UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

3D Augmented Reality

A.Y. 2022/2023

Animator Controller & Animations

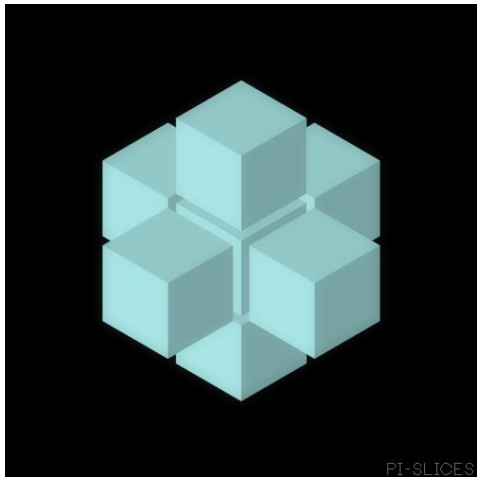
LAB experience 3

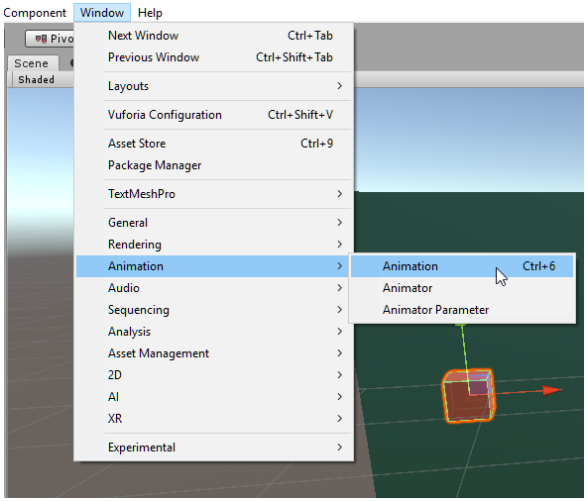
Elena Camuffo

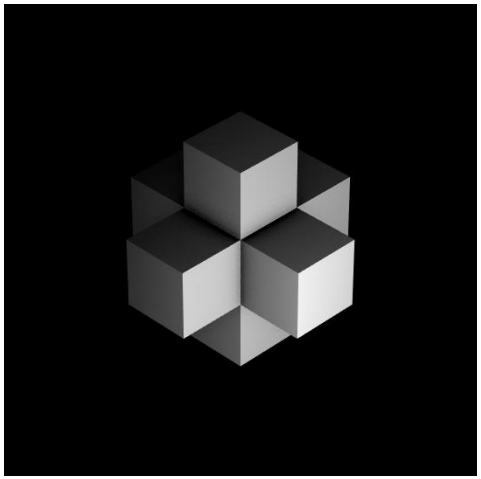
elena.camuffo@phd.unipd.it

Animations

- Unity allows us to **animate Game Objects** by means of the Animation window tab.



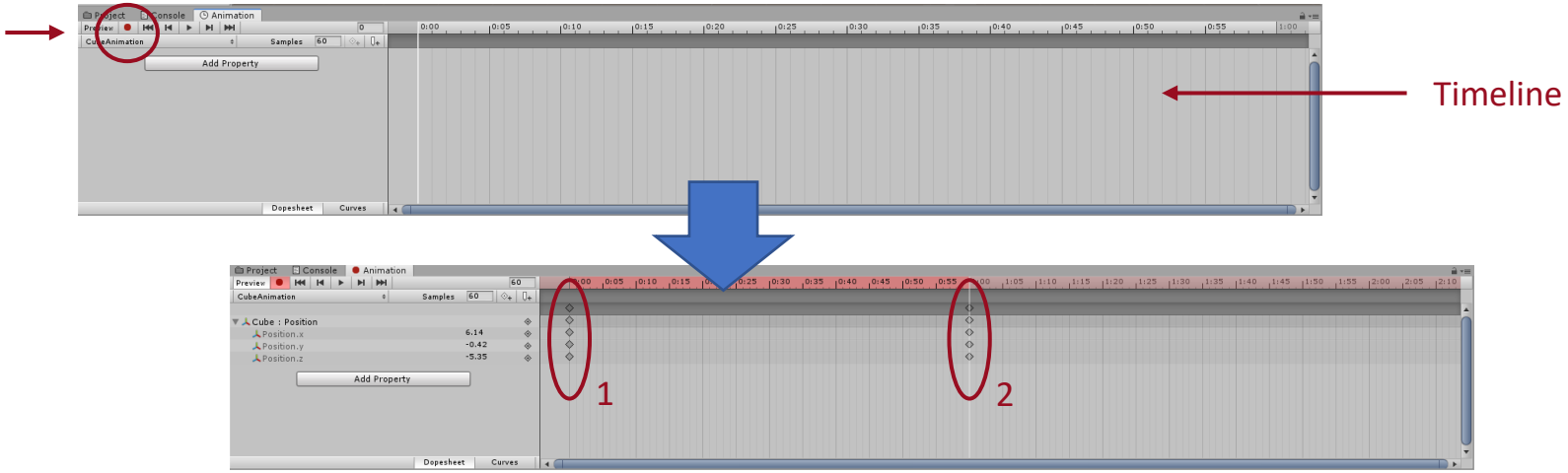




- The Animation **overwrites** Transform's position: if we add an animation to a Game Object, and then we move it in Scene view, it will go back to the first keyframe once we play the Game view.
- **To fix** this, animate Game Objects **inside a parent object,** so that the transformation will be *relative* to the parent Object coordinates and not to the world coordinates.
- You can exploit parent objects also to create **Nested Animations**.

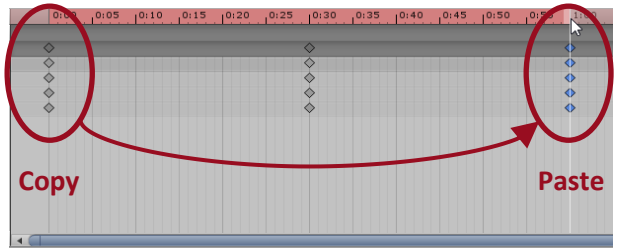
Create Animations

- To create an animation:
  1. Push the **Record button** and move the Game Object to the desired starting position.
  2. Change the position in the timeline and move the cube to the ending position.
  3. Click **Play** to visualize the animation.
- The dots in the timeline are called **keyframes.** They represent the set of Transform properties in that specific instant (in this case the 3 position components) and implement the concept of **Pose-to-Pose animation**.

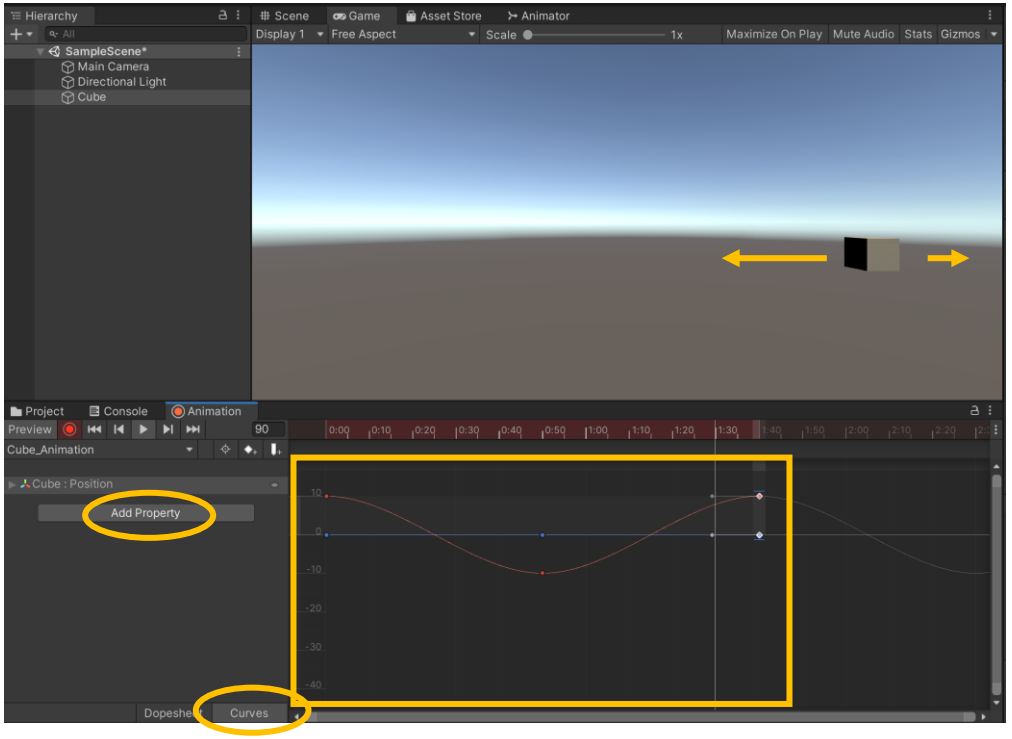


Create Animations

- You can operate on keyframes directly in the timeline. For example, if you want to **loop the animation,** copy and paste the initial keyframes at the end of the timeline, equally spaced from the second ones.

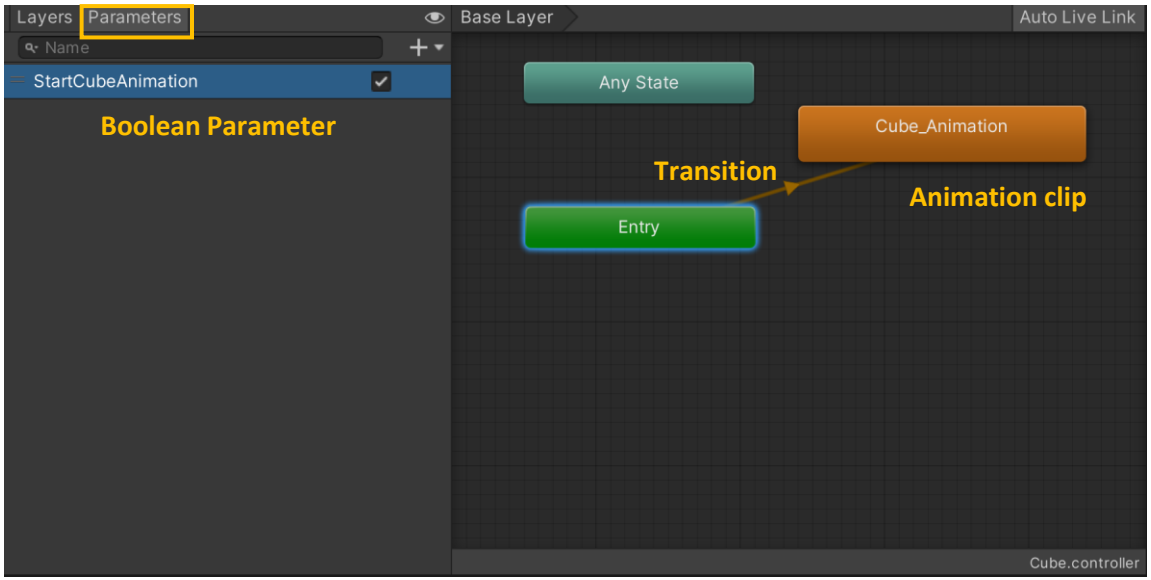


- You can also add new keypoints, saving different properties or values (click on the *Add Property* button).
- The **Curve Tab** shows the Spacetime diagram.

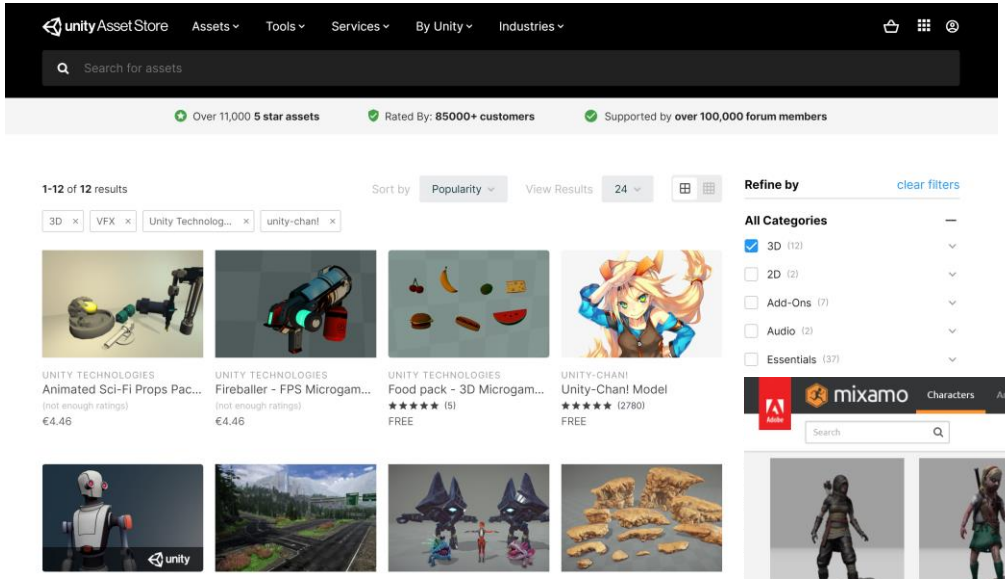


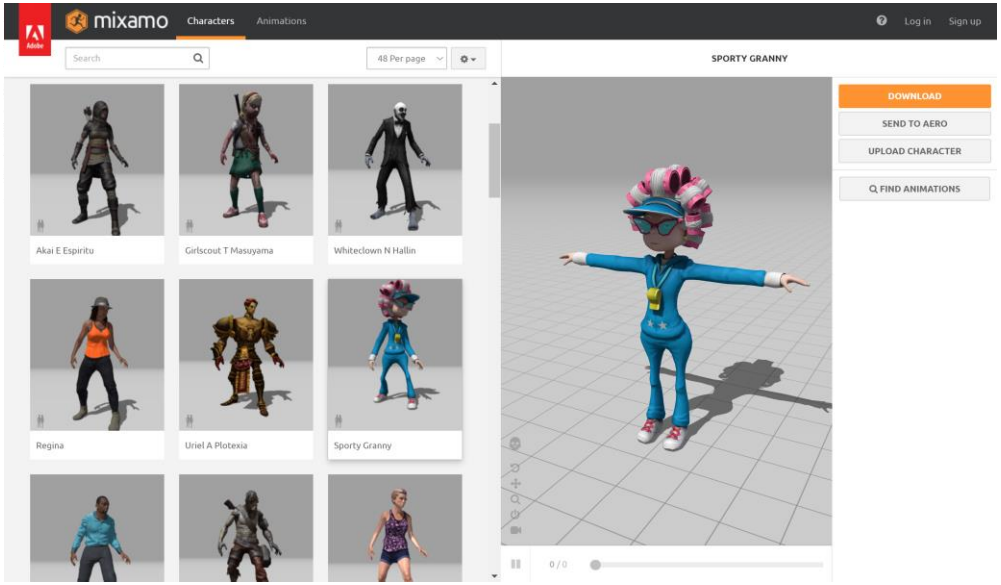
Animator Controller

- The **Animator controller** is an Asset which allows to manage animation clips of a Game Object and the Transitions between them using a **Finite State Machine.**
- The system of animations of a Game Object is represented through a **directed graph** where the animations correspond to the nodes and the transitions to the edges.
- The **Transitions** are controlled by some parameters (Boolean, integer etc.) via script.



Asset Store & Mixamo



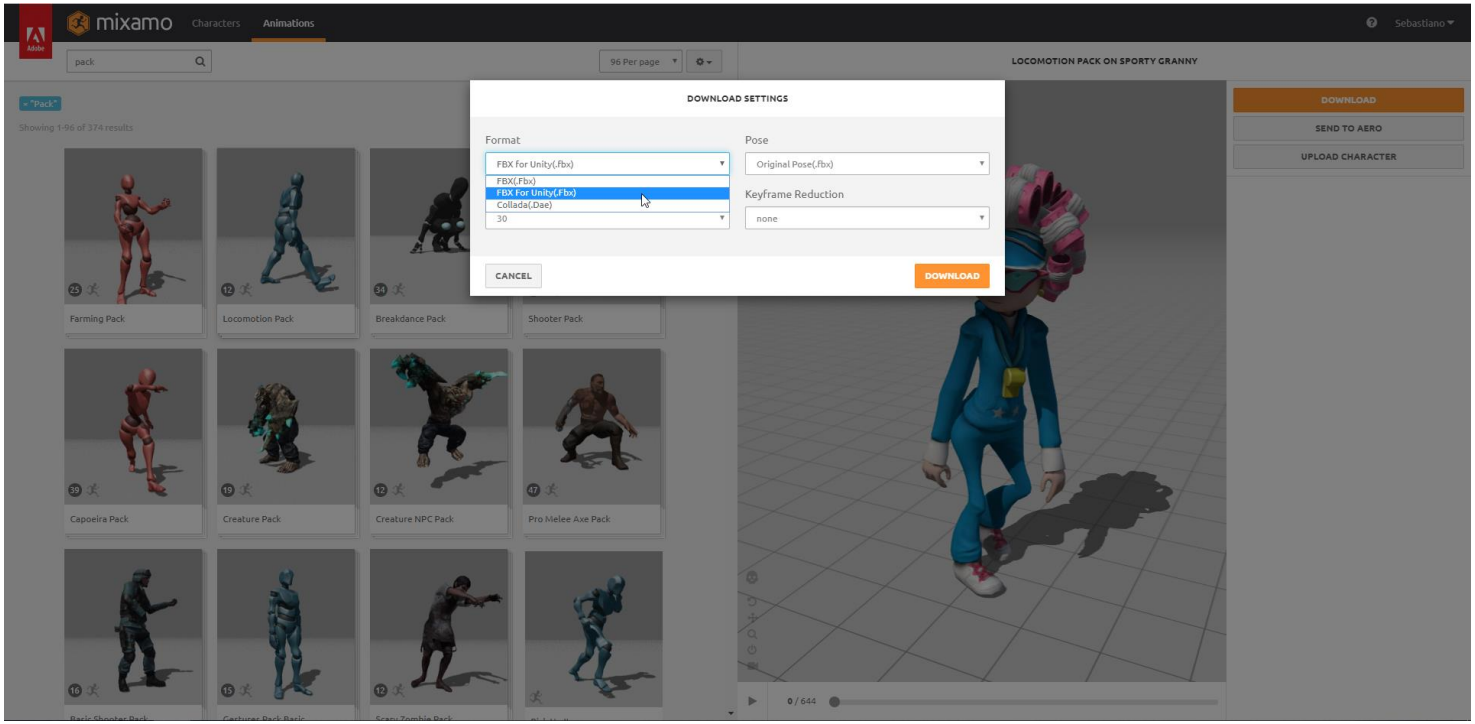


- Official **Unity Asset Store**: https://assetstore.unity.com/

- Adobe's Asset Store **Mixamo**: https://www.mixamo.com/

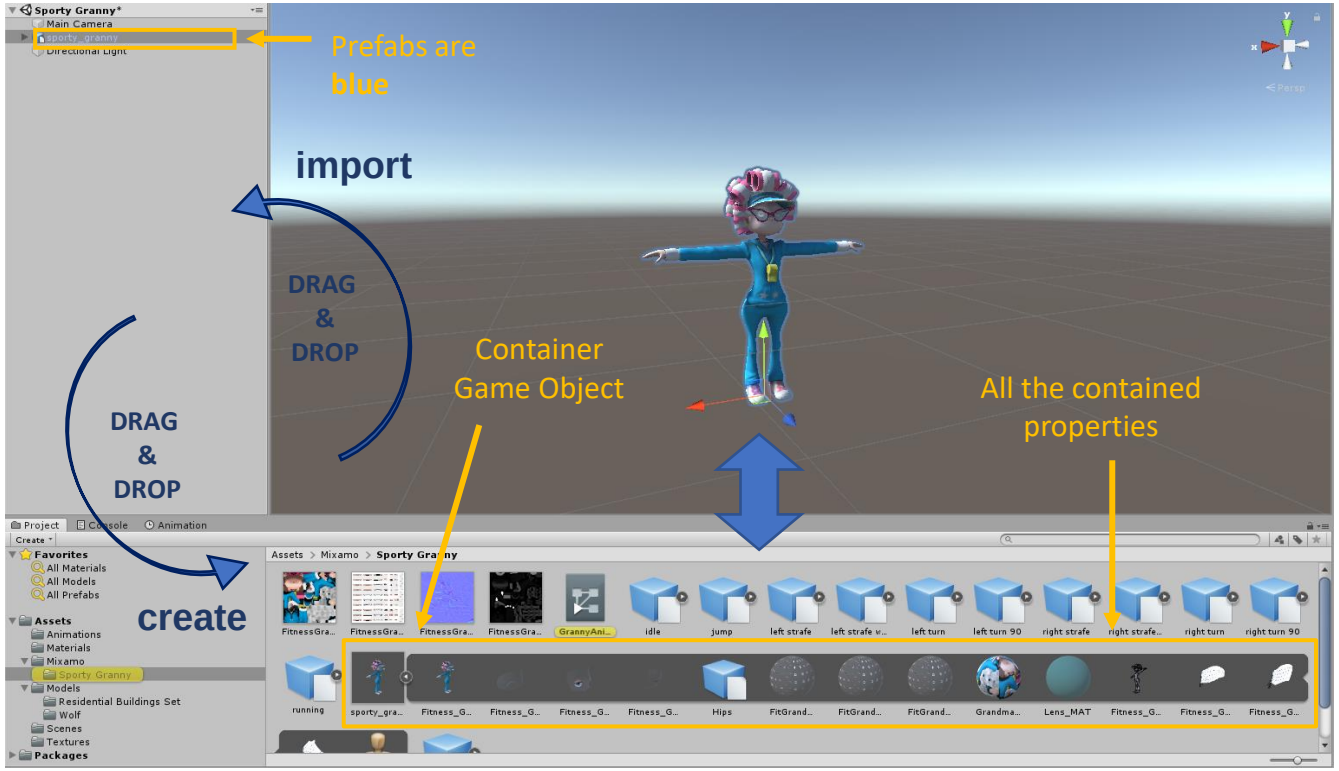
Download Characters and Animations

- From Mixamo you can download characters and animations (select *.Fbx* format).
- **Note:** you can also download single animations for your already downloaded characters. In that case, choose *Without Skin* in Pose menu.



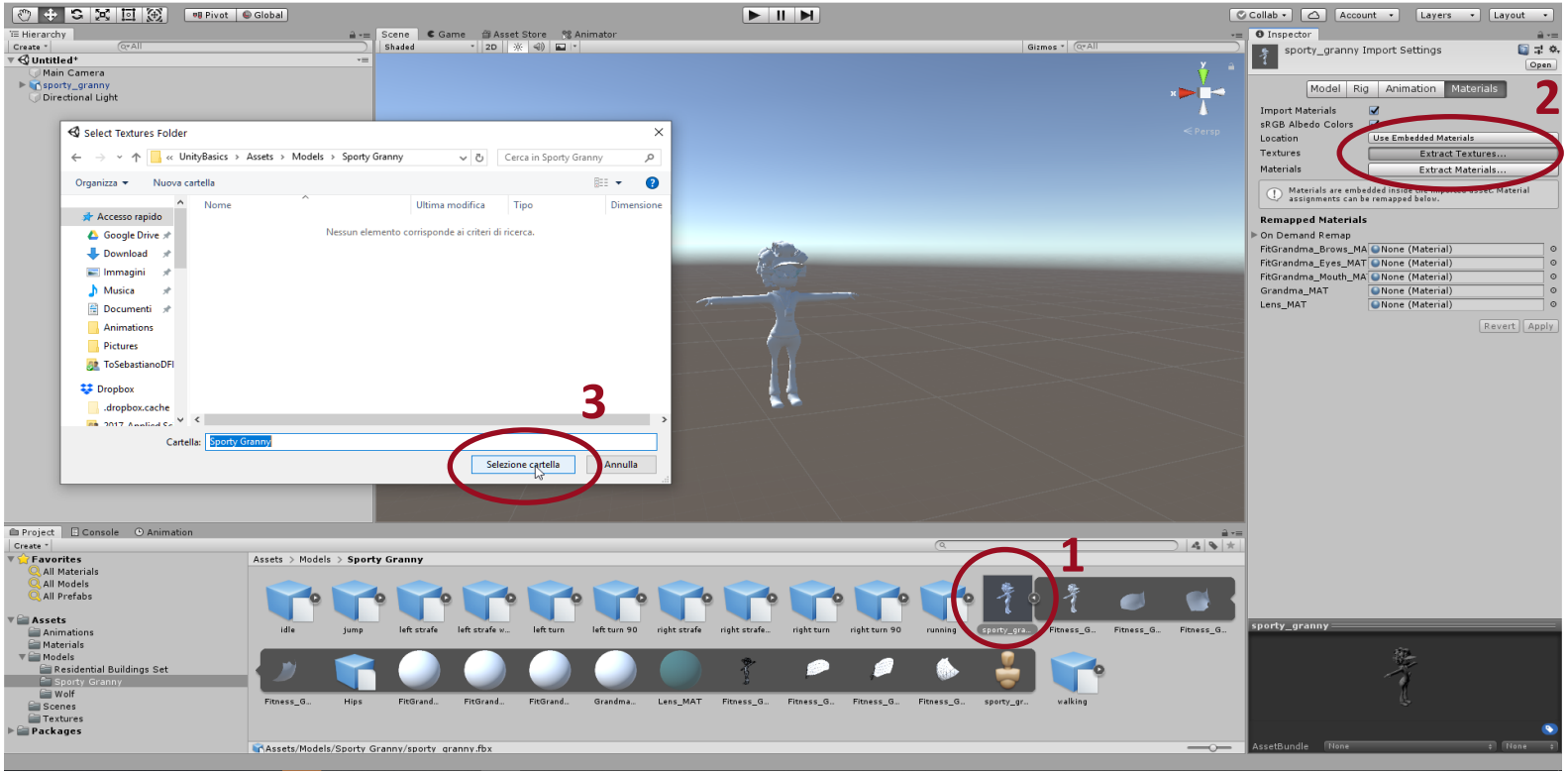
Prefabs

- A **Prefab** in Unity is a Game Object complete with all its components, property values, and children Game Objects, which can be **saved as a template** and reused as an Asset.

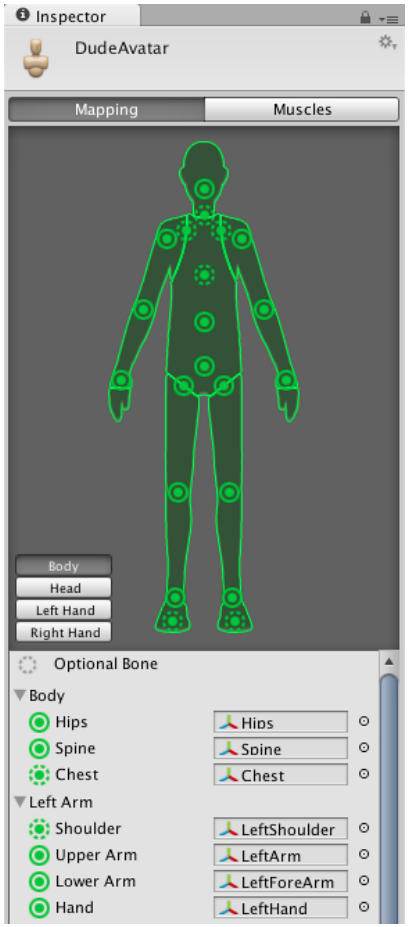


Add Texture

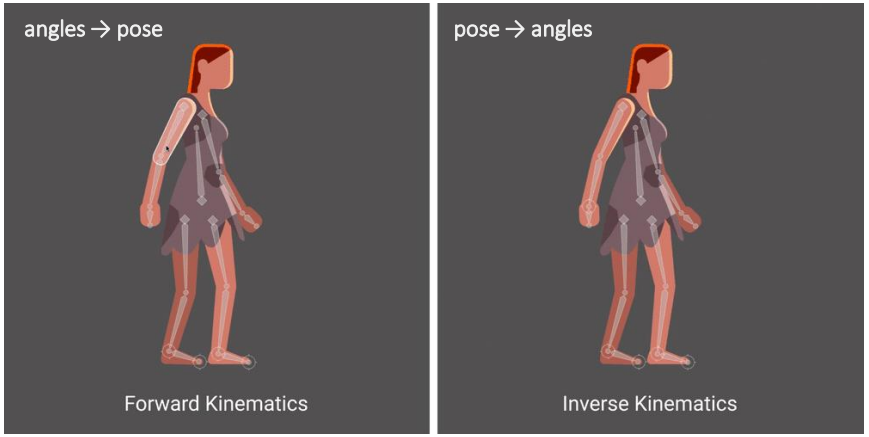
- Add the **Texture** to your character clicking on *Extract Texture* in the *Materials* tab.



Characters' inverse kinematics model

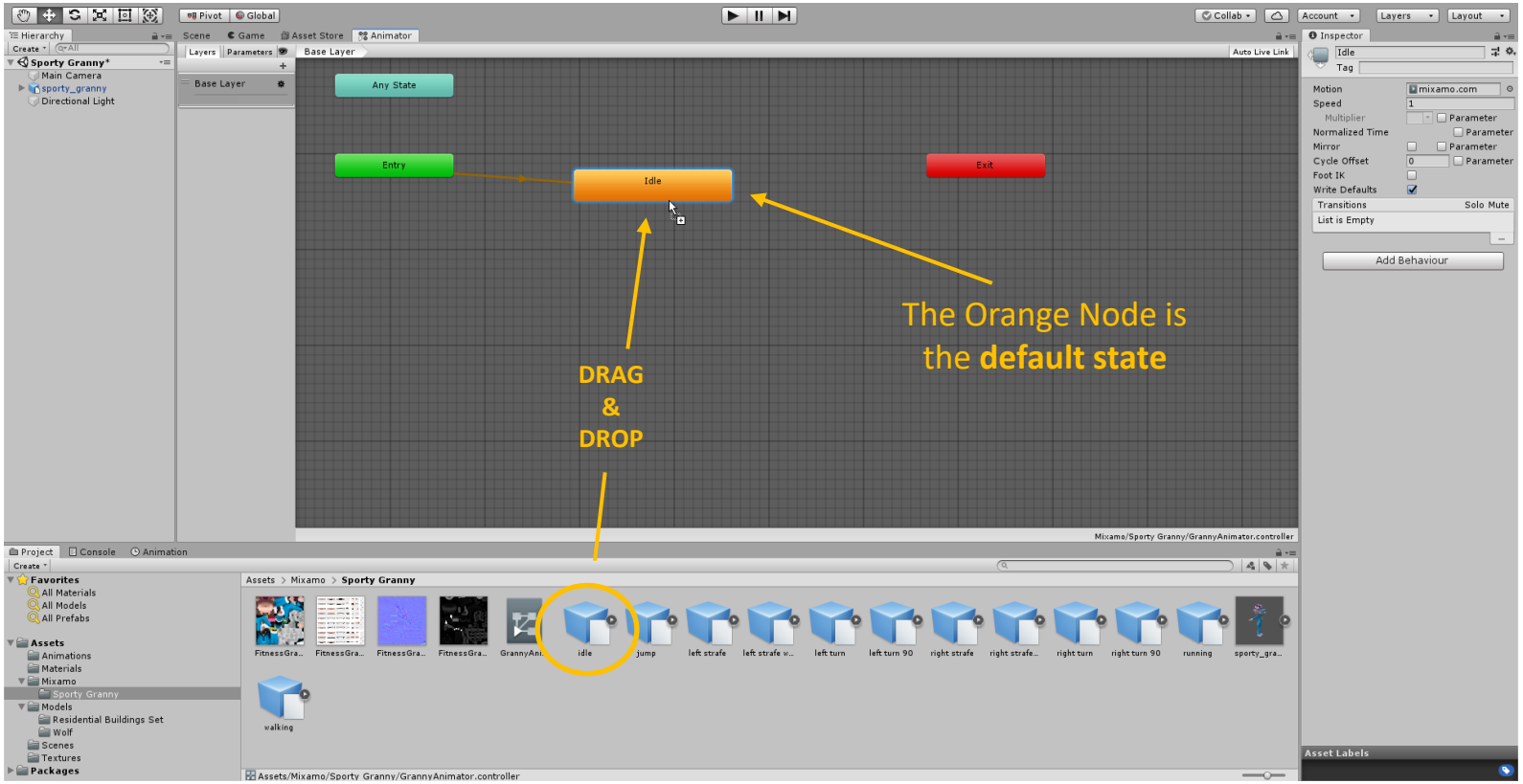


- In order to create or import an animation for your character, you need to have a **skeleton**, properly set. A skeleton defines the bones inside your mesh and their movement in relation to one another.
- The skeleton uses the **Inverse Kinematics Model** when animated.
- The process for creating the skeleton is known as **rigging.** *Mixamo* characters are just rigged.



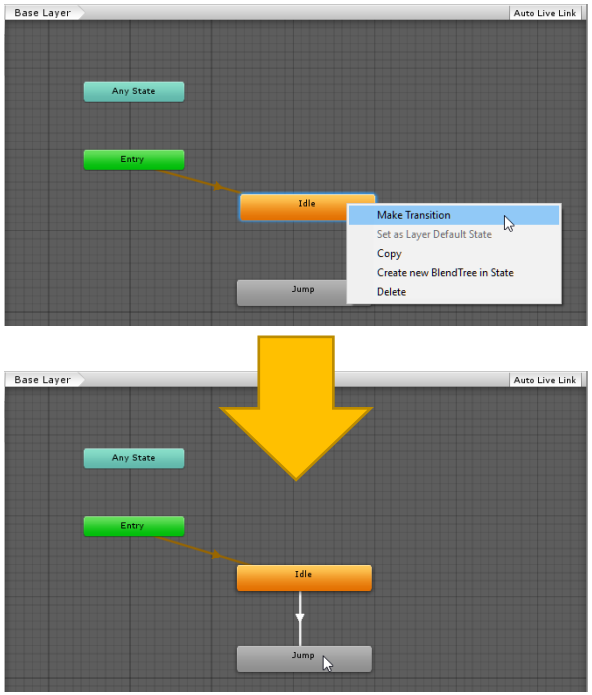
Add Animations

- You can add animations to the character by means of an **Animator Controller** component.



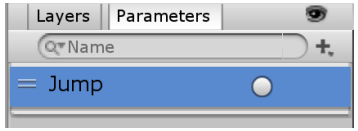
Make and Control Transitions

- Import various animations and link together with **Transitions**.
- To set the conditions that enable a specific Transition, select the Transition and modify the conditions in the Inspector panel, according to the Parameters.

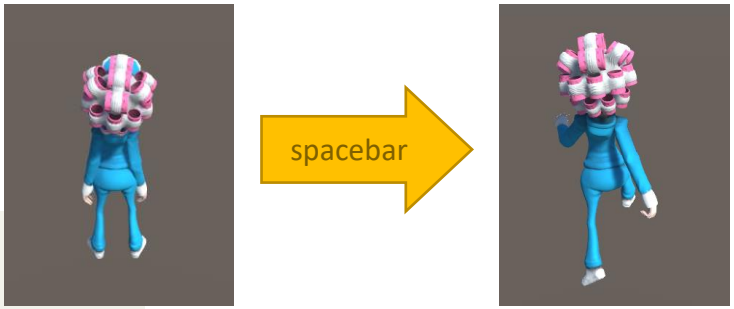


Has Exit Time condition means that the entering animation is delayed, leaving the previous one to complete

Animation **timing**



Script Example




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

public class GrannyControls : MonoBehaviour
{
    // Animator Controller component variable
    private Animator anim;

    // Start is called before the first frame update
    void Start()
    {
        //Get the Animator Controller of the attached Game Object
        anim = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
        // if Key 'Space' is pressed the 'Jump' trigger is activated
        if(Input.GetKeyDown("space"))
        {
            // Tell the animator the player is jumping
            anim.SetTrigger("Jump");
        }
    }
}
```

Start()

Load the Animator Controller component attached to the Game Object.

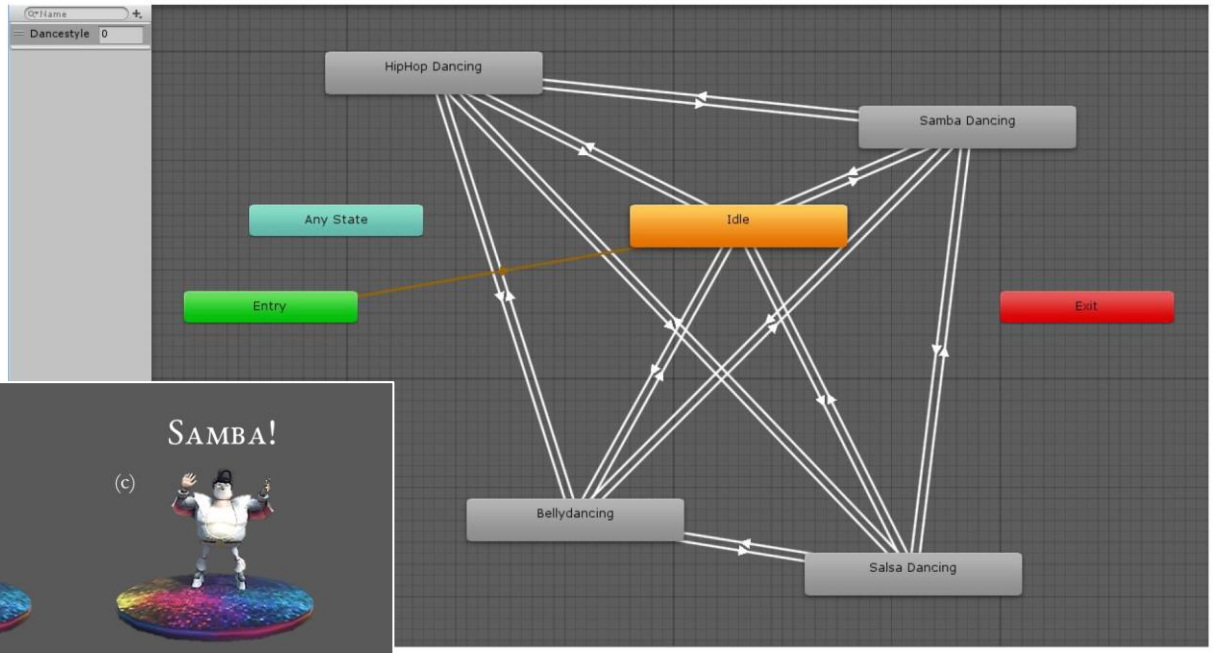
Update()

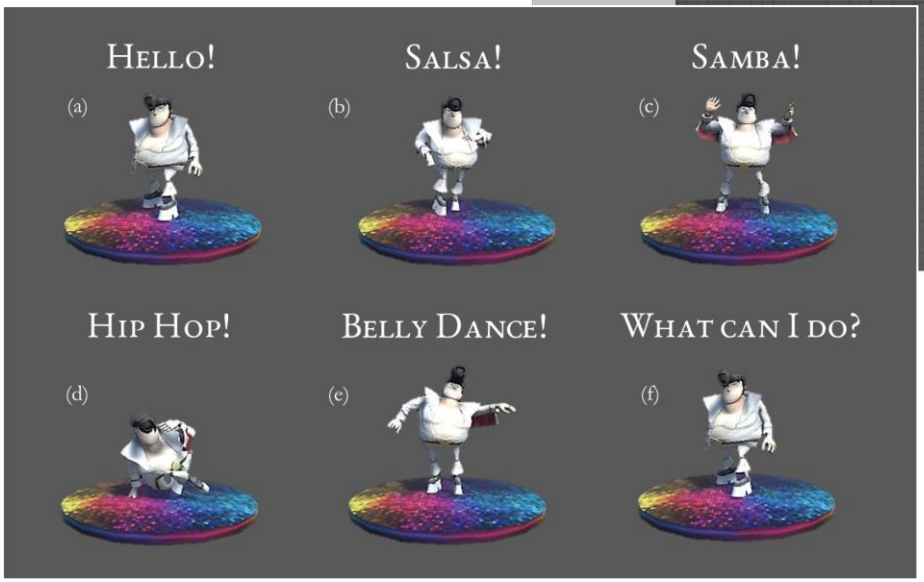
At each frame update, check whether the key 'space' is pressed. If it is, activate the *Jump* trigger, which makes the jump animation to start.

Note: Once the animation ends, the Animatior returns to the default state without the need for further parameters changes, but only a transition with an *exit state* set (property of trigger parameter).

MR Dance

Basic Version:
Change Dance according to the key pressed.





AR application adapted for Android, MS HoloLens and Magic Leap One with **different control mechanisms.**

MR Dance Script

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ChangeDance : MonoBehaviour
6  {
7      private Animator anim;
8
9      [SerializeField]
10     private TextMesh stile;
11
12     // Start is called before the first frame update
13     void Start()
14     {
15         // Get the animator component and set to the default 'idle'
16         anim = GetComponent<Animator>();
17         anim.SetInteger("Dancestyle", 0); //'idle' dancestyle
18     }
19
20     // Update is called once per frame
21     void Update()
22     {
23        // Update the Text and the dancestyle parameter
24        // according to the key pressed
25         if (Input.GetKey("0"))
26         {
27             anim.SetInteger("Dancestyle", 0); //'idle' dancestyle
28             stile.text = "What can I do?";
29         }
```

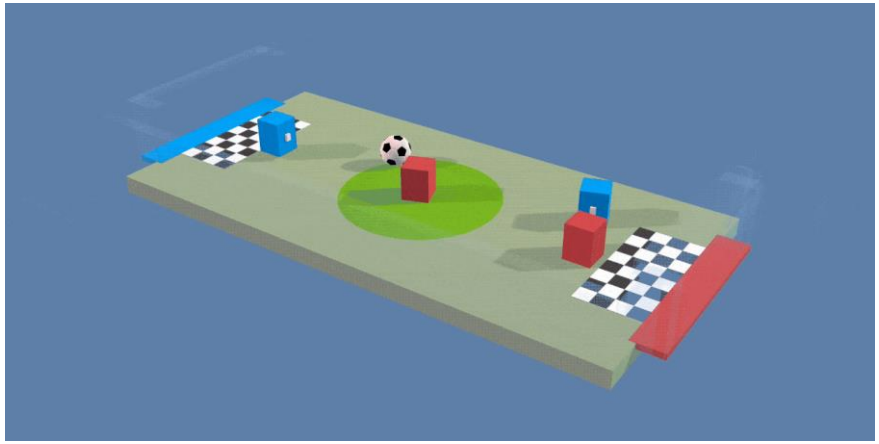
Update()

The parameter (int) of the Animator controller is Updated according to the Key pressed (e.g. if key 1 -> Salsa) and the text is updated accordingly.

```
30         if (Input.GetKey("1"))
31         {
32             anim.SetInteger("Dancestyle", 1); //'salsa' dancestyle
33              stile.text = "Salsa!";
34         }
35
36         if (Input.GetKey("2"))
37         {
38             anim.SetInteger("Dancestyle", 2); //'samba' dancestyle
39              stile.text="Samba!";
40         }
41
42         if (Input.GetKey("3"))
43         {
44             anim.SetInteger("Dancestyle", 3); //'hip hop' dancestyle
45              stile.text="Hip Hop!";
46         }
47
48         if (Input.GetKey("4"))
49         {
50             anim.SetInteger("Dancestyle", 4); //'belly dance' dancestyle
51              stile.text="Belly dance!";
52         }
53     }
54 }
```

Unity potentials

- Unity offers a great variety of tools and packages allowing to implement even more sophisticated graphical environments with **Terrains** and integrate **Machine Learning** with ML agents.



- Unity principally relies on python and tensorflow to implement **ML agents**.
- More information at:

  https://unity3d.com/how-to/unity-machine-learning-agents

EXERCISE: Sporty Granny

1. **Organize the environment with prefabs**

   Open the *SportyGranny_Exercise* project. The scene is set starting from the prefabs in the *LowPolyEnvironment* folder. You can add elements according to your taste, setting them properly as static objects. (they have been downloaded from the *Asset store*).

2. **Sporty Granny prefab**

   Add the *SportyGranny* prefab to your scene and make it a dynamic Game Object (The Sporty Granny has been downloaded from *Mixamo,* as well as the animations associated).

   Suggestion: Remember static and dynamic Game Objects' behaviors from Lab Experience #2.

3. **Add the Granny Animation**

   Then add the *Walking* animation to the animator controller and link it to the default state with back-and-forth transitions, controlled by a parameter.

4. **Make the Granny walk**

   Complete the *GrannyControl* script, writing code to control the transition to *walking* state, whenever the Granny is moving around.

EXERCISE: Sporty Granny

4. **(ADVANCED) Rotate accordingly**

   Complete the movement of the Granny making the camera follow her and adding code to rotate according to the chosen direction of walk. (you can do it in many different ways).

5. **(OPTIONAL) Fitness cube bouncing**

   Import the *FitnessCube* prefab and create an animation to make it bounce according to the "Principles of Animation".

   https://www.youtube.com/watch?v=uDqjIdI4bF4&list=WL&index=3&t=341s