

# *Santa Claus*

*traditional hero of our hearts*



*Manual*

## Merry christmas!

This Santa Claus character you purchased from the Unity Asset Store is a medium polycount game character with a painterly classic look. It is Mechanim humanoid compatible and counts around 10 k polygons (roughly 15k tris).

The character has a HIK (Maya Human IK) compatible skeleton with some extra joints (like hat and facial joints in eyelids and eyes).

Glasses are a separate object.

Santa comes with blendshapes for facial expression and phonemes and material presets and textures for Unity 5 Standard Shader PBR Materials ,  
Unity 4 Bump/spec(A) + normalmap  
mobile unlit and  
vertexlit shader usage.

The package also includes an iClone 5 avatar file for users of the animation program iClone (can also be imported into iClone 6).

The FBX files can be opened and edited in any standard 3d application like Cinema4d, Maya or Blender. Main texture is 4k resolution (it is set to a lower import resolution for the Unity use).

Props texture sizes are lower depending on the nature of the props.

The T-stance is a bit offset in the shoulders to make him compatible with normal proportion motion files because of Santa's wide hips .

Santa comes with several props (cane, candycane, present, sack closed, sack open and a snow landscape). He also comes with a script component that attaches the cane/candycane/present and sack to his hands at runtime in Unity and with a buch of custom animations with and without root motion that should cover a lot of possible game usages. The webplayer demo is included in the asset. The folder "Demo" can be safely deleted from the asset if not needed.

All script are encapsulated in namespace "SantaClaus" to avoid possible software conflicts with other assets.

On the following pages I will explain the files in the package in detail.

Kind regards,

Oliver Wuensch (creator)

email: [support@wuenschonline.de](mailto:support@wuenschonline.de)



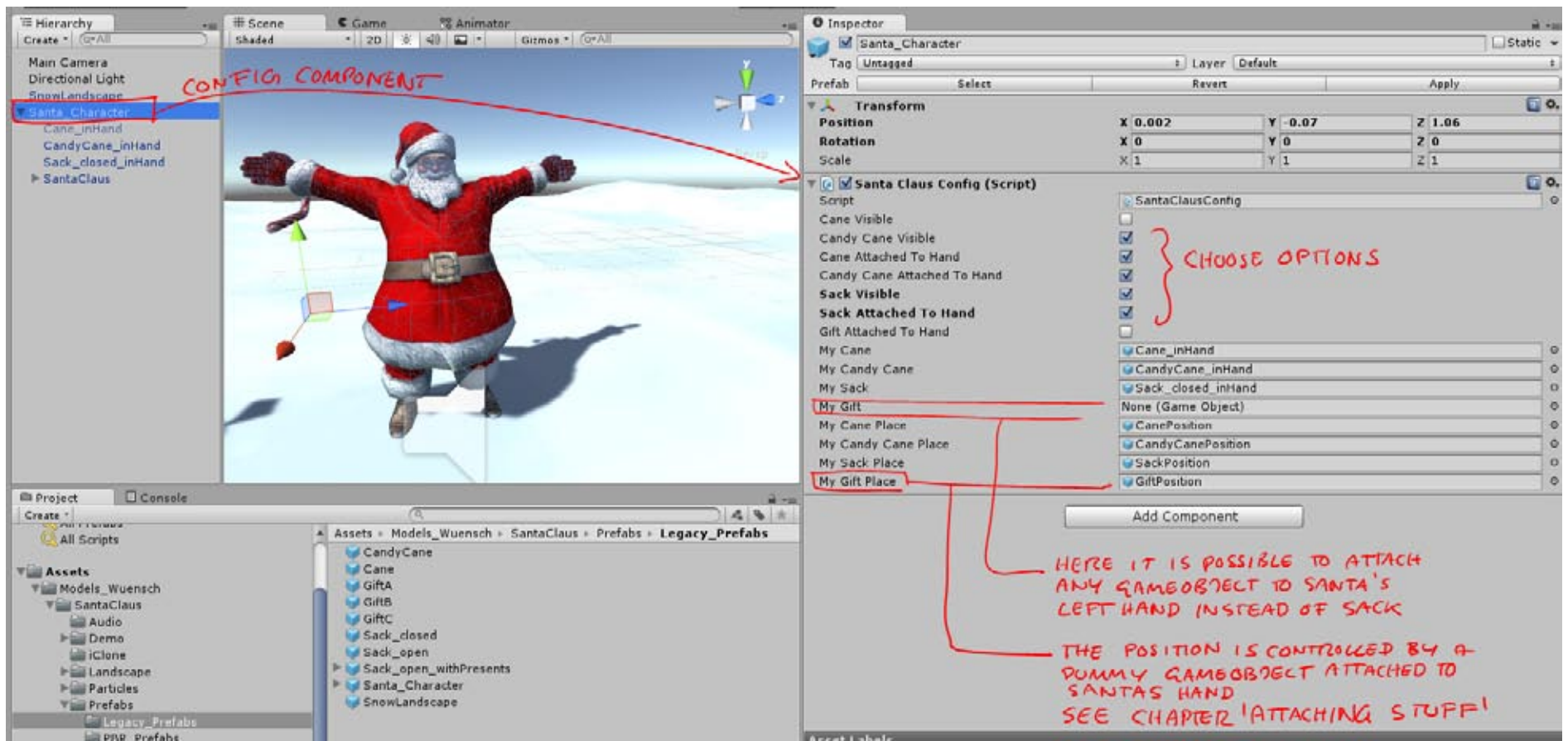
# Table of contents:



- 1) **Quick start**
- 2) The asset folder structure and other stuff that is good to know
- 3) Attaching stuff to hands
- 4) The Santa Config Script Component and using Santa Config with C# at runtime or with Hutong Playmaker
- 5) about animations
- 6) about iClone usage
- 7) about optimizing Santa for games

# 1: Quick start

- 0) open a new project in Unity, import the Santa Claus asset via the asset store window and create a new empty scene for Santa
- 1) go to Models\_Wuensch-> SantaClaus-> Prefabs->PBR\_Prefabs (or legacy prefabs for non-PBR versions)
- 2) Drag prefab Santa\_Character into your scene or hierarchy window to give Santa something to walk on.  
Drag prefab Santa\_Character into your scene or hierarchy window.  
Santa's SantaConfig script is attached to the top level SantaCharacter gameobject  
Santa's animator is attached to the SantaClaus gameobject inside.
- 3) select Santa\_Character in Hierarchy
- 4) configure the boolean switches in Santa Claus Config as desired (Sack in hand active, Sack visible active, candycane visible, candycane in hand active, cane not visible)



5) Open the animator window. The animator from the demo is linked to the prefab. It is a very simple setup for demo purposes. The mechanim variable **IndexAni** triggers the animation at runtime. Every animation has a different IndexAni value defined in the transition condition.

Set **Indexani** to the value **6**, that is the index for the running with sack in place animation. Press play, and Santa will run in place in the scene and game view with sack in hand and candycane in the other hand.

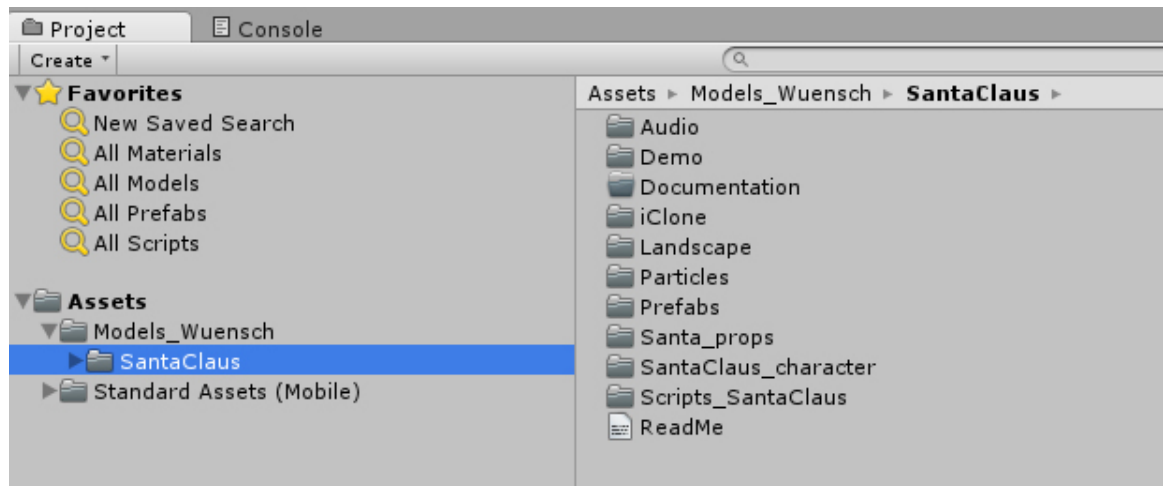
If you want to setup a “real” mechanim animator to use Santa as Player you can check out one of the many great mechanim tutorials available for Unity.

I recommend this one for a start:

[https://unity3d.com/learn/tutorials/modules/beginner/animation/animator-controller?playlist=17099](https://unity3d.com/learn/tutorials/modules/beginner/animation/animator-component?playlist=17099)



## 2: The asset folder structure and other stuff that is good to know



Everyone has his own system to file assets.

That's why I explain mine here:-)

On the top level you find SantaClaus inside the **Models\_Wuensch** folder. Inside you will find the following folders:

**Audio** contains all the non-professional (but hopefully charming) Santa Claus voice recordings samples I did. If you like my voice you are free to use them as you like.

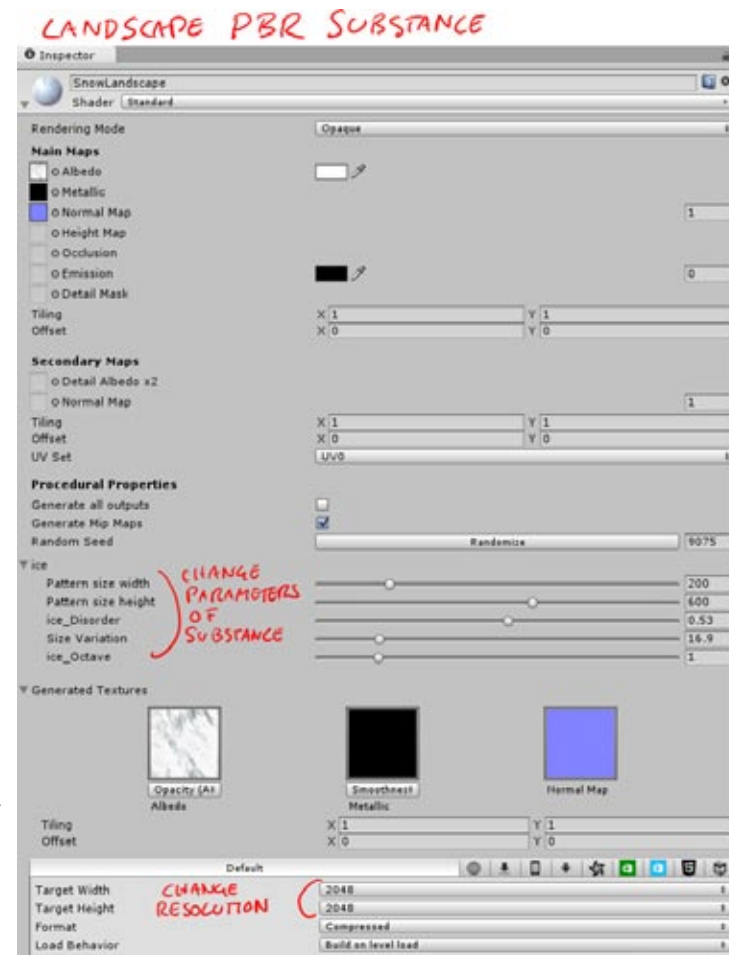
**Demo** contains all files that belong to the webplayer demo I did to show off the asset. You can delete the folder if you don't need to look at it, it is independent from the rest of the asset. If you want to know how I triggered Santa in the demo, take a look at it. I have 10 years of experience in several multimedia programming languages but am not a professional coder and use Unity for only 3 years now, so my solutions might be a bit clumsy here and there. But they are commented and should be easy to follow for someone with basic C# knowledge.

**Documentation:** you will find this PDF and a mini- tutorial for Hutong Playmaker users here.

**iClone:** contains a -zip file with an iclone 5 -avatar file of Santa plus props so iClone users can animate and lipsync santa right away. Santa is setup with phonemes for lipsync. Also check out the custom morph controls to animate the facial expressions. If you are no iClone user you can delete this folder from your project safely.

**Landscape** contains the ground mesh Santa is standing on, with two wintery skyboxes and textures and materials. For Unity 5 there is a PBR substance file included instead of regular textures in the textures folder.


Check out the prefab Snowlandscape\_PBR to see how it is set up and to play with the parameters.



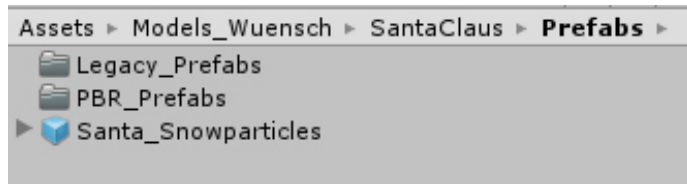
**Particles:** here you can find a snow particle system created with the highly recommended asset Particle Playground.

The particle effect is NOT optimized for mobile. It also contains the free-to-use runtime of Particle Playground.

For safety reasons to avoid possible script errors I modified the runtime scripts and enclosed them in their own namespace ( `santaClaus.ParticlePlayground`).

If you want to use the snow as a base for your own snow and own Particle Playground you can edit it in Particle Playground. In that case use the included **Playground Preset - Santa\_Snowparticles.unitypackage** that is inside the folder to install the editable preset. 

**Prefabs:** Santa and all props are already prepared as prefabs with a Unity 5 PBR version (**PBR\_Prefabs** and a Unity 4 spec/bump version (**Legacy:Prefabs**).



**Santa\_Character** (legacy shaders) and **Santa\_Character\_PBR** (Unity 5 standard shader) is the complete Santa with Scripts and a demo animator ready to be used right away. Simply drag the prefab into your scene and you are ready to go.

There is also a Prefab for the **Snowparticles** here, simply drag it into the scene to let it snow. The snow particles are NOT optimized for mobile.

**Santa\_props:** in this folder the models, textures and materials for Santa's props are stored that are being referenced by the prefabs.

Each prop's folder contains subfolders with the textures and materials. In the materials folders there are prepared materials for **legacy bumped spec**, **vertexlit** and **unlit** and **Unity 5 PBR** (vertexlit and unlit are very mobile friendly fast shaders when used with Santa.)

An exception is the gift parcel, here for Unity 5 PBR there is a **Substance (Gift\_PSB.sbar)** available instead of normal textures. You can modify the color and other parameters and create a variety of different materials with it if you want. Check out the gift prefabs to see how it is applied.

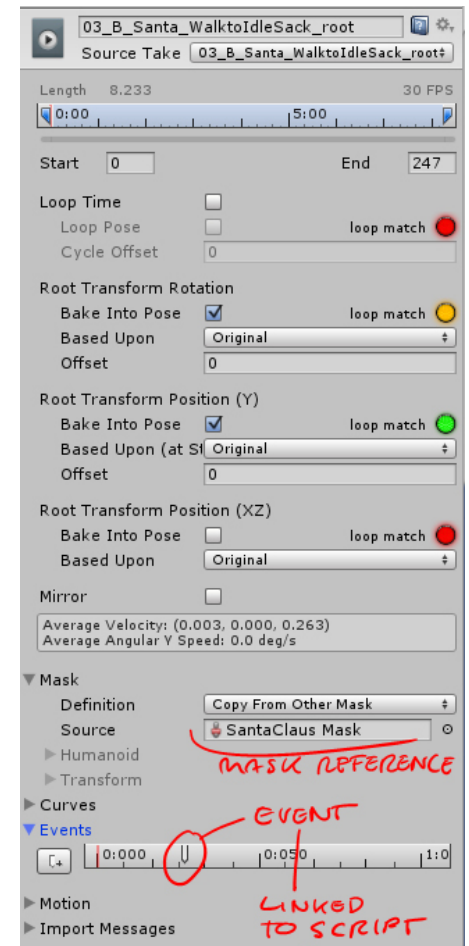
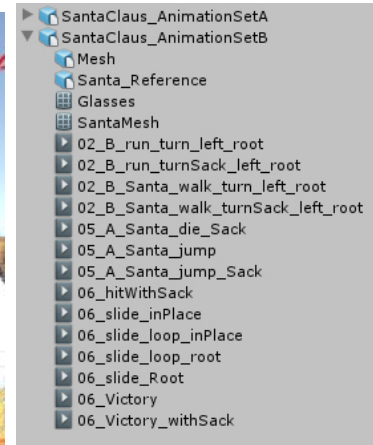
Finally there is the

**SantaClaus\_character** folder. In here Santa's raw material can be found. There is **SantaClaus.fbx** with the basic model, **Textures** and **Materials** and the **Animation** folder. You can delete or ignore the .fbm folders. These are automatically created when importing FBX files in spite of me choosing the option not to import them in Unity FBX settings. resistance is futile.

In the Materials you will again find all the different materials for **PBR, sec/bump, vertexlit and unlit shaders**.

In **Animations folder** there are 2 or 3 sets of FBX animations (as I am writing this there are 2, but I am planning to do a third one with the rest of the talking animations as update). If you unfold the FBX Sets you can see the single animation clips that can be dragged into Animator. If you check out the FBX in Inspector you can see what options I chose for the motions. Here you can change options, for example if you want a certain animation that is not looping to loop.

**All animations reference Santa Claus mask** one level above to make sure the extra bone animations in the face and hat and the morphs are available in all the animations. The animations where Santa puts his sack down (Santa\_WalktoIdleSack) also have one **event** attached here that is linked to a little script to tell Santa\_Config to let go of the sack in the demo scene.





Finally there is

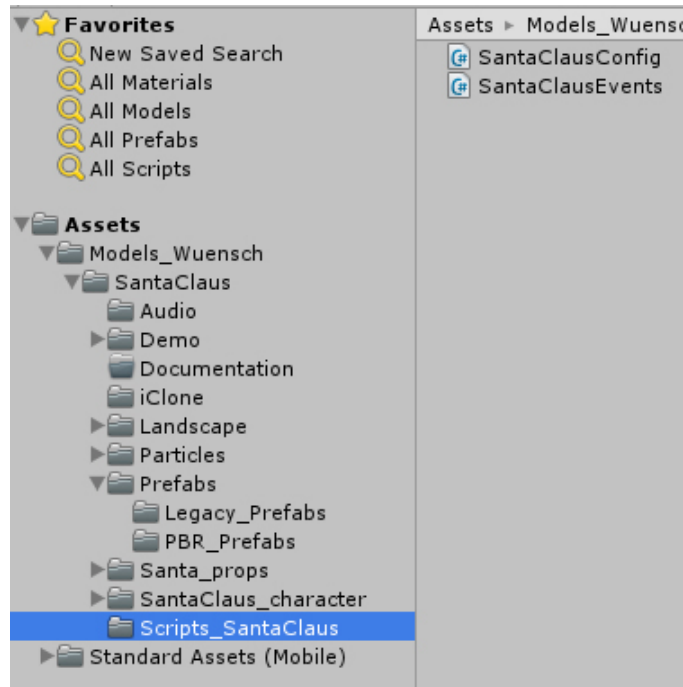
**Scripts\_SantaClaus**, the folder containing the 2 scripts relevant for the Santa character.

SantaClausConfig is the script that manages the props in santa's hand, SantaClausEvents is the events called from the animation clip.

There is not much in there apart from OnDropSack, the function that passes the event to SantaClausConfig to drop the sack.

Feel free to add your own events to make Santa come alive.

SantaClausEvents is in Unity namespace so it can be called by Mechanim event and uses SantaClaus namespace to be able to communicate with SantaClausConfig.



### 3: Attaching stuff to hands

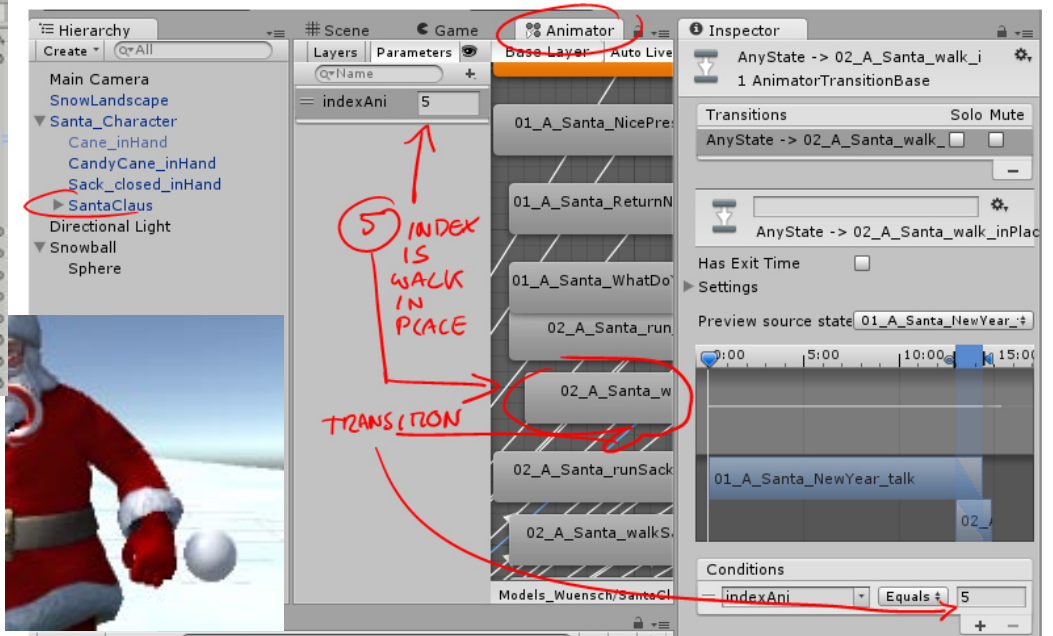
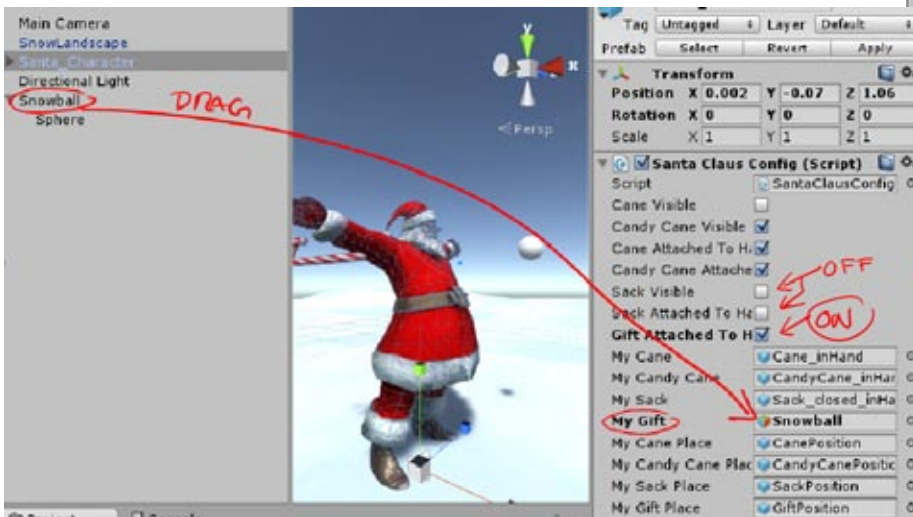
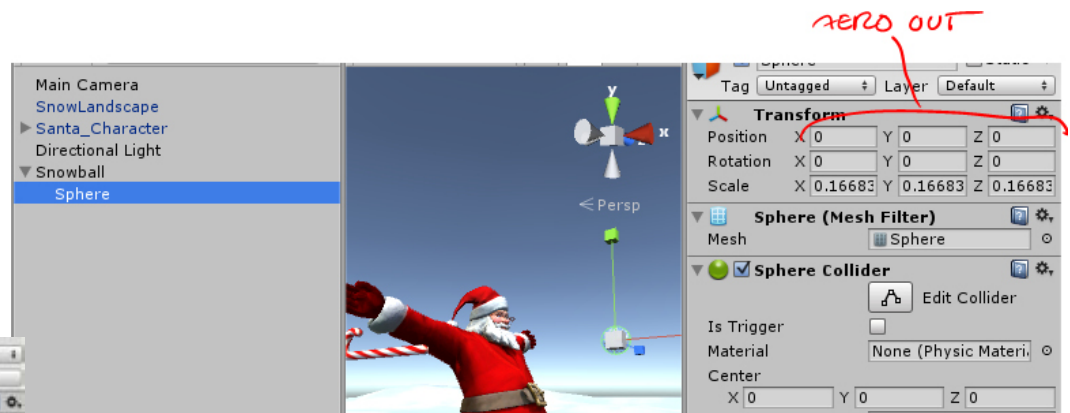
Santa is set up so that gameobjects can be dynamically attached to his hands at runtime. You might want to have him hold your company product, or a weapon like a snowball-- or whatever you can come up with.

You can easily achieve this by following this mini tutorial:

- 1) create an empty scene, drag the landscape and Santa\_Character from the prefabs to the hierarchy window.
- 2) Select SantaClaus object inside Santa\_Character in the Hierarchy and in Animator
- 3) create a simple **sphere** from the menu GameObject->3DObjects->Sphere, move it in front of Santa and scale it down to the size of a snowball.

4) create an **empty GameObject** and name it “**Snowball**”, make the sphere a child of this Gameobject and **zero out** the sphere’s position in the inspector transform properties (type in 0 at XY and Z).

5) select **Santa\_Character** and drag Snowball into the **My Gift** variable slot of the **Santa Claus Config** component. Turn on the “**Gift attached to Hand**” option and turn off Sack Visible



6) Select SantaClaus in Hierarchy, open **Animator** and change the Animator variable **indexAni** to **5** (which is the index for the “walk in place” animation) press play and watch as the snowball is attached to hand. The sphere is not in an appropriate position in the hand but following somewhere near it.

7) Now while the animation is playing **press the “Pause” button**, then rotate and move the scene view (alt key and right and middle mouse button) , and **transform and scale the sphere object** (which is a child of GiftPosition and Snowball now while the game plays ) until it sits at the right position in Santa’s hand.

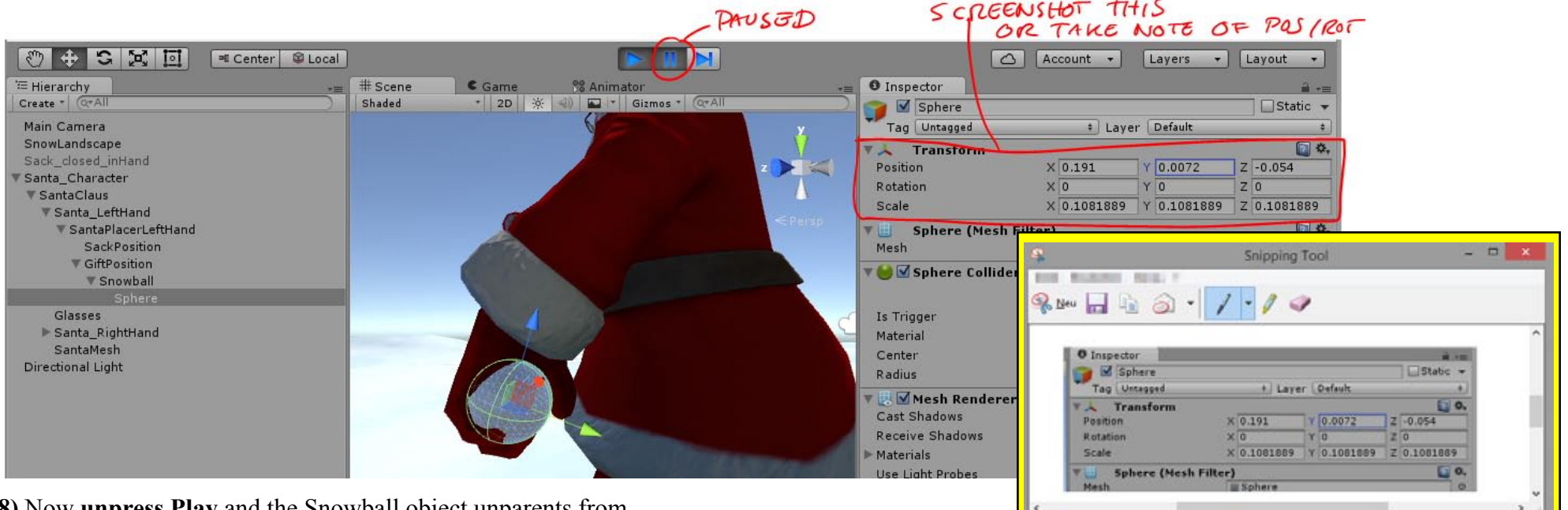
The hand in that animation was not really posed for a snowball, but it will do for tutorial purposes.

Unpress Pause and watch if it looks right in movement.

**Do not stop the animation**, press Pause again and make a screenshot of the sphere’s transform properties.

Integrated in Windows there is a nice tool named “Snipping Tool” which I use all the time for stuff like this, as you can grab a small area of the screen with it and the screenshot is available on screen right away.

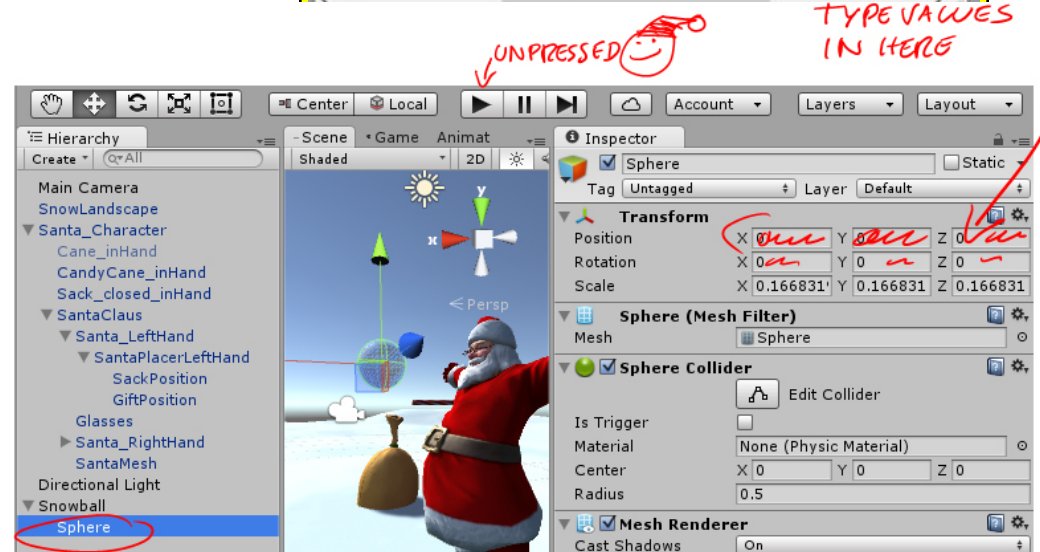
Or use old school pen an paper and write down the transform parameters (and for objects that are not spheres also write down the rotations).



8) Now **unpress Play** and the Snowball object unparents from GiftPosition and your sphere’s transforms are reset to zero.

Finally **type the snapshot transforms and rotation values into the now unparented sphere’s transform and rotation properties** and you have a snowball that you can place into Santa’s hand via script whenever you want and unparent it, simply by triggering the Santa Config variables via script or playmaker. And since the sphere is not parented directly to Santa’s hand you can still animate it independently or turn on physics on during gameplay. Very flexible.

Press **Play** to check if the ball is in Santa’s hand during anima-  
tion-- otherwise you have made a typing mistake :)



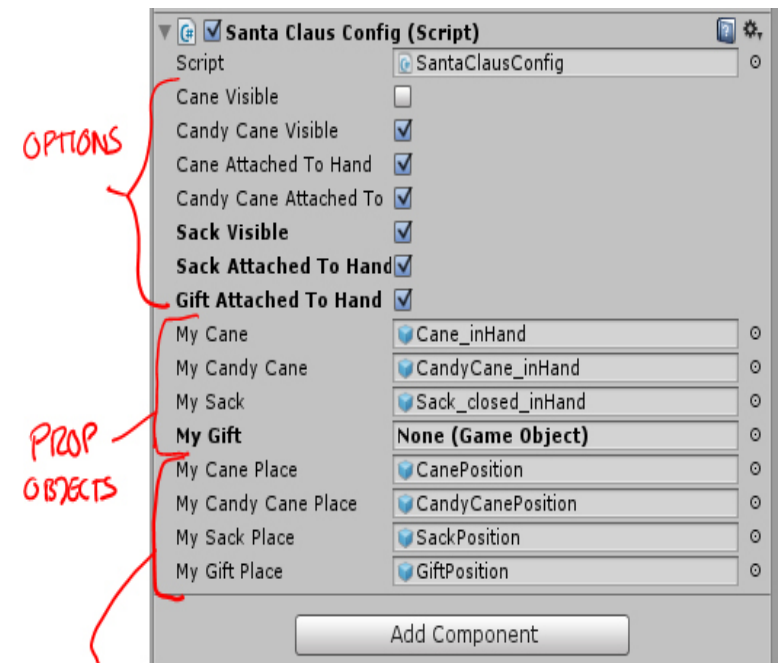
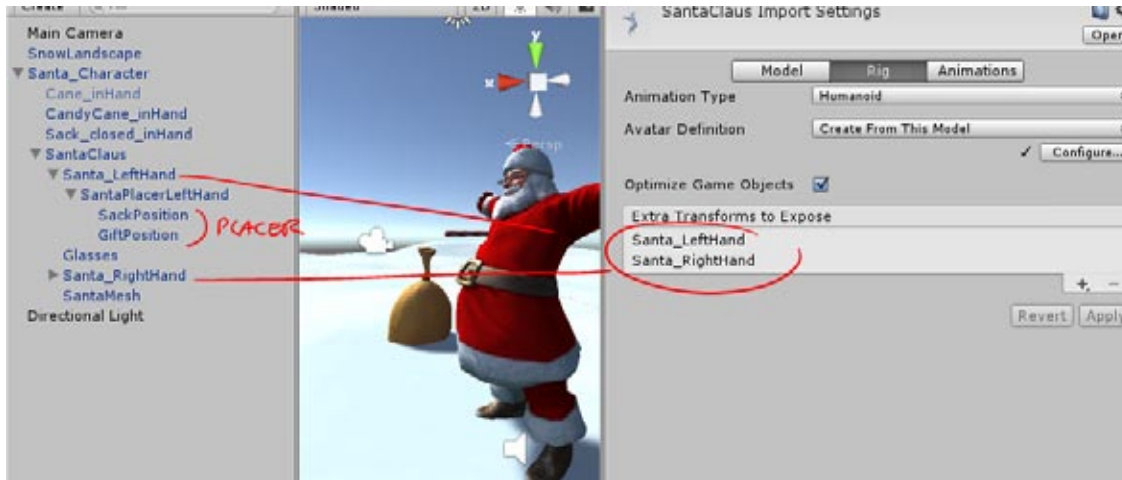
## 4: The Santa Config Script Component and using Santa Config with C# at runtime

The **Santa Claus Config** component is attached to the Santa\_Character game object of the Santa\_Character prefab.

The public variables names are pretty self explanatory (I hope).

**My Cane**, **My Candy Cane** etc. are links to the used prop models.

**My Cane Place** etc. are empty game object helpers to define the correct position for attaching the props. They are children of Santa's exposed hand game objects that I declared in FBX imports **Rig** tab.



The boolean options toggle the visibility and the parenting of the props.

If for example **Cane Attached To Hand** is active, when you press play the cane prop will be a child of the hand by using the helping placer gameobjects like shown in the previous chapter.

If the option is inactive the prop will be unparented.

You can make Santa leave his props behind if you want by using this unparenting variables. If you turn it on again the props will automatically be back in his hands.


You can manipulate the variables of Santa Claus Config with C# or via Hutong Playmaker (if you have that asset).

I have an extra PDF with a short tutorial included in the documentation for Playmaker users.

## Namespace:

Namespaces are a programming help to prevent conflicts between scripts of different origins because of using the same class names. I am using a lot of 3rd party assets in my own projects myself and conflicting class names can be really annoying.

To make sure that Santa does not conflict with other assets SantaClausConfig and the scripts from the demo folder are all running in namespace **SantaClaus**.

I also modified the Particle playground runtime used for the snow to run in SantaClaus namespace so that users who have Particle playground installed will not experience conflicts. 

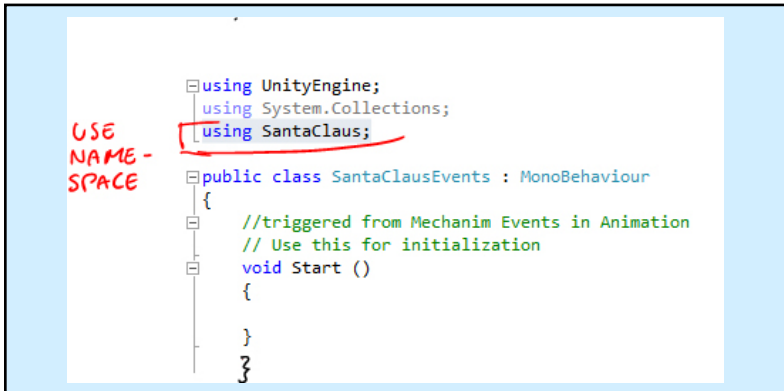
### because of that:

**If you want to reach SantaClausConfig from your own Scripts you have to tell your script to use the SantaClaus namespace.**

All you have to do is write the line

### using SantaClaus;

At the beginning of your script before the class is declared. You can check the SantaClausEvents script in the scripts folder for reference.



```
using UnityEngine;
using System.Collections;
using SantaClaus;

public class SantaClausEvents : MonoBehaviour
{
    //triggered from Mechanim Events in Animation
    // Use this for initialization
    void Start ()
    {
    }
}
```

You can then access the Script and it's properties like this in one of your own functions (in C#):

```
//find SantaConfig component and assign it to a variable

var mySantaConfig = GameObject.FindObjectOfType(typeof(SantaClausConfig)) as SantaClausConfig;

//change some of the scripts properties. Properties names are self explanatory. Look them up in the SantaConfig script.

mySantaConfig.SackVisible = true;
mySantaConfig.SackAttachedToHand = true;
mySantaConfig.CaneVisible = false;
```

Note that the variable names shown in the Inspector contain spaces. Unity does this automatically to make them easier to read but it can create confusion. **The real names of the variables in the script when you call them do NOT have spaces.** *My Cane* in Inspector is **MyCane** in one word in reality.

## 5: about animations

Santa has a **HumanIK** compatible skeleton so you can animate Santa with any program that supports this without rigging him manually. You find Santa's **animations** in 2 (or 3) fbx files in the Animation folder inside SantaClaus\_character.

Since the animations contain animated **blendshapes** and **animated extra joints** (in the hat and face) it is important to activate the mesh and the extra joints on import. That is why the animations that I created for Santa **all use the same SantaClaus mask**.

Santa is **Mechanim humanoid compatible**, so you can use all sorts of humanoid animations in Unity to enhance his motion set.

Santa has **relatively wide hips** for a man. If you use other mechainim animations it might be necessary to offset the shoulder rotations of the motion clips so that Santa's arms do not intersect the hips when moving.

One way to achieve this is to simply offset the other mechainim humanoid animation's avatar shoulder rotations at the import settings-> Rig->Configure->Mapping.

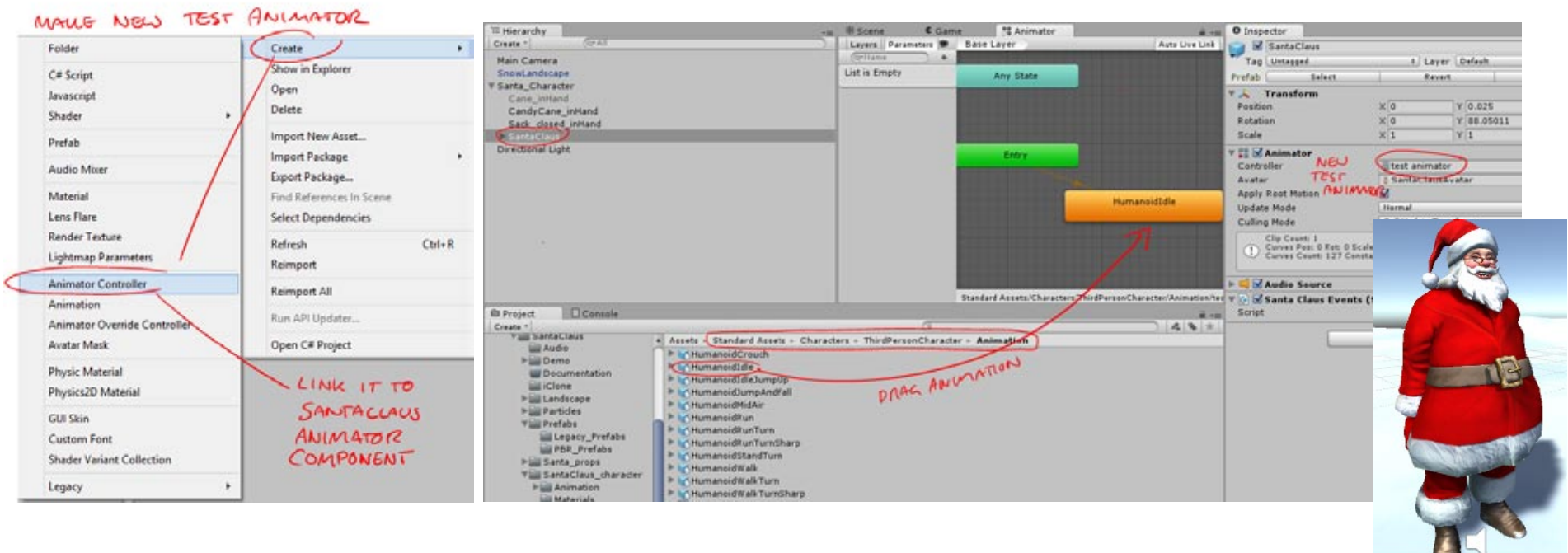
I am showing this here in a Mini- tutorial with an animation file from the free Standard Assets 1.0 from the Unity asset store:

1) Import Unity standard assets (or only the Characters->ThirdPersonCharacters->Animations from the standard asset) and the Santa asset into a new project, Make a new scene and place Santa (and the landscape) from the prefabs in the scene.

2) Create a **new animator** for test purposes (right click in Project ->Create->Animator Controller) and name it "**Test Animator**".

**Break Santa\_Character prefab instance** (menu Game Objects-> Break Prefab Instance) for test purpose and apply Test Animator to the Animator component on Santa-Claus.

3) Open the Animator from the Window menu and drag the **Humanoid Idle Motion** from the standard assets into the Animator Base Layer. It will automatically become the active animation as it is the only one. Press play and watch Santa's Arms go through his hips.



4) One way to correct this is to **change the shoulder rotations of the avatar** of the animation that you want to retarget to Santa. If you use the same motion for other characters make a copy of the motion file first or you will change also the other characters animations.

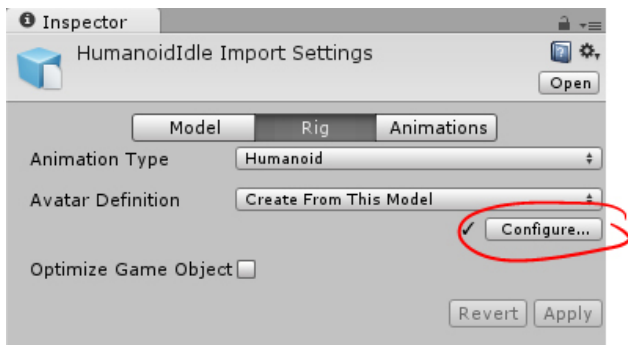
Select HumanoidIdle.fbx from the standard assets and in inspector go to the Rig Tab and press configure.

Then in **Mapping** select the shoulders and **change Z rotation** for both arms in transforms until it looks something like the screenshot. Depending on the nature of your animation it might also be a good idea to manipulate the hands so that they are parallel to the ground again. Unity will tell you the character is not in T-Pose. Ignore that and press Apply and Done, then press Play and check the animation on Santa.

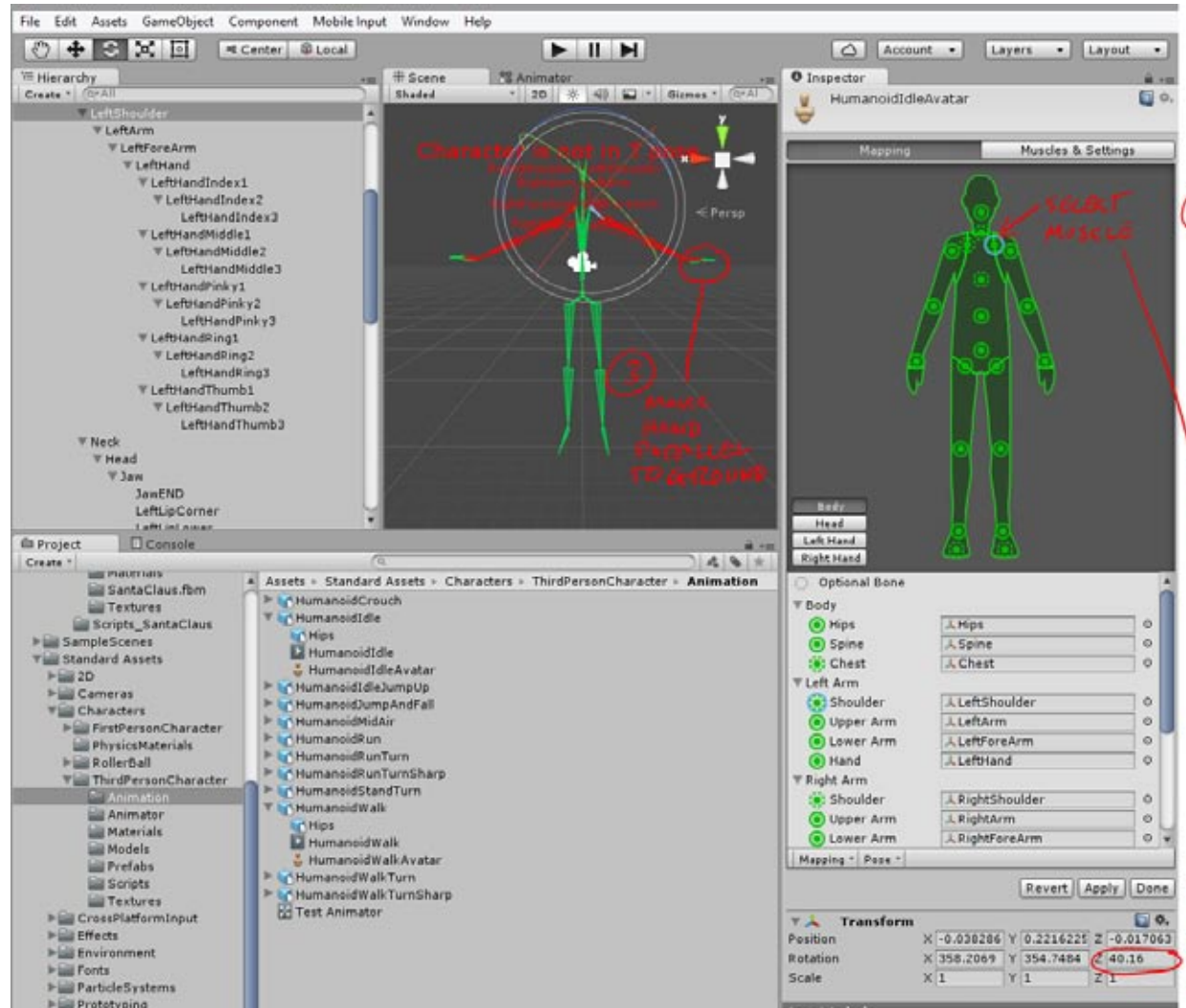
The arms should be much better now. Finetune the offsets to make it perfect.

In my case I also rotated the hands X axis so that the hands are really parallel to the ground.

You might also want to try to split up the offset of the rotation Z by offsetting the shoulders and also the upper arms and see if that looks more natural. In the end it depends on the nature of the motion you want to retarget. If it looks right when playing it is right for your project.



WITH ENFORCE T-POSE YOU CAN RESET YOUR OFFSETS WHEN NOT SATISFIED.



CHANGE ROTATION

## 6: about iClone usage

iClone is an autorigging and animation program. I like it because of its speed and superb retargeting capabilities between all sorts of humanoid animation files.

Included in the asset is an iClone avatar file, so if you own iClone and the 3dExchange Pipeline PlugIn you can animate and lipsync Santa in iClone and export additional custom Santa animations to Unity.

Simply unzip the avatar file and open it in iClone and you can begin animating right away.

Santa has a Spring constraint setup for the hat.

Special facial features can be animated via the Facepuppet 3DX\_Custom Face Profiles 1 to 4.

When you export your new Santa animations with 3d exchange pay attention to the following:

To export the animations and be compatible with this asset, choose the **Motionbuilder preset** in 3Dexchange FBX export (not the Unity preset).

Include the mesh data to make sure the facial animations are also exported (if you have made use of that).

Make sure the exported file's name is "SantaClaus" and nothing else.

You can rename the file as soon as it is written to the hard disk to any name you like.

Just make sure its name is "SantaClaus" on export.

The reason is that 3dExchange automatically encapsulates the FBX export in a nullobject with the name of the file and Unity will not be able to map the extra joint and blendshape animations if the name of this null object is different.

When you import your file in Unity, choose humanoid and assign the "SantaClausAvatar" as the source for Avatar Definition.

Use the SantaClaus mask with each motion to make sure all extra joint and face animations will be mapped.



## 7: about optimizing Santa for games

Santa's diffuse texture has been created with heavy padding, You can scale it down to 256 x 256 pixel dimensions and it won't show seams. Of course details will get lost, but depending on how you want to use him it might be a nice way of saving memory.

Santa's textures and prop meshes have not been combined to make the asset as flexible as possible. If you want to optimize your project for mobile it might be a good idea to use a mesh combination and texture atlas tool to reduce the draw calls. There are plenty of solutions in the asset store for this, even good free ones.

I recommend simpleLOD, it is a very powerful asset to optimize and combine meshes and textures and it is simple to use.



256 x 256  
DIFFUSE  
AND  
NORMAL MAP  
AND  
STILL  
AN  
ACCEPTABLE  
SANTA  
(IF YOU  
DON'T DO  
CLOSE UPS :))