MultiFPS – Project workflow

This document describes the structure of MultiFPS, naming conventions for assets, and how to work with animated models.

Contents

AnimatorControllers and AnimatorOverrideControllers

There are two base animator controllers that MultiFPS uses.

1. Item_BaseController – as name suggests this controller is a base for items animator controllers, they override this controller with their own animations. For example, in one animator for knife in place where we have stabbing animation we can have in another animator melee attack with gun like pistol, sniper shotgun etc. So, we can launch those different animations in different weapons using the same code and we don't have to create every time each animator controller from scratch for each weapon. But we Can do this if we want, when weapon is much different than the rest and does not match our base animator controller. Shotgun has its own controller since it is reloaded differently than the rest of the guns, so it needed a different approach to animating.

First person perspective arms model also uses the same animator controller as a base, since arms animations in FPP directly correspond to item animations, weapon is being reloaded, so hands need to play performing reload animations.

Naming convention: Item_ItemName, for example "Item_pistol", "Item_rocketLauncher"

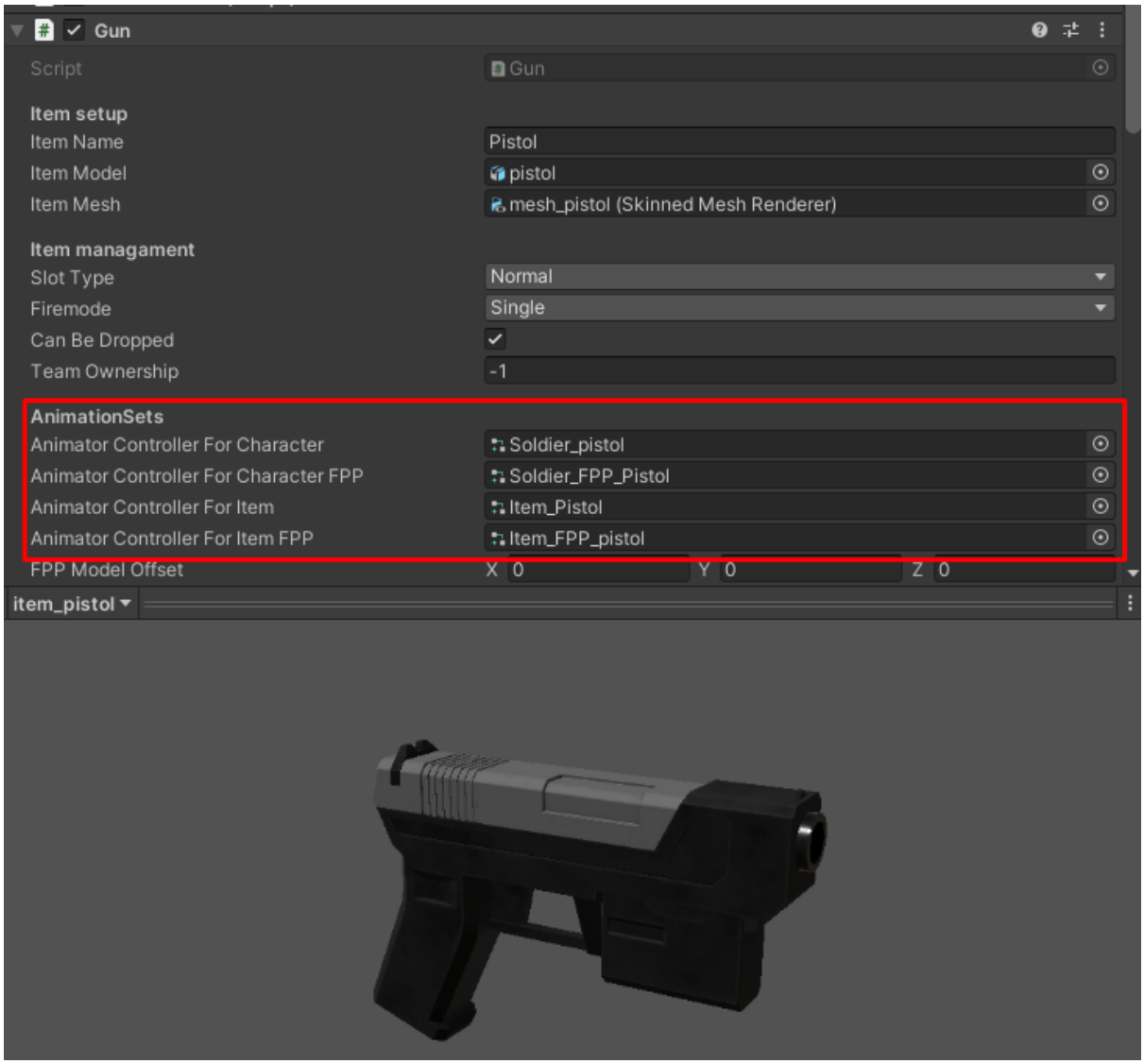
Naming convention for items viewed from first person perspective: Item_FPP_ItemName, for example: "Item_FPP_pistol", "Item_FPP_rocketLauncher"

2. Soldier_EmptyHanded – it is base controller for humanoid character model. It contains slots for animating walking, crouching, running, looking up/down (to achieve that, animation skeleton is separated in to two parts using AvatarMask). It also has a slot for receiving damage, punching and reloading item animations. Items can be placed in character animator AnimatorOverrideMask that overrides Soldier_EmptyHanded controller with its own unique animations. Thanks to that each item has different melee attack animation, reload animations, looking up/down animations, and it is also possible to change movement animations dependent on item for example heavy item could implement heavier walking animations, but this is not used anywhere for now.

Naming convention: Soldier_ItemName , for example "Soldier_pistol", "Soldier_rocketLauncher"

In this example "Soldier_Pistol" is controller that overrides "Soldier_EmptyHanded" controller with animations specific to pistol.

Each of those animator controllers are assigned to item prefab, like in this screenshot:



Here for example we have prefab for pistol, and in section “AnimationSets” it stores animator contollers for:

Animator Controller For Character – Animator that humanoid character model will use to be animated appropriately to pistol

Animator Controller For Character FPP – Animator that FPP arms model will use to be animated for using pistol

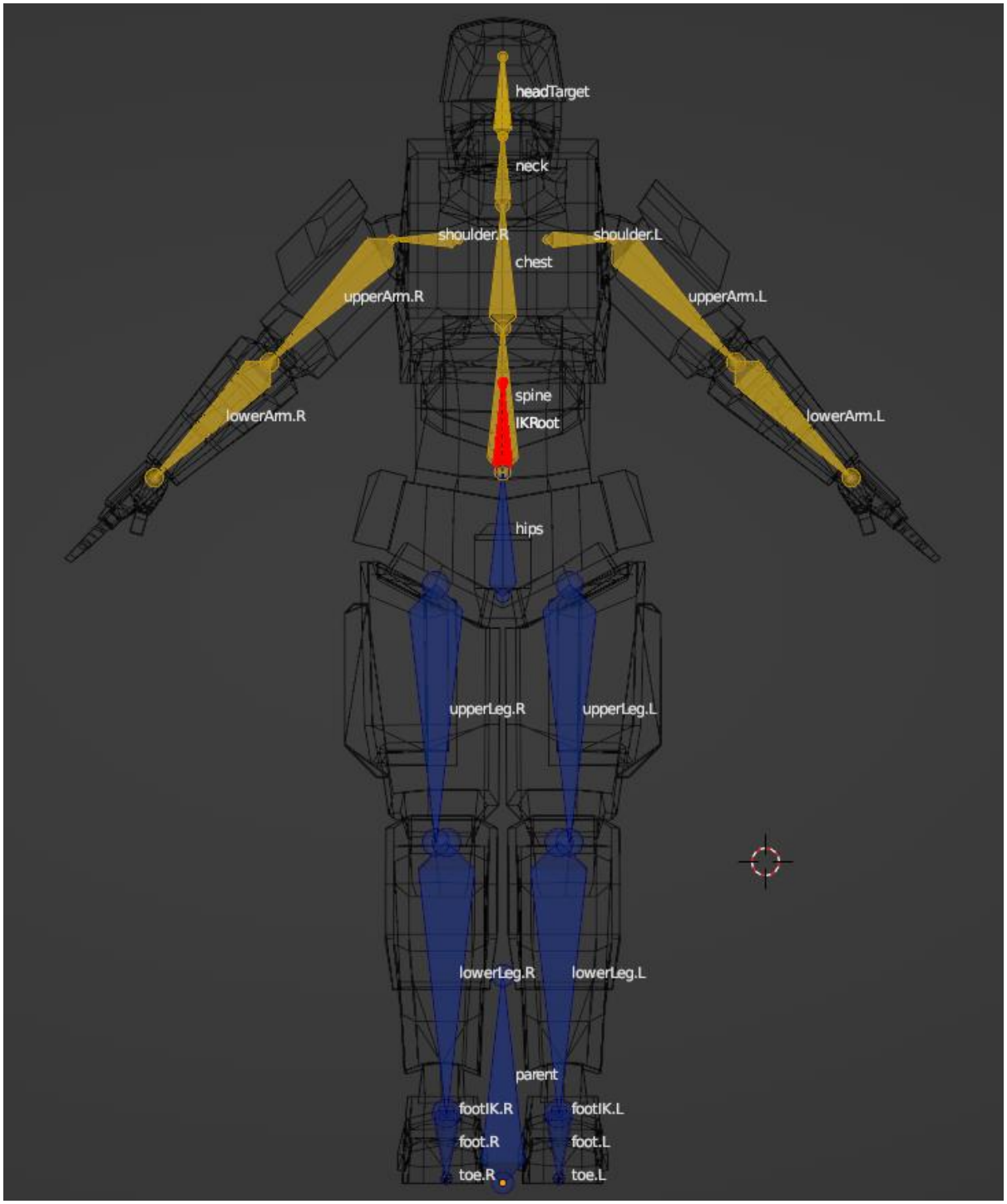
Animator Controller For Item – Animator controller for item being hold by full humanoid model, so here we just animate moving parts when player shoots or reloads

Animator Controller For Item FPP - Animator controller for item being hold in first person perspective by arms model.

Character animations

Blender files for all animated assets are included in this package in file “MultiFPS source files.zip”

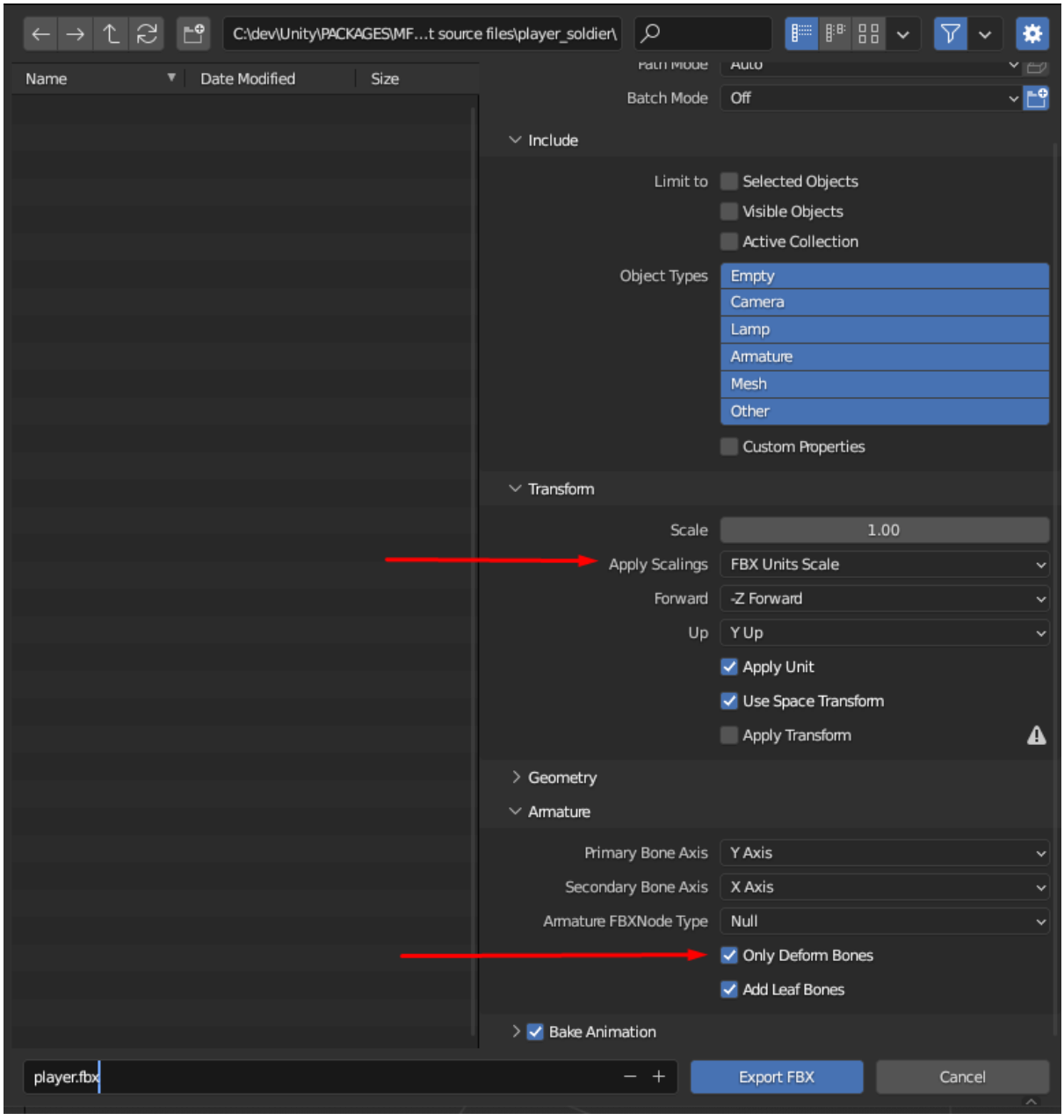
Character models use custom rig, which is different than humanoid rig due to existence of one additional bone, which is colored in red on image below. This bone is constraint to never rotate. Thanks to that we can animate upper body independently from lower body and we can achieve similar results to if we were using IK system, but without using IK. So for example for every weapon we can use the same movement animations, and only have special animations for upper body for each item.



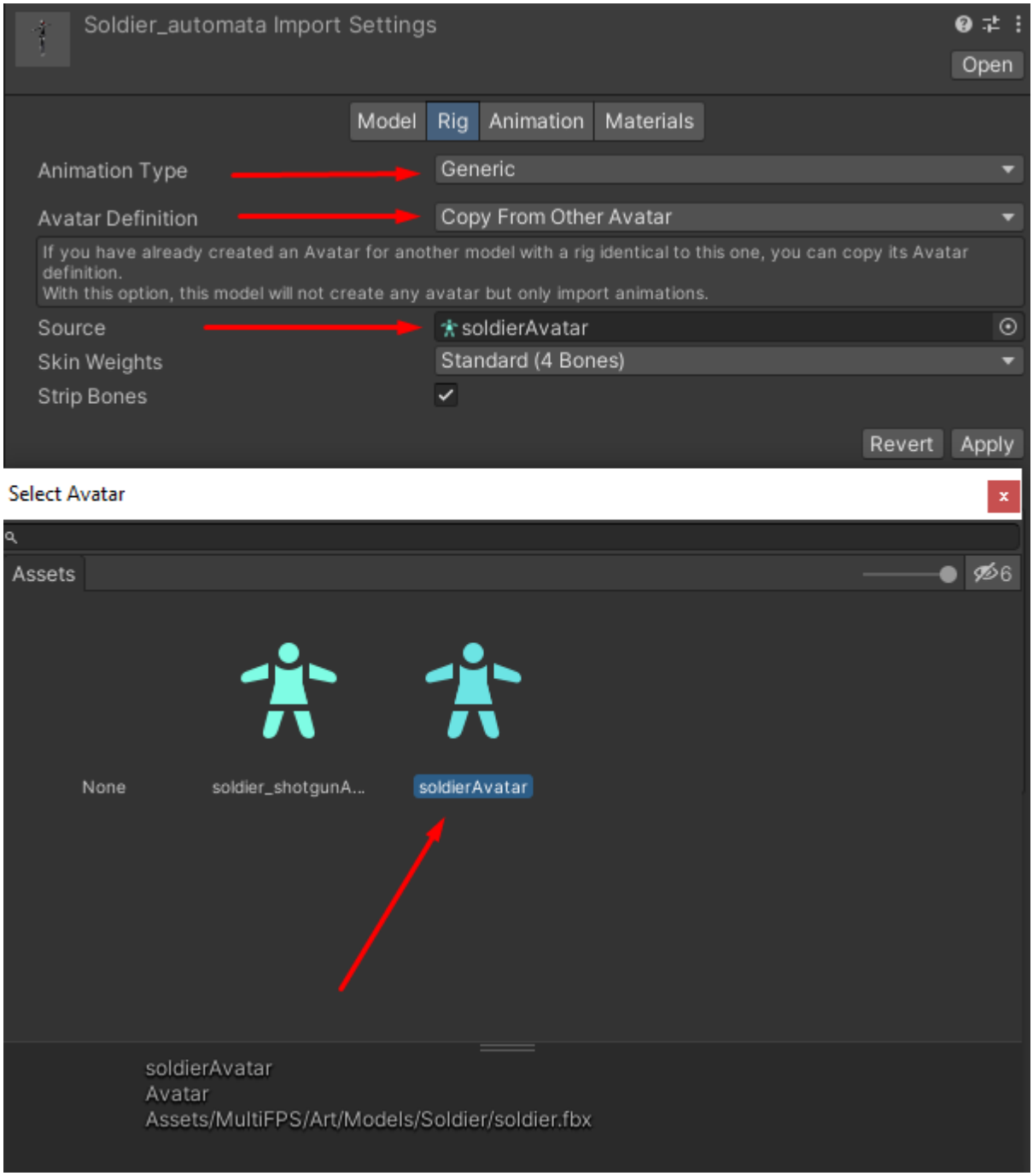
In Unity, file “soldier_AvatarMask_upperBody” has selected all of the upper body bones, colored with yellow on image above, and this file is assigned to described previously Soldier_EmptyHanded animator controller so it can differentiate between upper and lower body bones.

Exporting animated assets

If You want Your custom model to work with MultiFPS animations, they must have same rigs, and when exporting, set option “Apply Scalings” to “FBX Units Scale”, and check option “Only deform bones” in Armature option group. We don’t need IK bones in unity engine.



If You exported character, then later in Unity select Your FBX model, go to Rig tab, select animation type as “Generic”, then Avatar definition as “Copy from other avatar”, and as a source select existing “soldierAvatar” file. Thanks to that MultiFPS character animations will work with Your character model.



Thank You for purchasing MultiFPS

Contact:

Email: desnetware@gmail.com

Website

Youtube channel

AssetStore publisher page