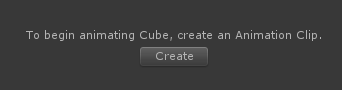
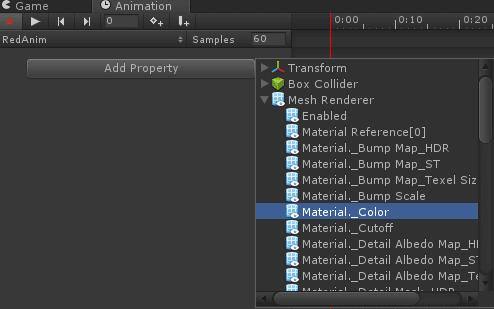
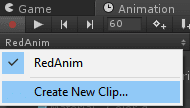
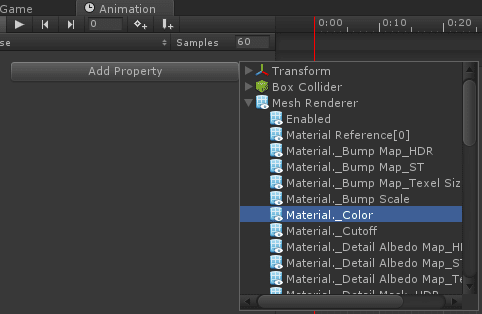
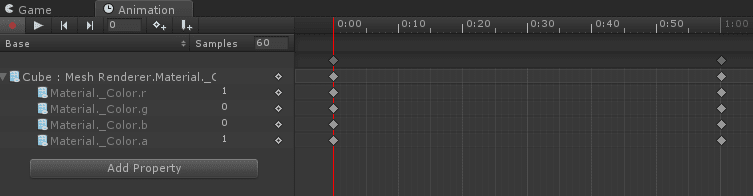
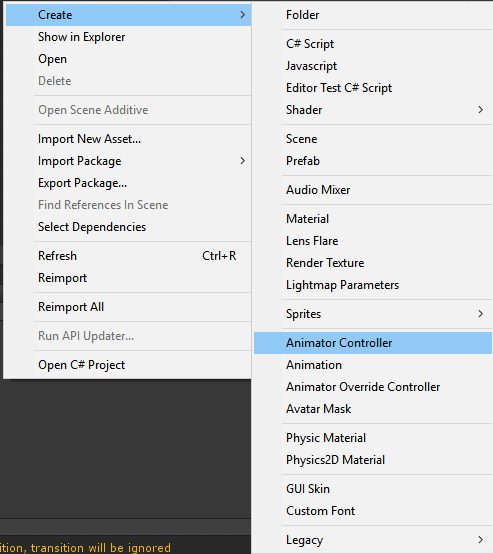
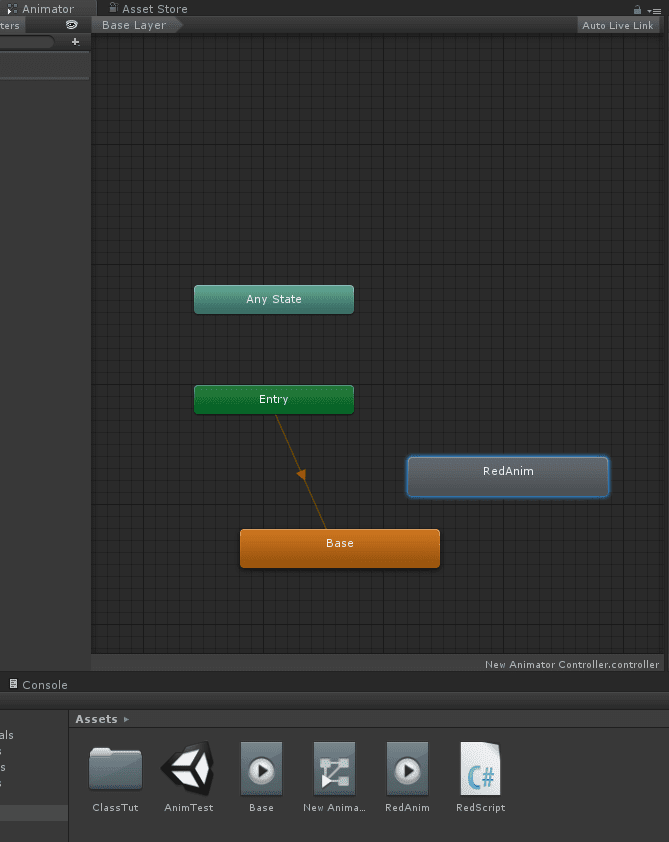
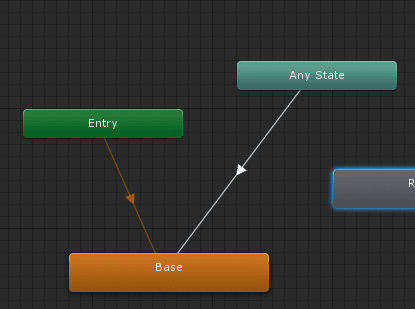
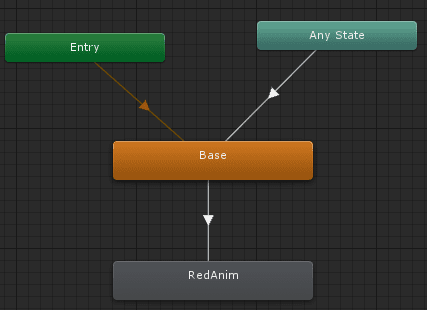
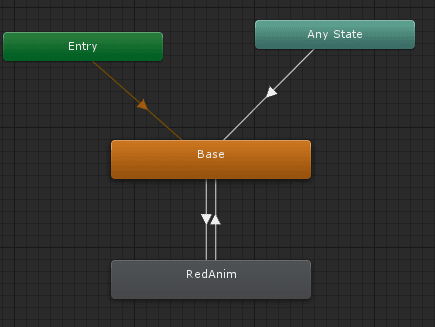
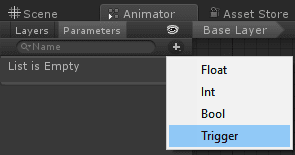
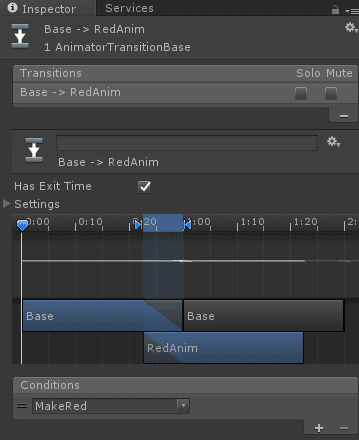
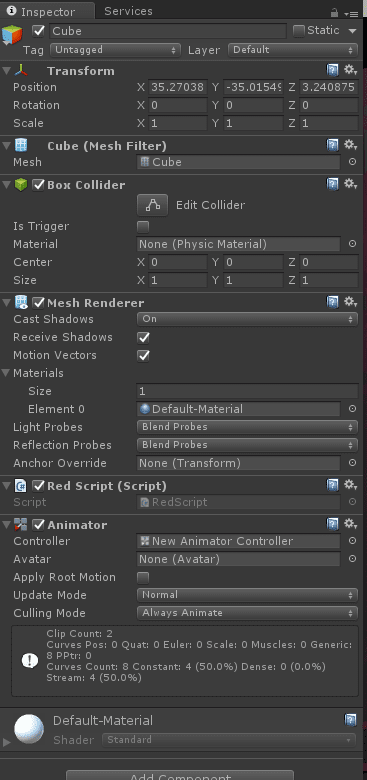
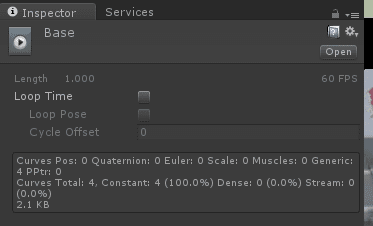
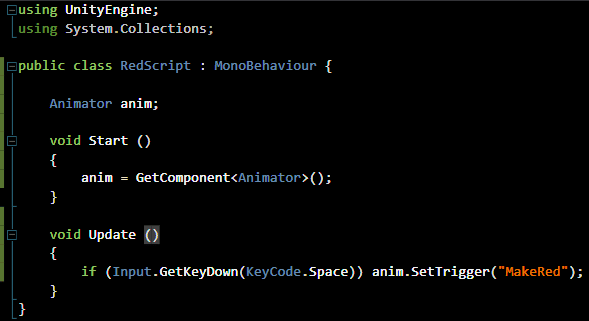
How to Make an Animation Controller

Assuming  you know most of the basics of [Unity](http://www.studica.com/unity/unity-5-professional-edition-for-students-and-faculty.html?ex_ref=news_BLOG&utm_source=blog&utm_medium=blogpost&utm_campaign=UTAnim), such as creating Game Objects and adding components to them.

1. Create a new Scene
2. Create a 3D Cube Game Object and position your camera to be looking at it
3. Go to the Window menu and click on “Animation”. This will open the animation window
4. In the animation window, click the “Create” button in the middle of the window.  
   
5. A window will come up asking you to specify a filename. Name it whatever you want. I named mine “RedAnim”.
6. In the animation window, click the “Add Property” button. Choose “Mesh Renderer”. Find “Material.\_Color” and scroll right in the window and choose the “+” icon to add it to the list of properties.  
   
7. This is all we need for the base animation. It will ensure that our cube is white.
8. Moving along in this Unity tutorial, we need to make another animation for our base color. Click the “RedAnim” header button and choose “Create New Clip…” as seen here:  
   
9. Another dialogue box will come up asking you what to save the file as. Name it “Base”. We’re going to add the same color property again by clicking “Add Property” and choosing the Mesh Renderer → Material.\_Color option.  
   
10. Make sure your timeline is at 0:00 and change the g and b values to 0.  
      
    This will make the transition from red back to white look more natural.
11. Now, we need to create the animation controller. Right-click in the project window and click Create → Animator Controller. Name it whatever you want.  
    
12. Double click on the animator controller to open the animator controller editor. Once it’s open, drag and drop both of your animations into the animator controller window.  
    
13. The next part of this Unity tutorial is to create transition logic for our animations. Right-click on “Any State” and choose “Make Transition”, Move your mouse on top of the “Base” node and left click on it. This will connect an arrow from Any State onto the Base Animation, like such:  
    
14. Now, we want to be able to transition from our Base state to our RedAnim state. Right-click on “Base” and choose “Make Transition” and click on “RedAnim”.  
    
15. Now, we also want to be able to go back to our base state from RedAnim. So right-click on RedAnim and choose “Make Transition” and connect it to the “Base” node.  
    
16. This flowchart is basically what we’re looking for. However, we still need to create a variable that we can use in our code to be able to make the transition happen. To do this. Click the “+” icon near the top-left of the animator controller window and choose “Trigger” from the drop down:  
    
17. Name the trigger “MakeRed”. We’ll be able to call on this trigger via C#. However, we need to tell the transitions that we want to use this trigger to make a transition happen. Left click on the arrow going **from base to RedAnim**. You’ll notice the transition come up in the inspector on the right side of Unity. There is a section called “Conditions”. Click the “+” icon in this section and it will add the “MakeRed” trigger to the list of conditions.  
    
18. We also need to do this on the transition **from RedAnim to Base**. Left-click on that arrow and do the same thing. Now Unity will know to only make these transitions happen if the MakeRed trigger is activated. **Notice how we use a single trigger for both transitions**. This works because each transition will only happen if we are already in that transition. So, if we are on the Base animation, then MakeRed can **only** cause the transition from Base to RedAnim to happen. The same is true for if we are on the RedAnim animation.
19. At this point, we need to put the animator controller on our cube. Drag and drop the animator controller you created into the cube object in the inspector. The cube’s component list should look like this:  
    
20. We need to change one more thing on our animations before we create the script. Left click on the animation in your project window. In the inspector, uncheck the “Loop Time” box. This will ensure that looping is turned off on these animations. **Do this for both animations**.  
      
    **Note that you may not always want to disable looping, depending on what you intend for your animation.**
21. Now you need to create a new C# script named RedScript. This script is incredibly simple. It looks like this:  
    
22. All this script does is grab an Animator component on the object. It then listens for the user to press Spacebar. If spacebar is pressed, then we set the MakeRed trigger, which will then fire off our animation transitions.
23. Save the script and attach it to the cube in your scene. Now, if you play the game and press spacebar. You should see the cube change to red. If you press spacebar again, it will change back to white. You can do this endlessly and it will keep making the same transitions occur.**If you leave your animator controller node tree open while the game runs, you can see how your animations are behaving.**This can be helpful for troubleshooting potential issues with your animation logic.