Restaurant Simulator Document

Jake Shin

Table of Contents

1. Introduction 3
2. 3D Characters 4
3. Apply Animations to Character 10
4. Apply a set of animations (more than 1) to Character 14
5. How to see the list of animations 17
6. Make a character move to a designated destination 20
7. Conditional transition of animation 23
8. Generate Scenario 26
9. Human Robot Interaction 29
10. Future Works 30

1. Introduction

- The main goal of this project was to help user to generate various human 3D animations that convey realistic and potential human and serving robot's behaviors in the restaurant environment.

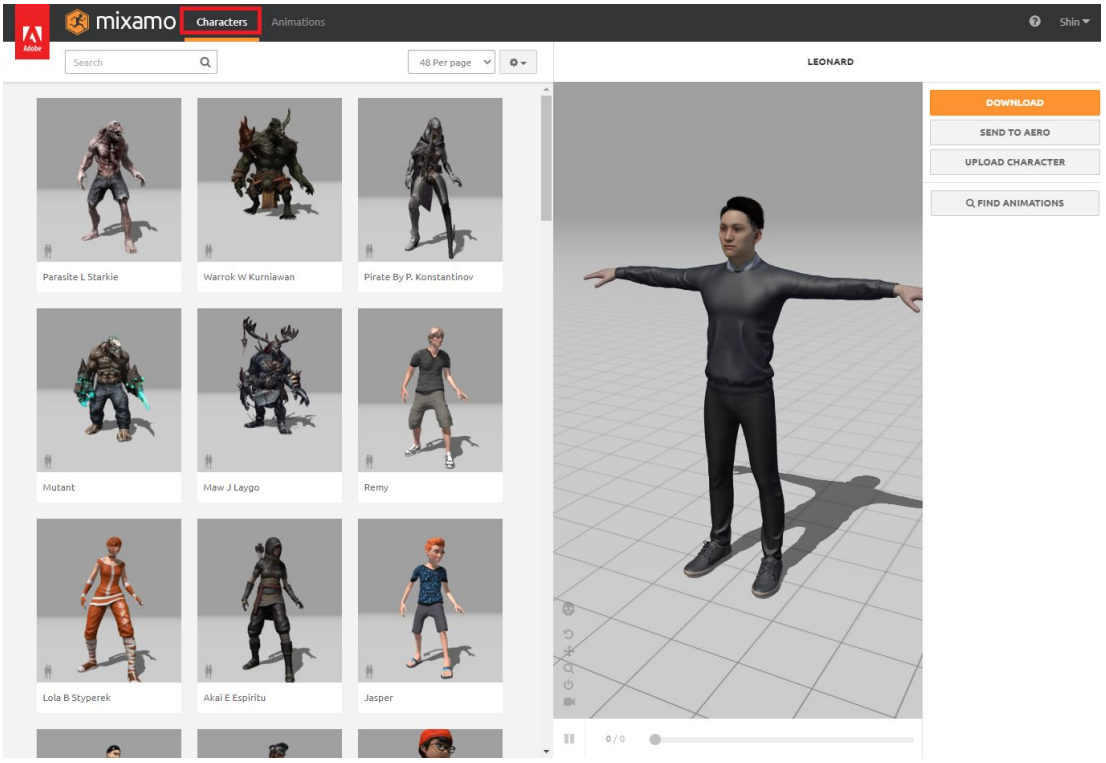
- Unity3D, 3D Game Engine that controls Characters/Scenarios, has been used to build this project
- Blender is used for creating custom animations for 3D human characters
- Unity Asset Store has been used to Download/Import restaurant & food asset
- Adobe Mixamo has been used to import pre-built 3D human characters
- This document contains all information about how to manage this simulator, but you need to individually learn how to use a **Blender** to create 3D animation

2. 3D Characters

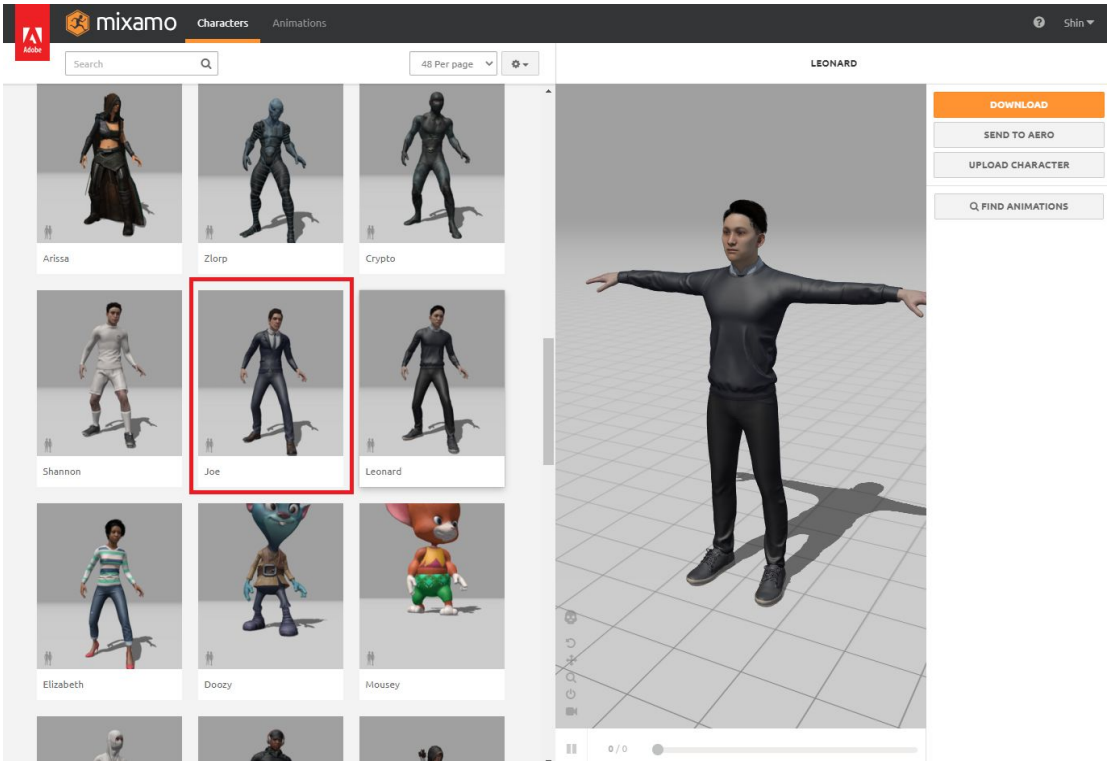
- For this project, 3D characters at Adobe Mixamo have been used.

- Steps to download/import 3D characters from the Adobe Mixamo

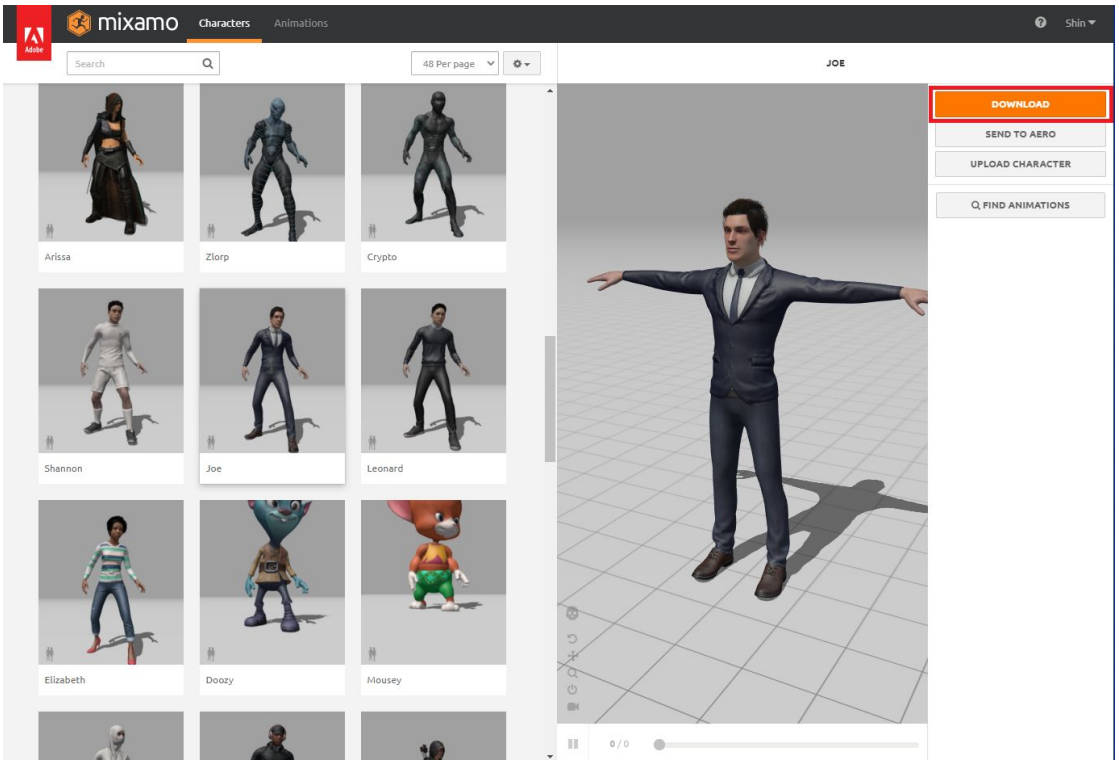
1. Go to Adobe Mixamo and login
2. Click the **Character** Tab



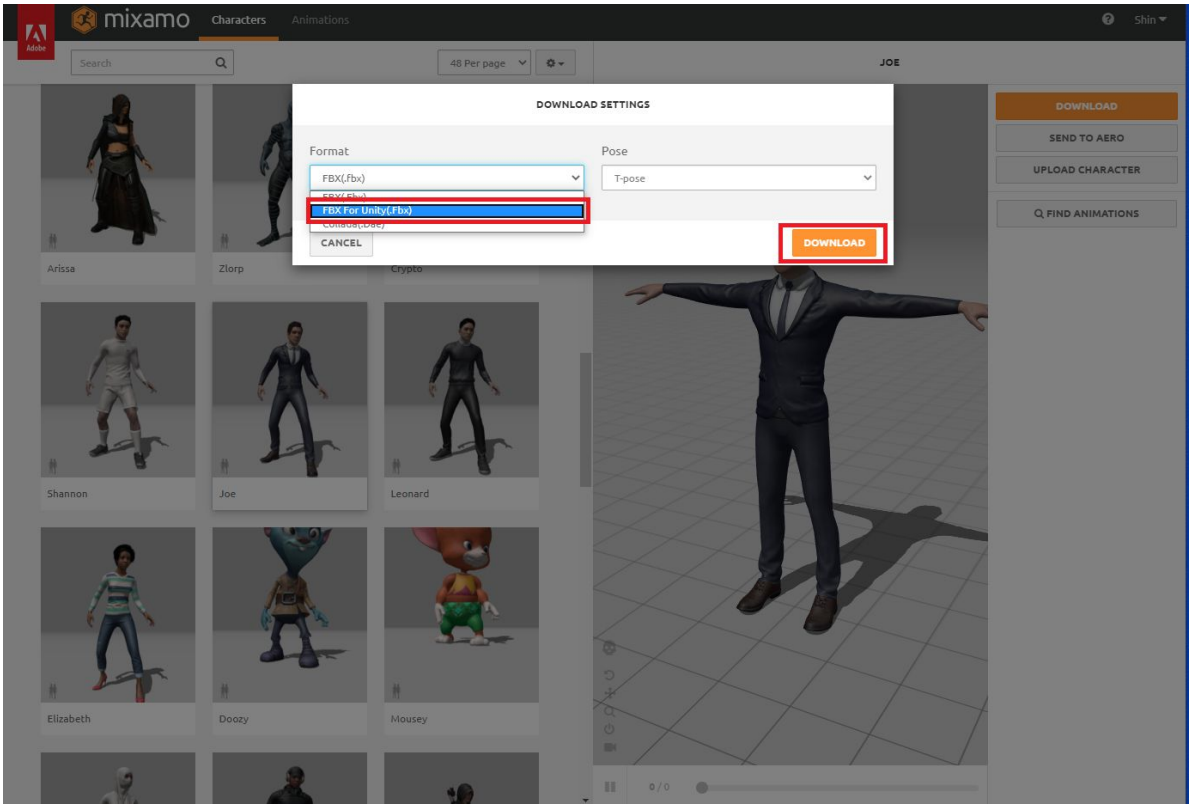
3. Click any 3D character you want to use



4. Click **Download**

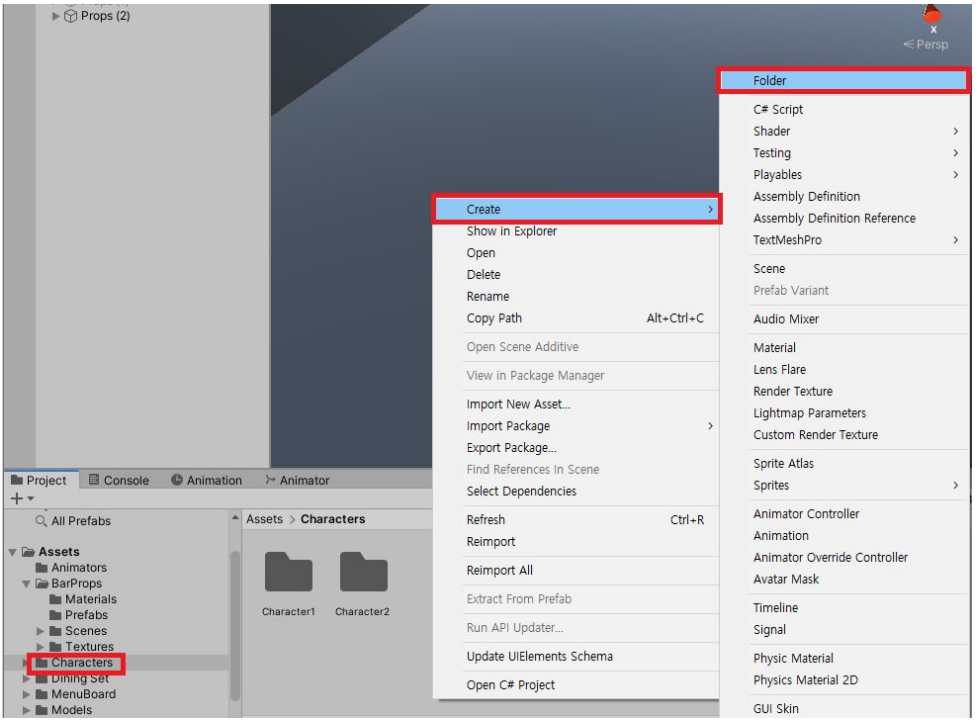


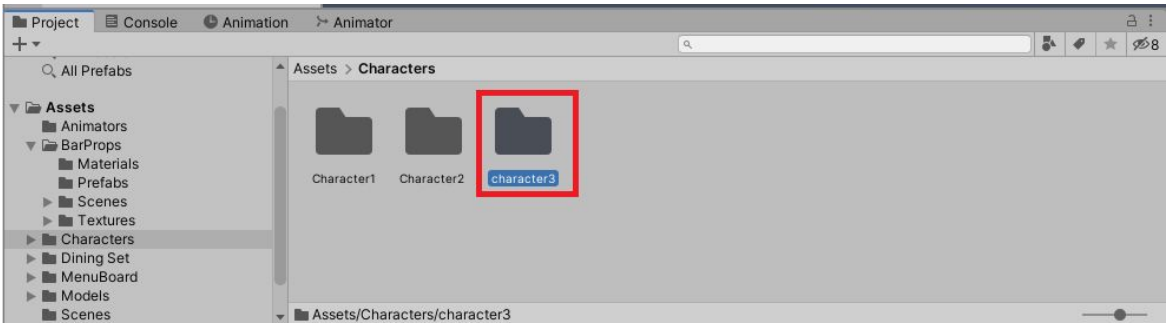
5. From **Format** tab, click **fbx for unity**, then click download



6. After download, you will have .fbx file that contains 3D character you selected.

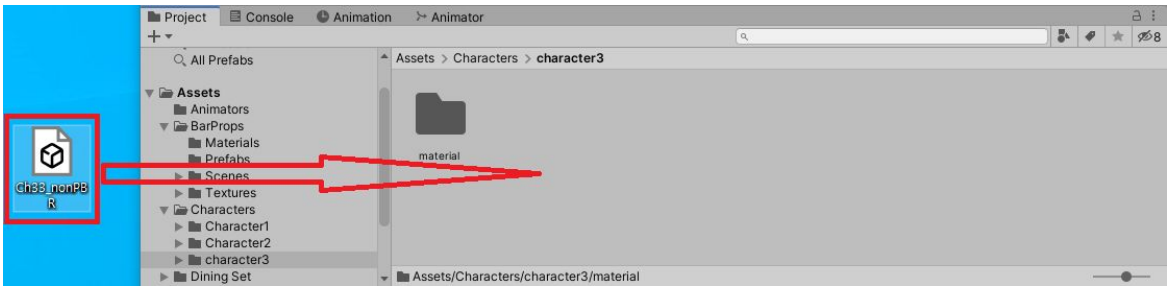
7. Now, open the Unity
8. From Project Menu go to **Characters** directory. Then create a new character folder (e.g charcter1, character2, …)



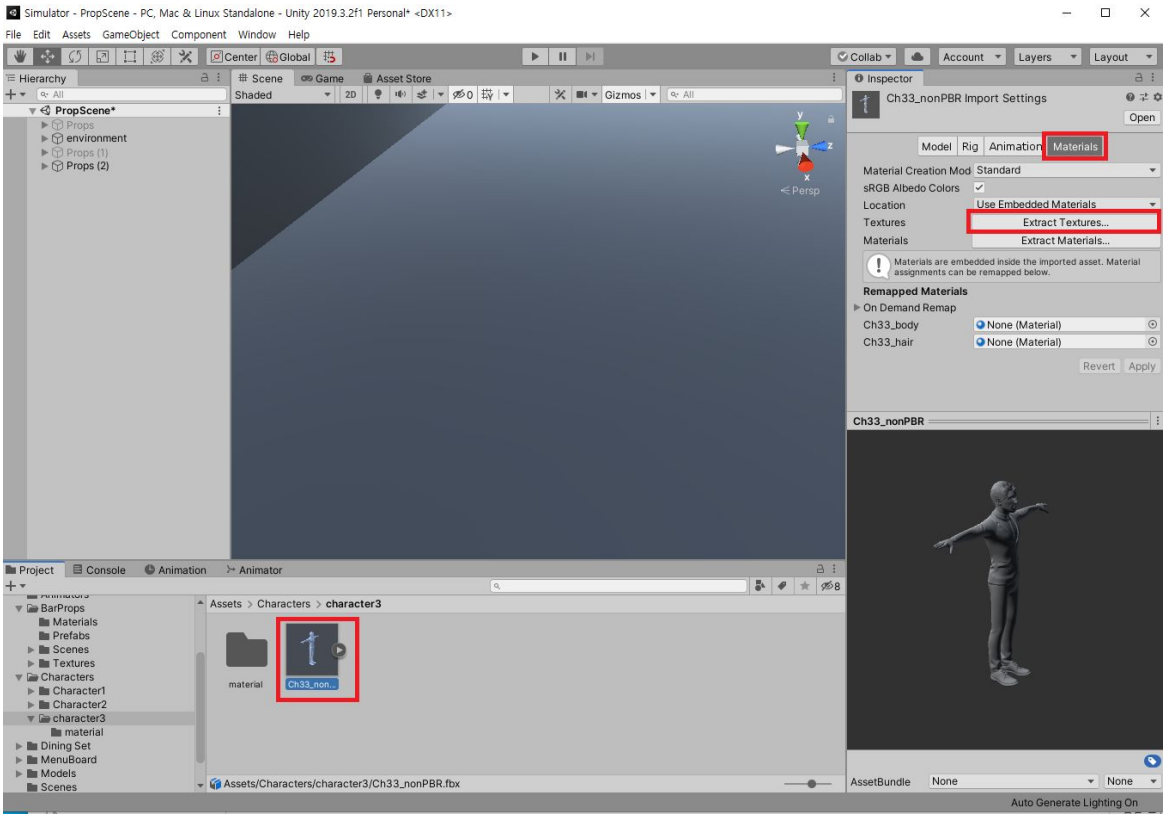


9. Go to the character folder, then create material folder.

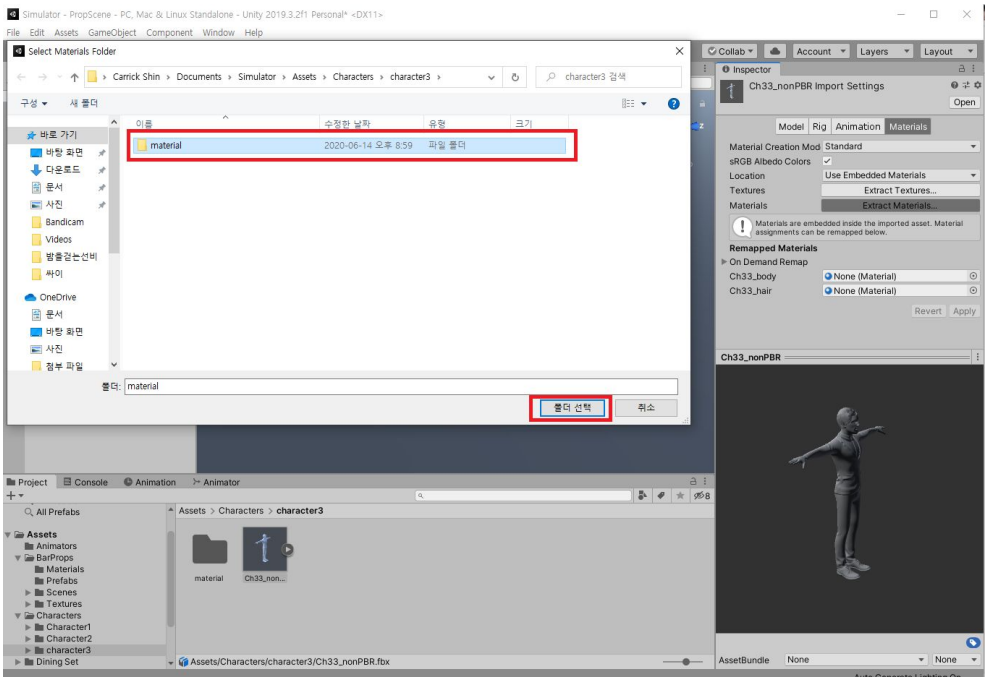
10. Drag & Drop fbx file you downloaded from Adobe Mixamo



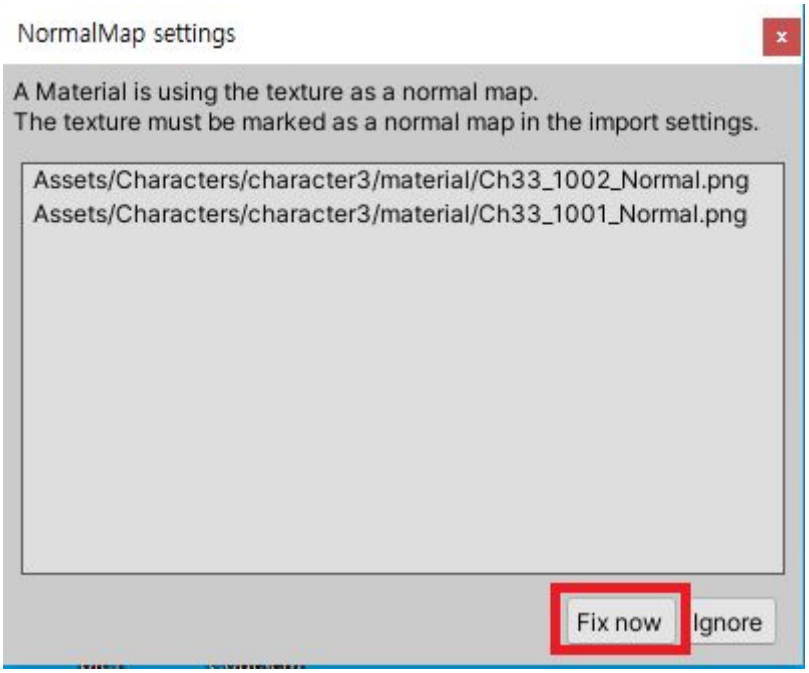
11. Click the fbx file. Then, on the **Inspector tab** on the right side, click **Material**, then click **Extract Textures**



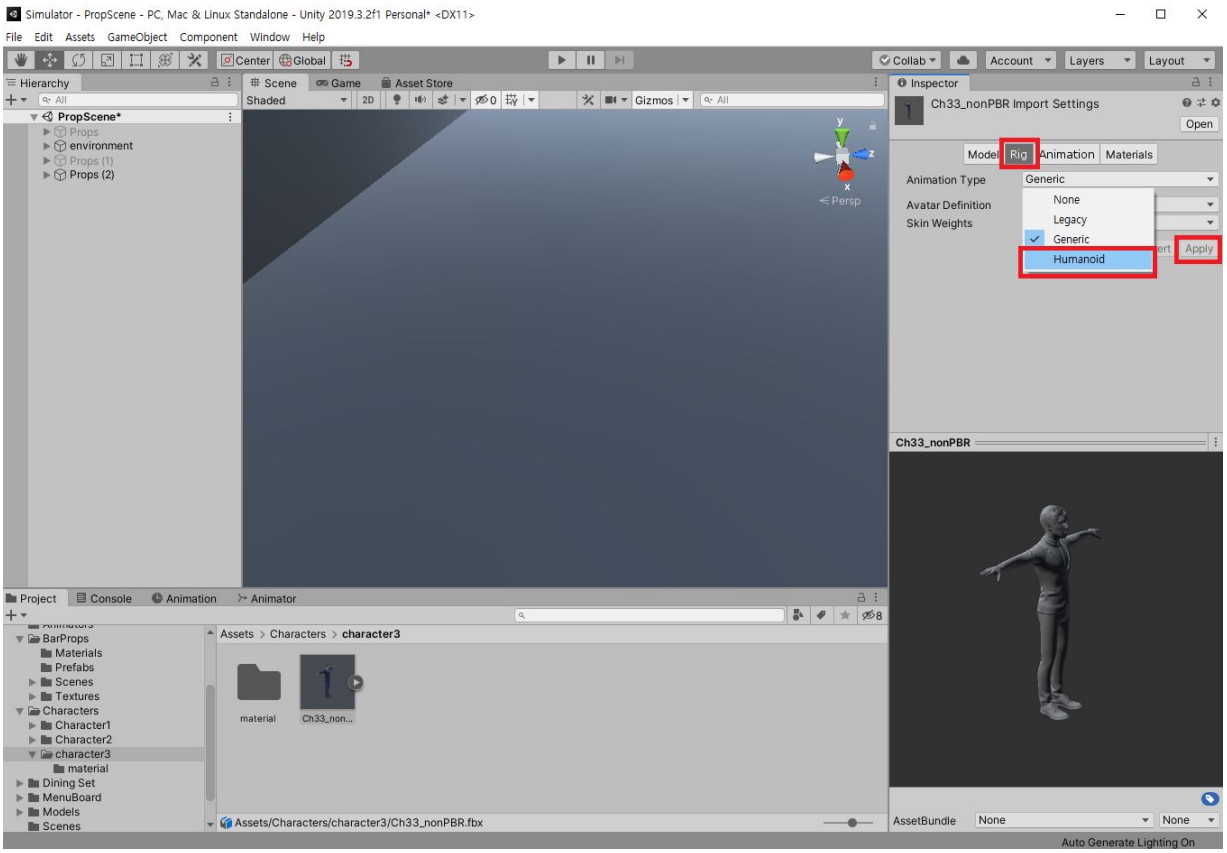
12. Select the **material folder** you created at step 9, then click **OK**



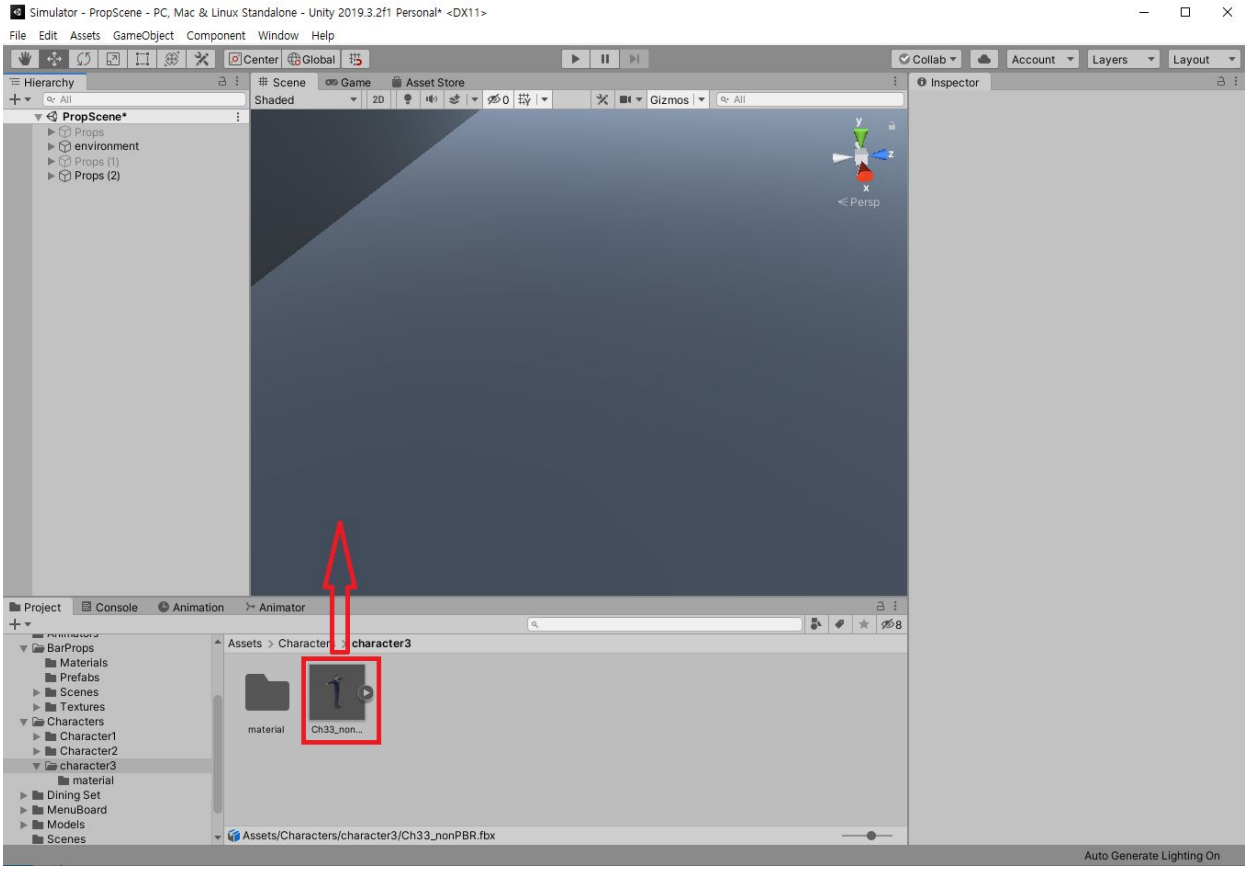
13. Click **Fix now** button



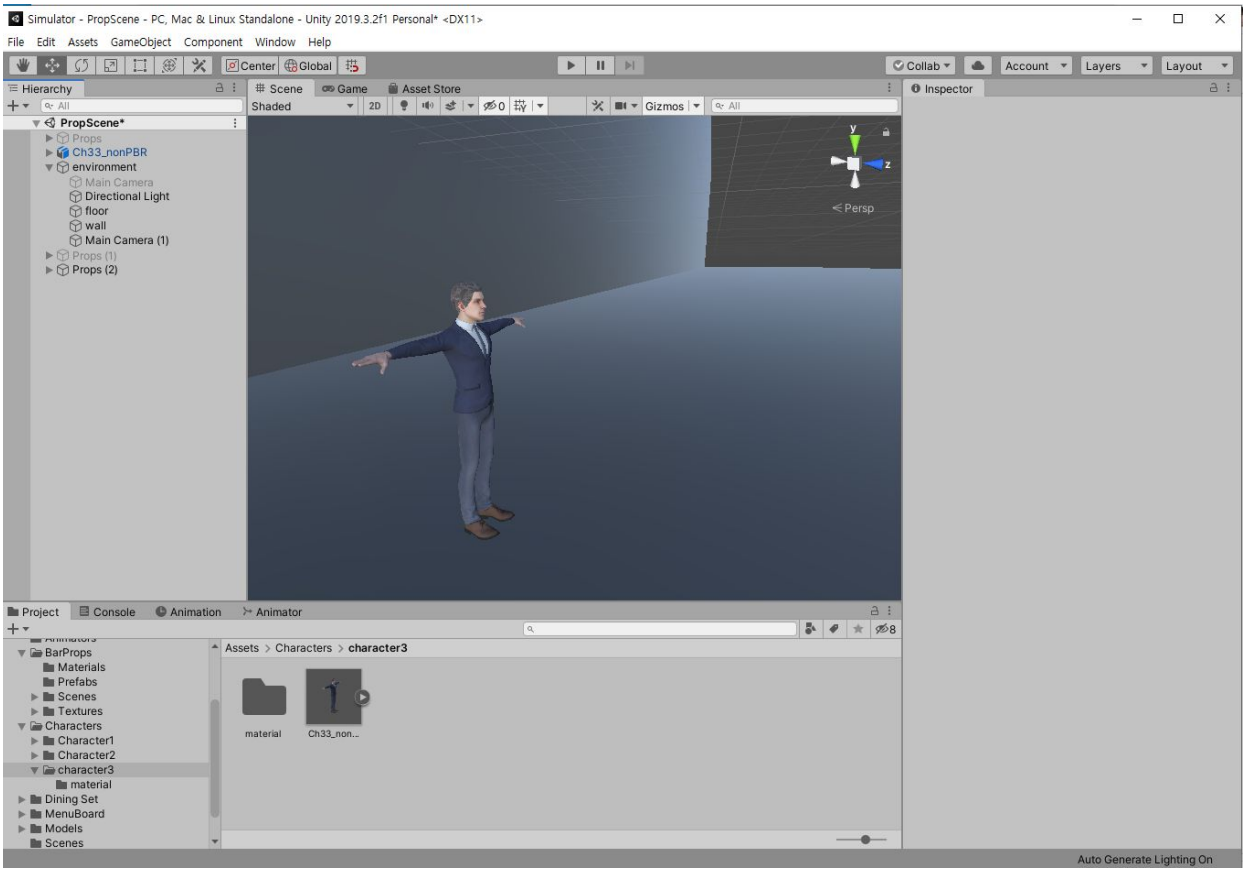
14. Click **Rig** button. From **Animation Type** menu, select **Humanoid,** then click **Apply**.



15. Now character is ready for use. Drag and drop the character into the scene.



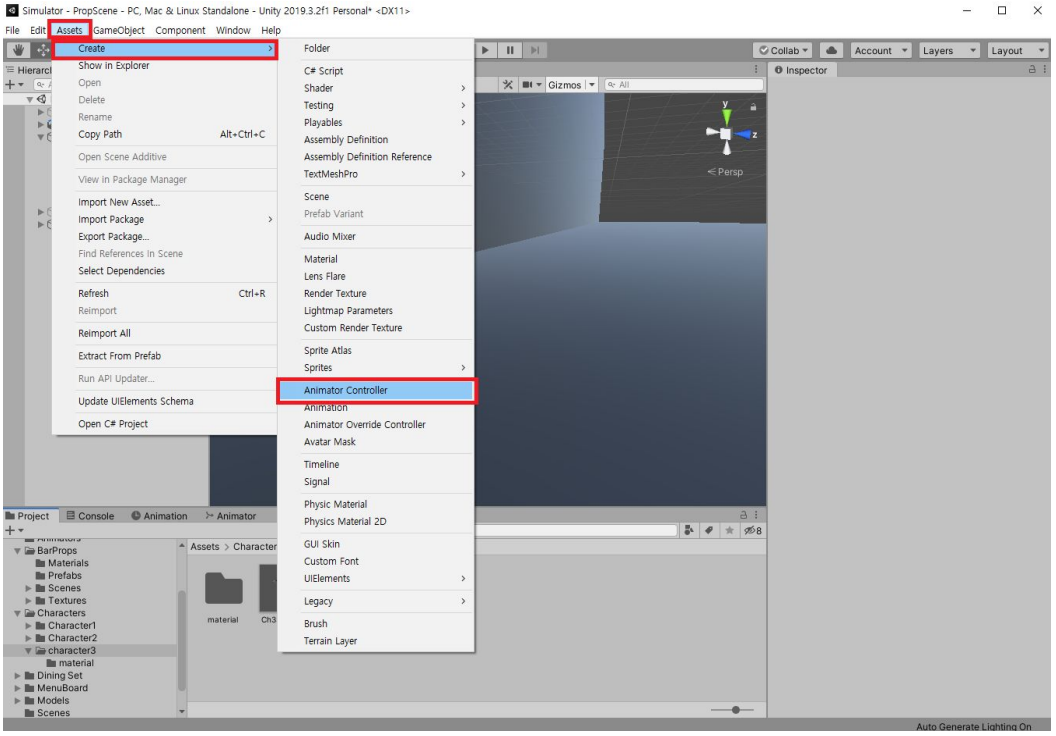
16. Character is in the scene now



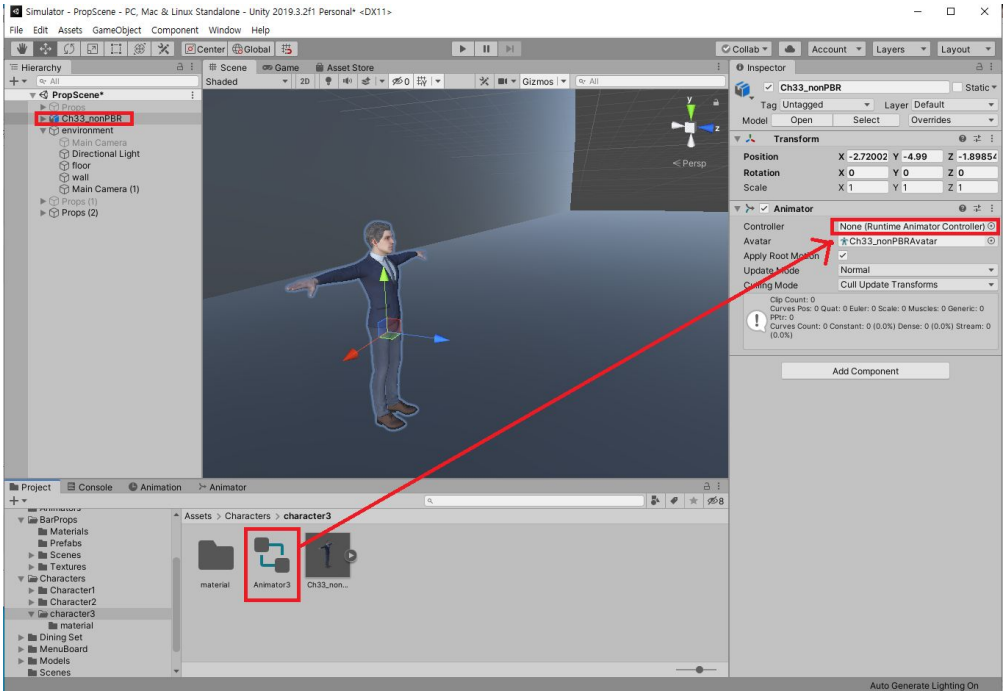
3. Apply Animations to Characters

- This step assumes that you have 3D character animation that you want to use in the Unity. Adobe Mixamo provides some built-in animations for each character, but you have to use a **Blender** if you want to make/use a custom 3D animation. There are many Blender tutorial on youtube how to make custom 3D animation, so I will not describe how to create 3D animation by a **Blender** at this document.

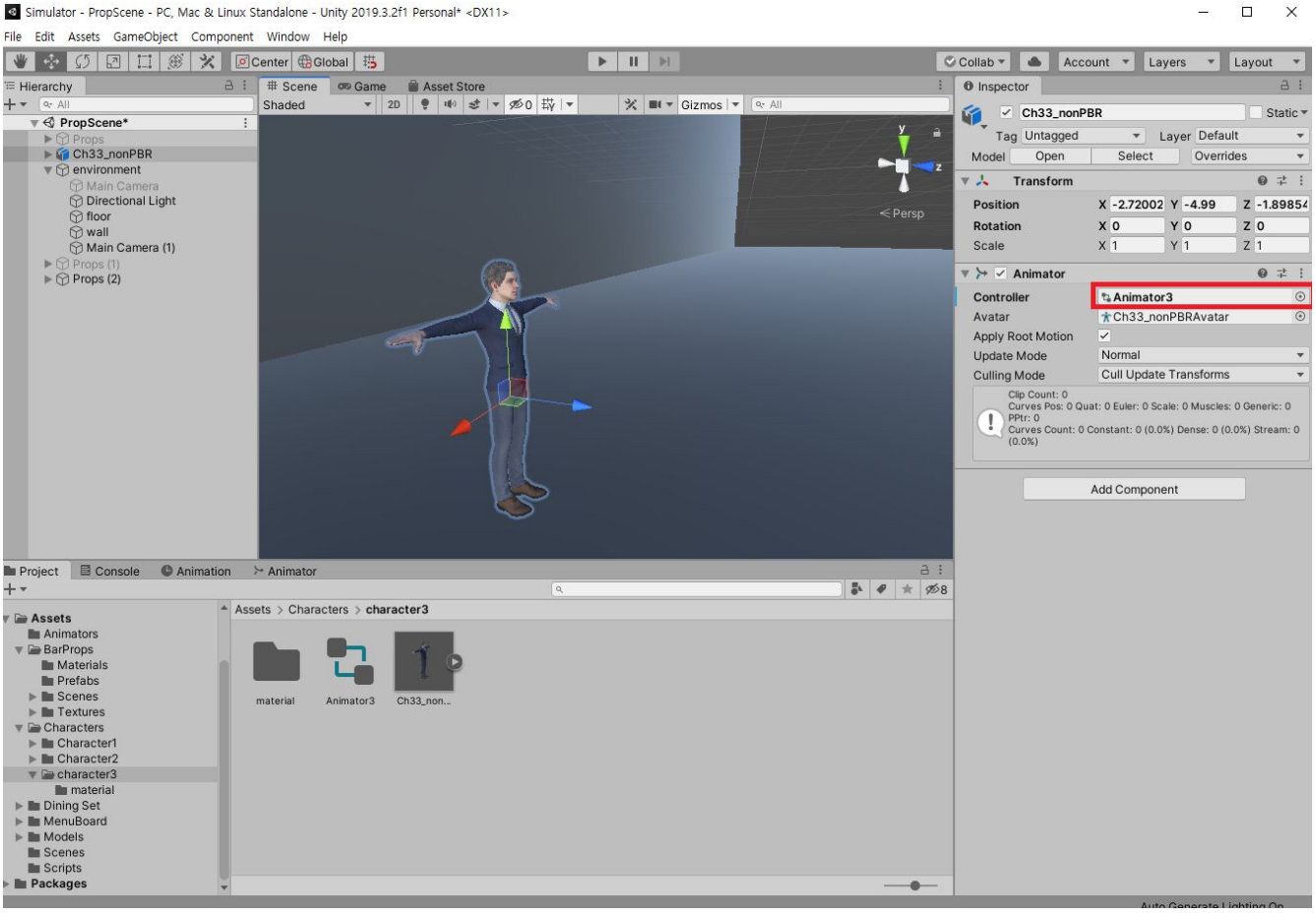
1. Click Assets - Create - Animator Controller



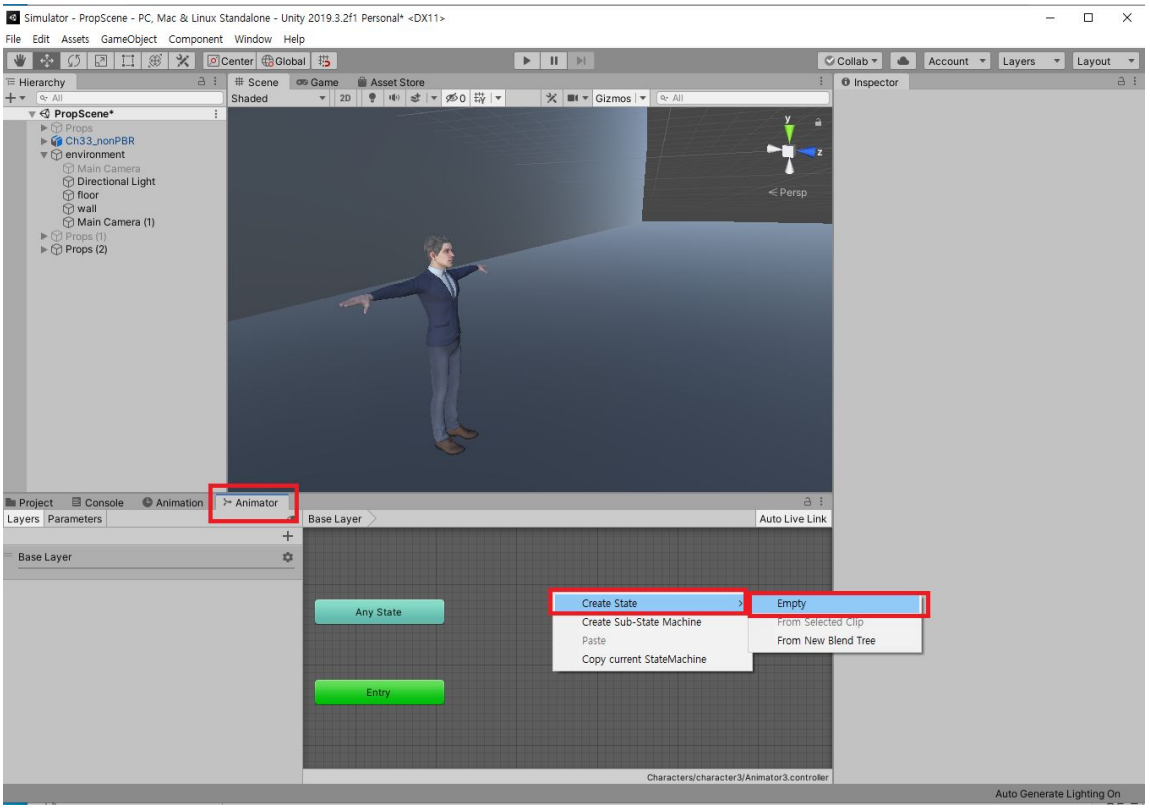
2. Click the 3D character in the scene. Then drag & drop **animator** you created at step 1 to **Animator - Controller**



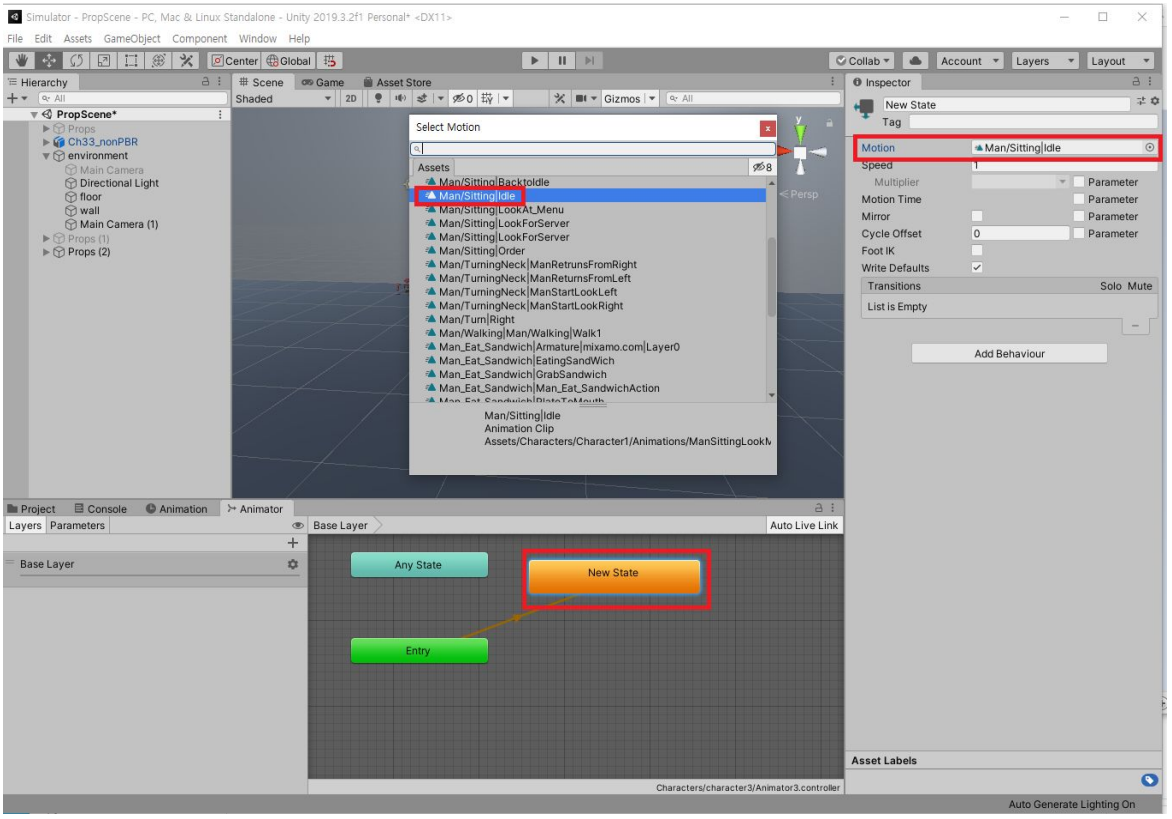
3. Double click the animator.



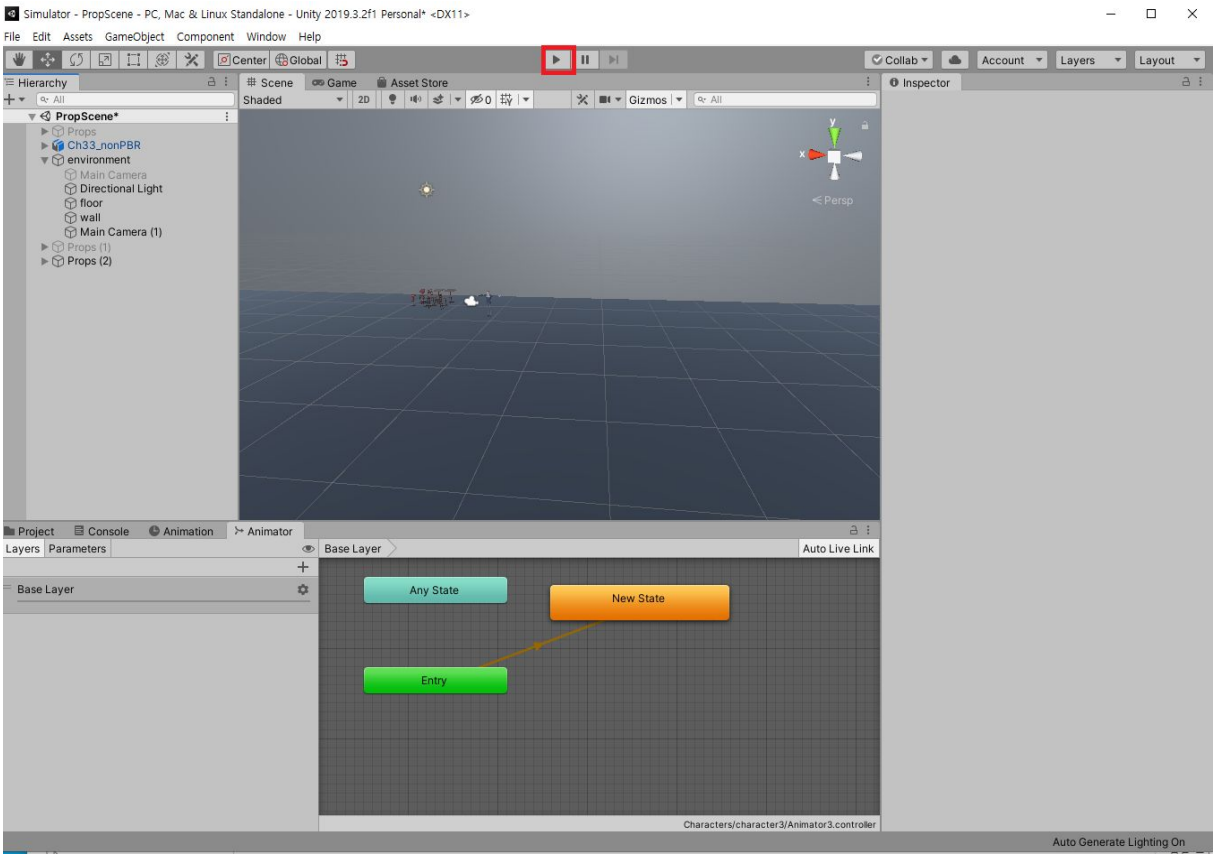
4. On the Animator Tab, right click the Base Layer, then click **Create State - Empty.** Then, **new state** icon will be generated.



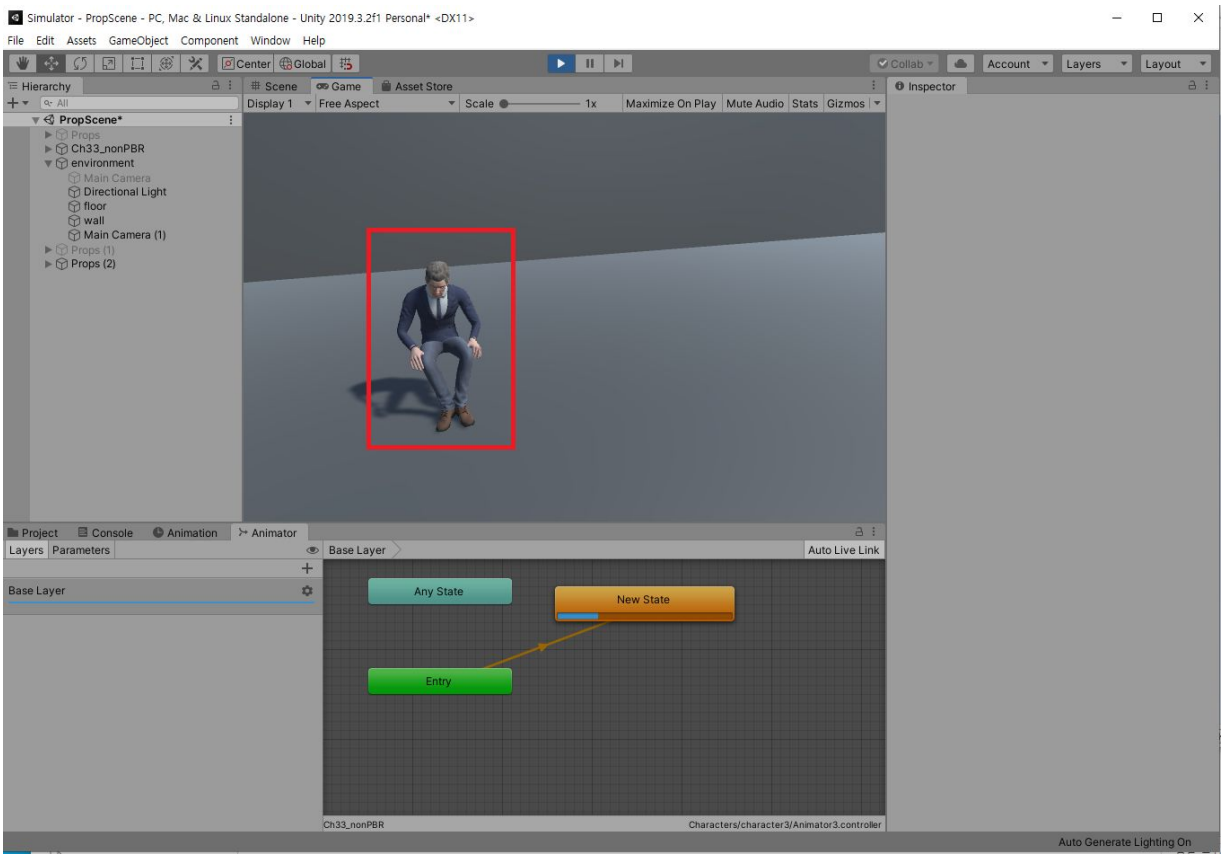
5. Click the orange-colored **new state** from the Animator Tab. From the **Inspector** tab, at the **Motion** section, click the icon on the right. Then, **Select Motion** will show all animations you have for this Unity project. Select the animation you want to use. For this document, I will select the animation that a character is sitting.



6. Run the Unity



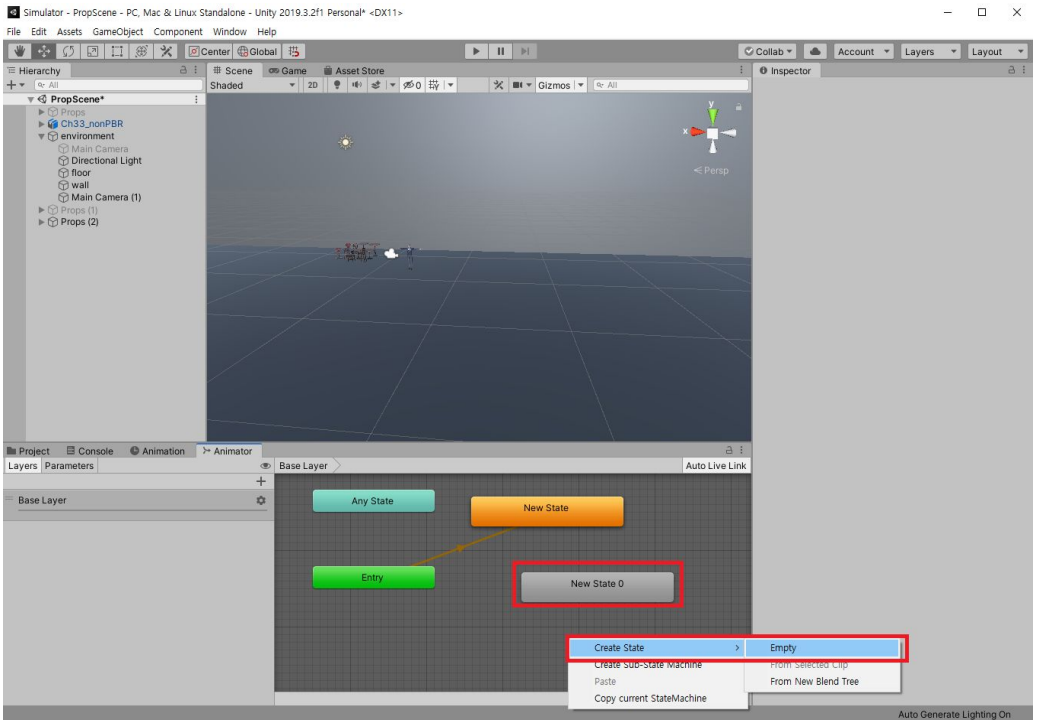
7. Then, we can see our 3D character is on sitting animation



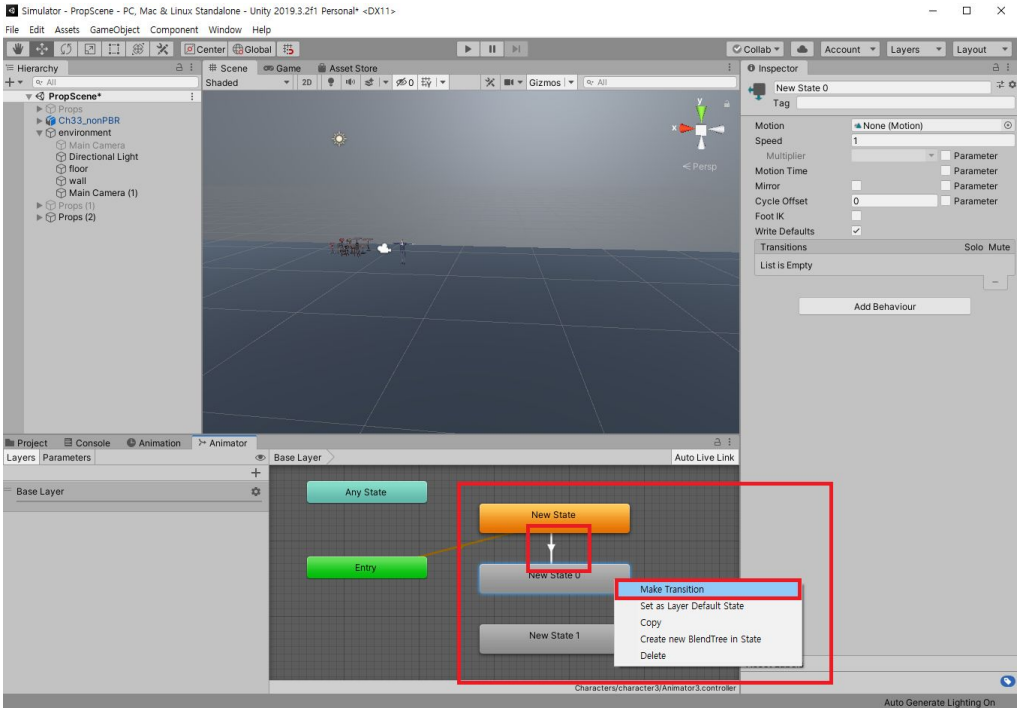
4. Apply a set of animations (more than 1) to Character

- Making 3D character to do just one animation (behavior) at a time is not enough to make a simulation in the restaurant. This chapter illustrates how to make a 3D character to do a set of animations

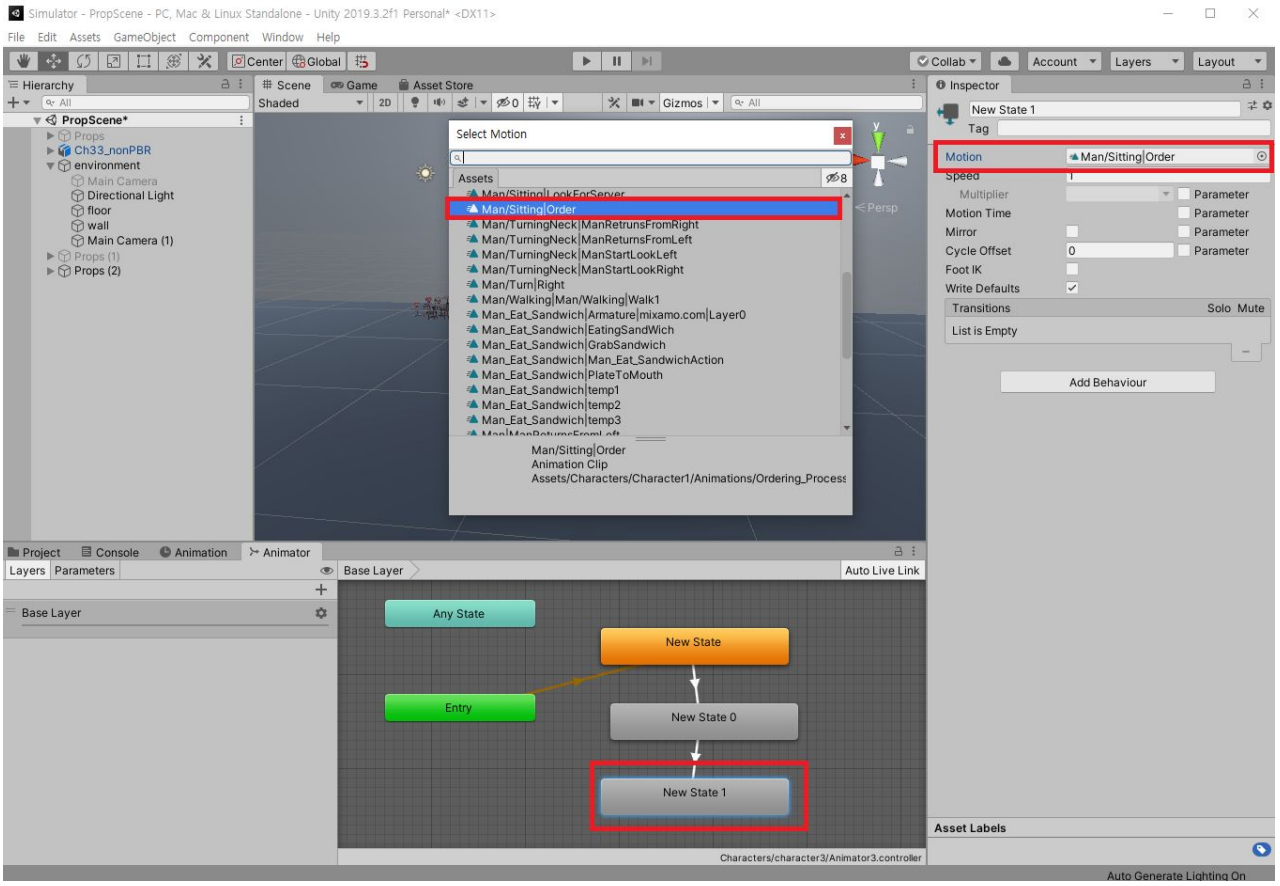
1. From Animator Tab, create 2 more new states by **Right Click - Create State - Empty**



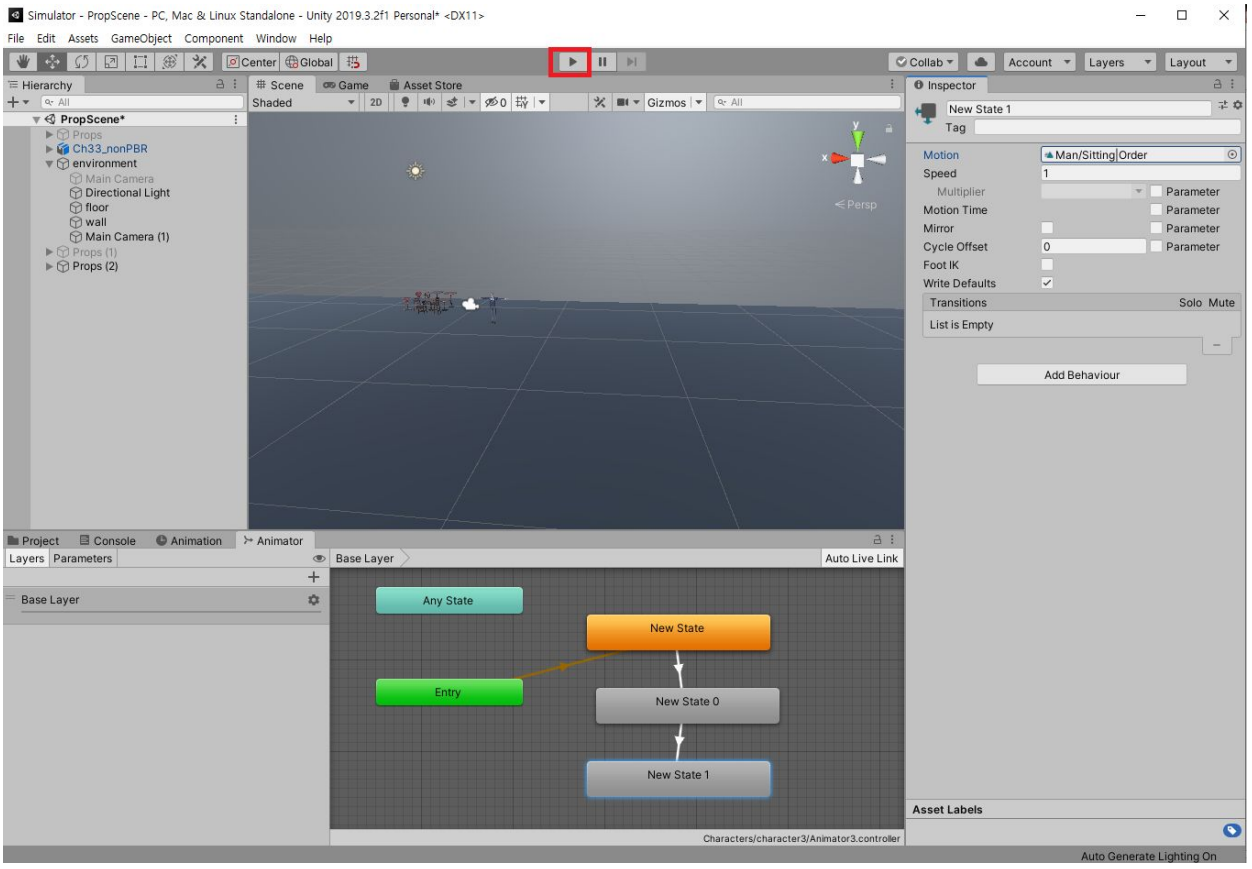
2. New State (One with orange color) is always the initial state. Right Click the **New State** - Click **Make Transition** - Click **New State 0**. Again, Right Click the **New State 0** - Click **Make Transition** - Click **New State 1**. Now the animator will operate in **New State** - **New State 0** - **New State 1** order.



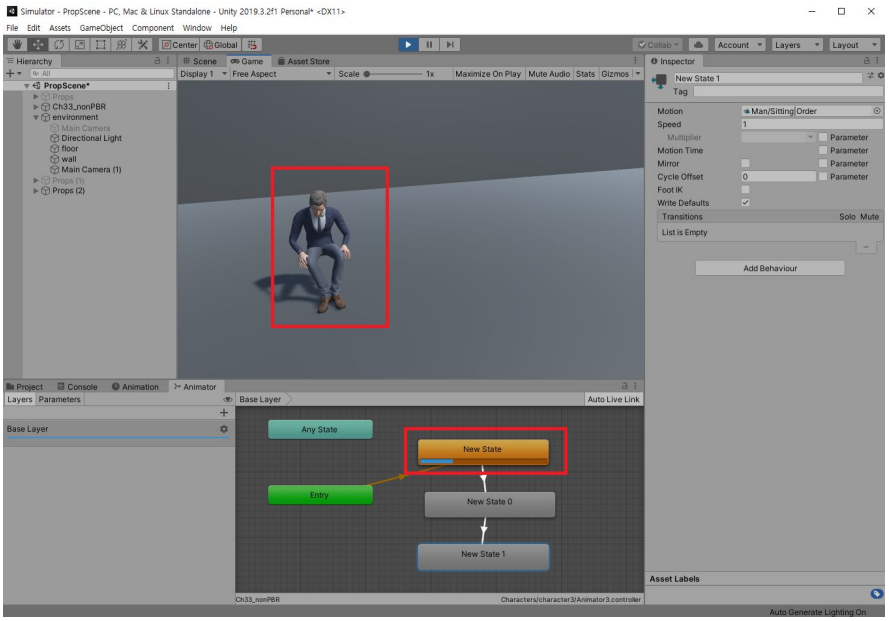
3. **New State 0** and **New State 1** contains no animation yet. Refer to the **Step 5** from Chapter 3, then select the animation for each state. For this example, I will select **LookingForServer** for **New State 0** and **Ordering** for **New State 1**.

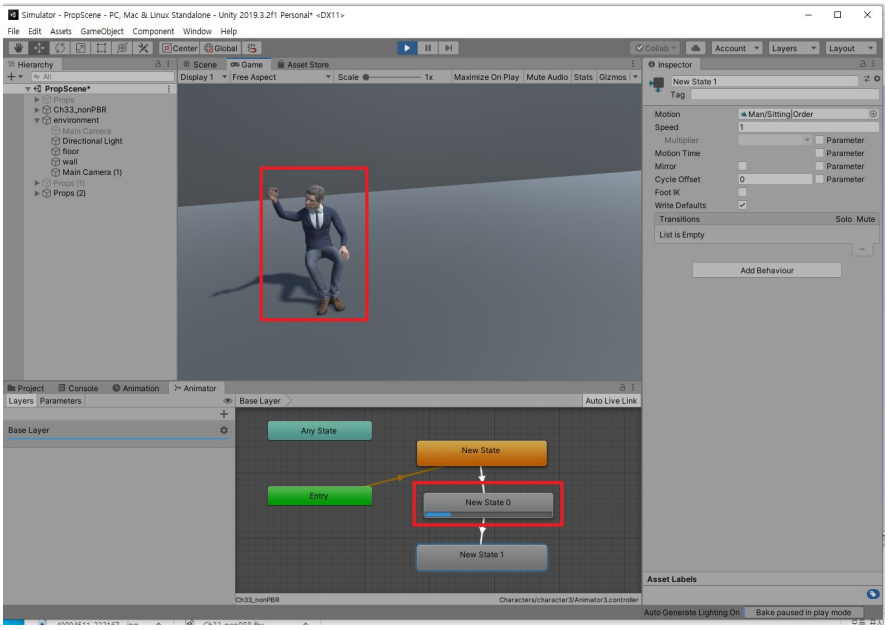


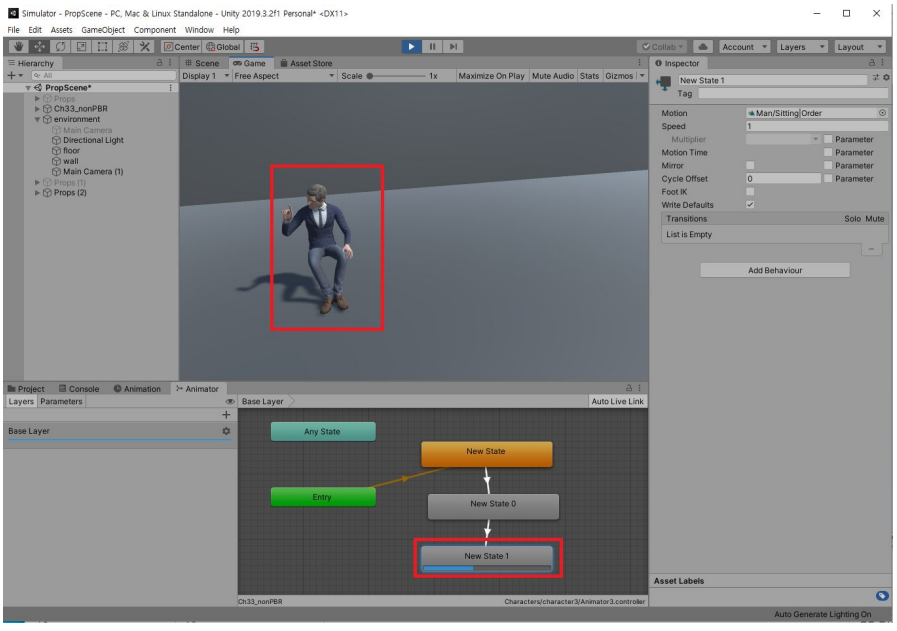
4. Run the Unity



5. Now we can see that 3D character changes animation as state is changed.

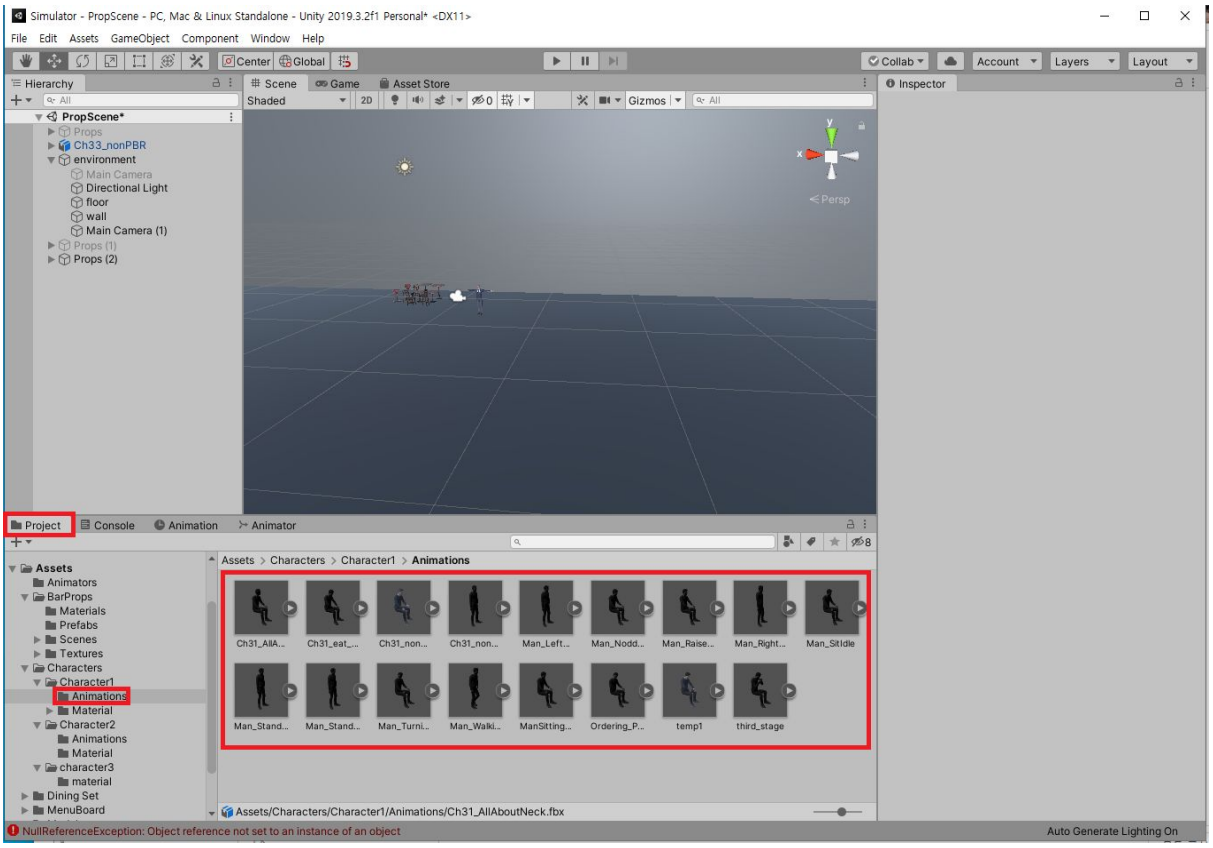




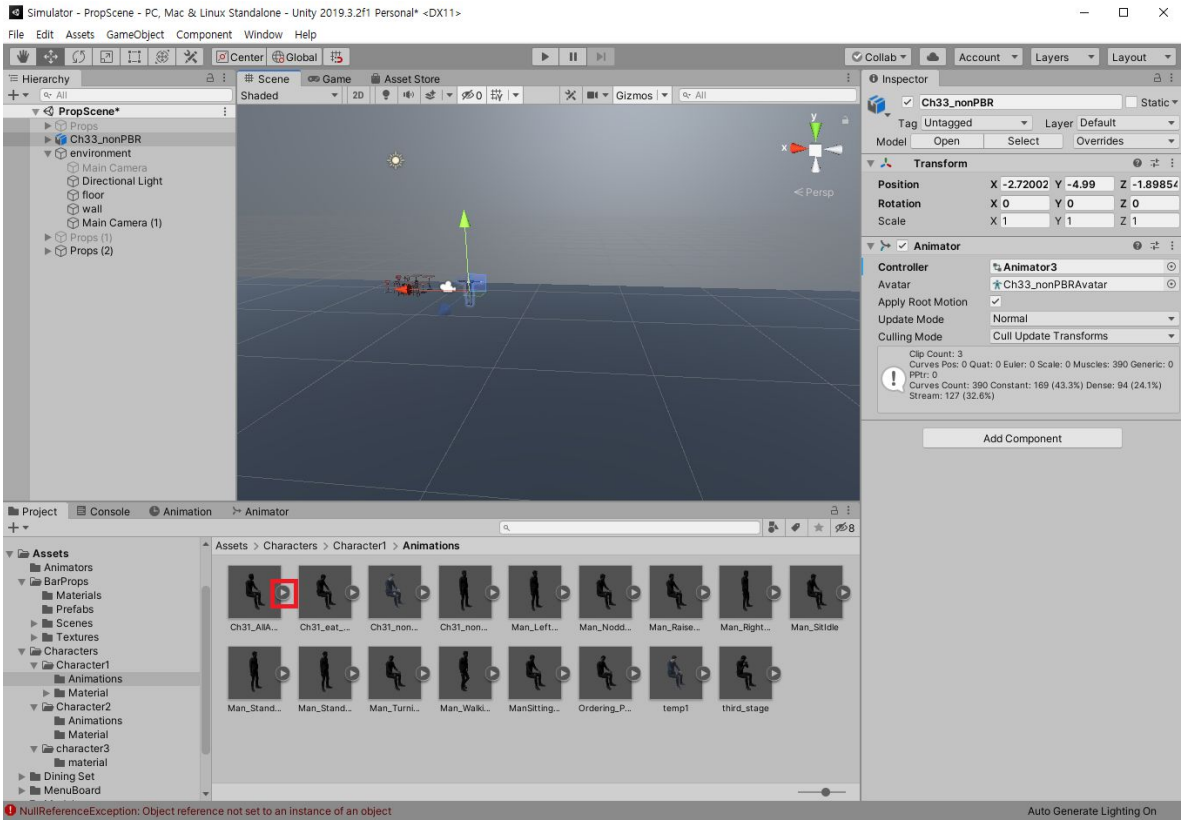


5. How to see the list of animations

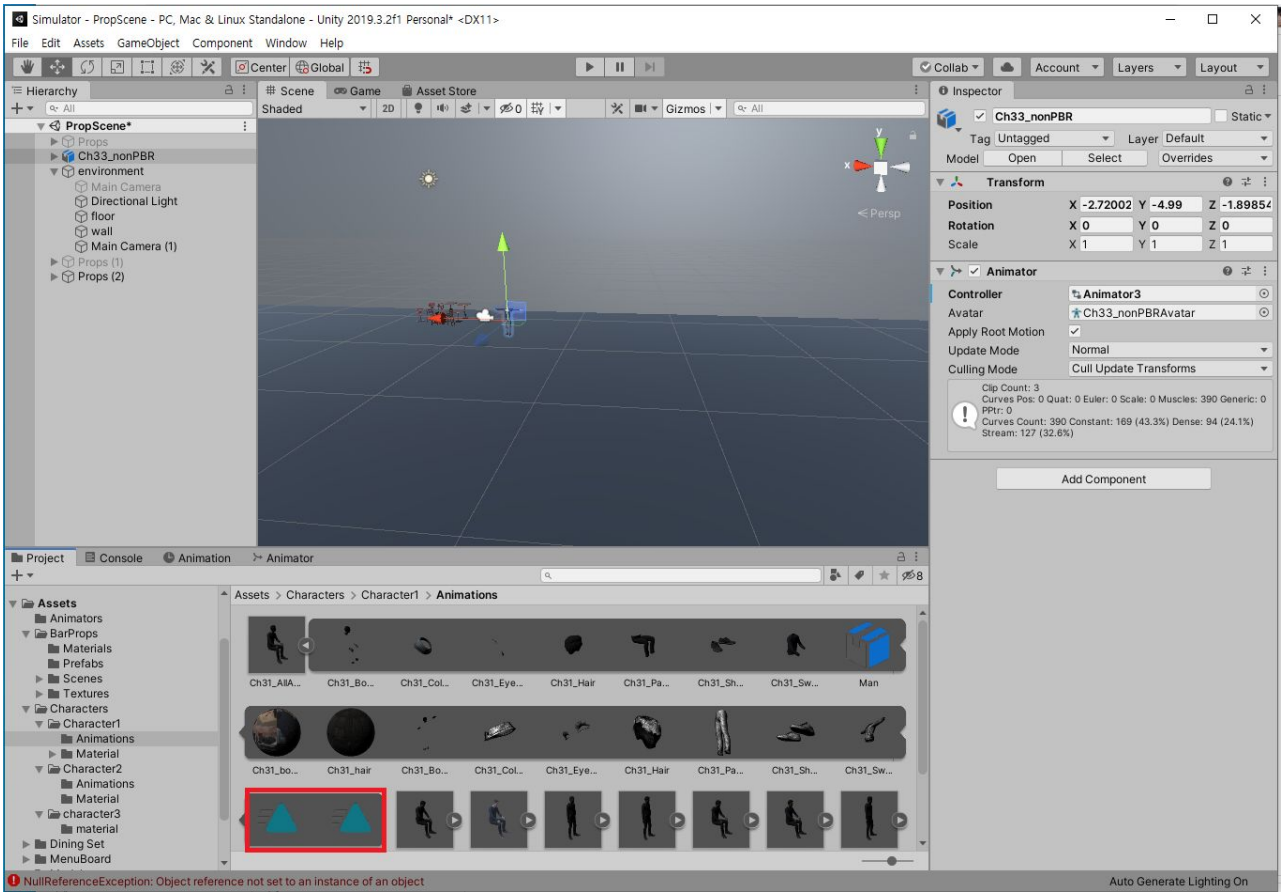
1. From the **Project** tab, click the **Characters** Folder. In **Characters** folder, there are different **character** folders. Each **character** folder, contains **Animations** folder. Click that folder.



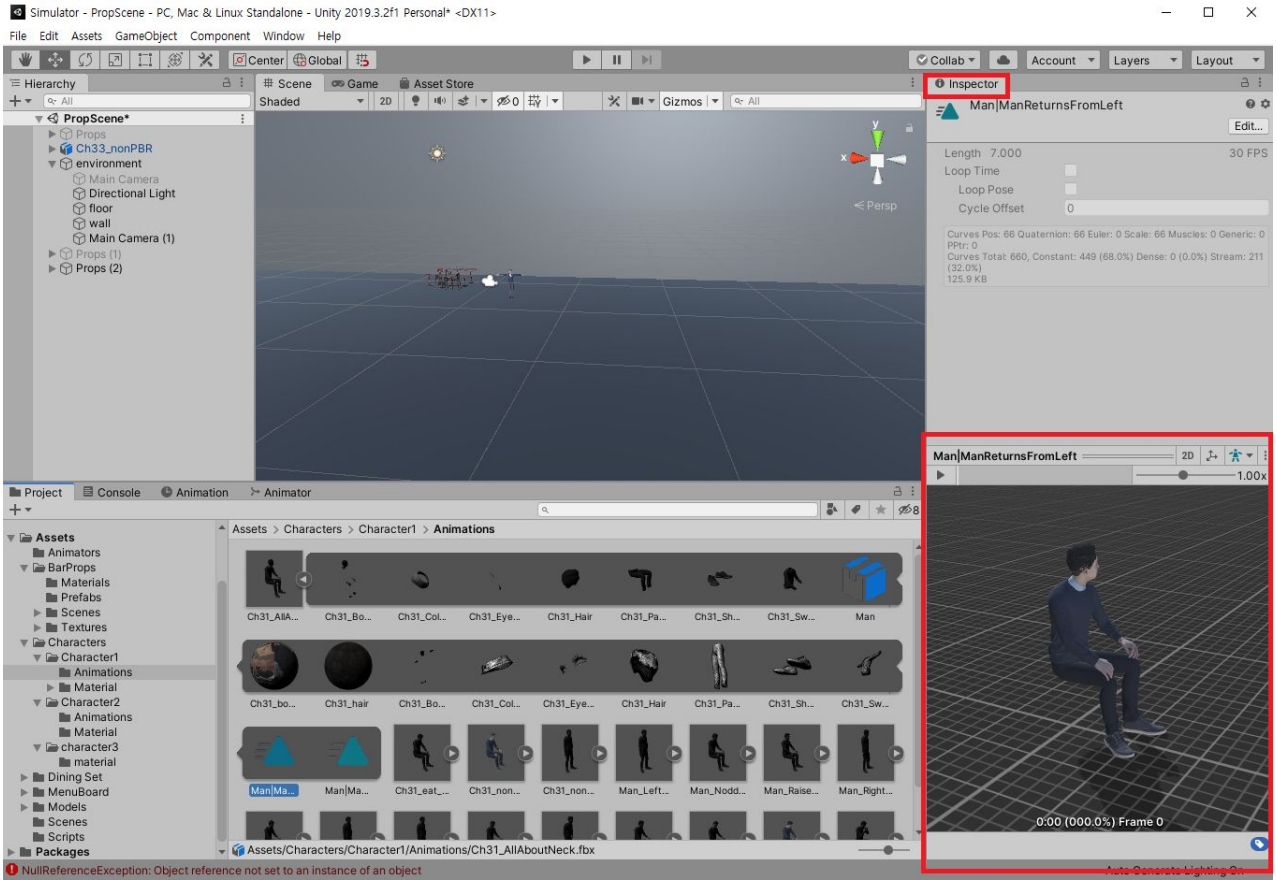
2. Click any animation file. Each file contains 1 to many animations.



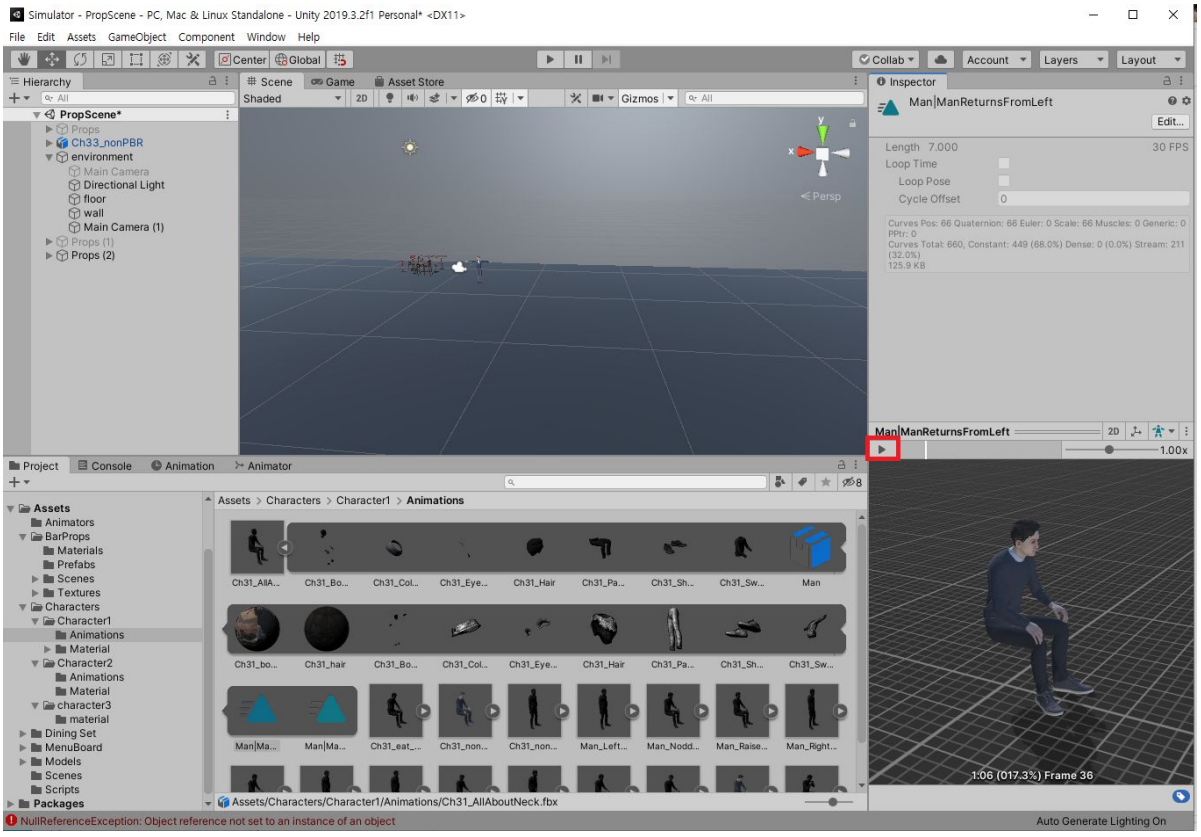
3. Icons with blue-green colored triangles are animation files. Click any of them to see its animation.



4. If you click the animation file, **Inspector** tab will show animation information.



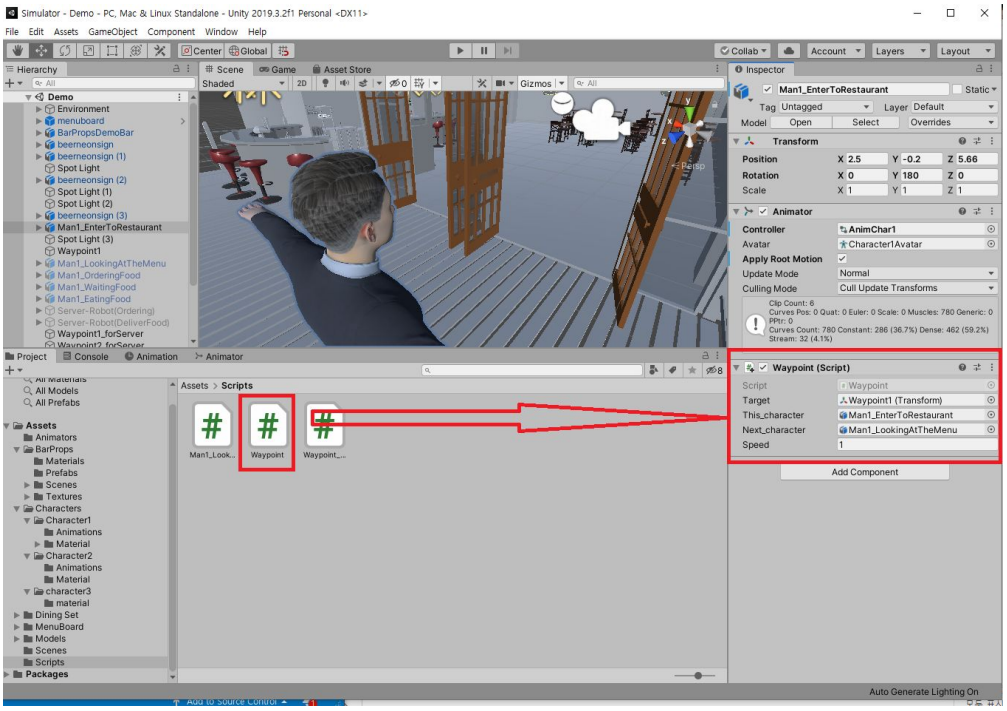
5. Click the Play button. Then, it will play the animation.



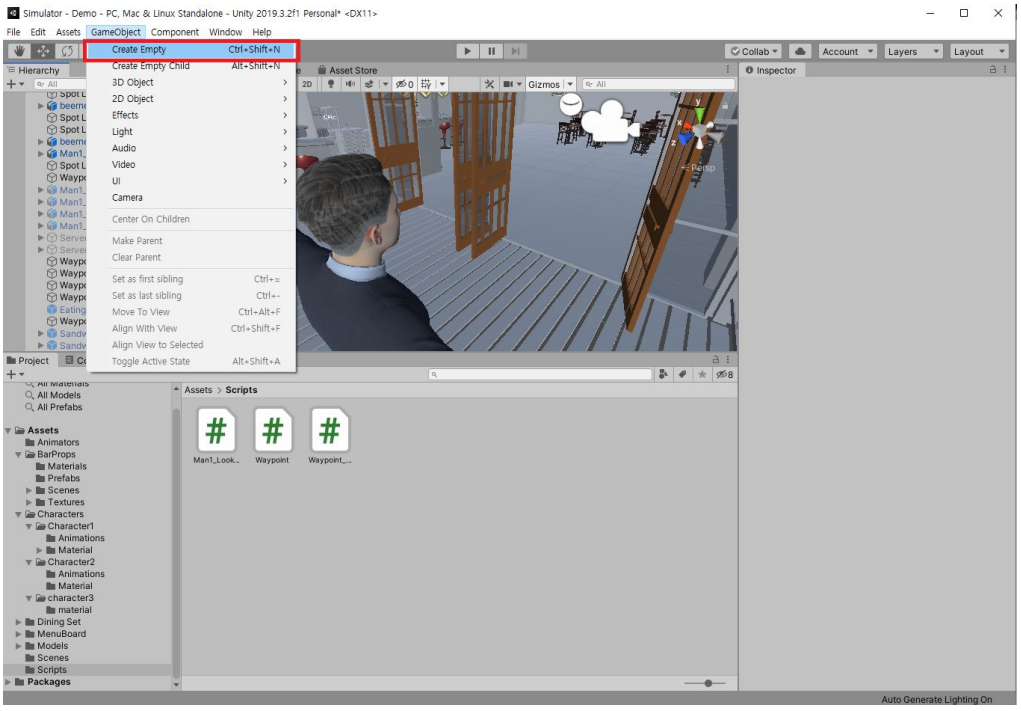
6. Make a character move to a designated destination

- This technique is usually used when 3D character is moving to a certain location. The base code for this technique is saved in the **Waypoint.cs** file at **Assets/Scripts** directory. For each character, when this technique is needed, refer to **Waypoint.cs** file, then properly modify and use it.

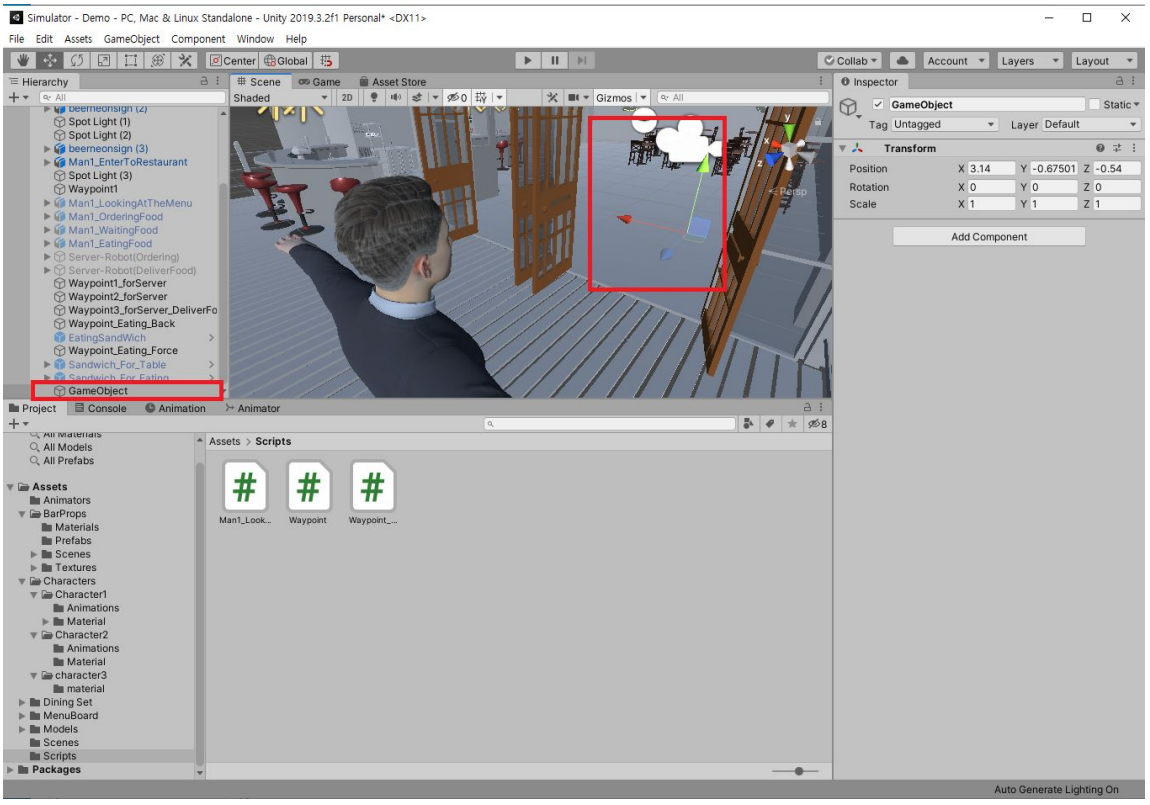
1. Drag and drop **Waypoint.cs** script at the character that you want to make a movement. Ignore **This_Character** and **Next_Character** sections for now, they will be explained at Chapter 8. Important sections are **Target** and **Speed**.



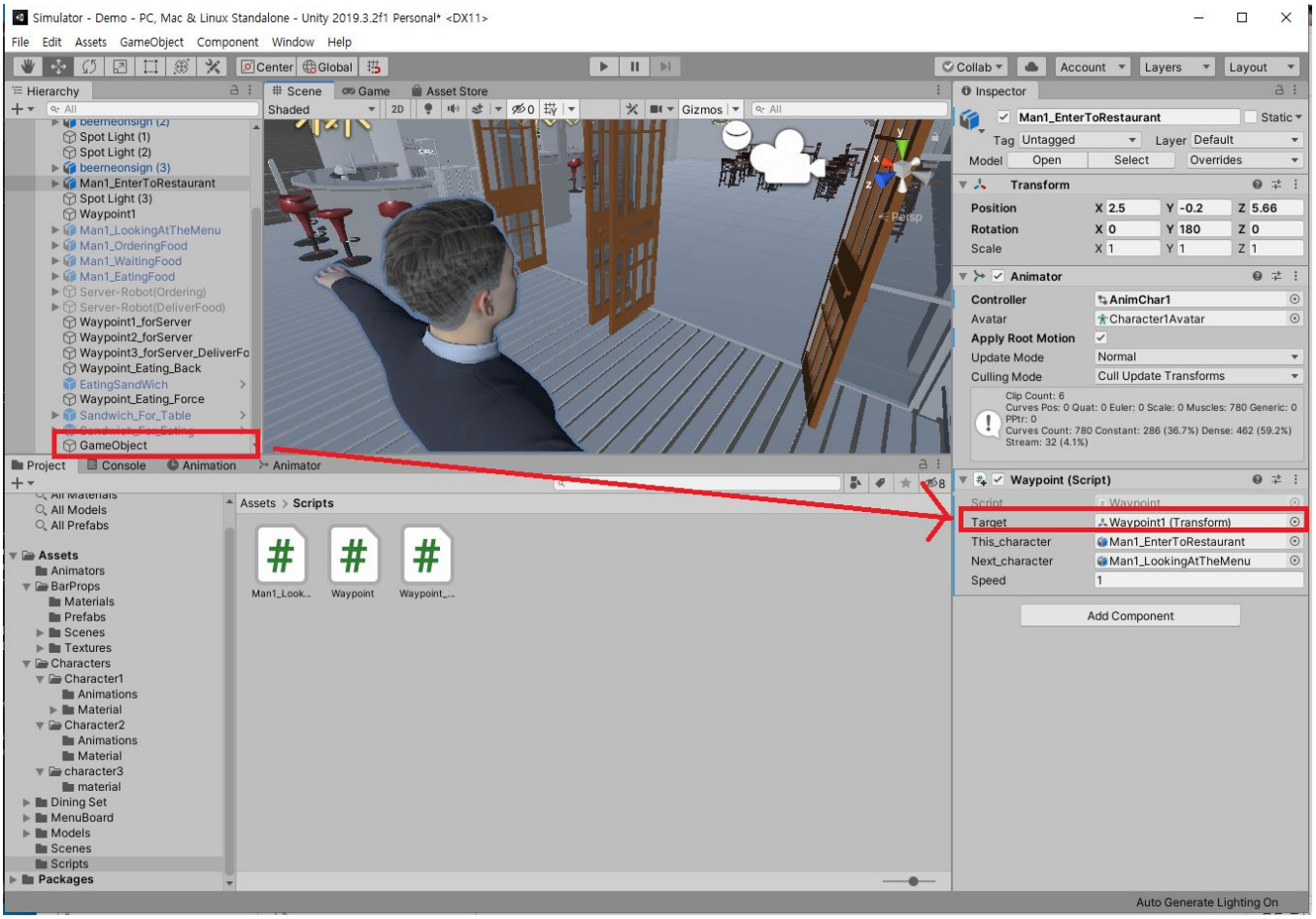
2. From the menu bar, click **GameObject - Empty**. Then, non-visual gameObject will appear.



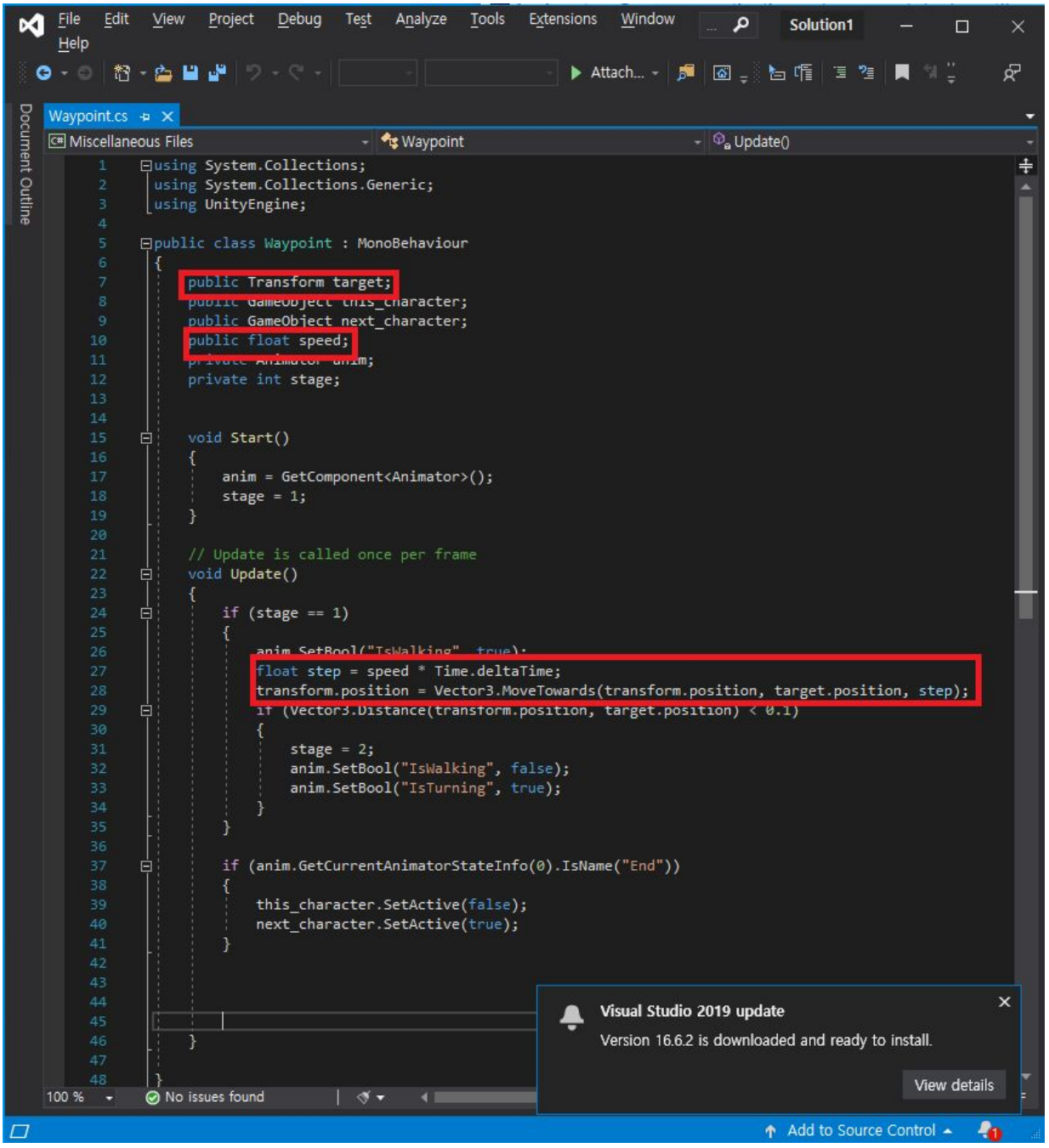
3. Place generated empty gameObject to the location that you want your 3D character arrive.



4. Drag & Drop this empty gameObject to 3D character's **Target** section.



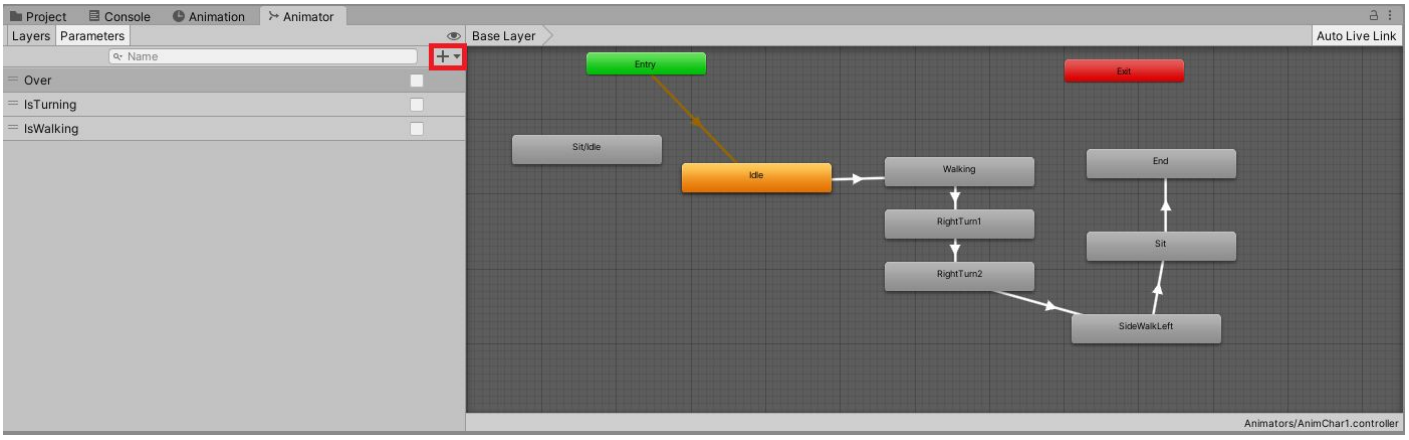
5. If you see the **Waypoint.cs** file, you can line 27 and line 28 make the character move to the empty GameObject's position.



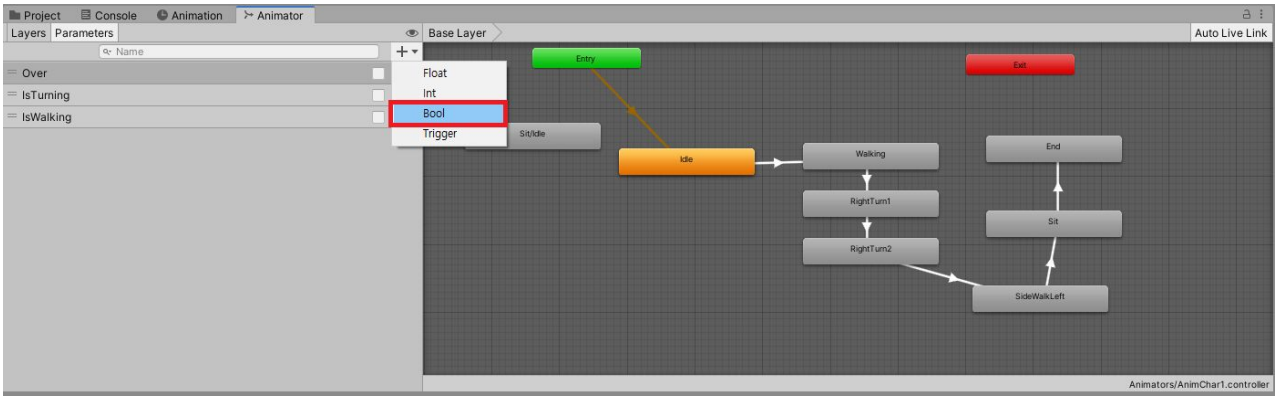
7. Conditional transition of animation

- Sometimes, animation needs to be repeated more than 1 time. For example, the character should repeat ‘walking’ animation until it arrives to the destination, and should not play other animation like ‘eating’ or ‘looking for the server’. This chapter introduce how to make a condition to change the animation.

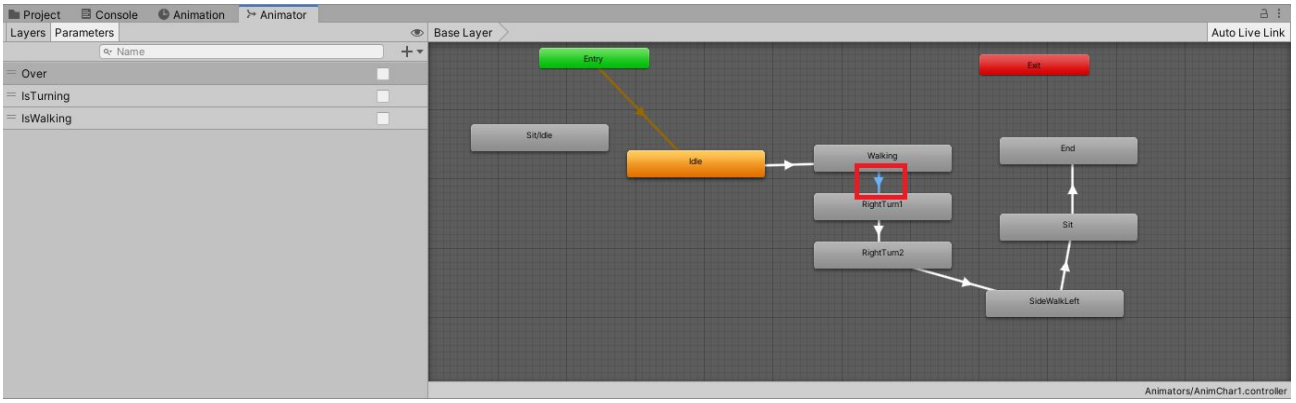
1. Image below describes Man’s action from entering the restaurant to sitting to the chair. Thus, animator should repeat Walking animation until the character arrives next to the chair. First of all, click the + button from **Animator - Parameters**



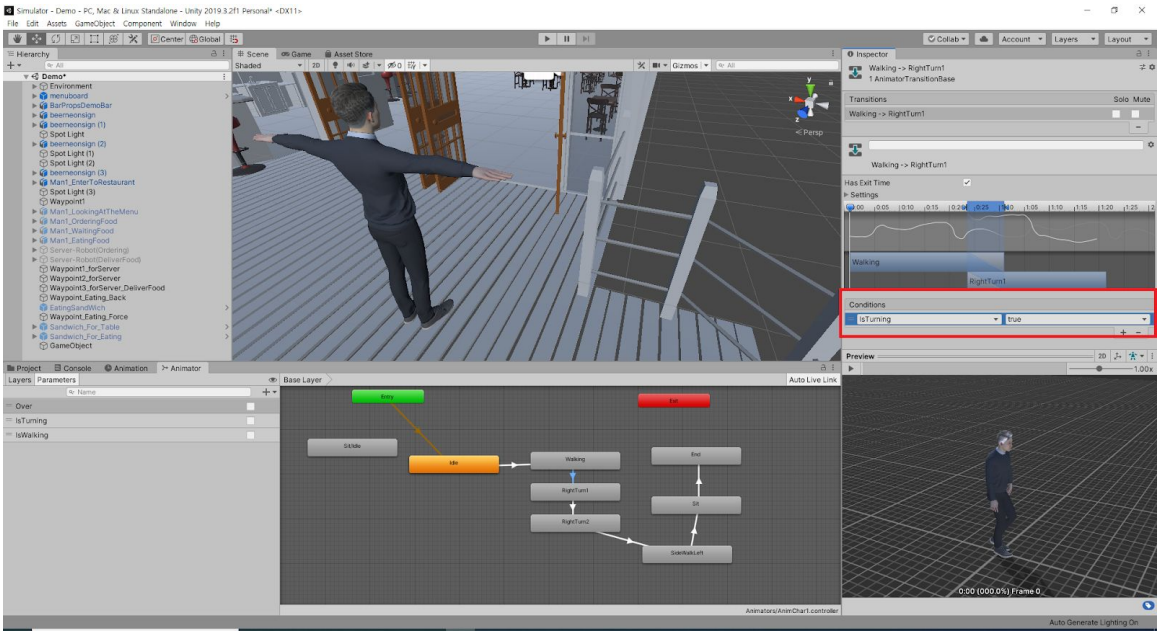
2. Click the **bool**, then change the name to **IsTurning**



3. From the **Base Layer** section, click the arrow between **Walking** and **RightTurn1**

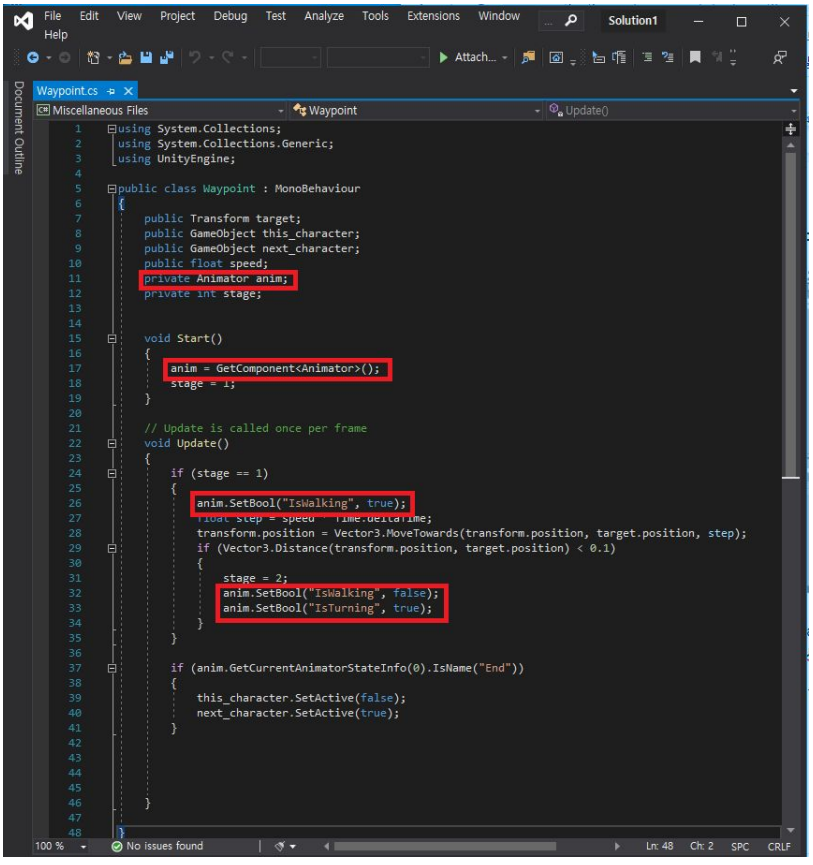


4. From **Inspector** tab's condition section, click +. Then, a new condition will be added. From a new condition, select **IsTurning** on left and **true** on right.

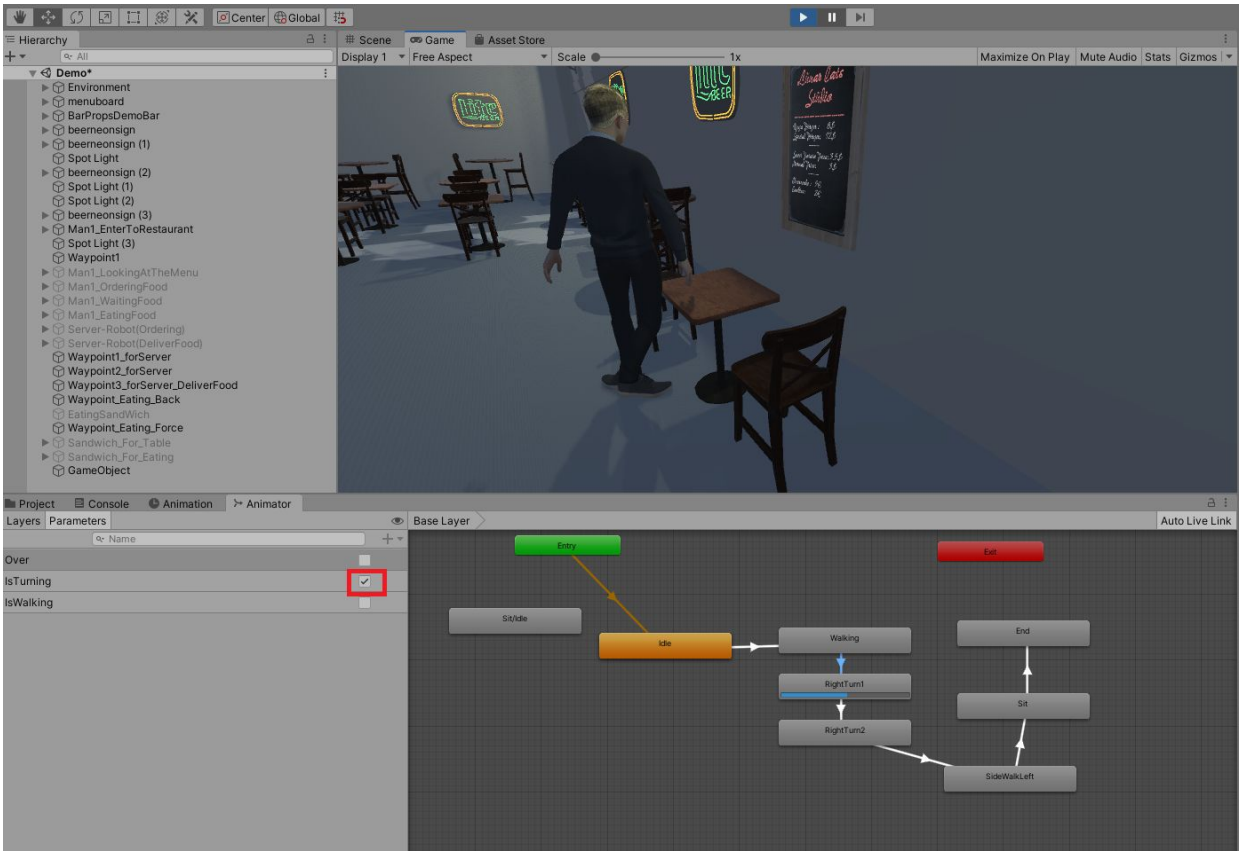


5. Now, the animation will make a transition from **Walking** to **RightTurn1** when **IsTurning** is true. By default **IsTurning** is set to False.

6. Now, let's see the Waypoint.cs again. We imported character's animator as *anim* at line 11 and line 17. We can change the parameter to either True or False by using anim.SetBool. At line 32 and 33, when character arrives very close to the chair, animator changes **IsTurning** parameter from False to True. (Also, IsWalking parameter False to True)

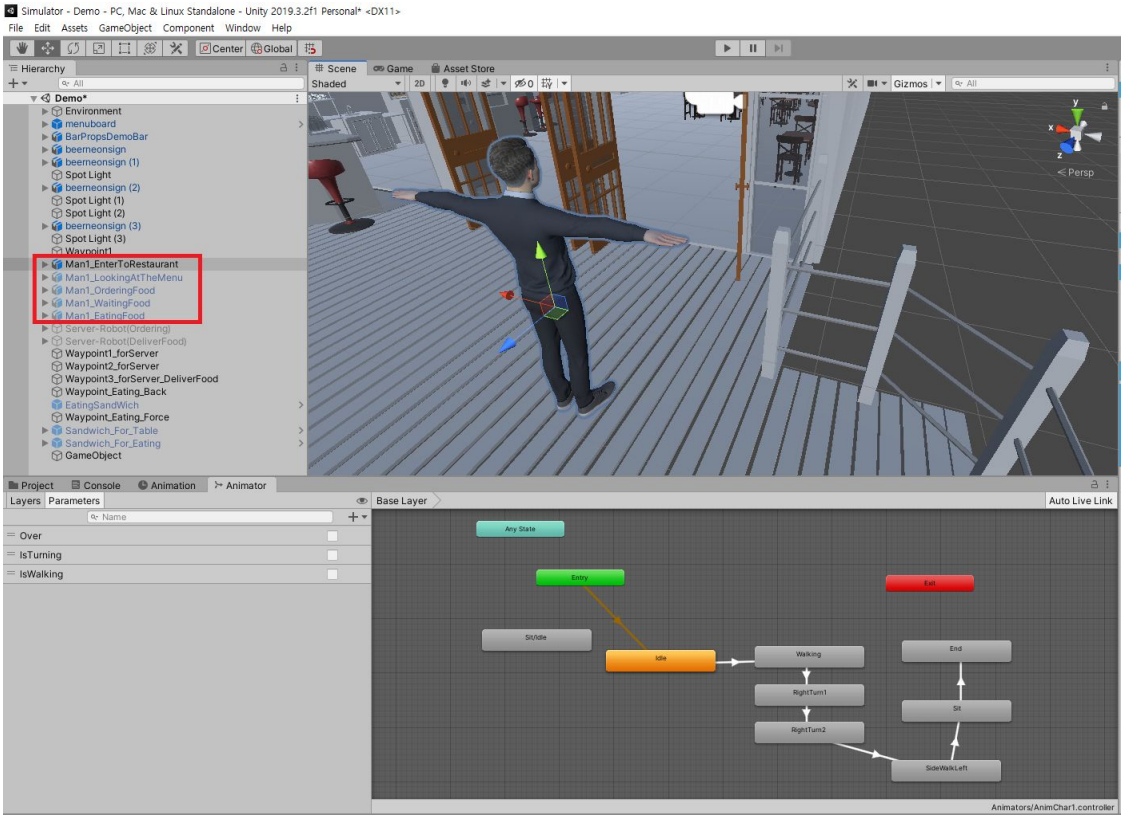


7. We can now see that the parameter **IsTurning** is set to True when the character arrives next to the chair, then starts to make a right turn.



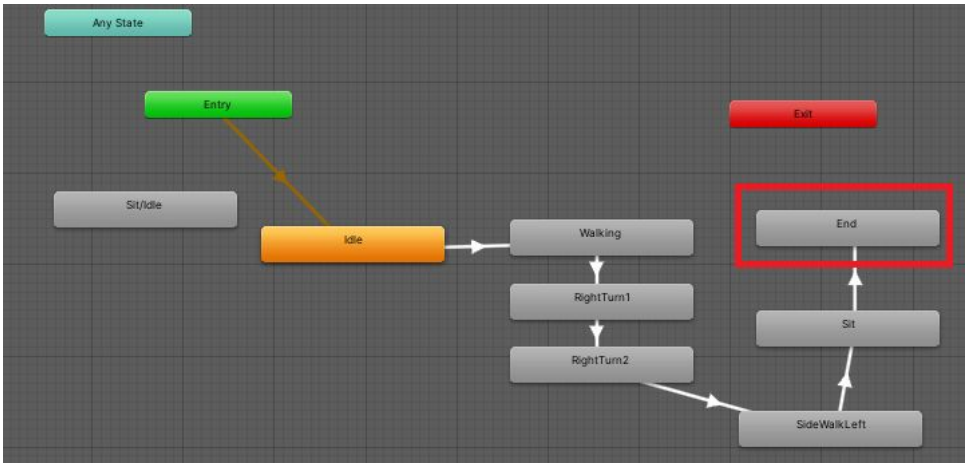
8. Generate Scenario

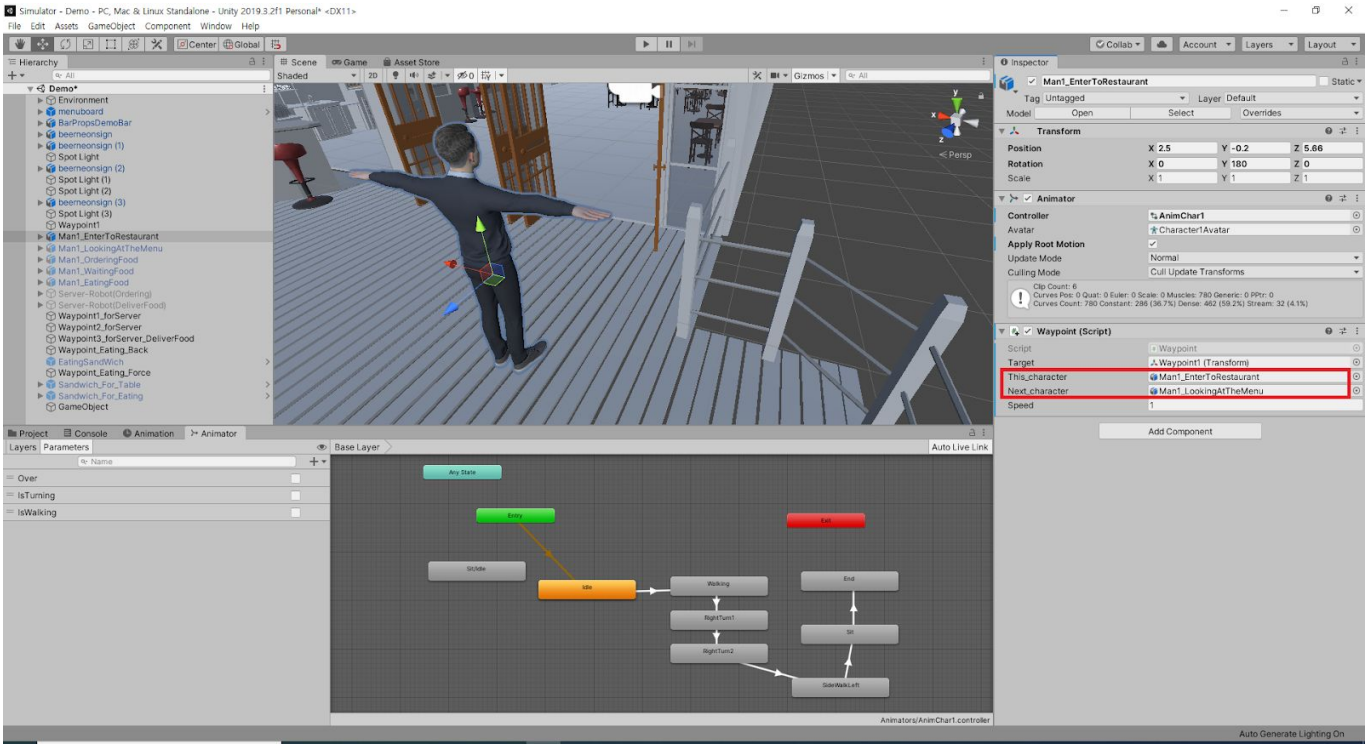
- Animator let the 3D character to play various animations sequentially. However, to avoid complexity, and make human-robot interaction or human-human interaction easier (will be explained at Chapter 9), it makes a scenario with several same characters.



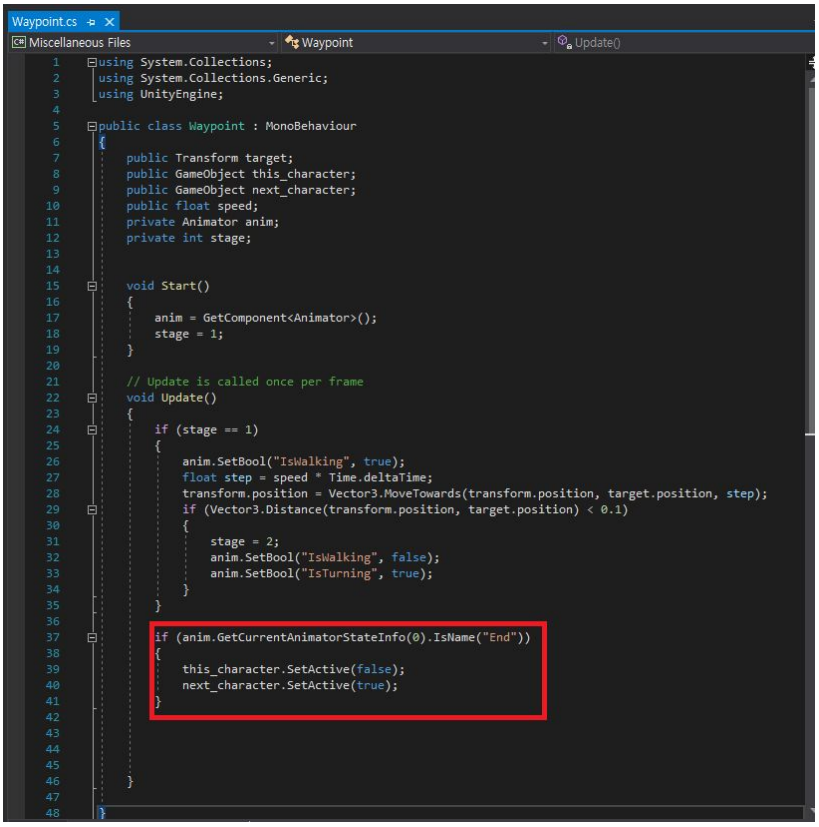
- For example, at this Scene, the man's total behaviors are divided into five sections:
    1) Enter to the restaurant
    2) Looking at the Menu
    3) Order the Food
    4) Wait For the Food
    5) Eat the food

- Also, notice that only 1) Enter to the restaurant is activated, and other 4 sections are deactivated. (Deactivated means that they are located at their own starting point, but cannot be seen at the Scene until they are activated. Animator does not work while the character is deactivated. Also, notice that all animator ends with **'End'** state.



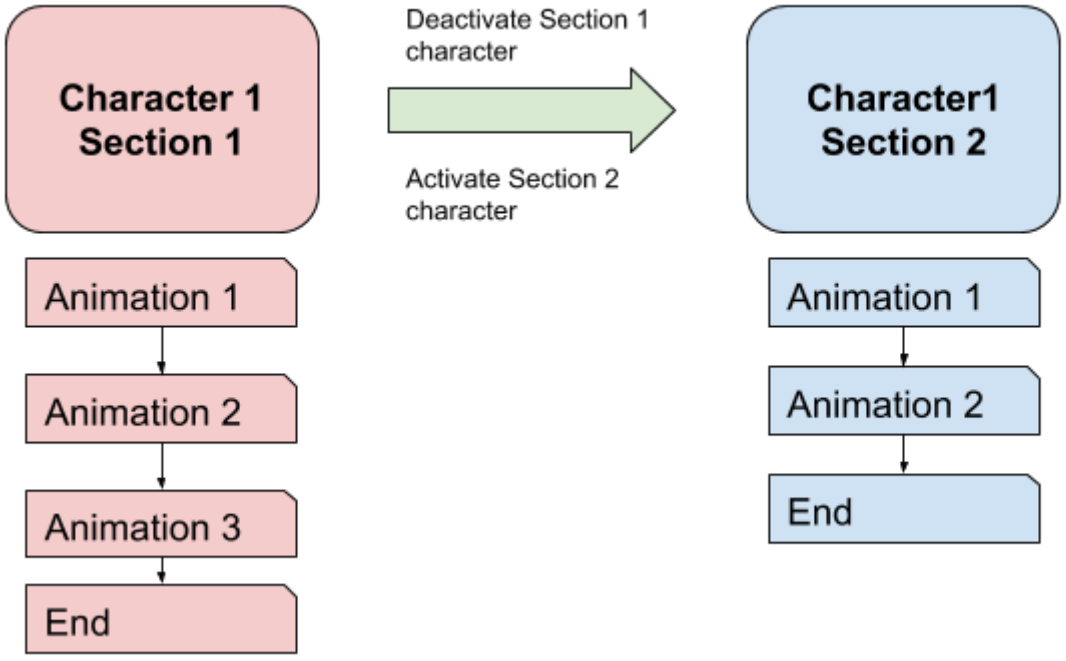


- From Inspector tab's Waypoint.cs section, set **'This_Character'** as the current section character by drag and drop, which will disappear after **End** state in the animator. Then, set **'Next_Character'** as the next section, which is the character with looking at the menu, which will be activated after **'This_Character'** is deactivated.



```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Waypoint : MonoBehaviour
6   {
7       public Transform target;
8       public GameObject this_character;
9       public GameObject next_character;
10      public float speed;
11      private Animator anim;
12      private int stage;
13
14
15      void Start()
16      {
17          anim = GetComponent<Animator>();
18          stage = 1;
19      }
20
21      // Update is called once per frame
22      void Update()
23      {
24          if (stage == 1)
25          {
26              anim.SetBool("IsWalking", true);
27              float step = speed * Time.deltaTime;
28              transform.position = Vector3.MoveTowards(transform.position, target.position, step);
29              if (Vector3.Distance(transform.position, target.position) < 0.1)
30              {
31                  stage = 2;
32                  anim.SetBool("IsWalking", false);
33                  anim.SetBool("IsTurning", true);
34              }
35          }
36
37          if (anim.GetCurrentAnimatorStateInfo(0).IsName("End"))
38          {
39              this_character.SetActive(false);
40              next_character.SetActive(true);
41          }
42
43
44
45
46      }
47
48  }
```

- Let's see the **Waypoint.cs** file again. From line 37 to 41, if this sections' animator is at the **End** state, it deactivate **this_character**, and activate **next_character**.



- This diagram summarize how the scenario is working.

9. Human-Robot Interaction

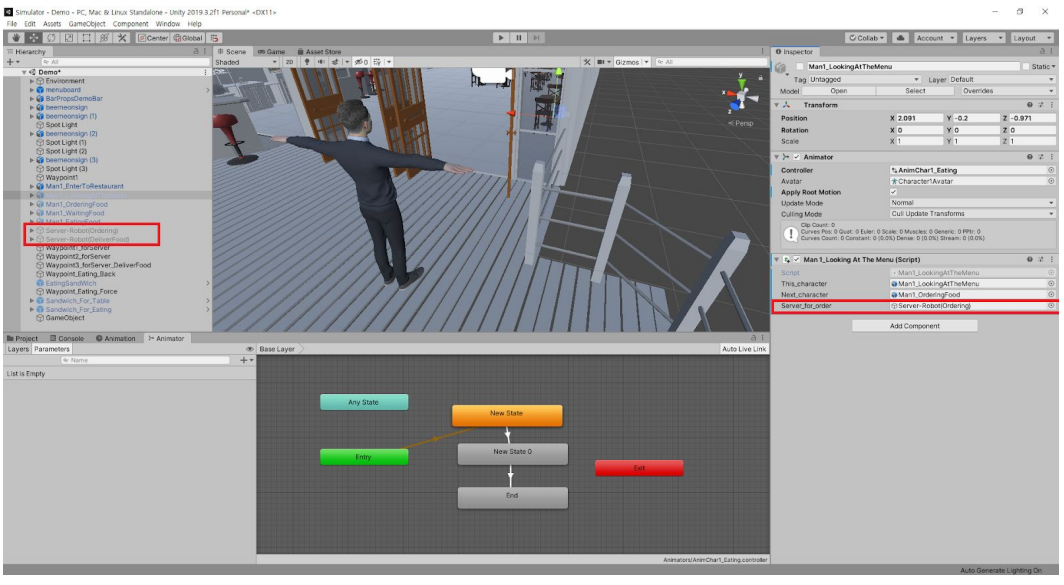
- We also want the robot to intervene in the middle of the scenario. If you want the robot to appear after the certain section in the scenario, just add one more GameObject for at the file, and let that robot object with its own animator activated after current section ends.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Man1_LookingAtTheMenu : MonoBehaviour
{
    private Animator anim;
    public GameObject this_character;
    public GameObject next_character;
    public GameObject server_for_order;


    // Start is called before the first frame update
    void Start()
    {
        anim = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
        if (anim.GetCurrentAnimatorStateInfo(0).IsName("End"))
        {
            this_character.SetActive(false);
            next_character.SetActive(true);
            server_for_order.SetActive(true);
        }
    }
}
```



- Please check each robot game objects to see how it is working. They are similar to human characters.

10. Future Works

- Unfortunately, this project is incomplete. So, I will illustrate works that I planned to do.
    1) To generate many scenes, more custom animations are needed, especially for eating part.
    2) Make more of robot's animation
    3) Make robot object more realistic (Buy one from the Unity Asset Store is an option)
    4) Human-to-Human interaction
    5) UI tools that automatically generates scenario after user selects several animations in order.