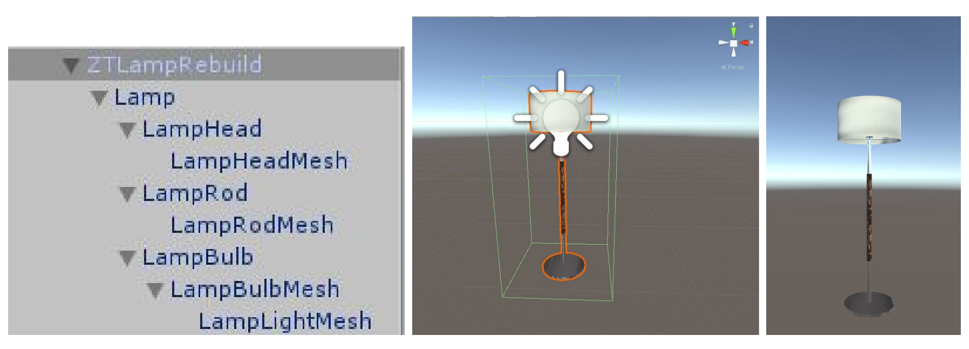
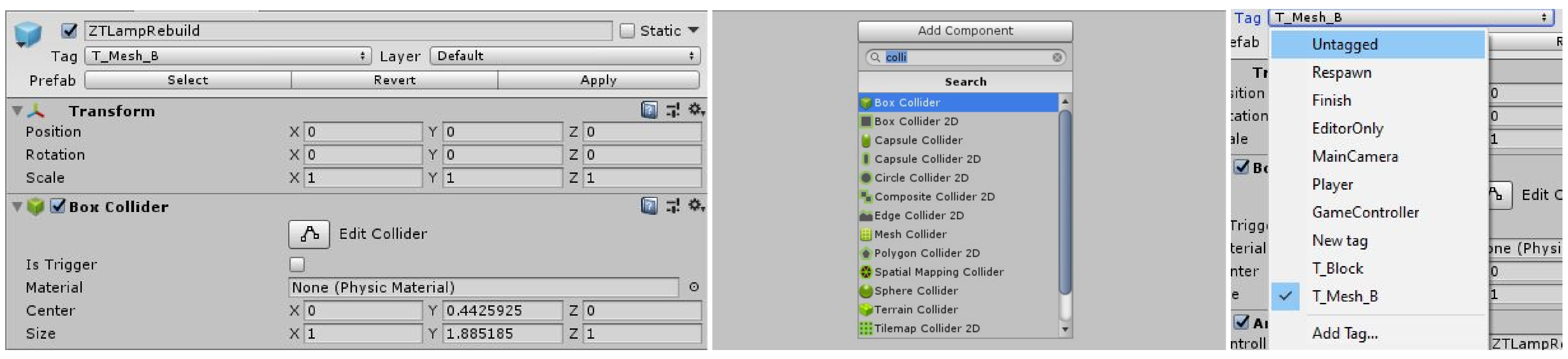
7 Days to Die Modding Unity, Modlet, XMLs & Localization Tutorial For Setting Up Animator For Switch Class With Lights

1. In order to set up an animator for switch class, you must first have to get the object together (meshes, textures, shaders, albedos etc.).ZTLampRebuild below was designed to be split up in three parts under Lamp: Lamp Head, Lamp Rod and Lamp Bulb. This ended up making a hierarchy, making them children under "Lamp." I then made ZTLampRebuild as an empty parent and dragged the entire set of children under it to make the hierarchy you see on the bottom left. This resulted in a final in-game image on the far right.

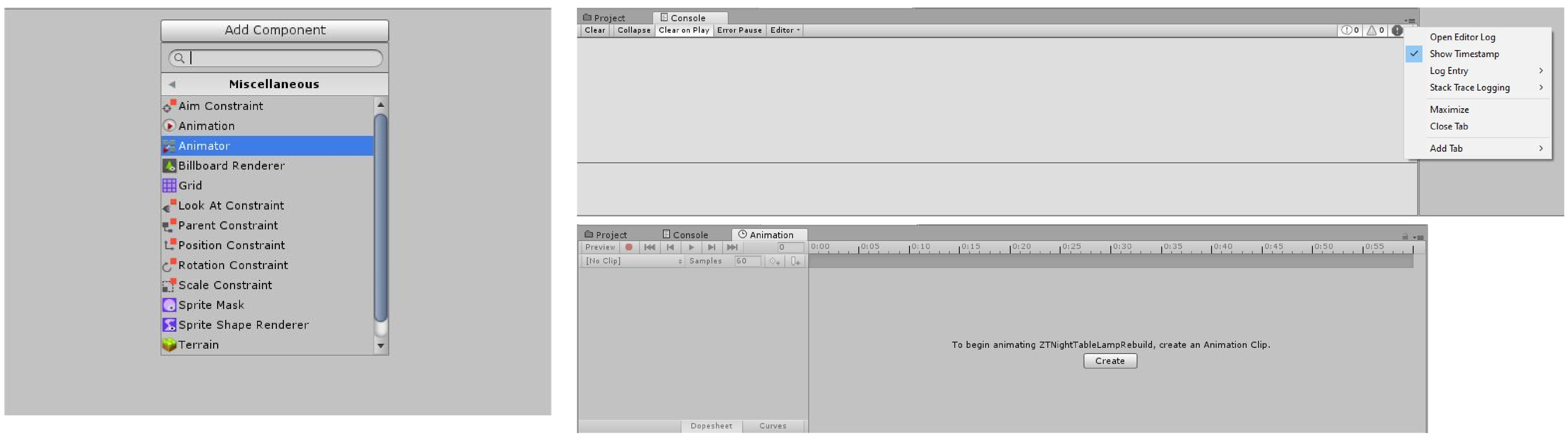


2. The next thing that's needed before moving on to the animator is clicking on the "ZTLampRebuild" and not only adding a tag, but also colliders. I've learned that we should be using box colliders. You need to "add component, search collider and add. Adjust the collider until it's as close to the object as possible without bleeding through. Depending on your object, you may need multiple colliders. Then you need to add a tag, which in most cases is T_Mesh_B, which stands for being able to interact with the object in-game (i.e. pressing e/y/triangle to turn off/on, close door / open door, talk to NPC, etc.).

   On a fresh game, you will need to manually add tags to the game. I am only aware of T_Block and T_Mesh_B. You have to have them spelled exactly like that, capitalization and all. However, other members of the Discord like Xyth and Guppy may be familiar with the tags, such as how Xyth explains on his YouTube videos like on this one - https://www.youtube.com/watch?v=0_MbTZJpAHM&list=PLrSIBJY8f7CFleBE2tji3NFO4tiedZmtt.

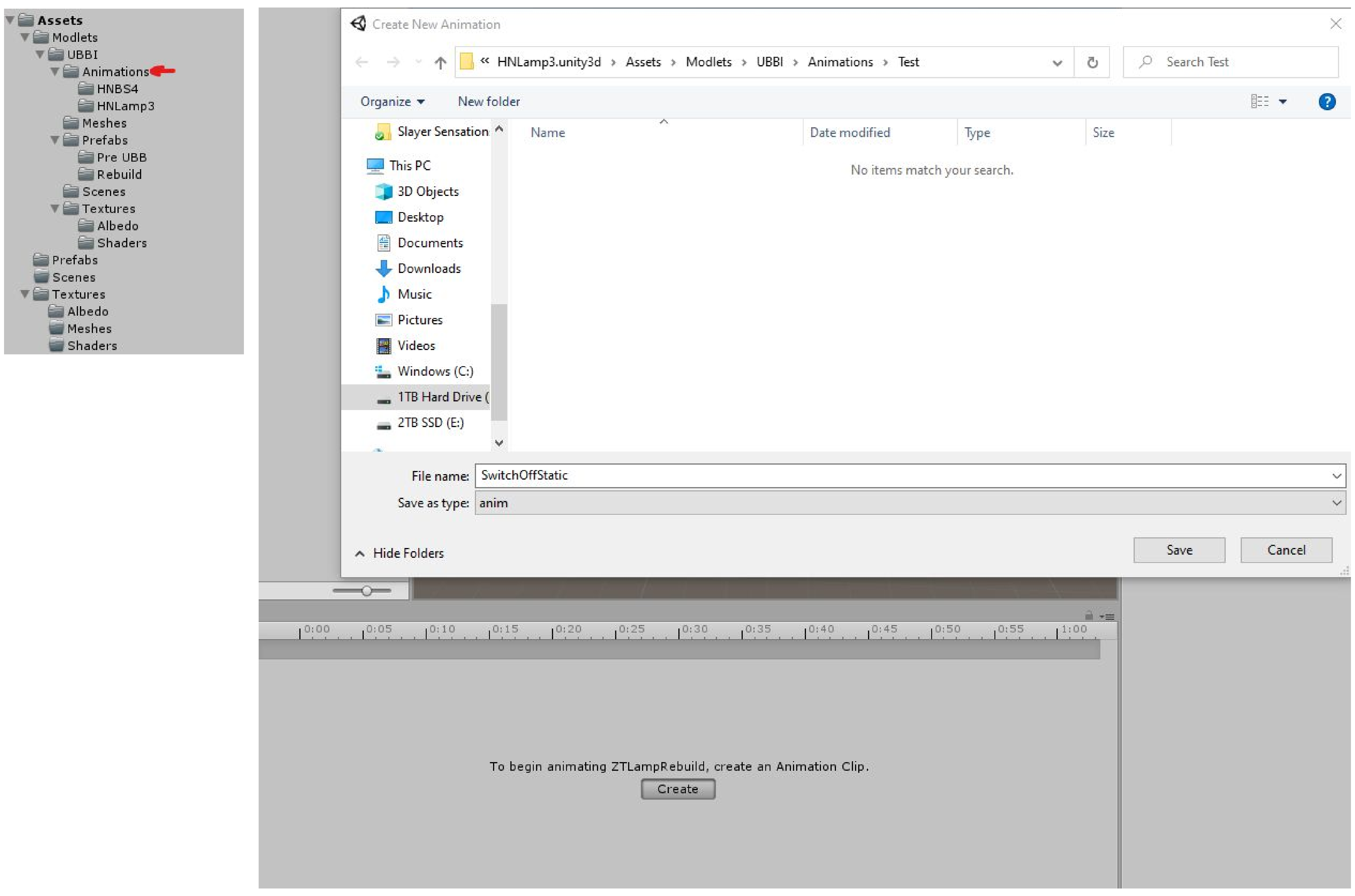


3. Now we must add an animator. First, make sure you click on the parent "ZTLampRebuild" and then on Inspector on the right, click add component, go to miscellaneous and then animator. Before you can keep going you then need to add a tab to the bottom section - i.e the animator tab. To do that, click on the tiny hamburger on the right and then add tab and click "animation." You should now see the new animation tab pop up below:

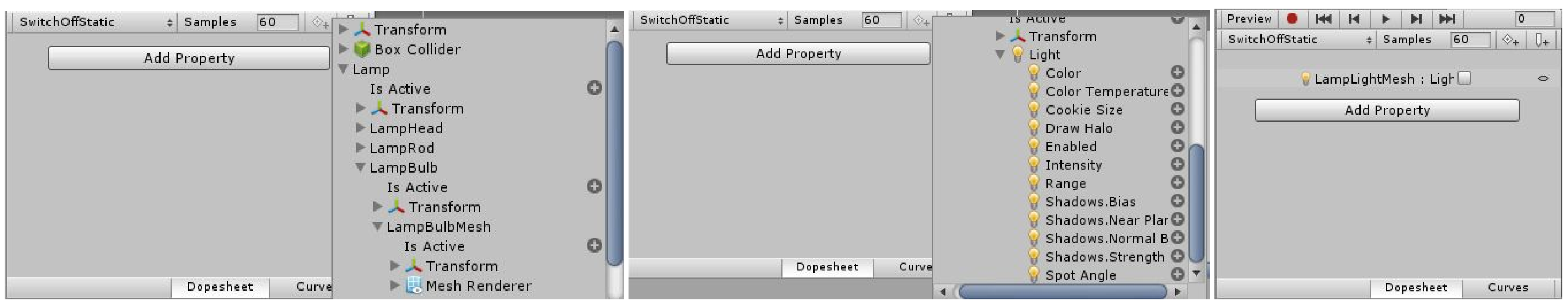


4. Before tapping "create" to create an animation clip, you must first create a folder to hold the animations you'll be creating for your light switch. If you see below, you'll notice I created a "Modlets" folder for my "Assets" folder. Under this folder I created "UBBI," which stands for Unnecessarily Beautiful But Immersive and then under that a subfolder for the object we're creating the animations for, "Lamp" or in this case HNLamp3.

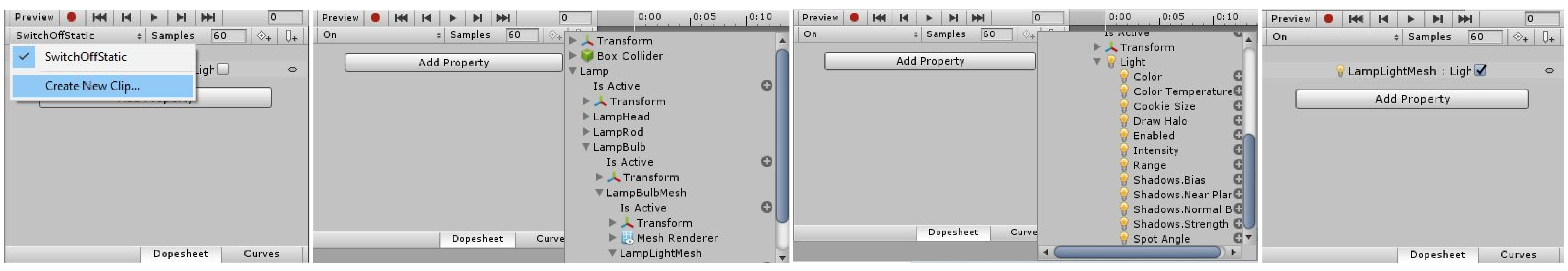
   Once you've done that, you tap create and name your first animation "SwitchOffStatic." Just make sure to place it in the new folder you created if it's now shown below. Then, we begin adding properties and the other three animations.



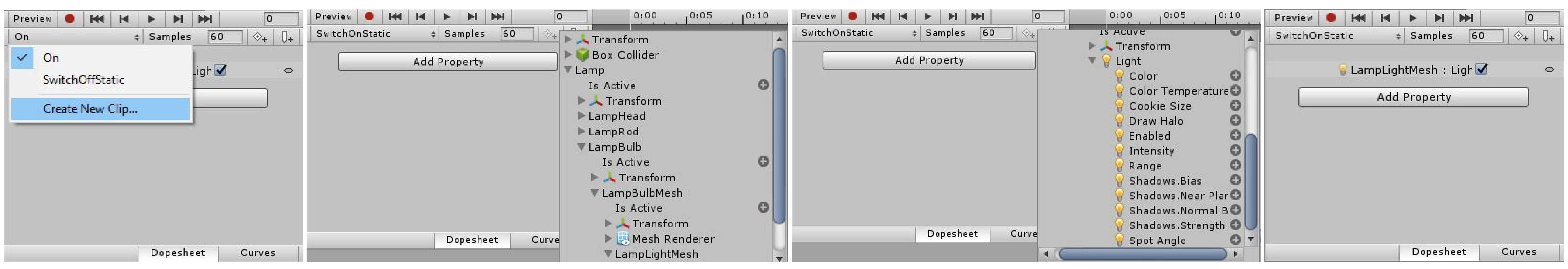
5. The next step is to add your property for 'SwitchOffStatic." In order to do that, you click "add property" and find where the light is actually located on your object. Once you find "Light" select it and then make sure to click on the plus sign next to "Enabled." Once added, make sure the checkbox is unchecked, which declares the light as "off."



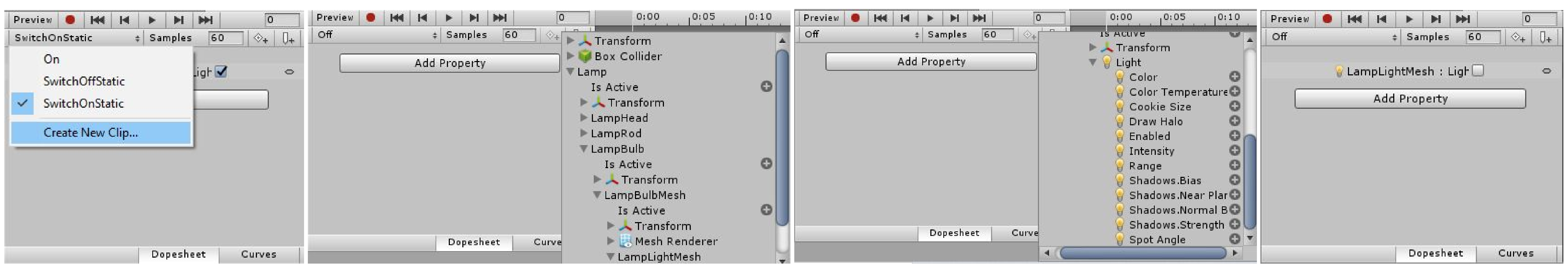
6. Next, we must create a new clip, which will be "On" by following similar steps as part 4. Once created, you'll be adding a property to "On." We'll be finding "Light" again and adding "Enabled," but this time, we will be enabling it by checking the checkbox.



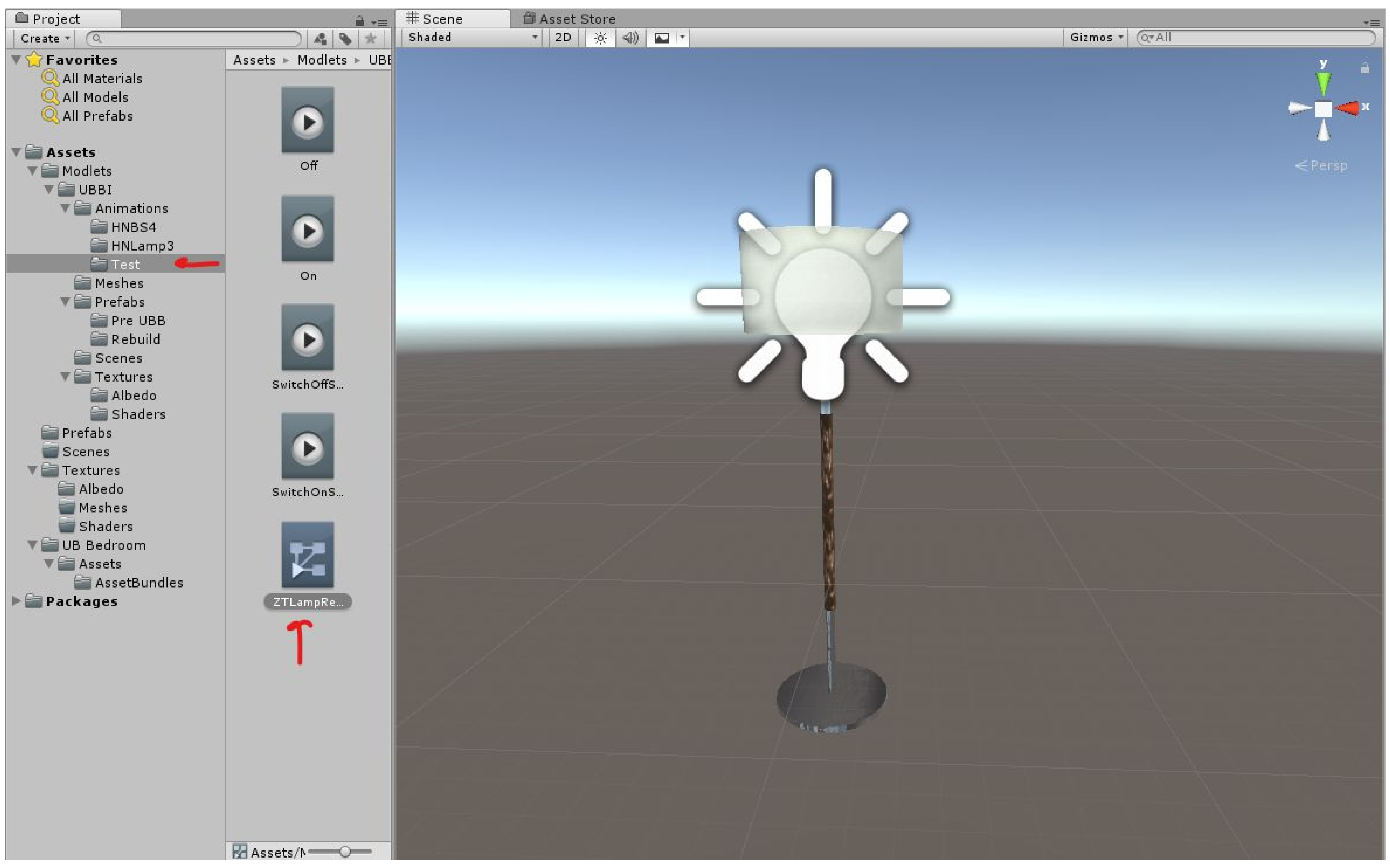
7. Next, we must create a new clip, which will be "SwitchOnStatic" by following similar steps as part 4. Once created, you'll be adding a property to "SwitchOnStatic." We'll be finding "Light" again and adding "Enabled," but this time, we will be enabling it by checking the checkbox again like part 6.



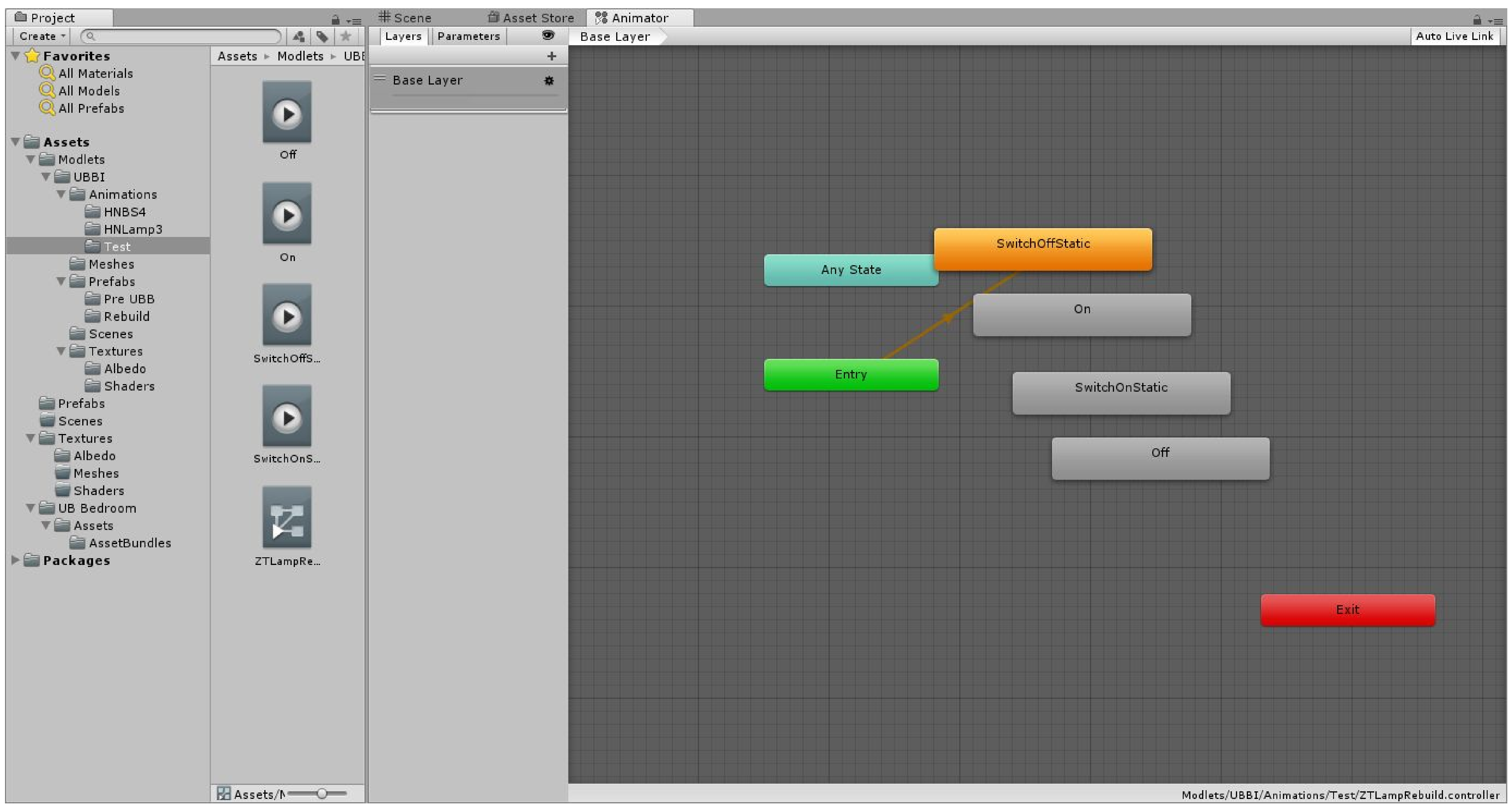
8. Then, we must create our last clip, which will be "Off" by following similar steps as part 4. Once created, you'll be adding a property to "Off." We'll be finding "Light" again and adding "Enabled," but this time, we will be making sure it is not enabled, by making sure the checkbox is unchecked like part 5.



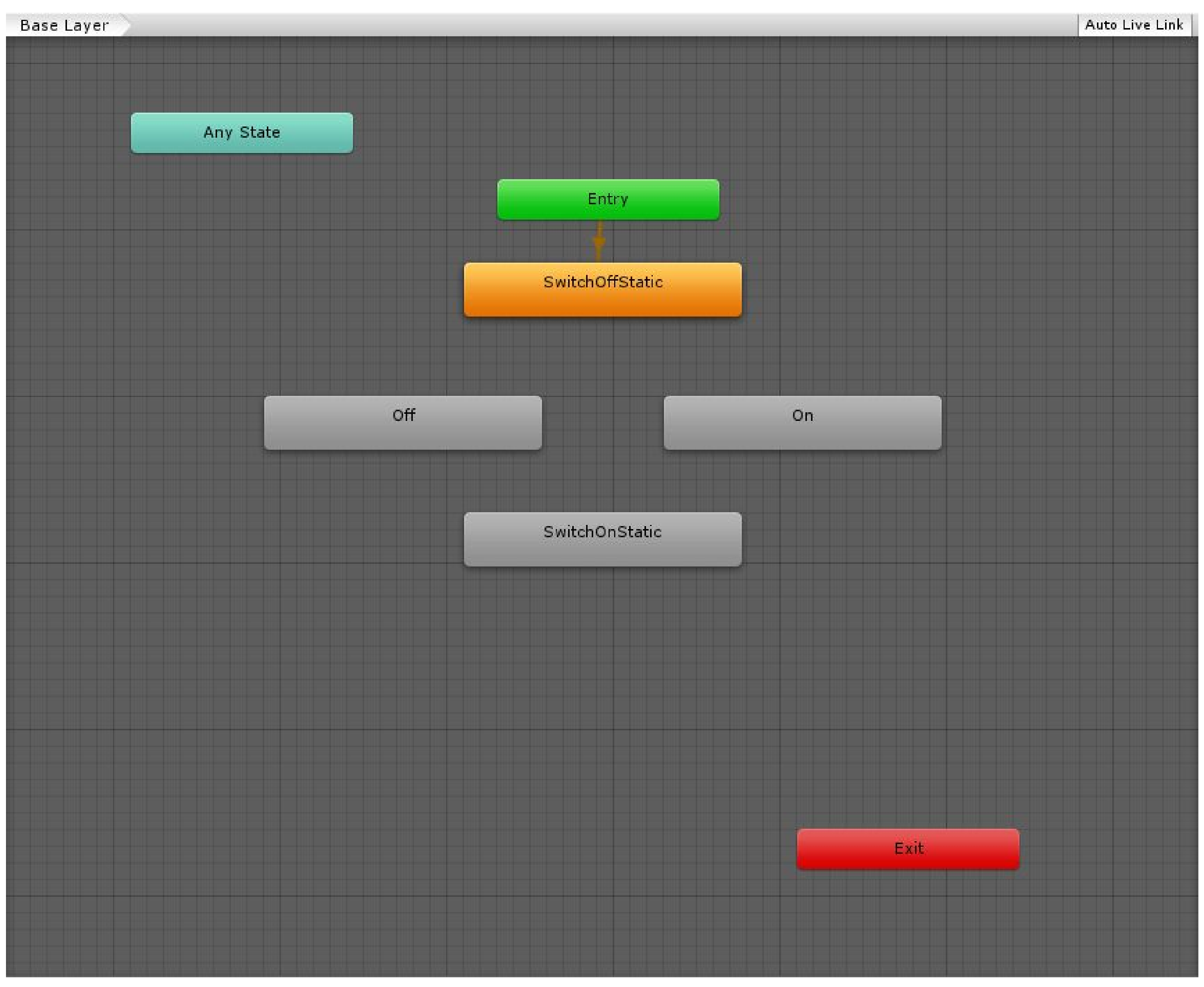
9. The next thing we need to do is go into the animator and really get this to all work. In order to do that, you must first add the animator tab by double clicking the actual animator that generated in your folder, which will look like this:

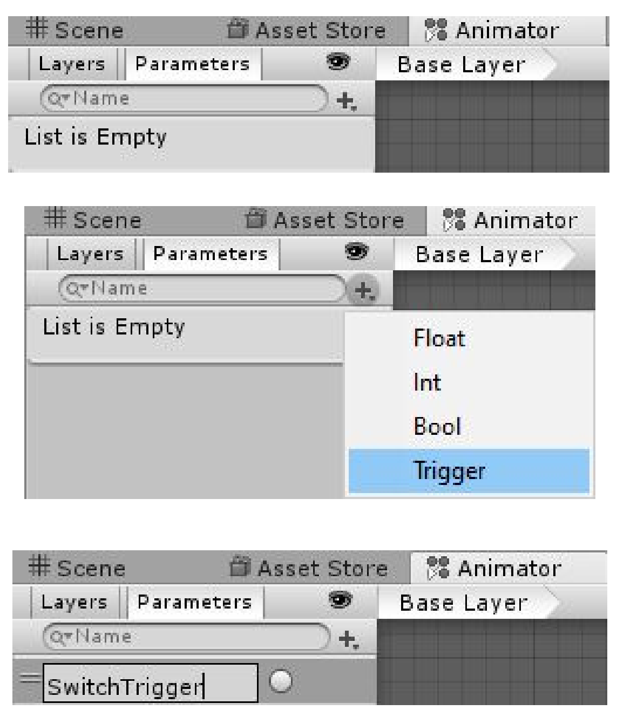


10. Once the animator tab is added, it will look a little like something below. You are required to have the turquoise/blue “Any State” block and the “Exit” block. By default, Entry will already have made a transition to “SwitchOffStatic,” since it was the first animation we created. Before we do anything else, let’s reorganize these a little bit.

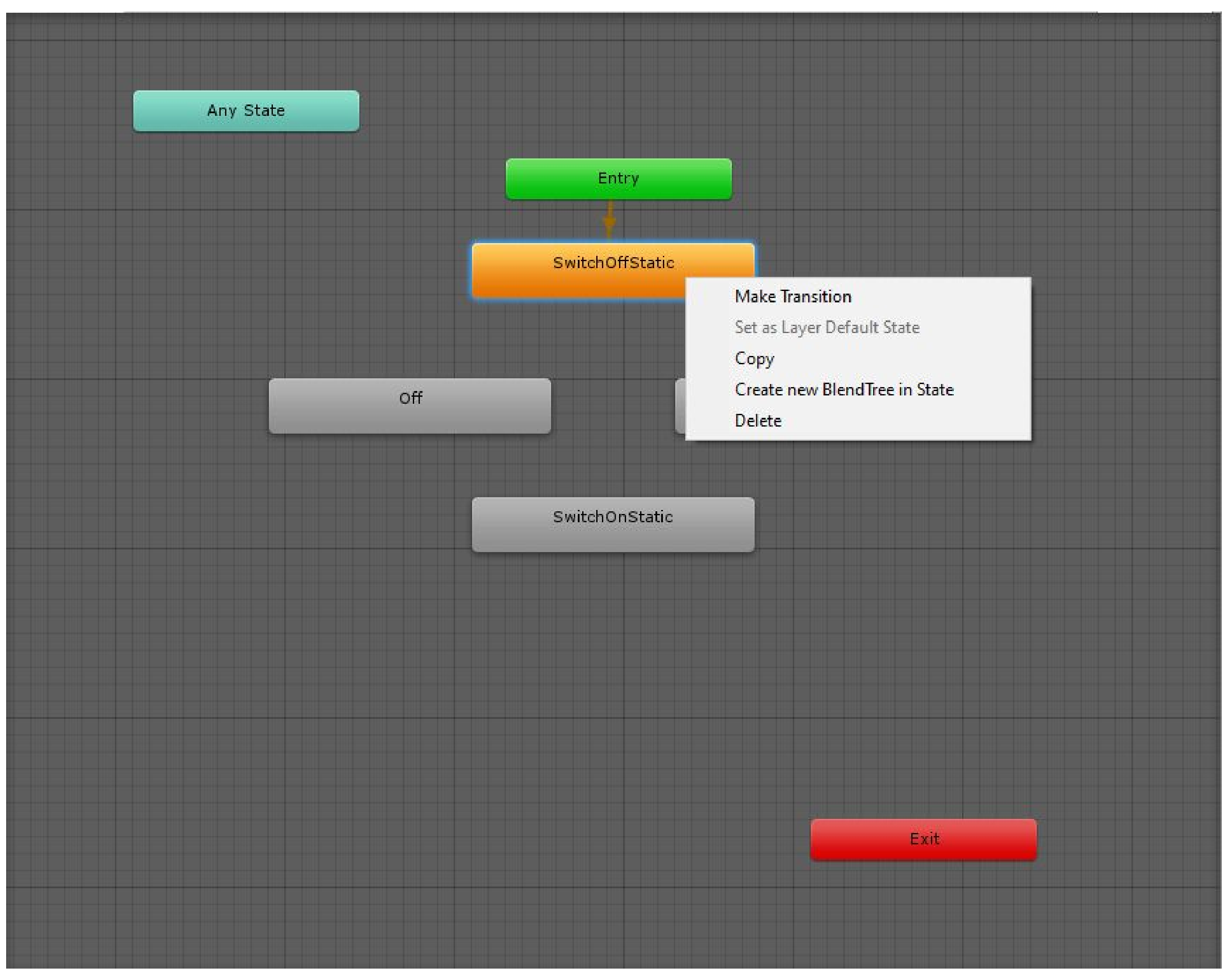


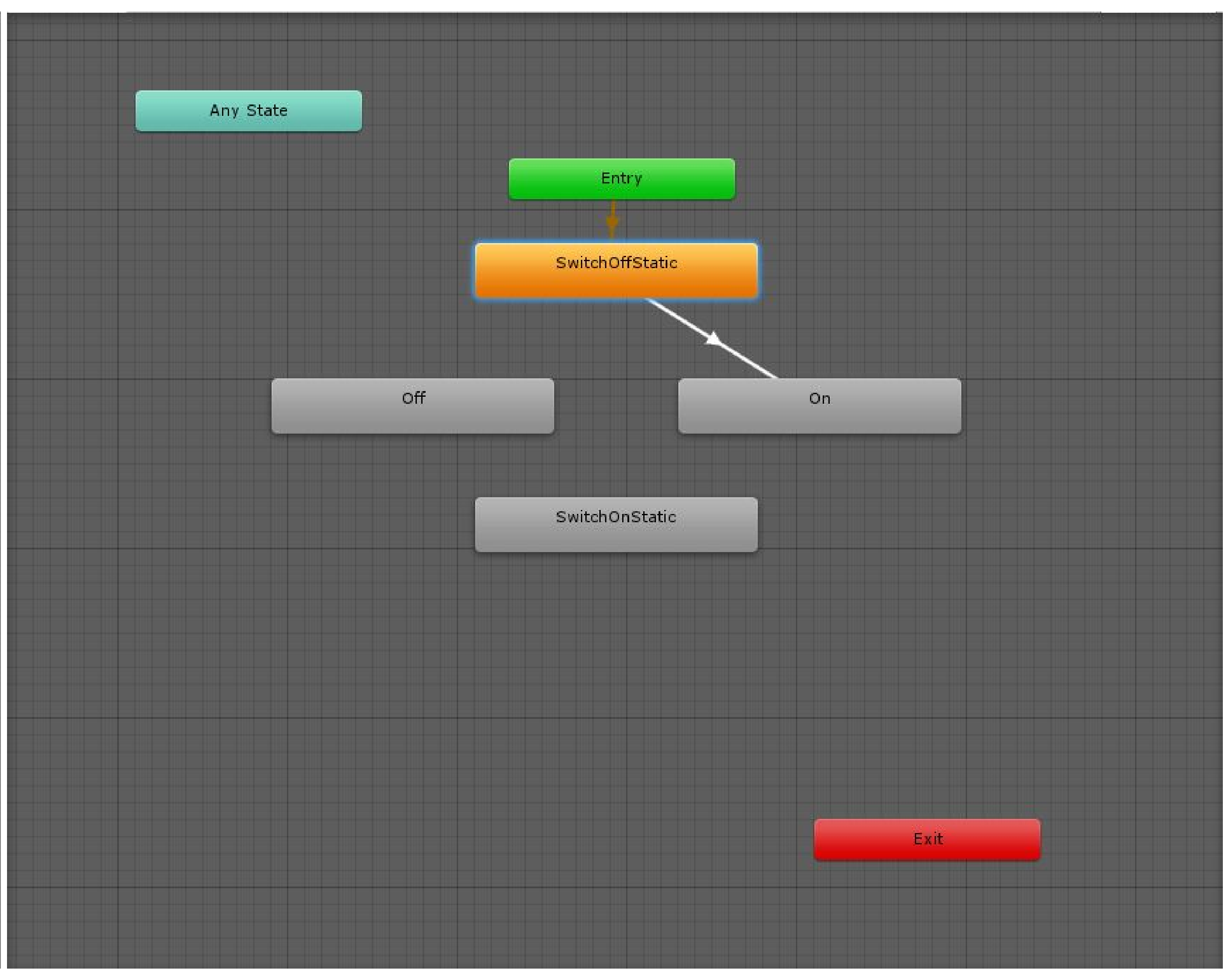
11. Once the animator looks similar to how I’ve organized it below, we can begin adding transitions. Before adding transitions, we must first create a “trigger” parameter titled “SwitchTrigger” (with exact spelling and capitalization).

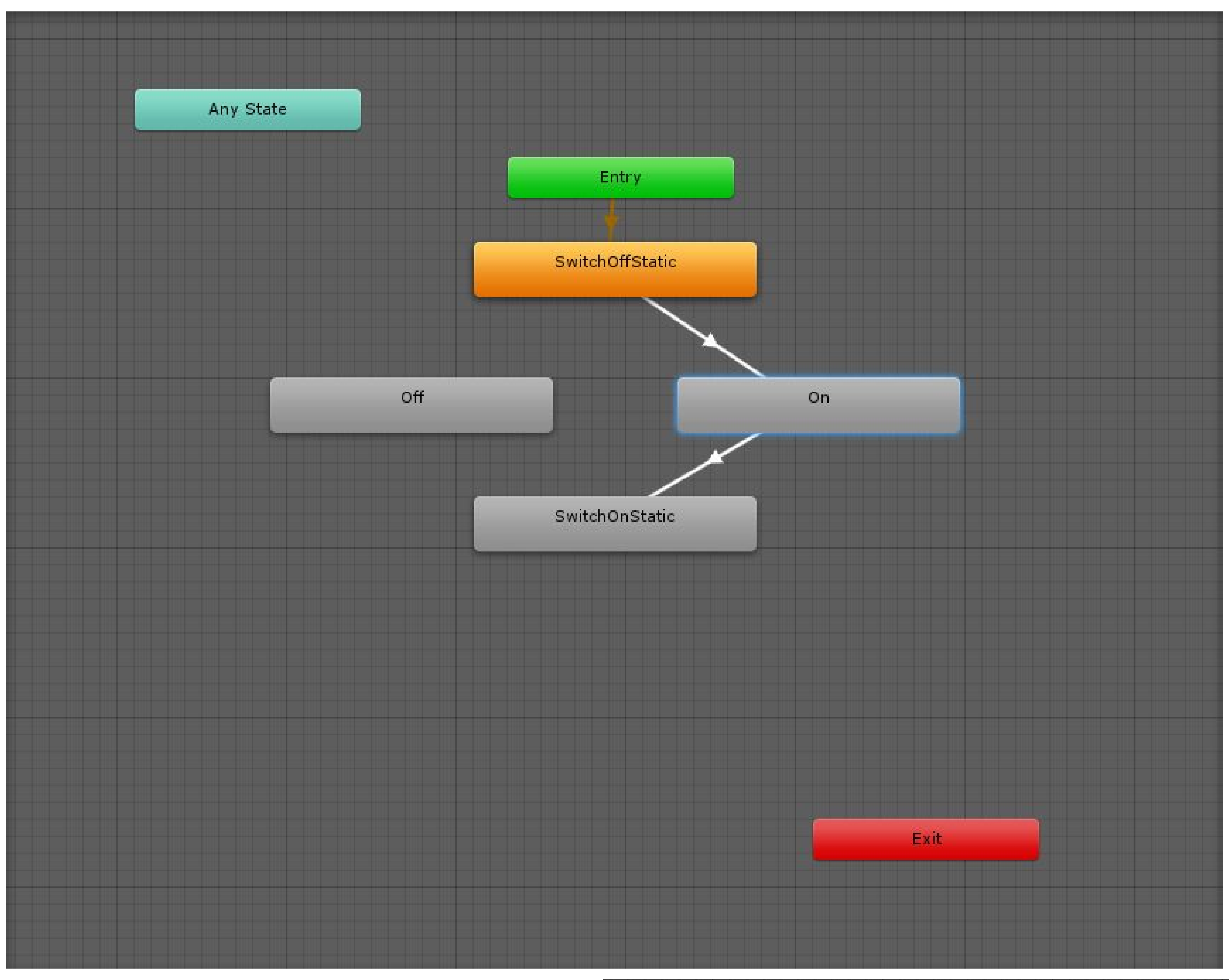


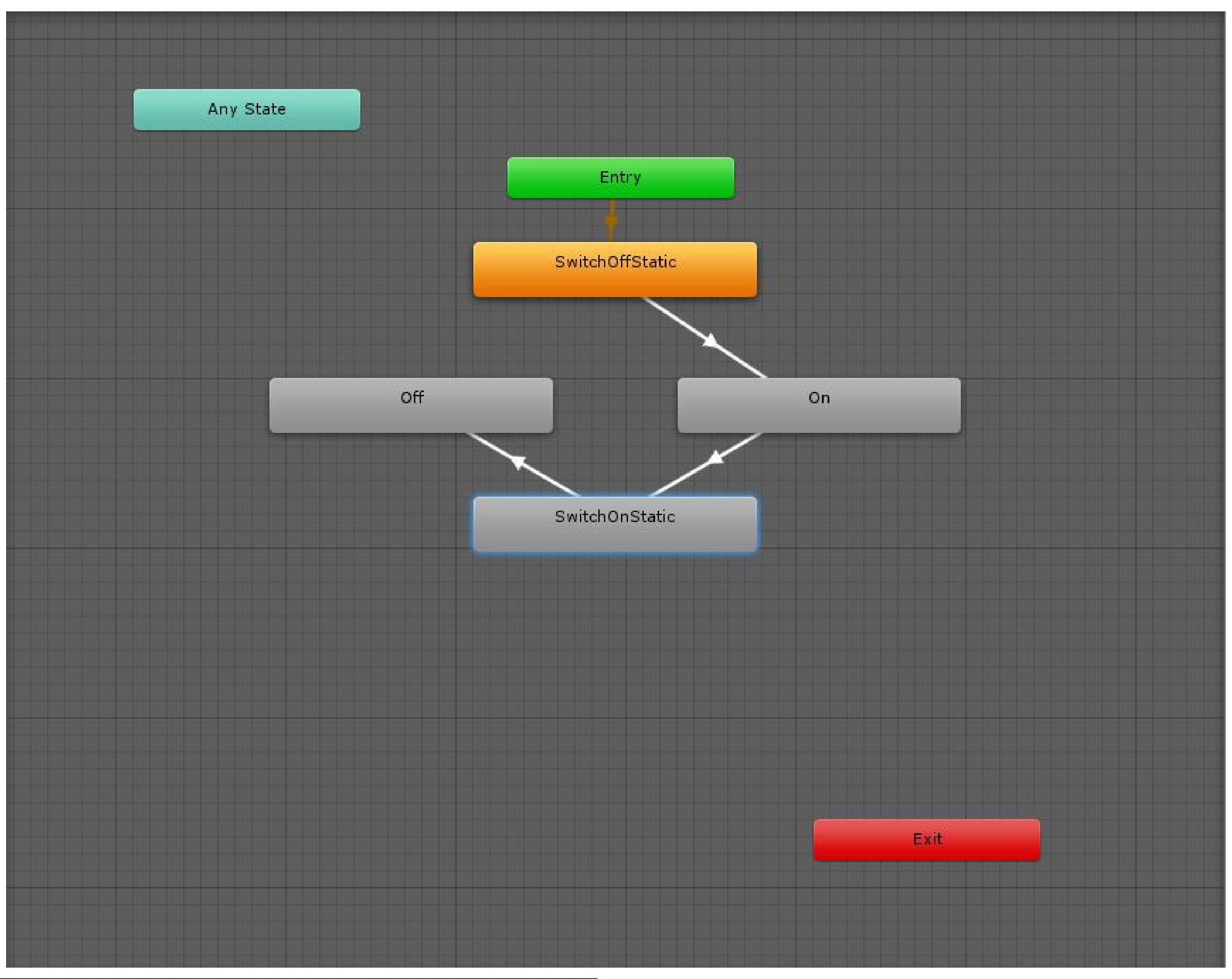


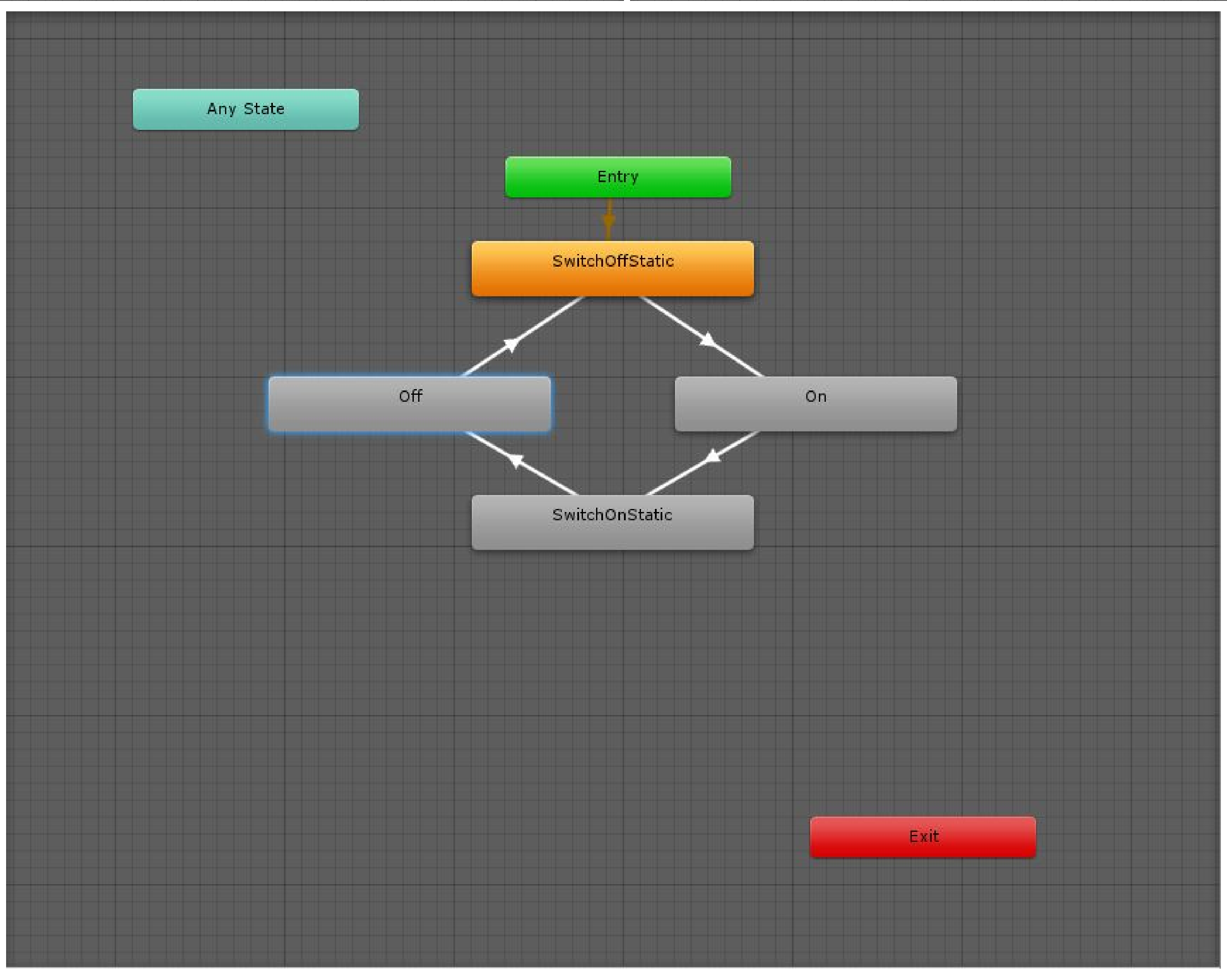
12. Now we will begin adding transitions. All you have to do is click “Make Transition” and carry it from “SwitchOffStatic” to “On,” and follow the same steps for “On” to “SwitchOnStatic,” “SwitchOnStatic” to “Off” and “Off” to “SwitchOffStatic.”



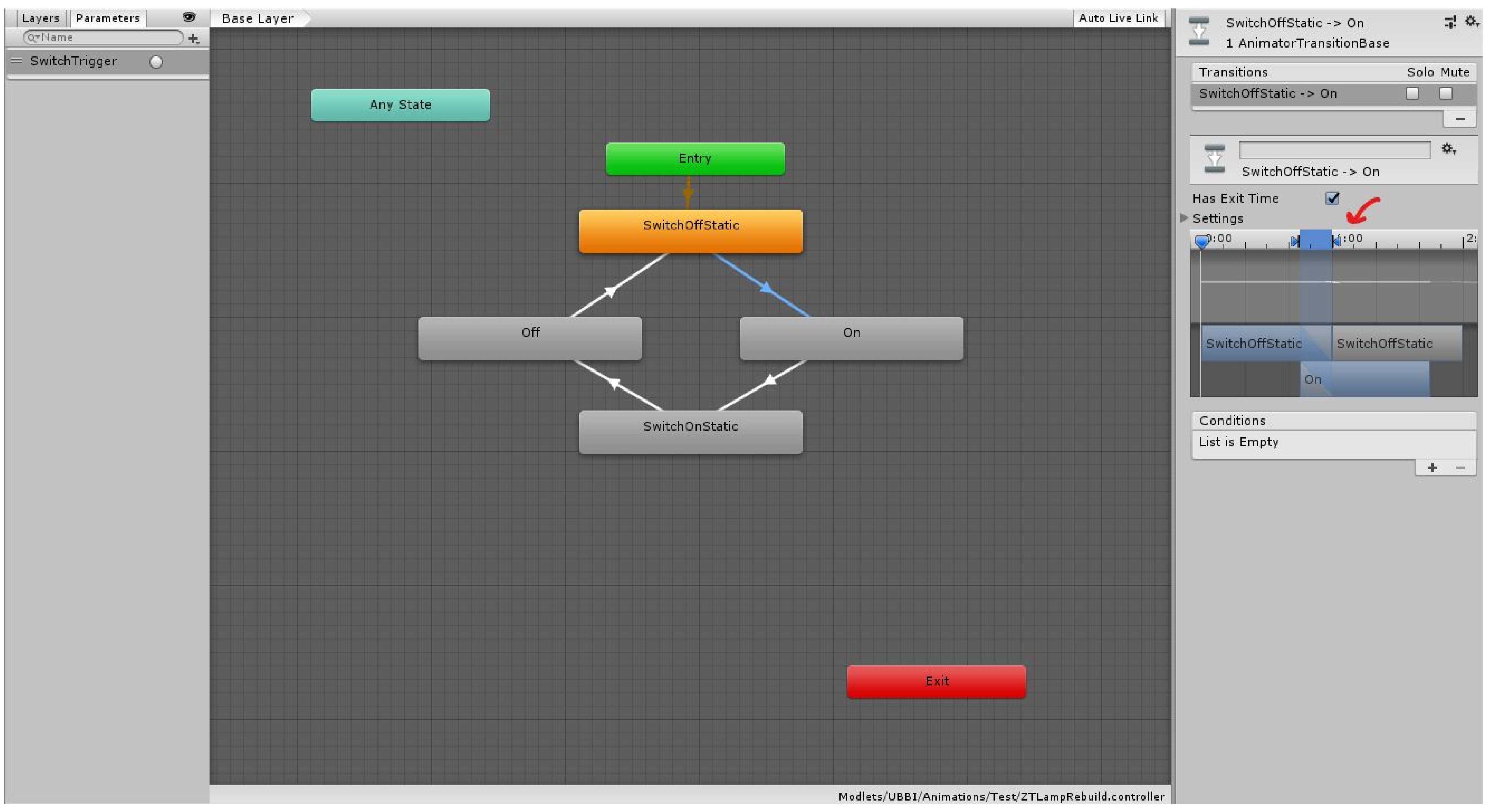




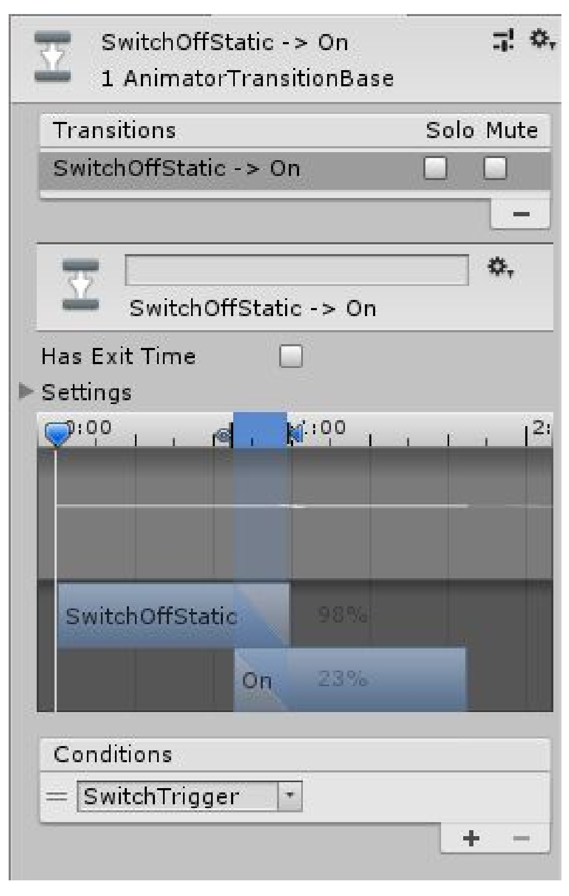




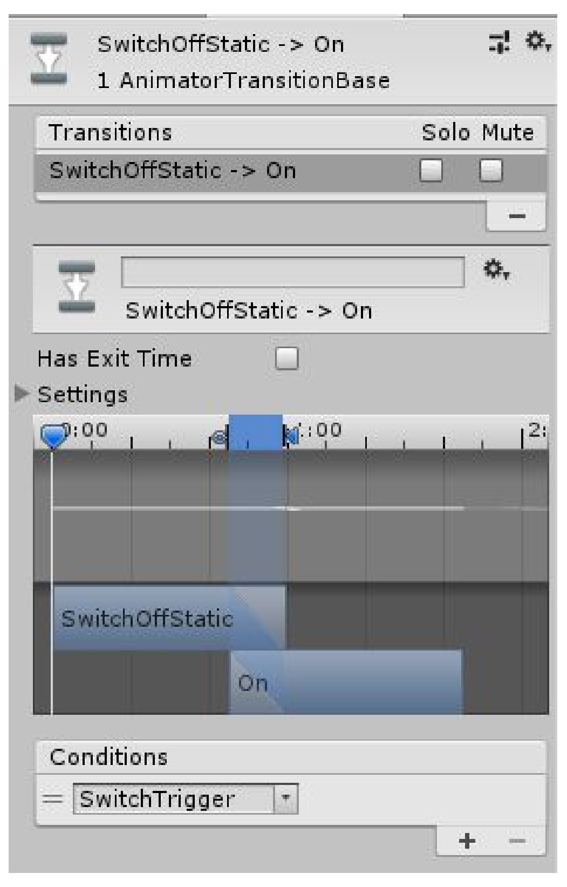
13. Once you have every transition created, you must now click on the arrows in-between the blocks, starting with the arrow in-between “SwitchOffStatic” and “On.” It should look a little like something below. Uncheck “Has Exit Time,” press the plus sign to add the SwitchTrigger condition and drag the right arrow to the left to 98% to make “SwitchOffStatic” on the upper right disappear. See the end result on the right side. The third version will be the way it should look.



SwitchOffStatic -> On
1 AnimatorTransitionBase
Transitions
SwitchOffStatic -> On
SwitchOffStatic -> On
Has Exit Time ☑
Conditions
List is Empty

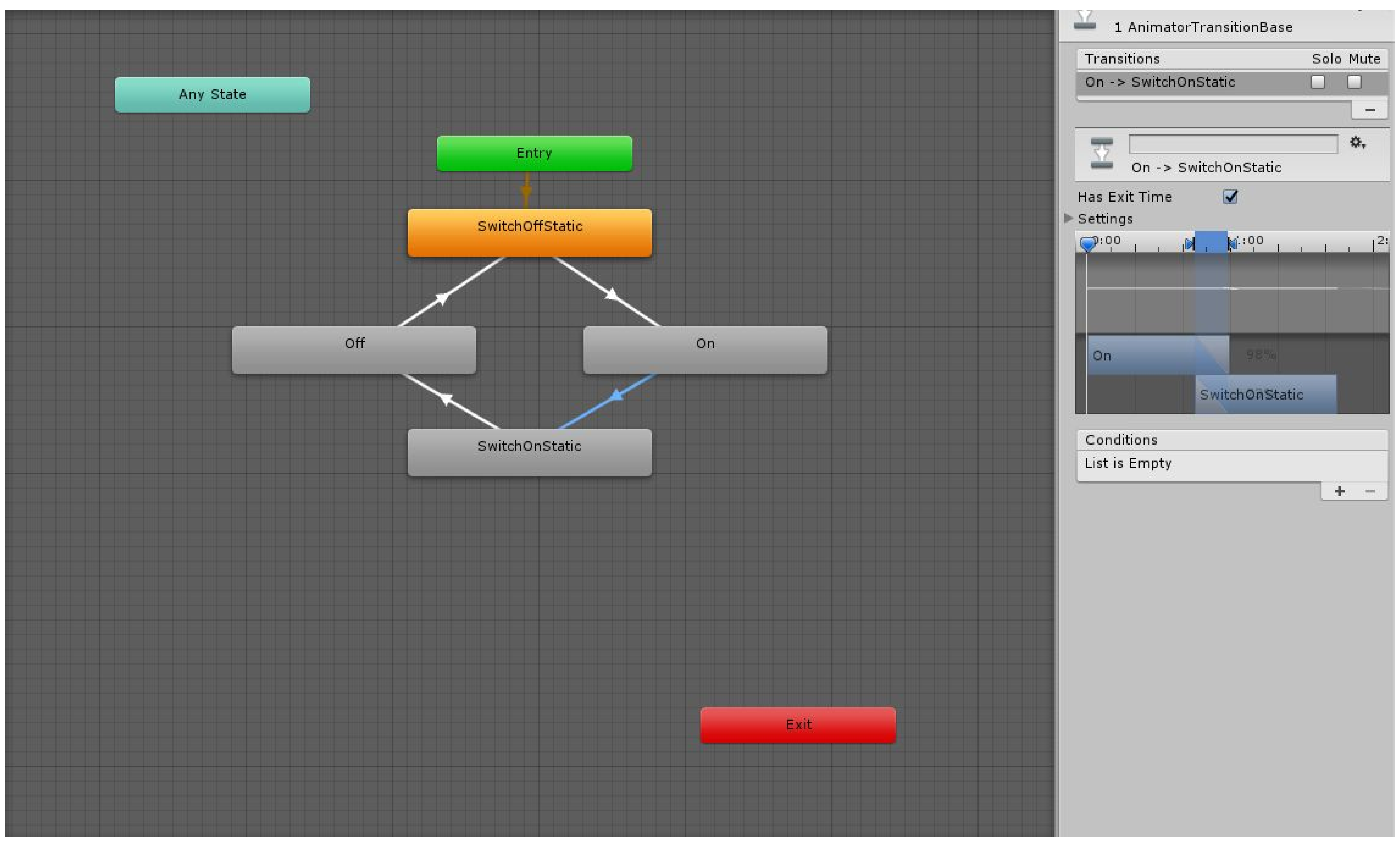


SwitchOffStatic -> On
1 AnimatorTransitionBase
Transitions
SwitchOffStatic -> On
SwitchOffStatic -> On
Has Exit Time ☐
Conditions
SwitchTrigger

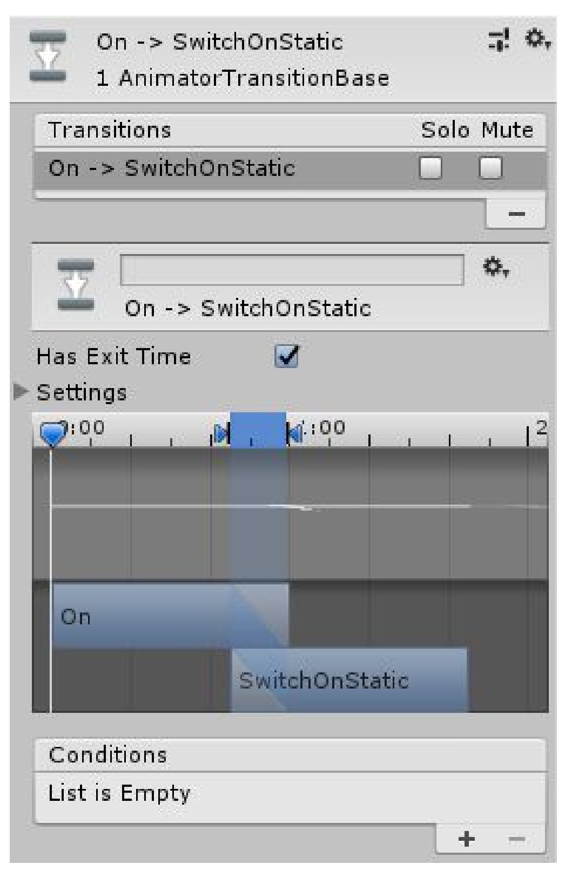


SwitchOffStatic -> On
1 AnimatorTransitionBase
Transitions
SwitchOffStatic -> On
SwitchOffStatic -> On
Has Exit Time ☐
Conditions
SwitchTrigger

14. Next, we will be clicking the arrow in-between “On” and “SwitchOnStatic.” It should look a little like something below. On this one, we keep “Has Exit Time” checked because it is exiting between “On” and “SwitchOnStatic” states. This block does not get a condition as we want “On” to automatically go to “SwitchOnStatic” to make the light stay on when turning it on in-game.

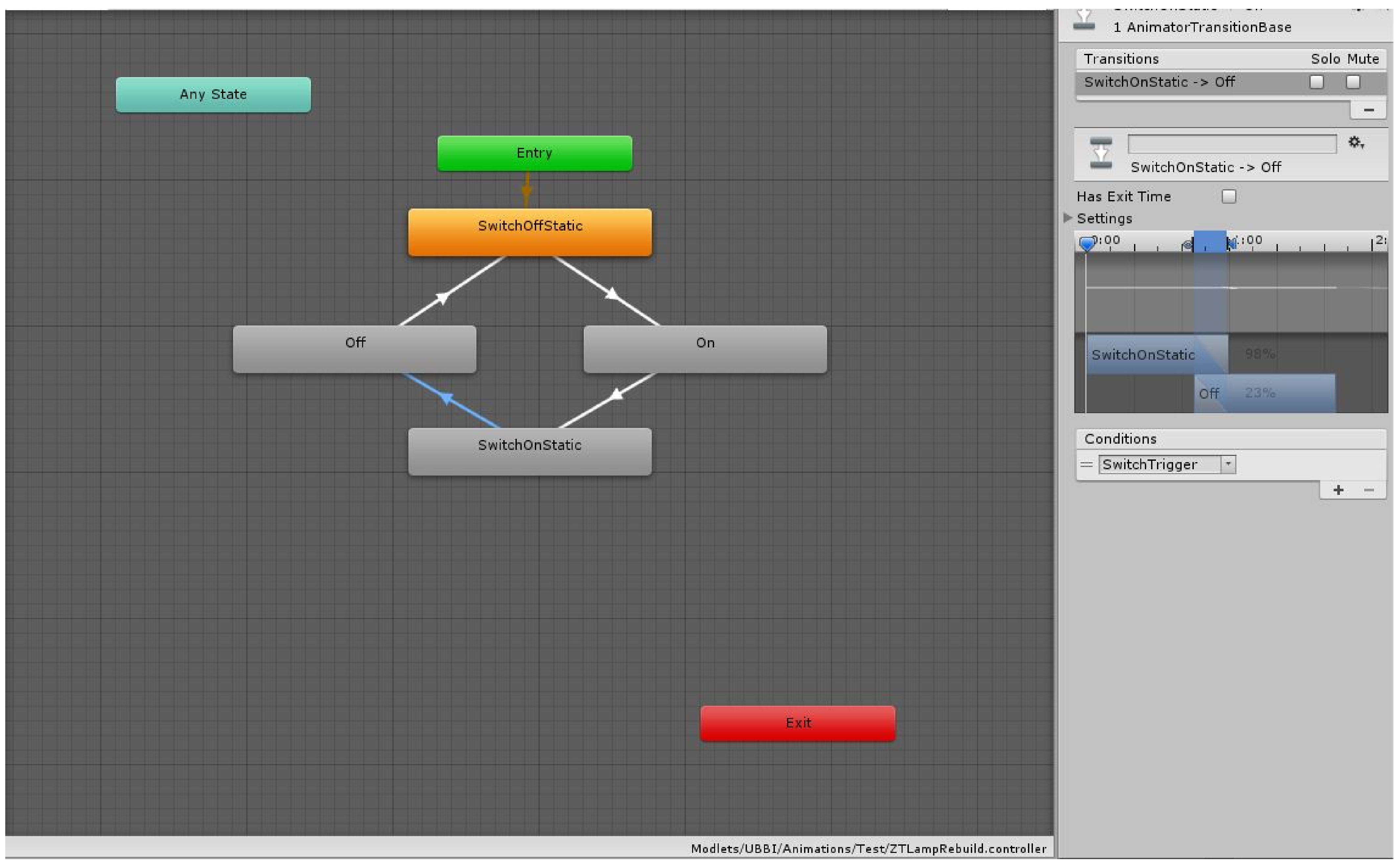


1 AnimatorTransitionBase
Transitions Solo Mute
On -> SwitchOnStatic
On -> SwitchOnStatic
Has Exit Time ✓
Settings
On 98%
SwitchOnStatic
Conditions
List is Empty

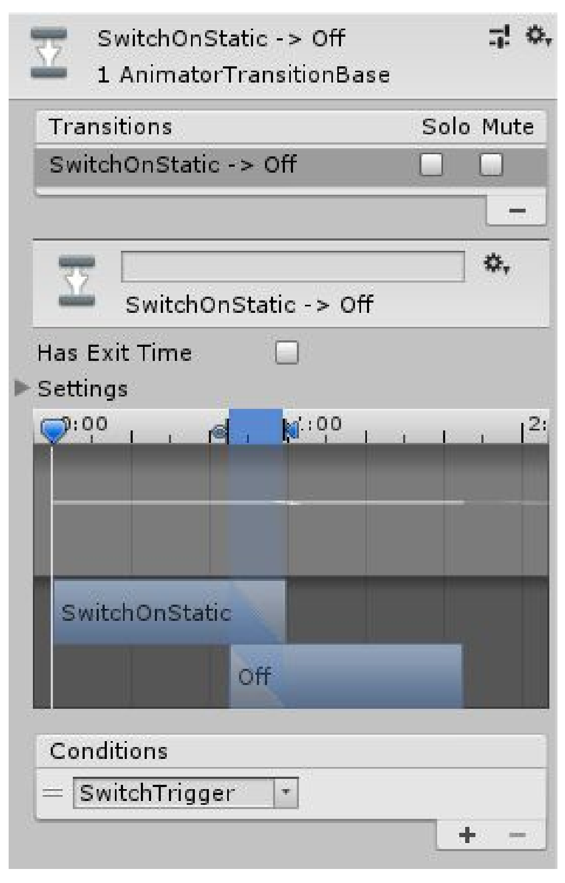


On -> SwitchOnStatic
1 AnimatorTransitionBase
Transitions Solo Mute
On -> SwitchOnStatic
On -> SwitchOnStatic
Has Exit Time ✓
Settings
On
SwitchOnStatic
Conditions
List is Empty

15. Next, we will be clicking the arrow in-between “SwitchOnStatic” and “Off.” It should look a little like something below. Just like part 13, we will be unchecking “Has Exit Time,” pressing the plus sign to add the SwitchTrigger condition and dragging the right arrow to the left to 98% to make “SwitchOffStatic” on the upper right disappear. See the end result on the right side.

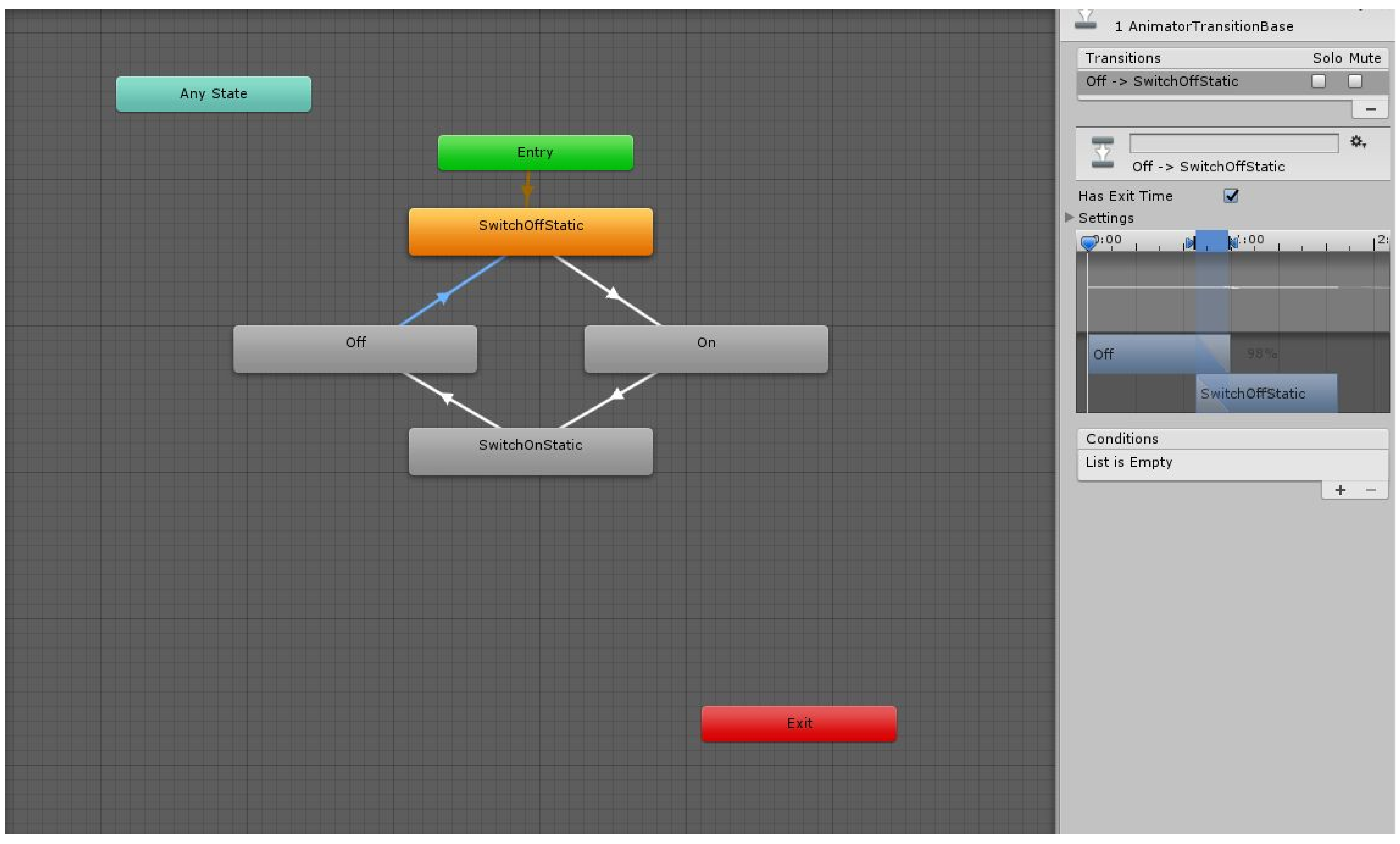


1 AnimatorTransitionBase
Transitions Solo Mute
SwitchOnStatic -> Off
SwitchOnStatic -> Off
Has Exit Time
Settings
SwitchOnStatic 98%
Off 23%
Conditions
SwitchTrigger

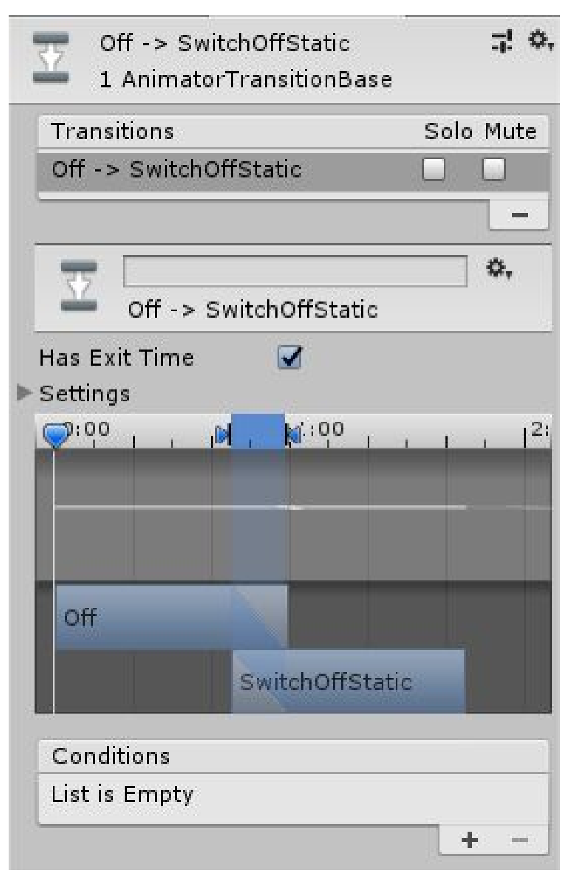


SwitchOnStatic -> Off
1 AnimatorTransitionBase
Transitions Solo Mute
SwitchOnStatic -> Off
SwitchOnStatic -> Off
Has Exit Time
Settings
SwitchOnStatic
Off
Conditions
SwitchTrigger

16. Finally, we will click the arrow in-between “Off” and “SwitchOffStatic.” It should look a little like something below. Just like part 14, we keep “Has Exit Time” checked because it is exiting between “Off” and “SwitchOffStatic” states. This block does not get a condition as we want “Off” to automatically go to “SwitchOffStatic” to make the light stay off when turning it off in-game.



1 AnimatorTransitionBase
Transitions Solo Mute
Off -> SwitchOffStatic
Off -> SwitchOffStatic
Has Exit Time ✓
Settings
Off 98%
SwitchOffStatic
Conditions
List is Empty

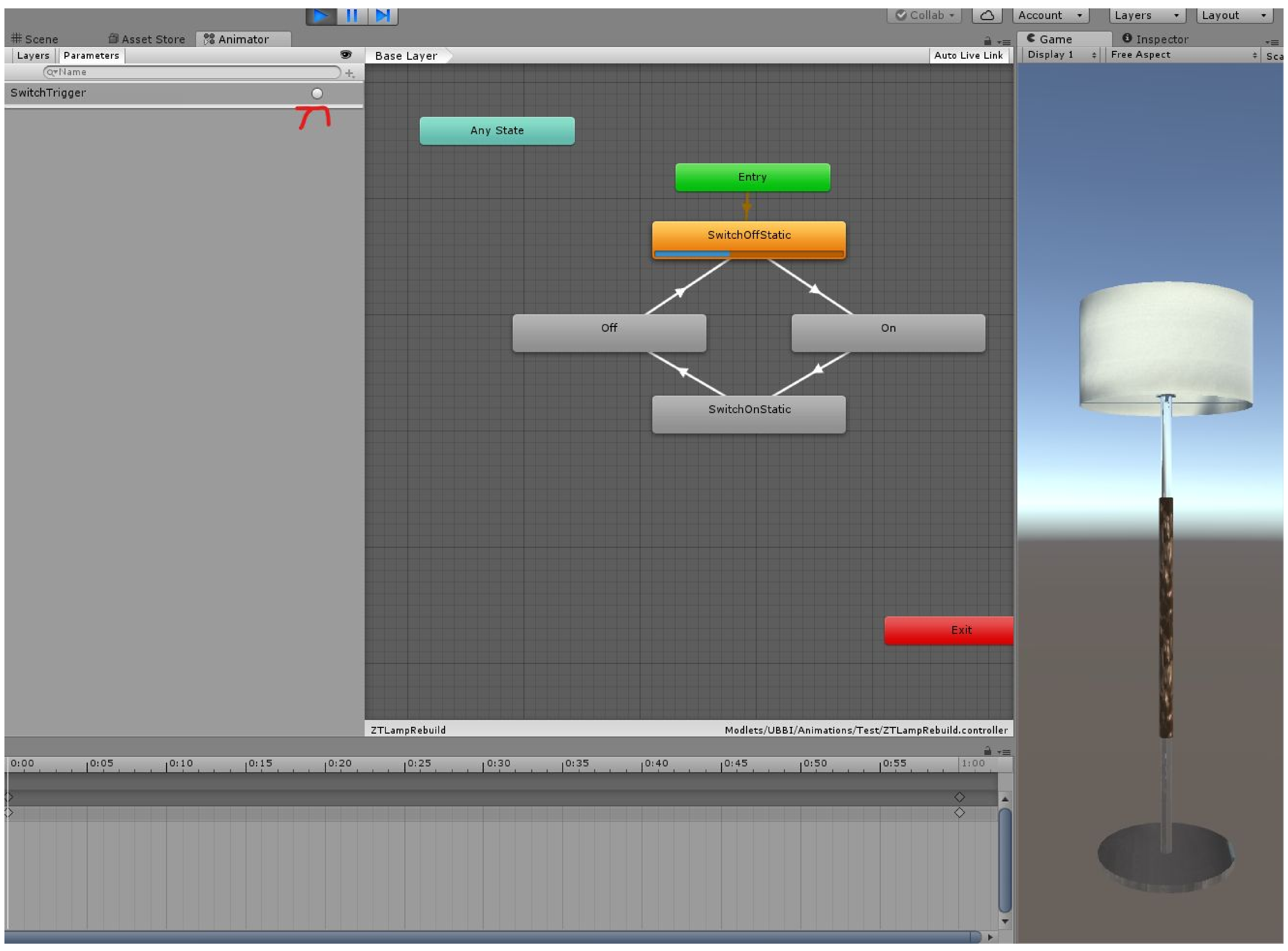


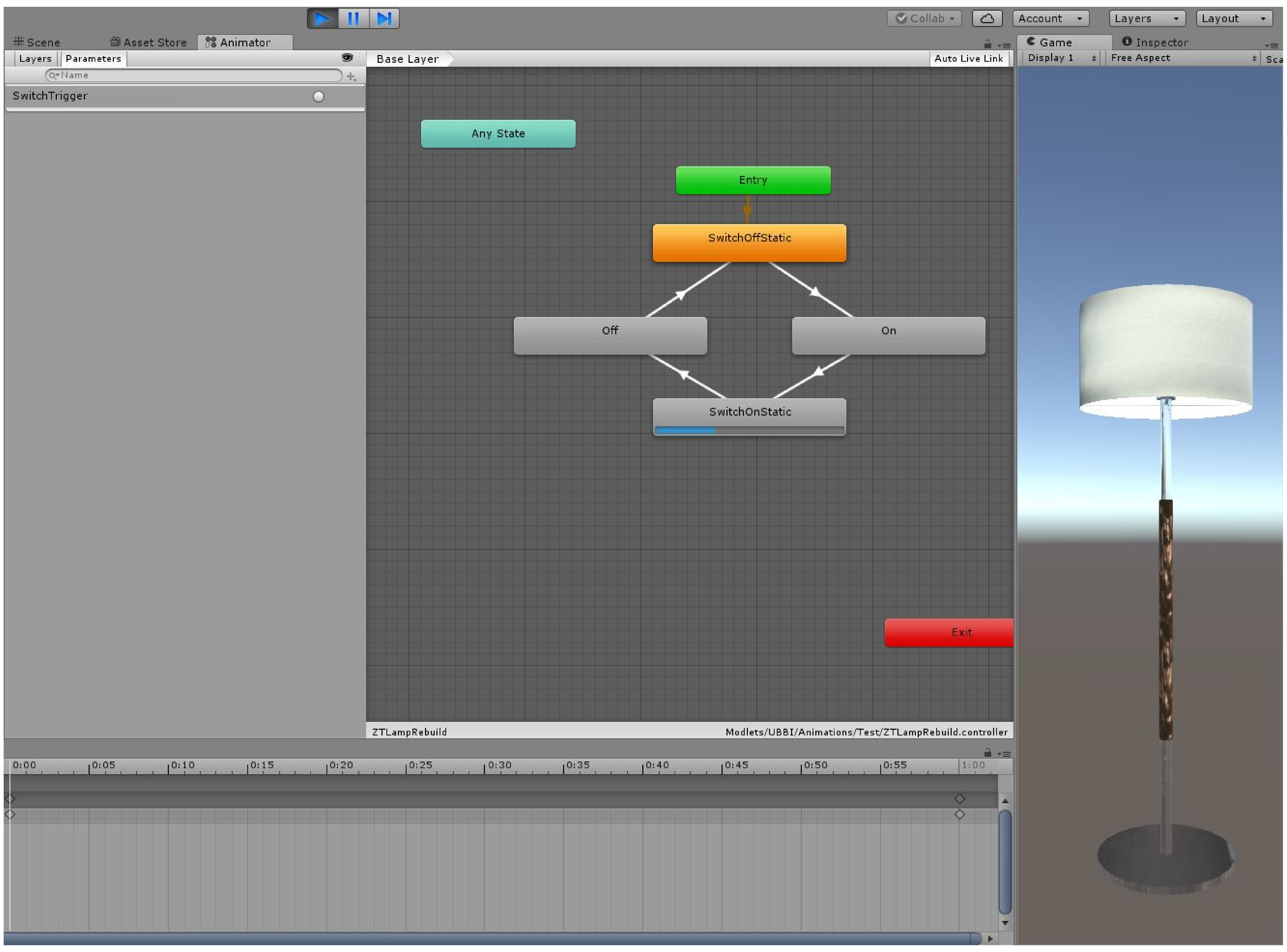
Off -> SwitchOffStatic
1 AnimatorTransitionBase
Transitions Solo Mute
Off -> SwitchOffStatic
Off -> SwitchOffStatic
Has Exit Time ✓
Settings
Off
SwitchOffStatic
Conditions
List is Empty

7 Days to Die Modding Unity, Modlet, XMLs & Localization Tutorial For Setting Up Animator For Switch Class With Lights

17. The last thing we have to do before being done with the animation is to test the light. Press play while in the animator. Click the trigger button to go through animations and you should see the light switch from off, to on and off again.

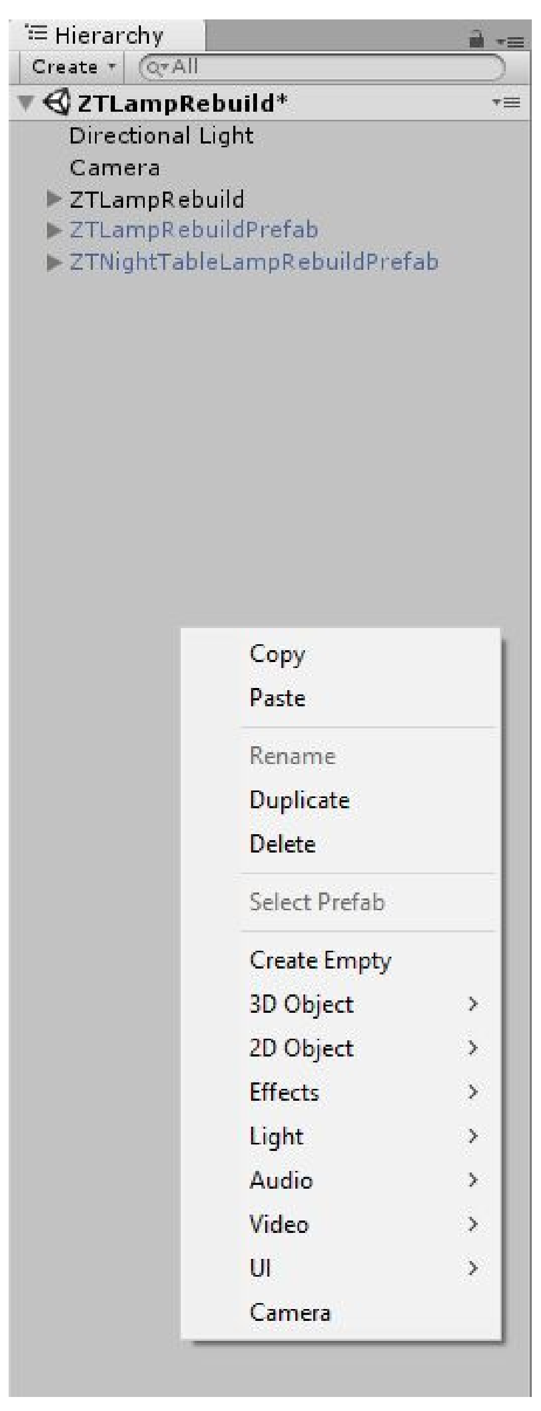
    If it's working, you'll click the "Switch Trigger" after the first block loops a couple of times and it'll automatically go to "On" and then move right away to "SwitchOnStatic" and then stay there until you hit "SwitchTrigger" again. Then once you hit it again, it'll loop from "SwitchOnStatic" to "Off" and ultimately to "SwitchOffStatic."

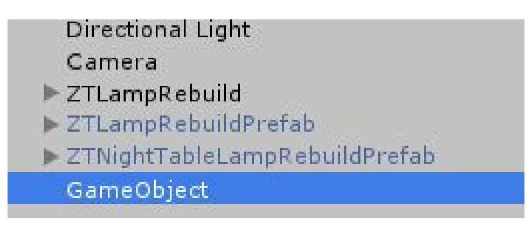


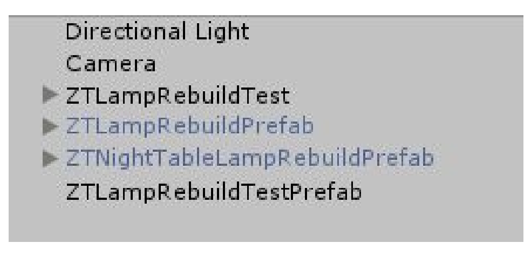


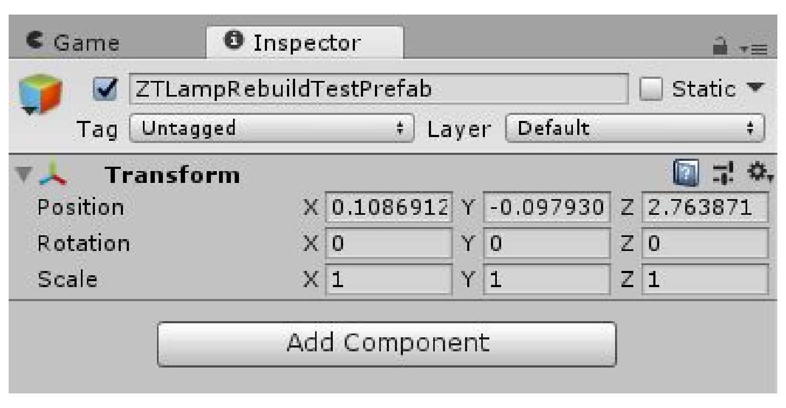
18. Once you confirm your animation works, move back to your hierarchy and create an empty object and title it "BlahBlahPrefab" or in this case, "ZTLampRebuildTestPrefab." The key is making sure the the child's name is "ZTLampRebuild" and the empty GameObject created as the "parent" is titled the same thing + Prefab, so it turns into "ZTLampRebuildTestPrefab." Once created, make sure the (X,Y,Z) position of the Prefab is set to 0 by hitting "reset."

Once created and prefab position reset to 0, grab "ZTLampRebuildTest" and move it into "ZTLampRebuildTestPrefab, so it matches the below hierarchy:





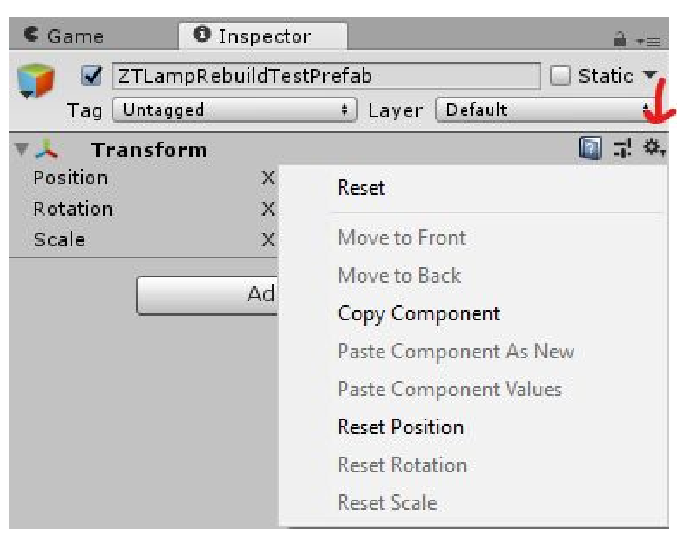


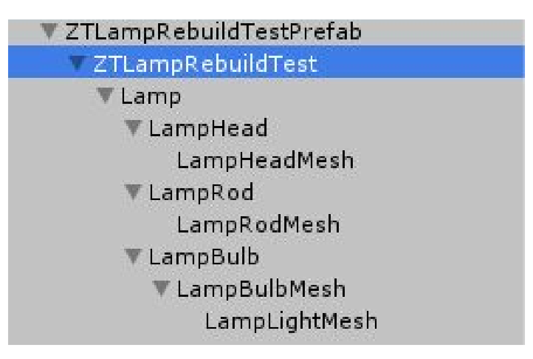


ZTLampRebuildTestPrefab

Tag Untagged Layer Default

	X	Y	Z
Position	0.1086912	-0.097930	2.763871
Rotation	0	0	0
Scale	1	1	1

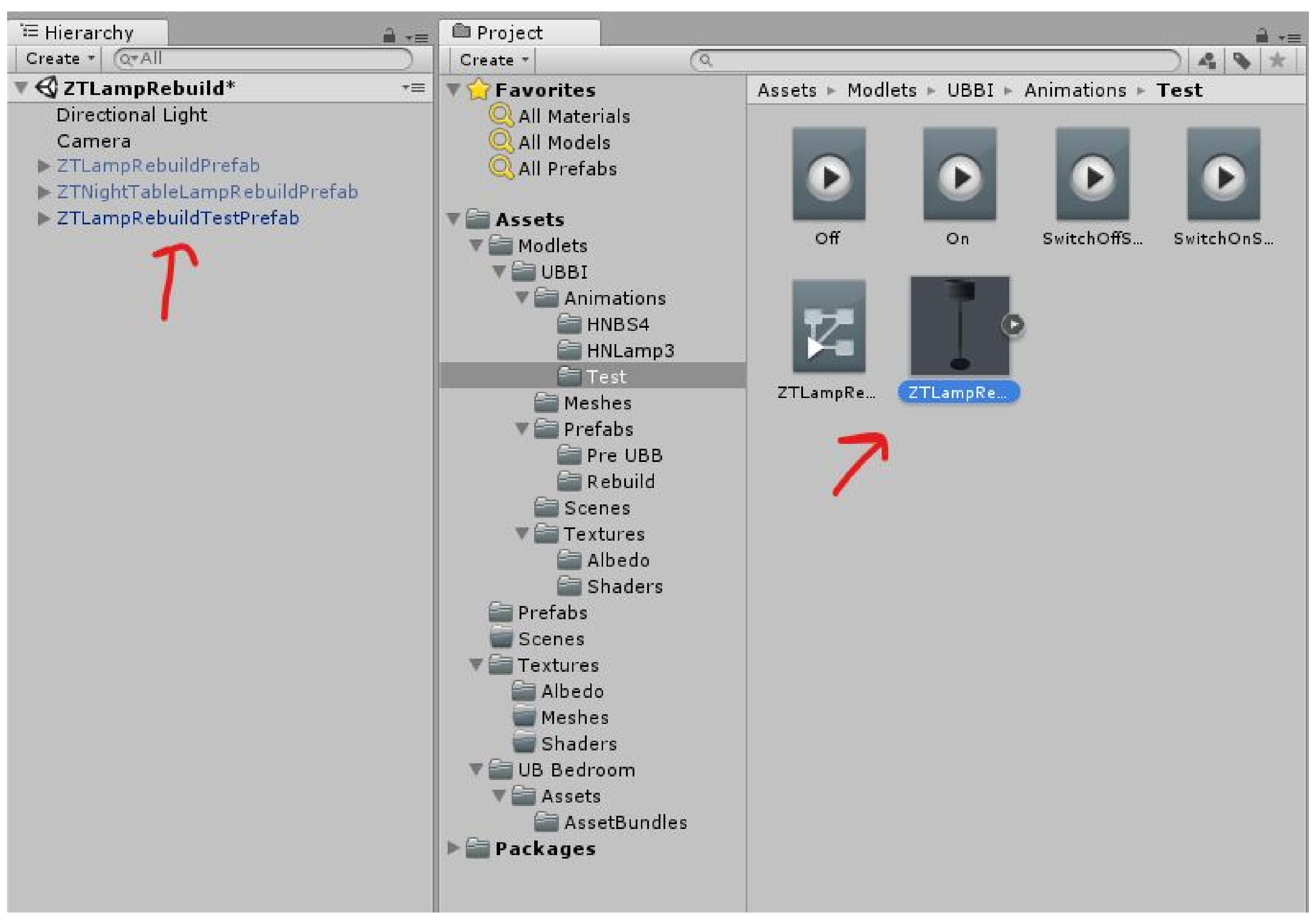


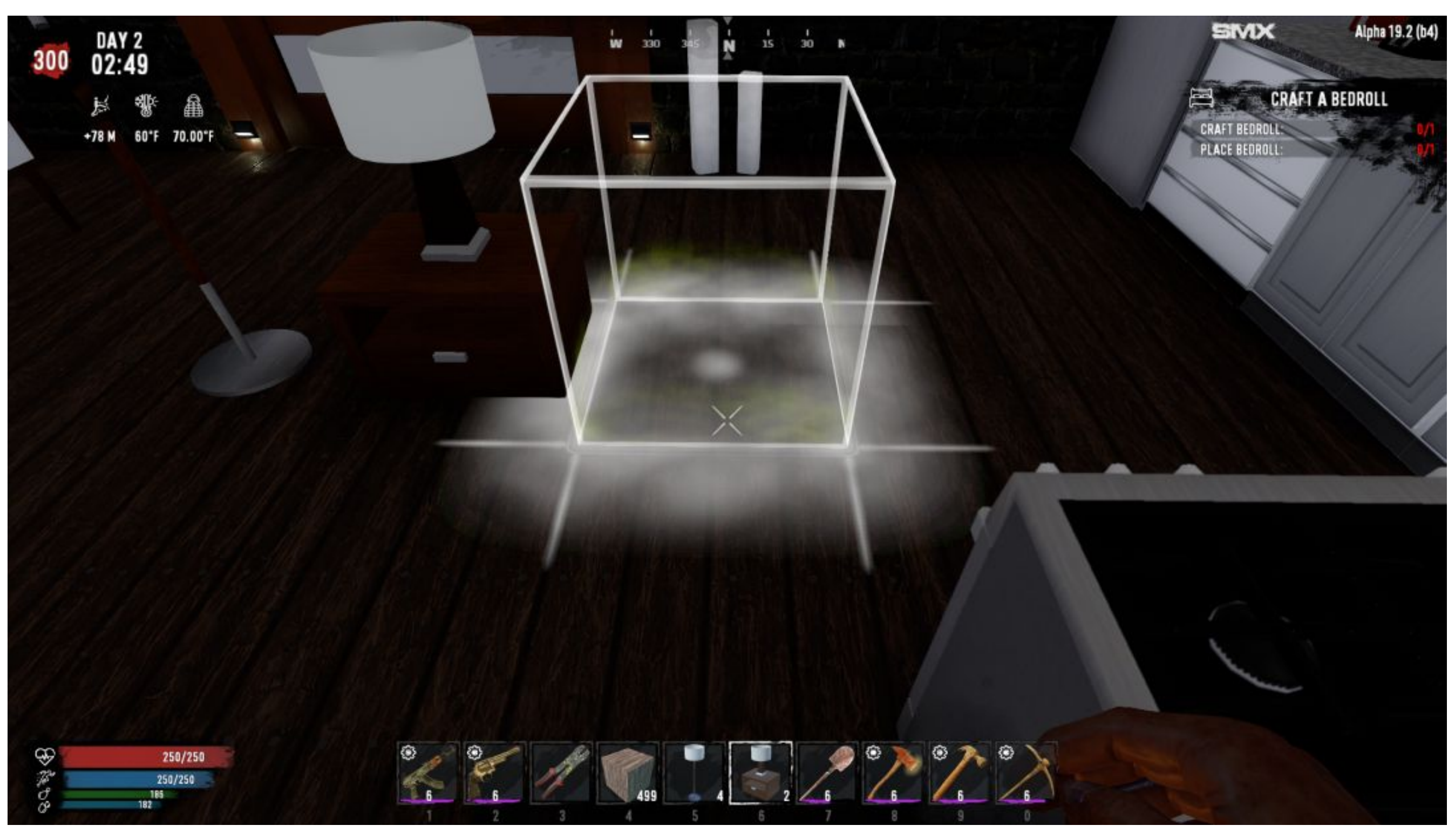


19. The next thing you want to do is grab and hold your finished prefab holding the entire object with proper hierarchy, animations, colliders, a tag, textures, etc. over to the project area like below and it will look like this:

What's important to know is that you should make a folder for Prefabs. I'm just putting this together as a test, so I just carried it into the same animation folder from before to show you.

Note: If the Prefab does not hold an image within its block like above, it will not show like in-game, as shown here next to this picture. If this occurs, just re-do the process and you should get it.



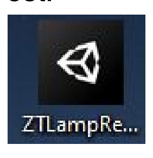


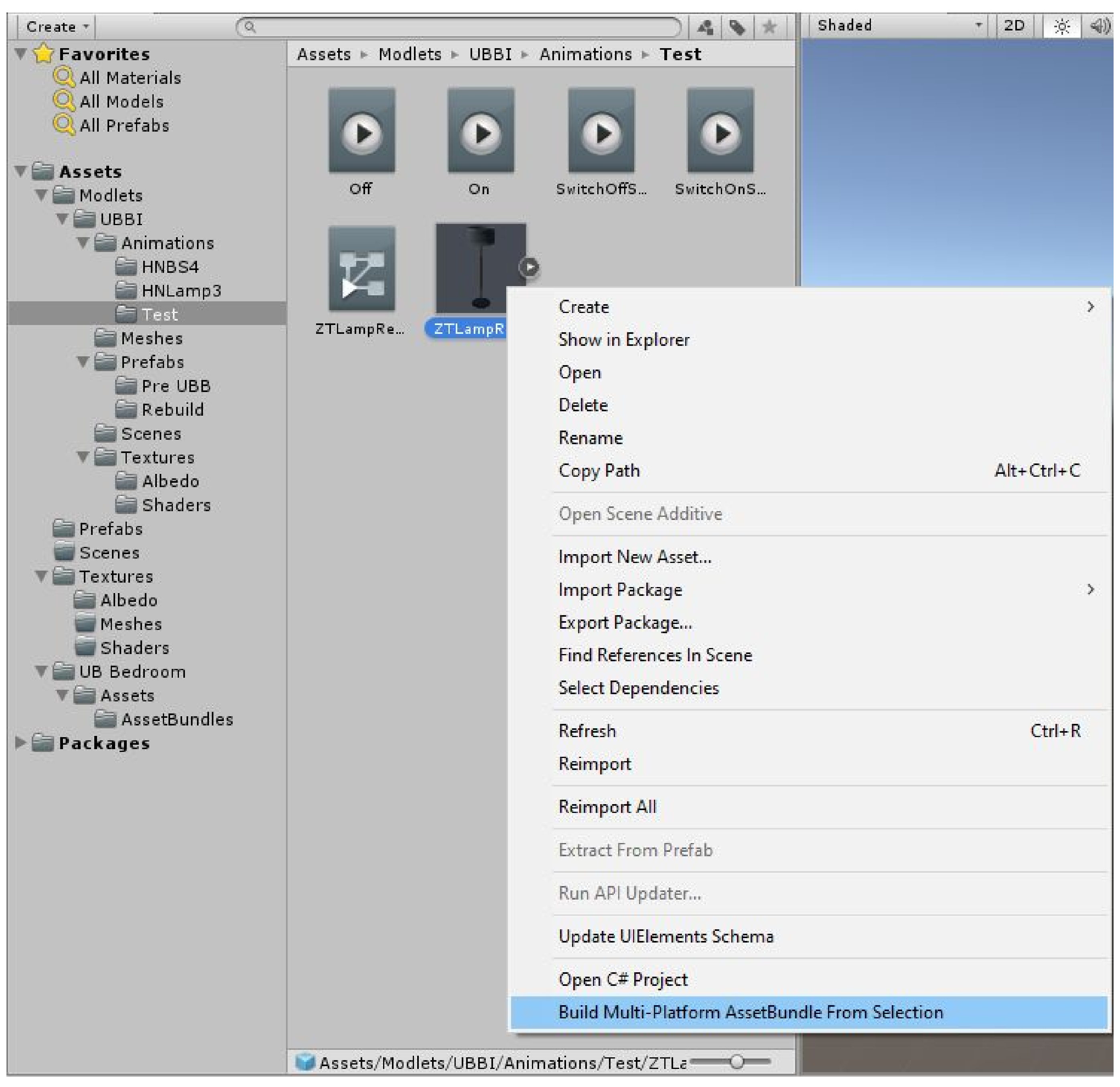
20. The final step is to make it a "Multi-Platform AssetBundle" by right clicking the Prefab we carried into the project folder. The only way to achieve this however, is to have a C# file titled "MultiPlatformExportAssetBundles." This can be retrieved from **Xyth**'s Templates and Utilities files on his Github found here - **https://github.com/7D2D/Templates-and-Utilities**.

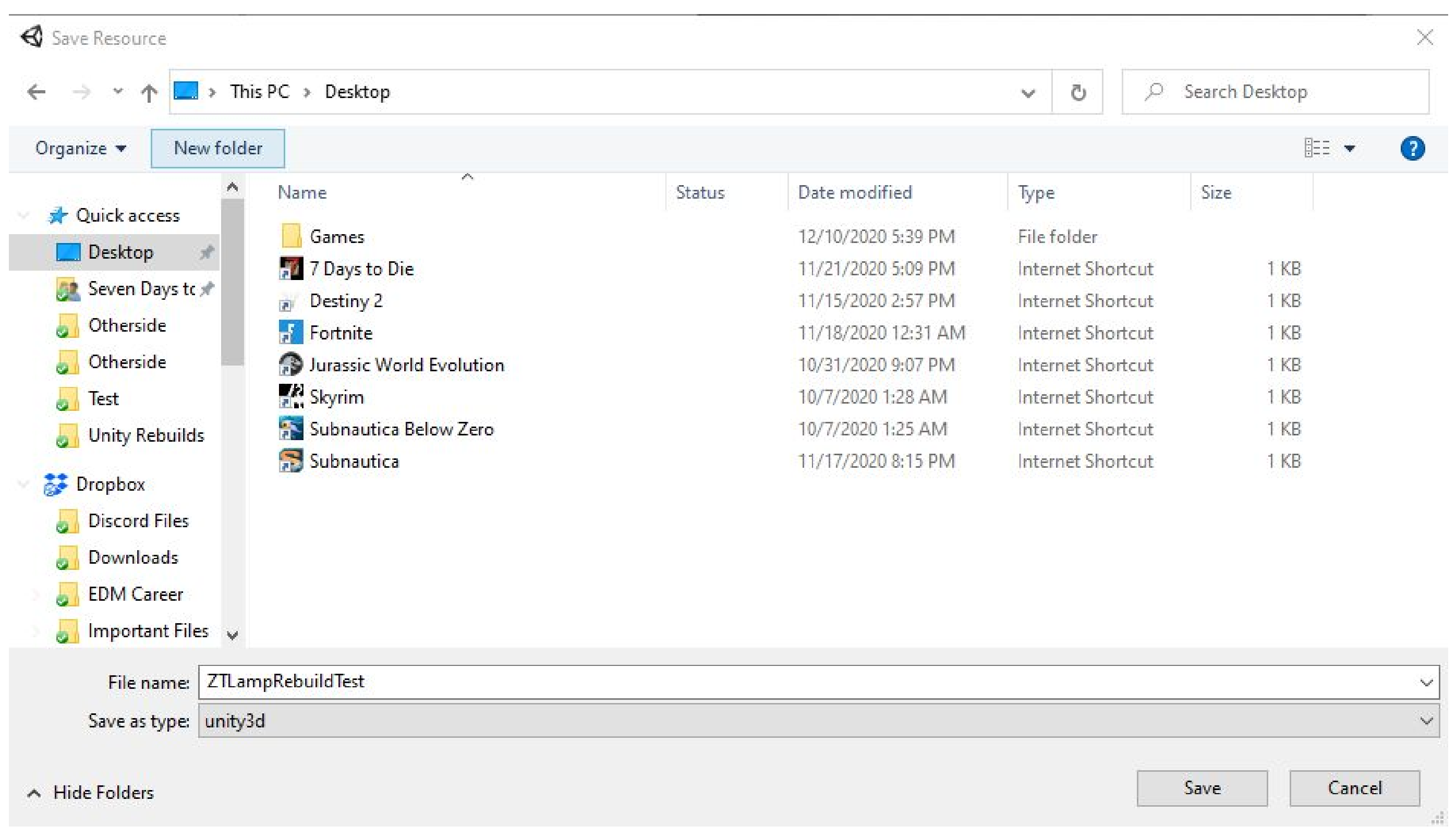
I've found that once you've added it, it doesn't matter where it is located in your "Assets," but it will look like this:

To meet naming conventions, give it the same name as the child directly under the ZTLampRebuildTestPrefab," but instead title the unity3d file as "ZTLampRebuildTest."

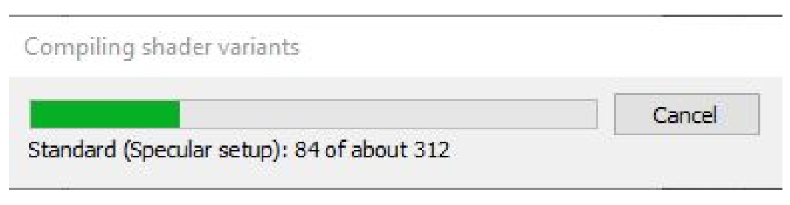
Depending on the size of the Prefab and object, you may have a lot to load before it's created, but it will do a little bit of loading, before it's done and looks like this:





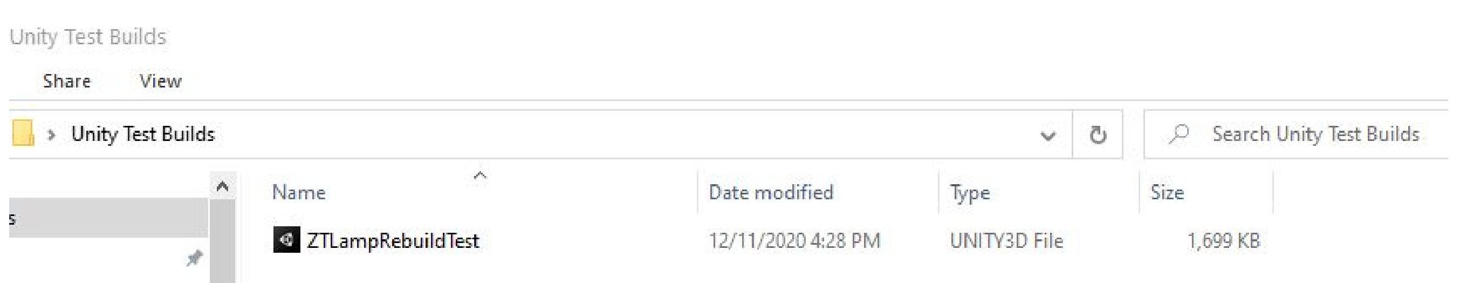


File name: ZTLampRebuildTest
Save as type: unity3d



Compiling shader variants

Standard (Specular setup): 84 of about 312



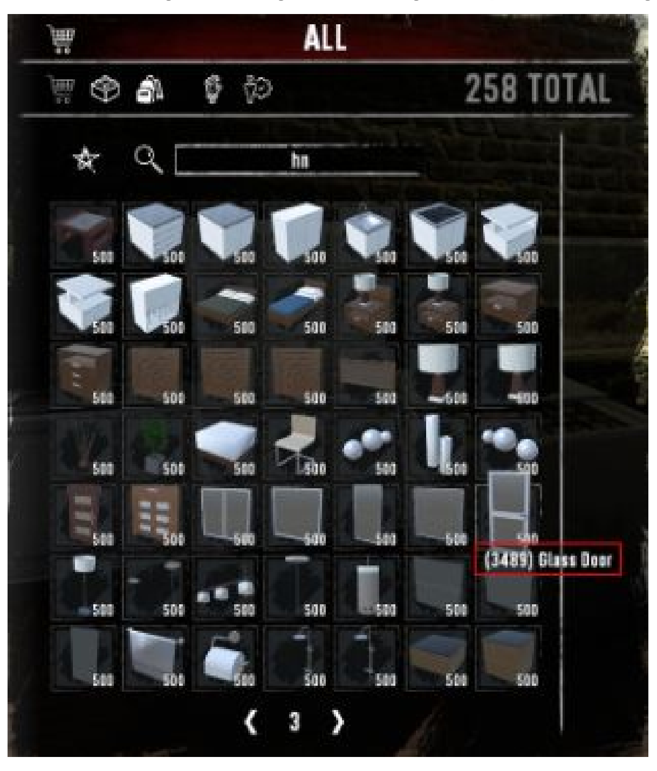
Name	Date modified	Type	Size
ZTLampRebuildTest	12/11/2020 4:28 PM	UNITY3D File	1,699 KB

21. Now that we have done all the hard work of getting the asset ready to be in-game and work like a light using the "switch class" to turn on/off using the wire tool, now we have to get it working inside a modlet. In other words, we now must make our modlet. To do so, we have to create folders in the appropriate hierarchy along with a ModInfo.xml file (Read Me.md is optional).

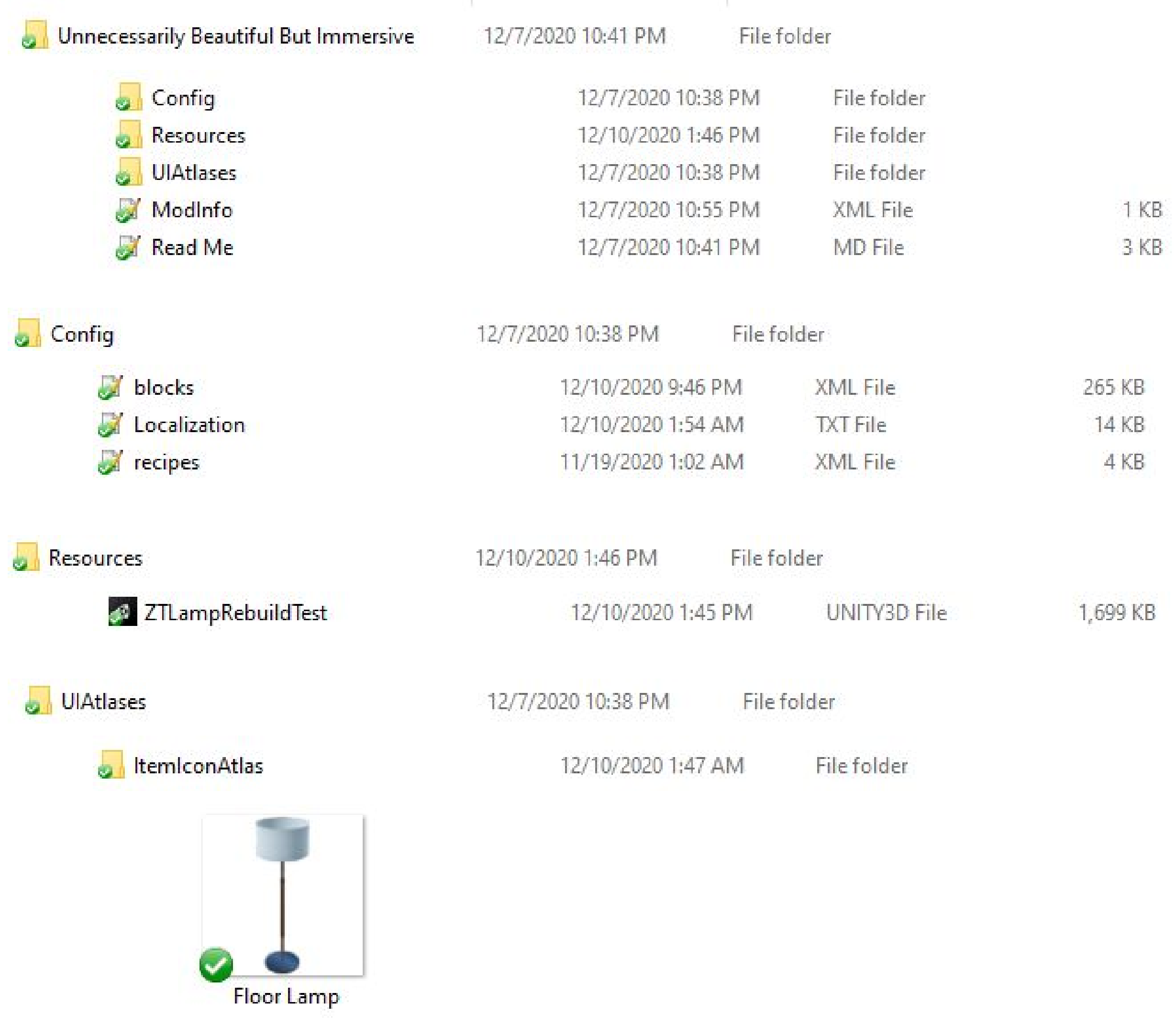
Config: The Config folder holds XML files like blocks.xml (which is important for what we've been working on), recipes.xml (also important if you want to make the item craftable) and txt files like a Localization.txt file. A Localization.txt file is optional to an extent, but it should be normalized with the rise in usage of health bar mods from Sirillion, Khaine and Konrad. With a Localization.txt file, you give proper names to everything you put into the game, which not only increases immersiveness and makes it easier for other modders to work with your content, but also gives a sense of professionalism to your modlet.

Resources: The Resources folder holds the Unity3D File that we created earlier, it must remain here to be called within the blocks.xml coding in order to be placed in the game and function as intended.

UIAtlases ->ItemIconAtlas: The UIAtlases folder holds the ItemIconAtlas folder, which dictates whether the item shows up as an icon in creative mode searching, crafting searching, recipe searching, workbench searching etc. As a note, you must make sure that the name of the icon, in this cas "Floor Lamp" has the same "CustomIcon" name set as "Floor Lamp" in order for it to show up.



See Modlet Hierarchy Below:



```
Unnecessarily Beautiful But Immersive   12/7/2020 10:41 PM    File folder

    Config                               12/7/2020 10:38 PM    File folder
    Resources                            12/10/2020 1:46 PM    File folder
    UIAtlases                            12/7/2020 10:38 PM    File folder
    ModInfo                              12/7/2020 10:55 PM    XML File        1 KB
    Read Me                              12/7/2020 10:41 PM    MD File         3 KB

Config                                   12/7/2020 10:38 PM    File folder

    blocks                               12/10/2020 9:46 PM    XML File      265 KB
    Localization                         12/10/2020 1:54 AM    TXT File       14 KB
    recipes                              11/19/2020 1:02 AM    XML File        4 KB

Resources                                12/10/2020 1:46 PM    File folder

    ZTLampRebuildTest                    12/10/2020 1:45 PM    UNITY3D File  1,699 KB

UIAtlases                                12/7/2020 10:38 PM    File folder

    ItemIconAtlas                        12/10/2020 1:47 AM    File folder

        Floor Lamp
```

22. Now that we understand the hierarchy, we must build the ModInfo.xml.

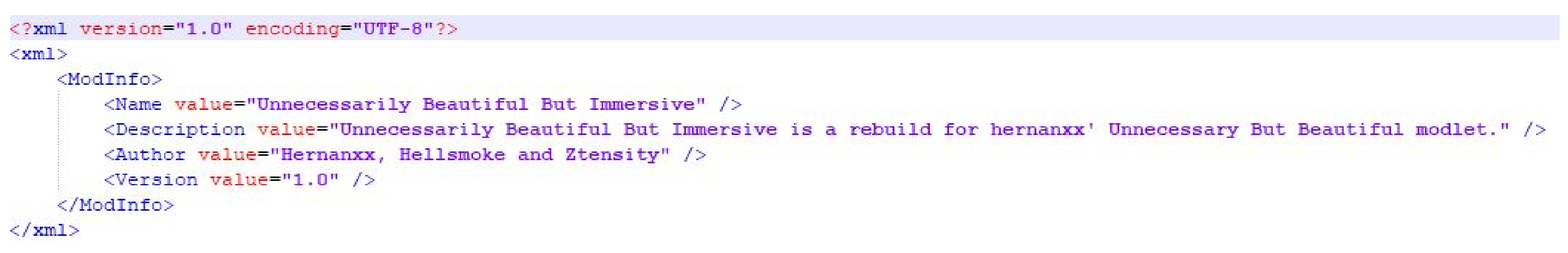
Name: The value here has to be the exact same name as the main folder that the entire modlet is sitting in, which is "Unnecessarily Beautiful But Immersive."

Description: The value here is similar to a read me, it does nothing but give an explanation of what it does. I'd say make it brief. You can make the bigger description in the Read Me.MD file.

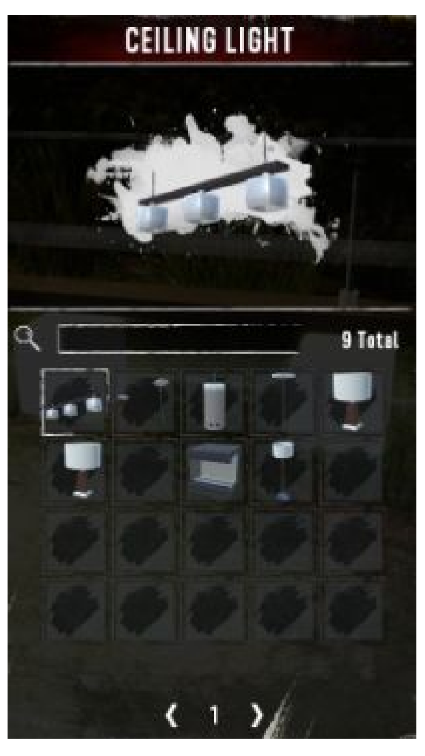
Author: The value here must be either just you or you and other people if credit is due to other authors.

Version: The value here will change as the modlet is updated. I use "1.0" as my first edition.

All other lines stay the same no matter what.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<xml>
    <ModInfo>
        <Name value="Unnecessarily Beautiful But Immersive" />
        <Description value="Unnecessarily Beautiful But Immersive is a rebuild for hernanxx' Unnecessary But Beautiful modlet." />
        <Author value="Hernanxx, Hellsmoke and Ztensity" />
        <Version value="1.0" />
    </ModInfo>
</xml>
```



23. The Read Me.MD or Read Me.txt is an optional file, but I prefer to use it instead of putting the notes in ModInfo.xml. Here, you can lay out easier to read sections like: About this Mod, Tips, Credits, etc. This file is useful in giving context for the modlet you've created, especially if it's a bigger, overhaul modlet.

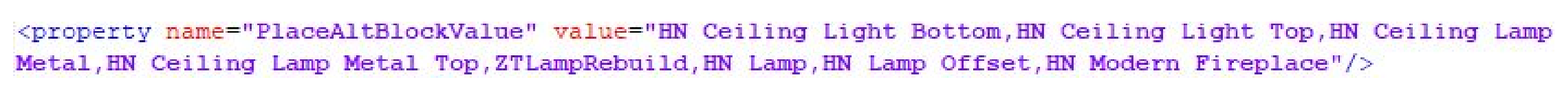
24. Now that we've figured out the hierarchy or mod structure, created a ModInfo.xml and/or a Read Me.MD file, we must now create our blocks.xml file.

The First "HH Lights" Block: This block holds "ZTLampRebuildTest" within it. In other words, in Creative Mode (CM), if you were to search "HN" you'd find "HN Lights."

CustomIcon: This property is set "HN Ceiling Light Bottom," which actually dictates which image from the ItemIconAtlas is being displayed when searching for "HN" to build at a workbench or in CM.

PlaceAltBlockValue: This property is important, as when you have multiple blocks put together here such as "ZTLampRebuildTest, HN Lamp, Ceiling Lamp, etc," the order in which you put these will make them appear a certain way when editing a block to pick which one to build. See on the right side to "Ceiling Light" (renamed from the Localization). As you can see, one space is missing in-between a lamp and a fireplace, that is because in this instance I had accidentally left "HN Lamp" in place of this value like "ZTLampRebuildTest, Ceiling Lamp, HN Lamp" when it should have only been "ZTLampRebuildTest,Ceiling Lamp." Where you put "ZTLampRebuildTest" will also dictate where it's listed in the list.

See what a longer version of the code looks like below:

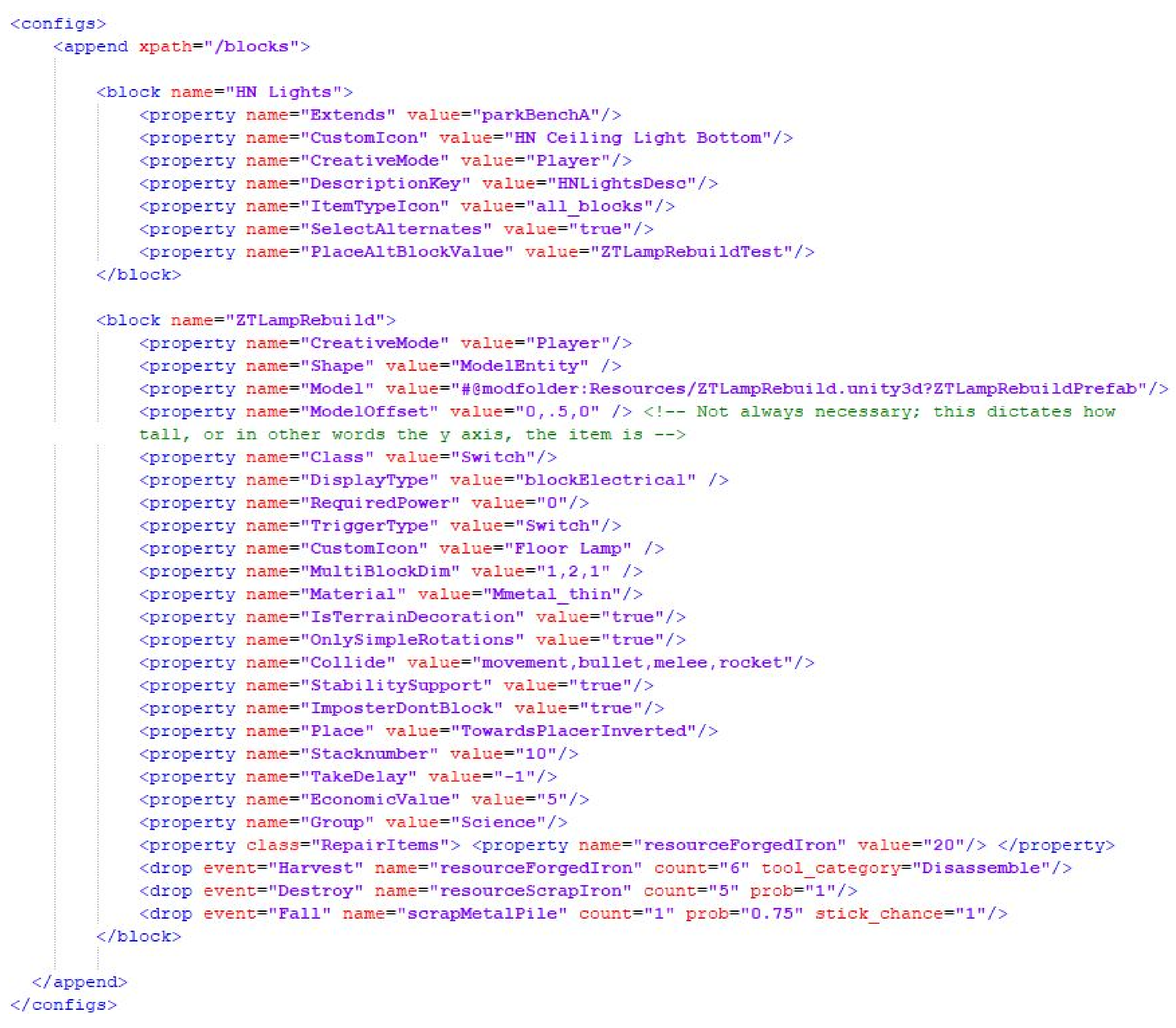


```
<property name="PlaceAltBlockValue" value="HN Ceiling Light Bottom,HN Ceiling Light Top,HN Ceiling Lamp
Metal,HN Ceiling Lamp Metal Top,ZTLampRebuild,HN Lamp,HN Lamp Offset,HN Modern Fireplace"/>
```

MultiBlockDim: This property reflects how much block space the object takes up in-game. For instance, my lamp took up 1 block in x, 2 blocks in y and 1 block in z. Look at those values as (X,Y,Z).

ModelOffset: This property actually determines how tall an object is for you to be able to click on it. The value of .5 was provided by Xyth and described to fit two blocks high or 2 blocks in the y direction.

See the coding necessary for the Prefab (blocks.xml file) we have created below:



```
<configs>
    <append xpath="/blocks">

        <block name="HN Lights">
            <property name="Extends" value="parkBenchA"/>
            <property name="CustomIcon" value="HN Ceiling Light Bottom"/>
            <property name="CreativeMode" value="Player"/>
            <property name="DescriptionKey" value="HNLightsDesc"/>
            <property name="ItemTypeIcon" value="all_blocks"/>
            <property name="SelectAlternates" value="true"/>
            <property name="PlaceAltBlockValue" value="ZTLampRebuildTest"/>
        </block>

        <block name="ZTLampRebuild">
            <property name="CreativeMode" value="Player"/>
            <property name="Shape" value="ModelEntity" />
            <property name="Model" value="#@modfolder:Resources/ZTLampRebuild.unity3d?ZTLampRebuildPrefab"/>
            <property name="ModelOffset" value="0,.5,0" /> <!-- Not always necessary; this dictates how
            tall, or in other words the y axis, the item is -->
            <property name="Class" value="Switch"/>
            <property name="DisplayType" value="blockElectrical" />
            <property name="RequiredPower" value="0"/>
            <property name="TriggerType" value="Switch"/>
            <property name="CustomIcon" value="Floor Lamp" />
            <property name="MultiBlockDim" value="1,2,1" />
            <property name="Material" value="Mmetal_thin"/>
            <property name="IsTerrainDecoration" value="true"/>
            <property name="OnlySimpleRotations" value="true"/>
            <property name="Collide" value="movement,bullet,melee,rocket"/>
            <property name="StabilitySupport" value="true"/>
            <property name="ImposterDontBlock" value="true"/>
            <property name="Place" value="TowardsPlacerInverted"/>
            <property name="Stacknumber" value="10"/>
            <property name="TakeDelay" value="-1"/>
            <property name="EconomicValue" value="5"/>
            <property name="Group" value="Science"/>
            <property class="RepairItems"> <property name="resourceForgedIron" value="20"/> </property>
            <drop event="Harvest" name="resourceForgedIron" count="6" tool_category="Disassemble"/>
            <drop event="Destroy" name="resourceScrapIron" count="5" prob="1"/>
            <drop event="Fall" name="scrapMetalPile" count="1" prob="0.75" stick_chance="1"/>
        </block>

    </append>
</configs>
```

A lot of the additional fields may need to be changed if you're making a different type of block like for running a door ("DoorSecure"), but for a switch, the above code should be sufficient with a few tweaks here and there. Watching Xyth's videos, tutorials and looking at other peoples' modlets will help you learn completely what needs to be changed vs. not.

In this block, what doesn't need changed are:

- <property name="CreativeMode" value="Player"/>
- <property name="Shape" value="ModelEntity" />
- <property name="Class" value="Switch"/>
- <property name="DisplayType" value="blockElectrical" />
- <property name="RequiredPower" value="0"/>
- <property name="TriggerType" value="Switch"/>
- <property name="Material" value="Mmetal_thin"/>
- <property name="IsTerrainDecoration" value="true"/>
- <property name="Collide" value="movement,bullet,melee,rocket"/>
- <property name="StabilitySupport" value="true"/>
- <property name="ImposterDontBlock" value="true"/>
- <property name="Place" value="TowardsPlacerInverted"/>
- <property name="TakeDelay" value="-1"/>
- <property name="Group" value="Science"/>

In this block, what does need to be changed are:

- <property name="Model" value="#@modfolder:Resources/ZTLampRebuild.unity3d?ZTLampRebuildPrefab"/>
    - What will need to change is - "ZTLampRebuild.unity3d?ZTLampRebuildPrefab"
    - What dictates this is the name of your Prefab. As long as you have followed all naming conventions during the Unity creation, the first part will be the "ZTLampRebuild" and then after unity3d, it will be "ZTLampRebuild" + Prefab.
- <property name="ModelOffset" value="0,.5,0" />
    - This property is actually entirely optional. If your object is a door, tall lamp etc. then you'll want this. If you're exceeding well beyond two blocks high and want the player to be able to use it from the same height (say a radio tower for example) you will have to figure out the offset based on watching videos from Xyth or reaching out to the Discord #unity channel.
    - If you are using this property for a door, a tall object like a lamp etc. basically an object that takes up two blocks high and you want the player to be able to interact with it on the entire block, you will want to keep this property exactly as is.
- <property name="CustomIcon" value="Floor Lamp" />
    - This value needs to be changed per the name of the same icon in your "ItemIconAtlas" folder.
- <property name="MultiBlockDim" value="1,2,1" />
    - This value will change based on how many blocks the object you're putting in the game is.
    - The blocks will be dictated by the X, Y, Z axis or in other words how long the object is by blocks, how tall the object is by blocks and how wide it is by blocks.
- <property name="Stacknumber" value="10"/>
    - This value determines how many can stack in your inventory at once.
    - For example, it would make more sense to have a lamp only be stackable immersively by say 2-10, while cash could allow stackable up to 6,000.
- <property name="EconomicValue" value="5"/>
    - This value will determine how valuable the object will be at Traders.
    - For example, this could make a big deal in online servers for online lobbies - an item people might fight people for.
- <property class="RepairItems"> <property name="resourceForgedIron" value="20"/> </property>
- <drop event="Harvest" name="resourceForgedIron" count="6" tool_category="Disassemble"/>
- <drop event="Destroy" name="resourceScrapIron" count="5" prob="1"/>
- <drop event="Fall" name="scrapMetalPile" count="1" prob="0.75" stick_chance="1"/>
    - All these fields can be changed based on what makes the most sense.
    - "RepairItems" dictates how the player will repair the object.
    - "Harvest" and "Destroy" dictate what you will retrieve when breaking the object.
    - "Fall" dictates what will replace the object when it's destroyed and the probability of that occurring.

```
<configs>
    <append xpath="/blocks">

        <block name="HN Lights">
            <property name="Extends" value="parkBenchA"/>
            <property name="CustomIcon" value="HN Ceiling Light Bottom"/>
            <property name="CreativeMode" value="Player"/>
            <property name="DescriptionKey" value="HNLightsDesc"/>
            <property name="ItemTypeIcon" value="all_blocks"/>
            <property name="SelectAlternates" value="true"/>
            <property name="PlaceAltBlockValue" value="ZTLampRebuildTest"/>
        </block>

        <block name="ZTLampRebuild">
            <property name="CreativeMode" value="Player"/>
            <property name="Shape" value="ModelEntity" />
            <property name="Model" value="#@modfolder:Resources/ZTLampRebuild.unity3d?ZTLampRebuildPrefab"/>
            <property name="ModelOffset" value="0,.5,0" /> <!-- Not always necessary; this dictates how
            tall, or in other words the y axis, the item is -->
            <property name="Class" value="Switch"/>
            <property name="DisplayType" value="blockElectrical" />
            <property name="RequiredPower" value="0"/>
            <property name="TriggerType" value="Switch"/>
            <property name="CustomIcon" value="Floor Lamp" />
            <property name="MultiBlockDim" value="1,2,1" />
            <property name="Material" value="Mmetal_thin"/>
            <property name="IsTerrainDecoration" value="true"/>
            <property name="OnlySimpleRotations" value="true"/>
            <property name="Collide" value="movement,bullet,melee,rocket"/>
            <property name="StabilitySupport" value="true"/>
            <property name="ImposterDontBlock" value="true"/>
            <property name="Place" value="TowardsPlacerInverted"/>
            <property name="Stacknumber" value="10"/>
            <property name="TakeDelay" value="-1"/>
            <property name="EconomicValue" value="5"/>
            <property name="Group" value="Science"/>
            <property class="RepairItems"> <property name="resourceForgedIron" value="20"/> </property>
            <drop event="Harvest" name="resourceForgedIron" count="6" tool_category="Disassemble"/>
            <drop event="Destroy" name="resourceScrapIron" count="5" prob="1"/>
            <drop event="Fall" name="scrapMetalPile" count="1" prob="0.75" stick_chance="1"/>
        </block>

    </append>
</configs>
```

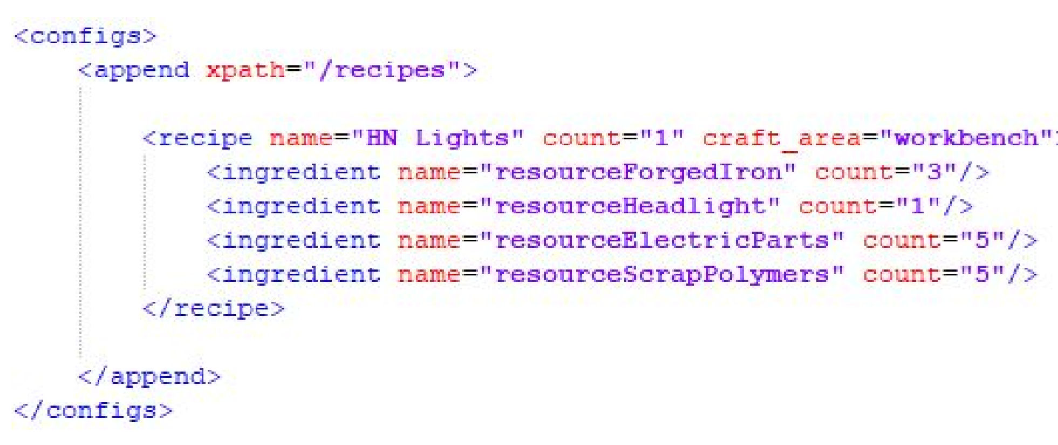
25. Now that we've created a blocks.xml file in order to get our block into the game, the next step is to make a recipes.xml if we want to make it optional to craft the object at a workbench. See code below:

Recipe Name:The recipe name can be anything. As we can localize it later. What's important is that whatever it is here, you can search it quickly in your crafting area by searching "HN Lights" even if it's localized as "Lights."

Craft Area: There are likely other options, but we use "workbench" to make the object only craftable through the workbench via this field.

Ingredient: Here is where you would list what resources are required to craft the item and then how many of that ingredient.

What's important to note is that since we've used "HN Lights" instead of "ZTLampRebuild," is what you craft is actually a block that can turn into multiple items like the lamp or another light, which was set by **PlaceAltBlockValue**.



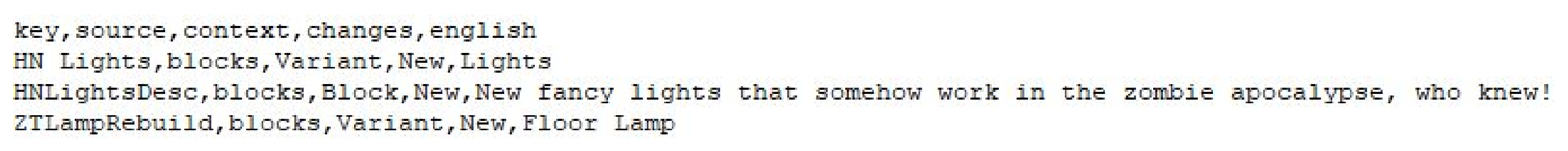
```
<configs>
    <append xpath="/recipes">

        <recipe name="HN Lights" count="1" craft_area="workbench">
            <ingredient name="resourceForgedIron" count="3"/>
            <ingredient name="resourceHeadlight" count="1"/>
            <ingredient name="resourceElectricParts" count="5"/>
            <ingredient name="resourceScrapPolymers" count="5"/>
        </recipe>

    </append>
</configs>
```

26. The final type of file to create and possibly the most important is a Localization.txt file. Most mod authors will say this is optional and not quite as important, but I will push for this needing to be normalized. This said, localizations actually dictate the names of everything you see in games from NPCs like the traders, zombies and animals to machines, vending machines and even things like UI elements. If you use a localization file in your modlet, it will increase the immersiveness of your mod and therefore greatly increase the likelihood of people trying out your mod and sticking it in their load order. See an example of the Localization.txt file below:

Localization.txt files act like CSV files, which should be viewable in Excel, Google Sheets, etc. However, the easiest way to tackle them is with Notepad++ or via **wrathmaniac**'s [7 Days to Die Mod editor](). I personally prefer to do localizations via Notepad++, however **wrathmaniac**'s mod editor does help you create mods, find block files etc. especially if you're new to everything.

First Line: This is the header. It can be very simplified, but I prefer to use either "key,source,context,changes,english" or "key,source,context,english."

My understanding is "key,source,context,changes,english" should be used when changing a localization for vanilla blocks, creatures, etc. Meanwhile "key,source,context,english" localizes blocks, creatures etc. added by your mod specifically. What's important is "changes," which will always be reflected as "New" when used.



```
key,source,context,changes,english
HN Lights,blocks,Variant,New,Lights
HNLightsDesc,blocks,Block,New,New fancy lights that somehow work in the zombie apocalypse, who knew!
ZTLampRebuild,blocks,Variant,New,Floor Lamp
```

The remaining lines will be based on what you want to localize. In this instance, we wanted to localize "HN Lights," "HNLightsDesc" (dictated by Description Key property in blocks.xml code) and "ZTLampRebuild." In order to make things more immersive, we changed HN Lights to "Lights," "HNLightsDesc" to say "New fancy lights that somehow work in the zombie apocalypse, who knew!" and "Floor Lamp."

This is how the header and the blocks correspond to one another (if in the excel sheet):

key	source	context	english	HN Lights
HN Lights	blocks	Variant	New	Lights
HNLightDesc	blocks	Block	New	New fancy lights lights...
ZTLampRebuild	blocks	Variant	New	Floor Lamp

YOU'RE READY TO TEST & PLAY!!

Now that you have created your object in Unity, created an entire modlet around it and put it in the correct folder, the last thing to do is test it. Carry the modlet into your "mods" folder, create a new save and load into the game. Press F1, type CM to active creative mode and search for your block and get a wire tool. As long as you do not get any red error lines or yellow error lines while loading in, as you place your object or use your object, you should be good to go! If it works, you can then either upload to Nexus, Forums, 7 Days to Die Mods website, share onto the Discord or keep it to yourself.

If you have any issues, just check with the Discord channels and the community can help you from there!

Congratulations!

Ztensity

My Modlets: Ztensity's Vanilla Creature Vanilla Renaming Modlet on Nexus / Ztensity's Creature Renaming Modlet Series on Forums

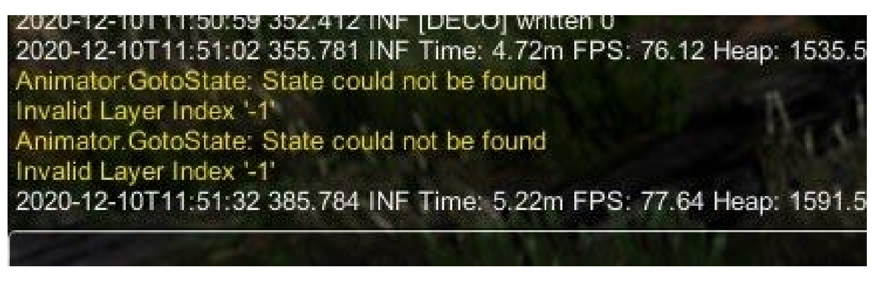
Shout Outs: I want to give shout outs to mod authors in Guppy's Discord like wrathmaniac, TSBX, Xyth, Mumpfy, sphereii, seminal, Khaine, chiko, Guppy, bdubyah, Zoro8183 and Ragsy for helping me learn the entire process of recipes, blocks, Localizations, Unity, "Switch Classes" etc. I couldn't have made my first mods, as much progress I have on my soon to be announced public project and this guide without everyone in that community.

Community Links & Resources:
- Wrathmaniac's 7 Days to Die Mod Editor: https://7daystodiemods.com/7-days-to-die-mod-edit/
- Sphereii's Mod Launcher: http://7d2dmodlauncher.org/
- 7 Days to Die Mods Website: https://7daystodiemods.com/
- 7 Days to Die Forums Mods Page: https://community.7daystodie.com/forum/27-mods/
- Guppy's 7 Days to Die Discord: https://discord.gg/Zh3SNaqHnn
- 7 Days to Die page on Nexus Mods: https://www.nexusmods.com/7daystodie
- Xyth's YouTube channel for 7 Days to Die videos (Unity Tutorials, showcases, etc.): https://www.youtube.com/channel/UCUAR3Bc_Z97lyT4zeI3-dcg

Bug Fixes & Extra Tips

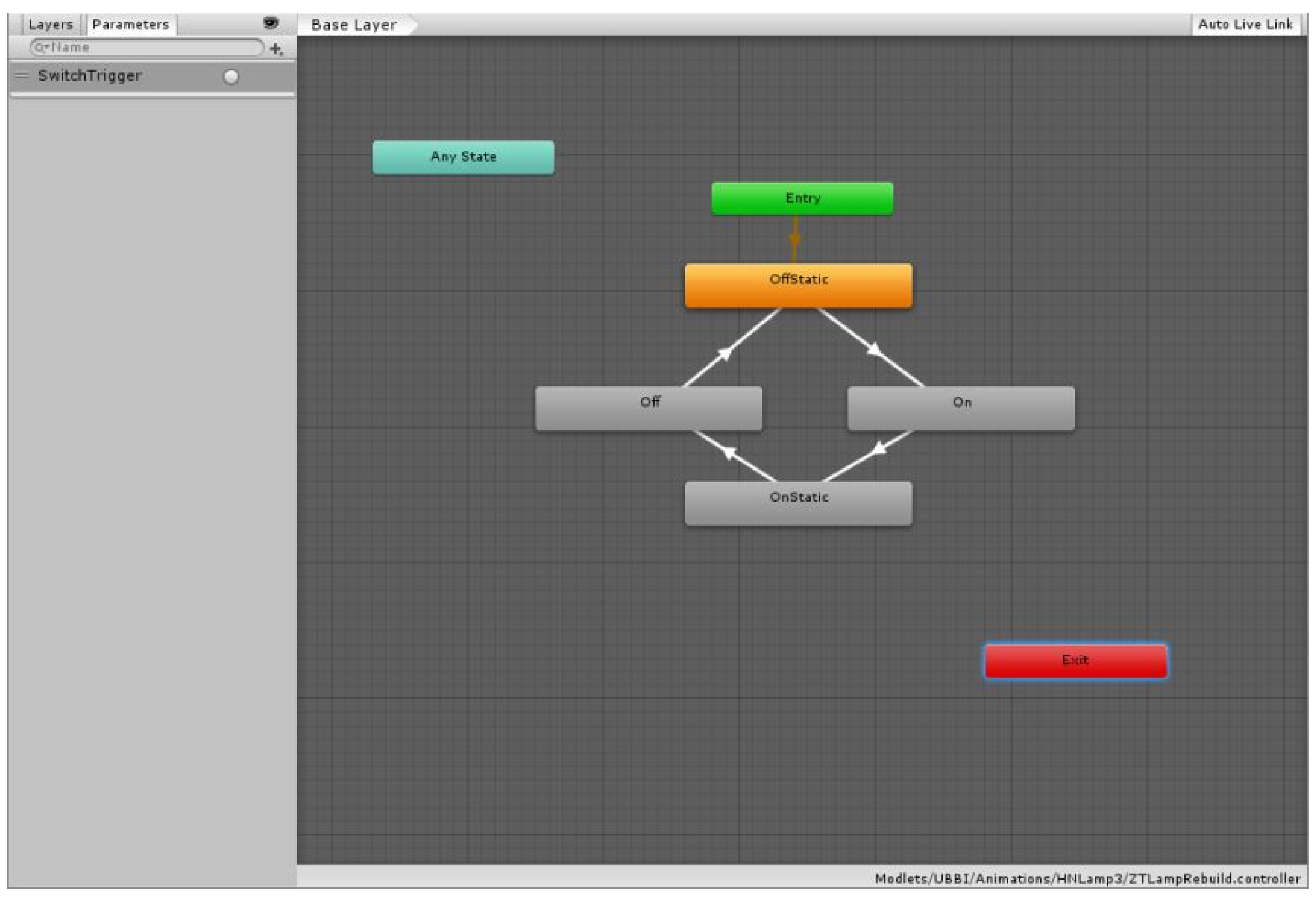
Yellow line errors in 7 Days to Die console caused by Animator:

In late testing, I found these lines in the console. After conversations with Xyth and Guppy, we found that it was due to the names I gave my animation clips.



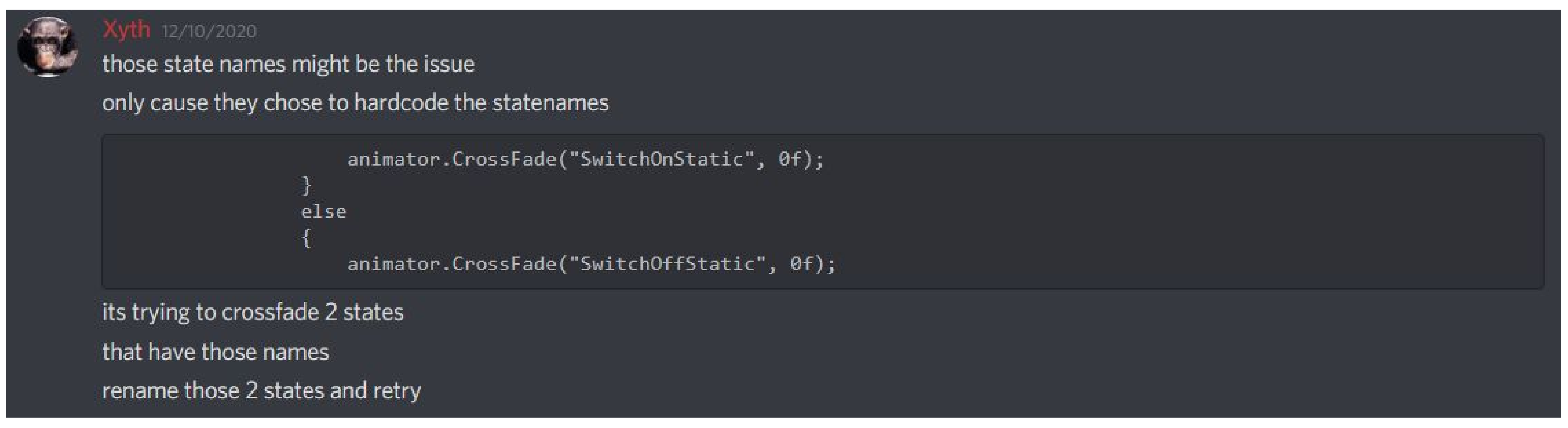
```
2020-12-10T11:50:59 352.412 INF [DECO] written 0
2020-12-10T11:51:02 355.781 INF Time: 4.72m FPS: 76.12 Heap: 1535.5
Animator.GotoState: State could not be found
Invalid Layer Index '-1'
Animator.GotoState: State could not be found
Invalid Layer Index '-1'
2020-12-10T11:51:32 385.784 INF Time: 5.22m FPS: 77.64 Heap: 1591.5
```

This was what the animation clips were:



SOLUTION:

What made it so that these yellow lines wouldn't show were to change names from "OffStatic" and "OnStatic" to "SwitchOffStatic" and "SwitchOnStatic."



Xyth 12/10/2020
those state names might be the issue
only cause they chose to hardcode the statenames

```
                    animator.CrossFade("SwitchOnStatic", 0f);
                }
                else
                {
                    animator.CrossFade("SwitchOffStatic", 0f);
```

its trying to crossfade 2 states
that have those names
rename those 2 states and retry

Yellow Line Information:

Yellow lines can be ignored, but if you are the mod author, I think it's best to make them disappear if possible.

An explanation from Xyth was that "yellow animator layer errors happen when the game code calls the objects animator in Unity, but that layer doesn't exist in that animator. -1 means the layer name specified in code was not found. If it called it by index, it should return the layer number that's missing, like when people don't add a pain layer to entities in A19."