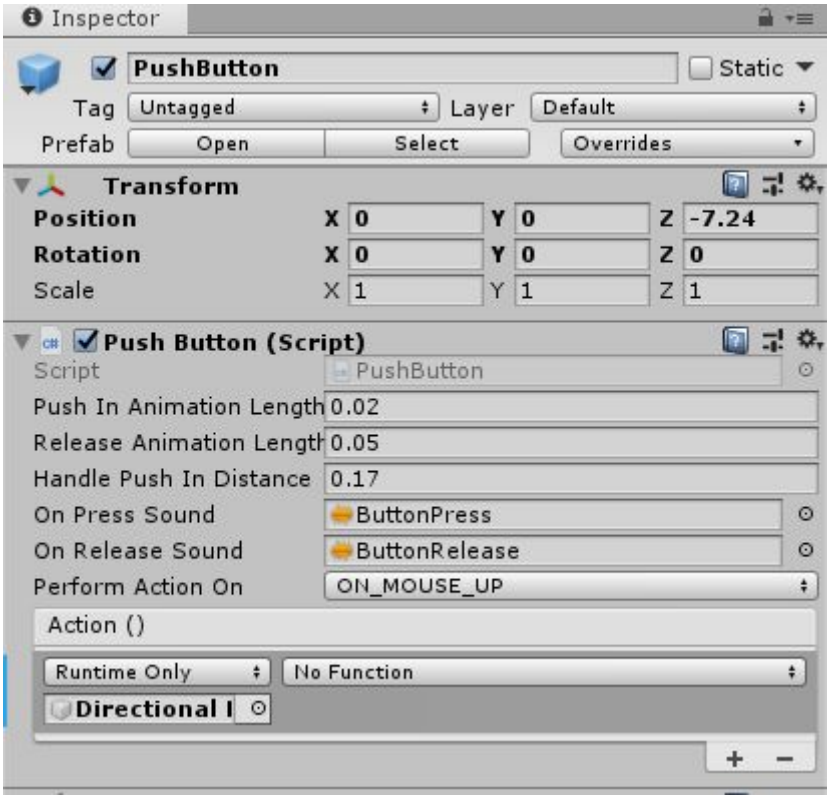
Turning off lights with a button

Add a PushButton (from the prefabs folder) to your scene and put it somewhere that the camera can see it.

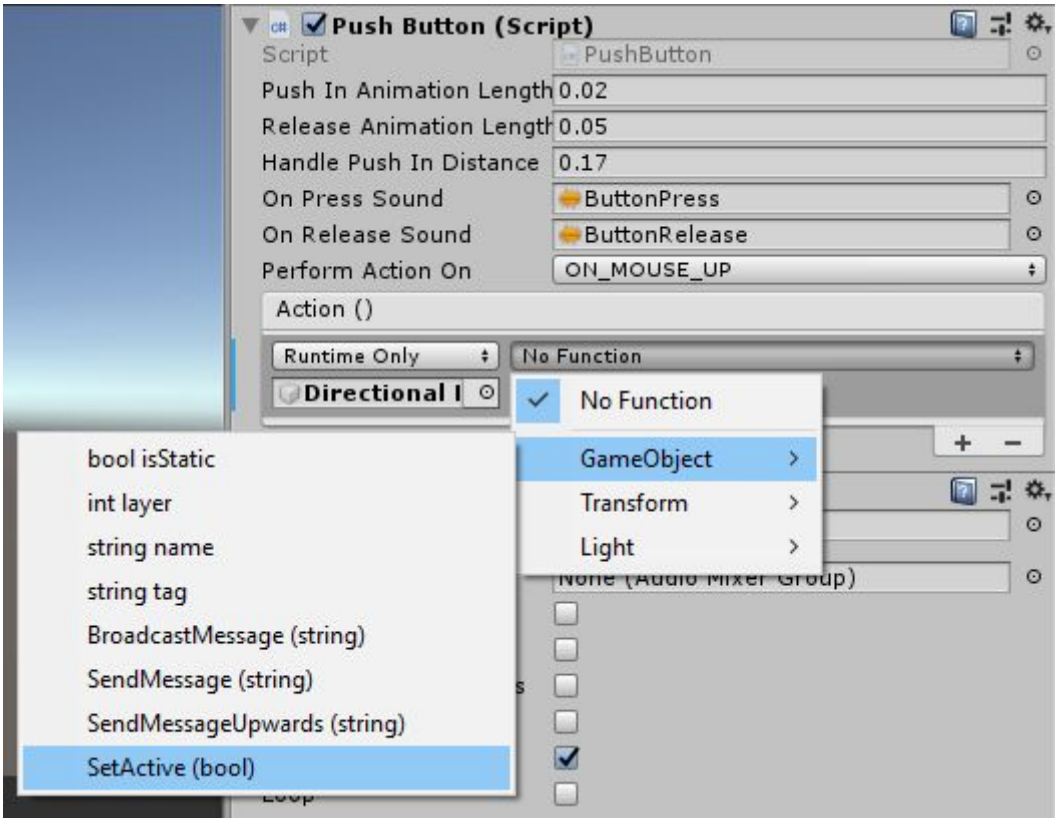


In this example, we'll be turning on and off the default directional light, but you can choose any light for your scene.

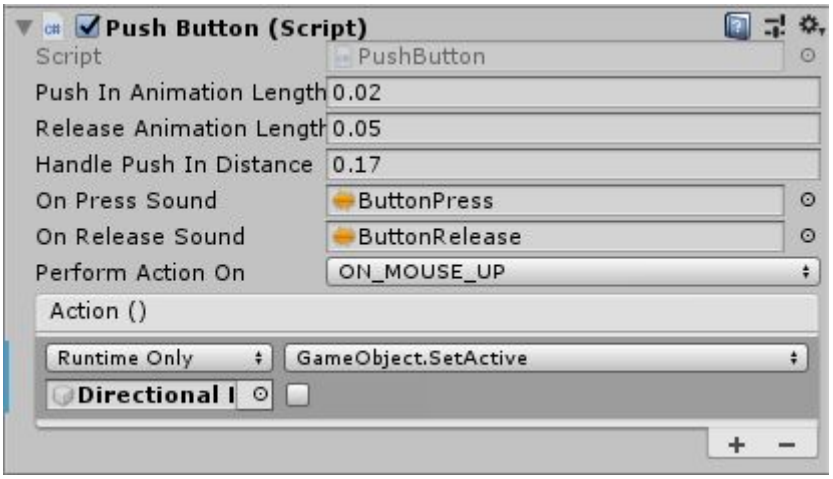
Select the PushButton and drag the Directional Light into the Action field of the Push Button component.



Click “No Function” and choose SetActive under Gameobject.

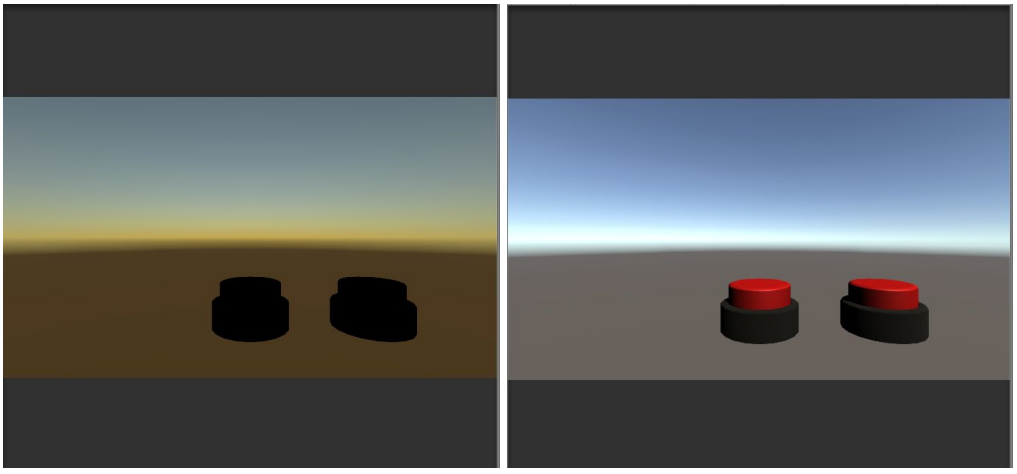


Make sure the checkbox is not checked so that the button makes the light turn off.



Run the scene and click on the button. The light should turn off.

Now, let's add a second button for turning the light back on. Duplicate the PushButton, move it to be next to the first one, and now check the box in the Action field. Run the scene, and you should be able to turn on and off the light with the buttons.



All settings

These tutorials covered the basics of the controls, but you can customize them further with these settings:

Knobs and sliders

Use curve	Toggles whether or not to use the taper curve. Turning this off saves some computation, so if the curve is linear you might as well turn it off.
Taper curve	Changes how the output value of the knob changes as the knob/slider position changes.

Handle grabbed scale	Makes the scale (size) of the model change when the knob/slider is grabbed with the mouse. Useful for giving feedback to the player as to what they clicked, but also gives a less realistic feel to the controls.

Knobs

Min angle	The angle relative to the forward vector that is how far back the knob can be turned.
Angle range	The knob's range of motion. How far forward from the min angle it can be turned.
Amount rotated	The current angle relative to the forward vector that the knob is set to.
Mouse drag sensitivity	Only for when in VERTICAL_DRAG mode. How much the mouse movement affects the knob rotation.
Interaction type	Changes how the knob is interacted with by the player. ROTATION_DRAG mode means the knob is turned by rotating the mouse around it, and VERTICAL_DRAG mode means moving the mouse up and down rotates it.

Sliders

Min Position	How far back on the Z axis relative to the object's pivot the slider handle can move.
Movement range	How far forward on the Z axis relative to the min position the slider handle can move.
Amount moved	The current position relative to the min position of the slider handle.

Buttons

Push in animation length	How long the button animates pushing in when pressed down.
Release animation length	How long the button animates returning to normal position when released.

Handle push in distance	How far on Y axis the handle part of the button moves when pushed down.
On press sound	The sound clip to play when the button is pressed (Only if an AudioListener is attached)
On release sound	The sound clip to play when the button is released (Only if an AudioListener is attached)

Extending functionality

The provided settings are only a starting point for customizing the controls. You can build on top of this asset package with your own models and code.

Changing the models

You can replace the model objects in the prefabs with your own models. The scripts require that a certain object in the prefab be called “handle”:

For knobs: handle is the object that turns.
For sliders: handle is the object that moves up and down.
For buttons: handle is the object that pushes in when clicked.

Note: make sure to update the collider size on the parent object when changing the models. Child objects of the prefab should not have colliders, only the parent.

Adding functionality in the code

To make knobs and sliders do things that are not included in the provided listeners (transform, mixer param and light), you can create your own listeners. First, create a new C# script. Add “using KnobsAsset;” to the top of your script. Then, make the script extend the “KnobListener” abstract class. Finally, implement the OnKnobValueChange method. The parameter of this method “knobPercentValue” is a number from 0 to 1 which is the position that the knob is turned to. Once you’ve finished making this script, attach it to a knob or slider.

Although the abstract class is called KnobListener, it works for both knobs and sliders.

Interacting with other devices than the mouse

The knobs, sliders and buttons are interactable by default with the mouse, but you can implement your own input code to make them interactable from another device. The knobs and sliders have public methods for changing their value, and buttons have methods for pressing and releasing them.