SWIMMING MOTION PACK

9/23/2016

Latest Documentation:
http://www.ootii.com/Unity/MotionPacks/Swimming/SwimmingUsersGuide.pdf

IMPORTANT

The Swimming Motion pack **requires** the following:

- Motion Controller v2.25 or higher
- Mixamo's free swimming animations (using Y Bot)

Contents

Demo Quick Start

This quick start is simply to get the demo up and running in a self-contained project.

1. Start a new Unity 5 Project.

2. Download and import the Motion Controller asset.

3. Download and import the Swimming Motion Pack asset.

4. Download Mixamo's animations using "Y Bot" (see this flow).

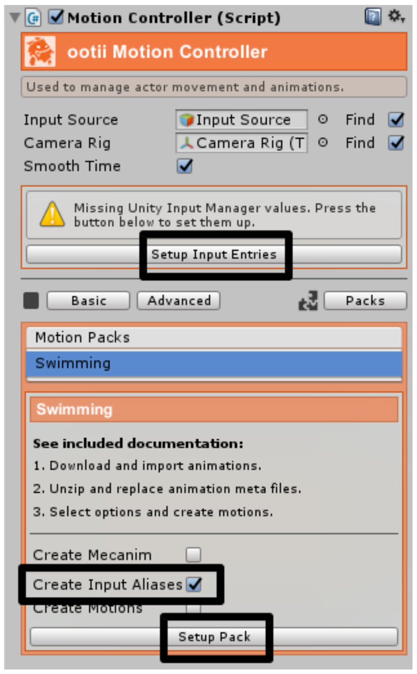
5.Unzip animations to project's Mixamo folder.
...\Assets\ootii\MotionControllerPacks\Swimming\Content\Animations\Mixamo

6. Unzip AnimationMeta.zip from pack's "Extras" folder to project's Mixamo folder.
...\Assets\ootii\MotionControllerPacks\Swimming\Content\Animations\Mixamo

7. Let Unity import the animations and meta data. Then, close and re-open Unity.

8. Open the "SMP_Demo" demo scene.
...\Assets\ootii\MotionControllerPacks\Swimming\Demos\Scenes\

9. Setup input entries on the Packs tab.


Motion Controller (Script)
ootii Motion Controller
Used to manage actor movement and animations.
Input Source: Input Source — Find ✓
Camera Rig: Camera Rig (T — Find ✓
Smooth Time: ✓
Missing Unity Input Manager values. Press the button below to set them up.

Motion Packs
Swimming
Swimming
See included documentation:
1. Download and import animations.
2. Unzip and replace animation meta files.
3. Select options and create motions.
Create Mecanim: ☐
Create Input Aliases: ✓
Create Motions: ☐

10. Press play.

DEFAULT CONTROLS

WASD-QE	= Move / swim
Shift	= Move / swim fast
Space	= Dive and exit
RMB	= Look

Custom Quick Start

This quick start assumes you've worked with the Motion Controller and that you have a Motion Controller enabled character in your scene already.

1. Open your MC enabled project and scene.

2. Download and import the Swimming Motion Pack asset.

3. Download Mixamo's animations using "Y Bot" (see this flow).

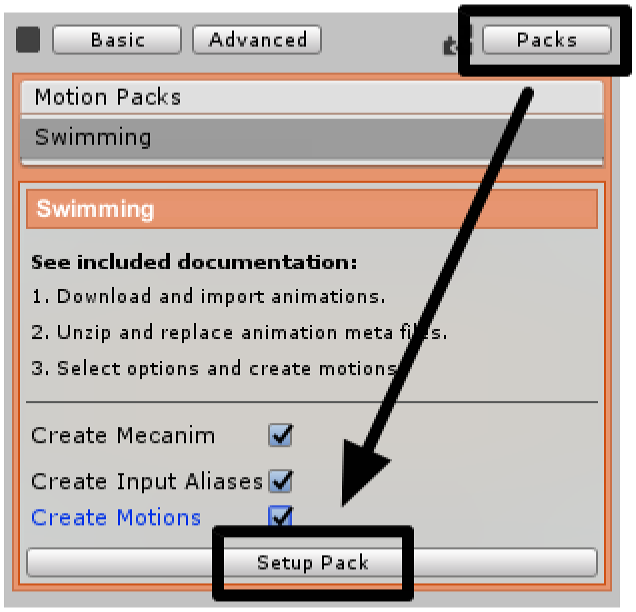
4.Unzip animations to project's Mixamo folder.
...\Assets\ootii\MotionControllerPacks\Swimming\Content\Animations\Mixamo

5. Unzip AnimationMeta.zip from pack's "Extras" folder to project's Mixamo folder.
...\Assets\ootii\MotionControllerPacks\Swimming\Content\Animations\Mixamo

6. Let Unity import the animations and meta data. Then, close and re-open Unity.

7. Open the scene.

8. Setup motion pack on the Packs tab.



See included documentation:
1. Download and import animations.
2. Unzip and replace animation meta files.
3. Select options and create motions.

Create Mecanim ✓
Create Input Aliases ✓
Create Motions ✓

9. Swimming motions are added to the Advanced tab.

10. Add water plane and set the layer to "Water" and ensure the plane has a collider.

Overview

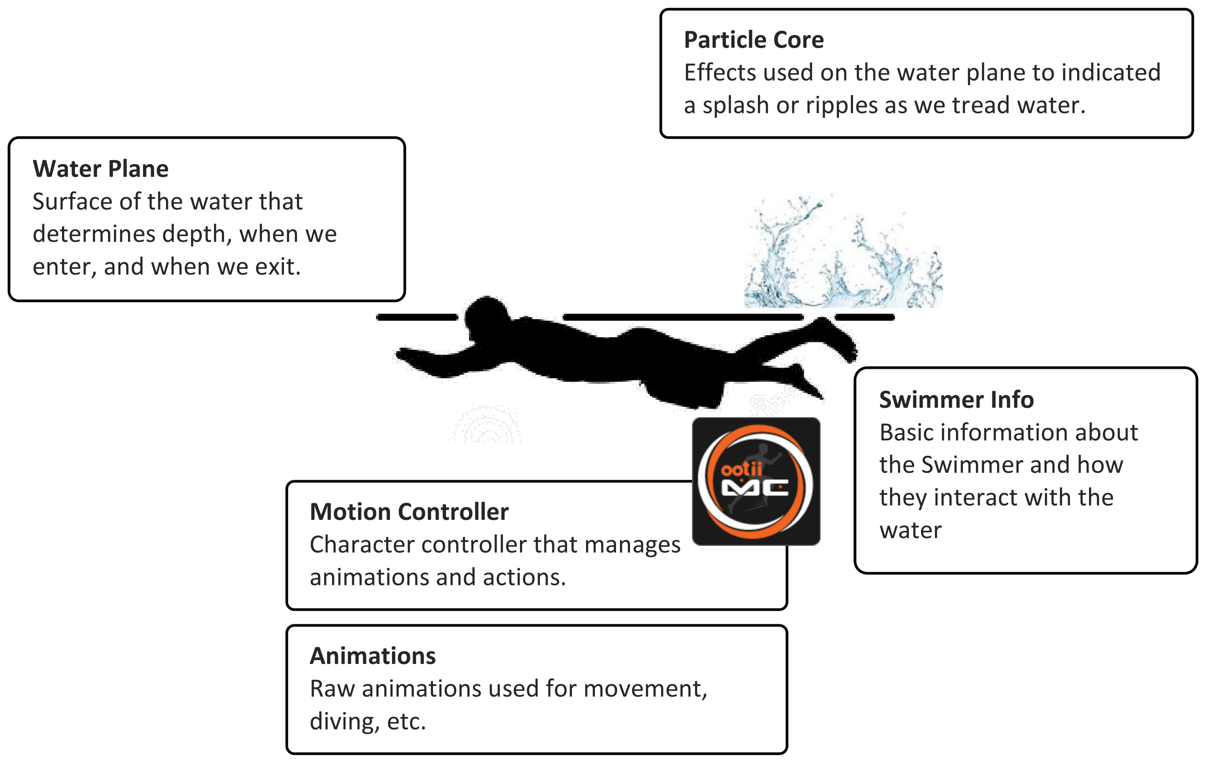
This asset is a bit different from my other assets as it's an add-on pack to the Motion Controller. So, the Motion Controller is required for this asset to work. It also requires Mixamo's swimming animations. When those are combined with this asset, your character gains the ability to dive, swim, and exit the water.

I've tried to create this asset in a very modular way. This way, you can use it as-is or customize it to work with your game. For example, you can use my particle effects or replace them with your own.

If you think this add-on is missing a key feature or option, let me know.

Key Components

With assets like this, there is typically several pieces that work together. This is a quick overview of those pieces and I'll go into them in more detail:



Particle Core
Effects used on the water plane to indicated a splash or ripples as we tread water.

Water Plane
Surface of the water that determines depth, when we enter, and when we exit.

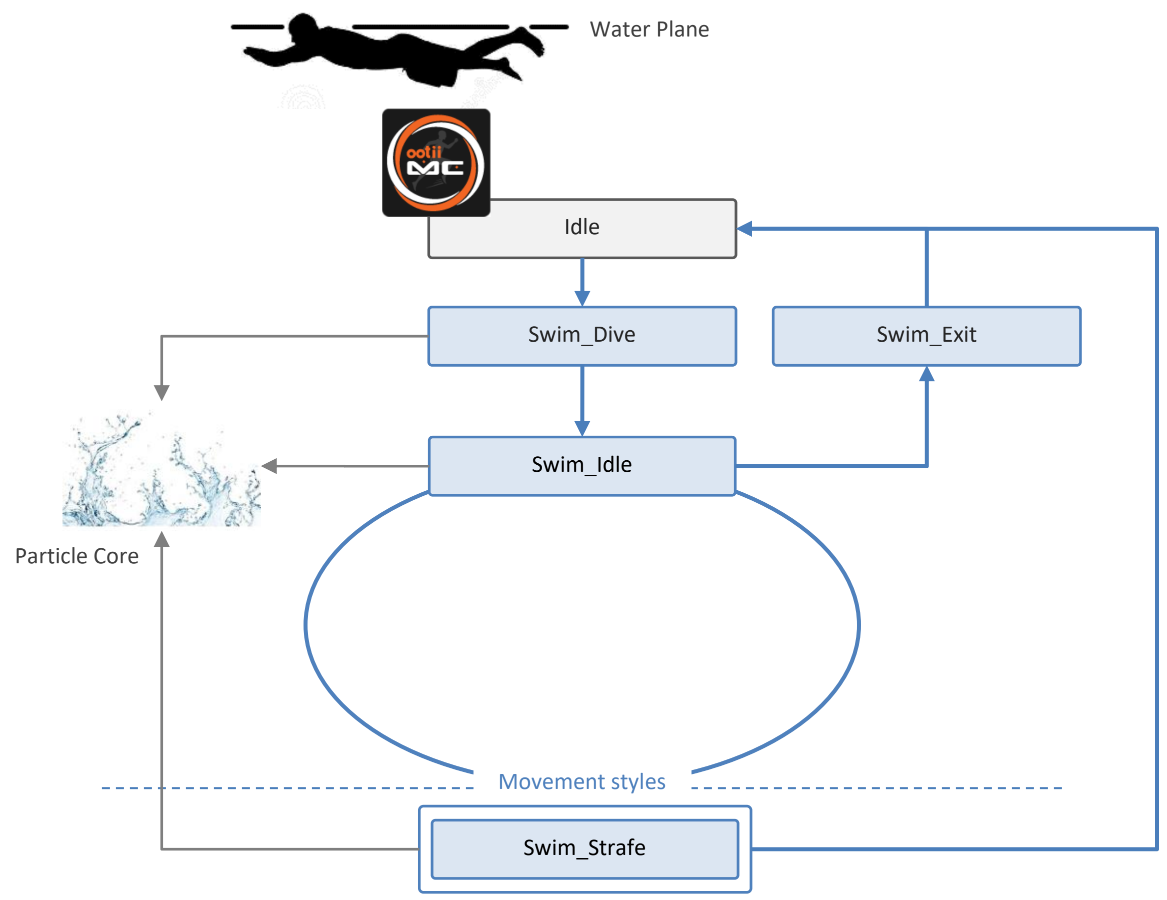
Swimmer Info
Basic information about the Swimmer and how they interact with the water

Motion Controller
Character controller that manages animations and actions.

Animations
Raw animations used for movement, diving, etc.

Process Flow

Just like the motions that come with the Motion Controller, the Swimming Motion Pack is composed of several motions. These motions (blue rectangles below) work together to create the full range of capabilities.



If you don't need a swim motion, want to change how a motion works, or want to add an additional swim motion... you do it just like any other motion. These swim motions are just custom motions that take advantage of Mixamo's animations.

I've created this pack based on how I think swimming should work. As you can imagine, there are lots of ways we could do it and your game may be different than mine. If you want the motions to act differently, you are welcome to change these motions or shoot me an email and there may be a feature or option I can add.

Motion Controller Motions

These are the motions included with the Swimming Motion Pack. This is just a brief description and I'll go into the properties of each motion in the Motion Details section.

Swim – Idle: Simple idle animation while swimming.

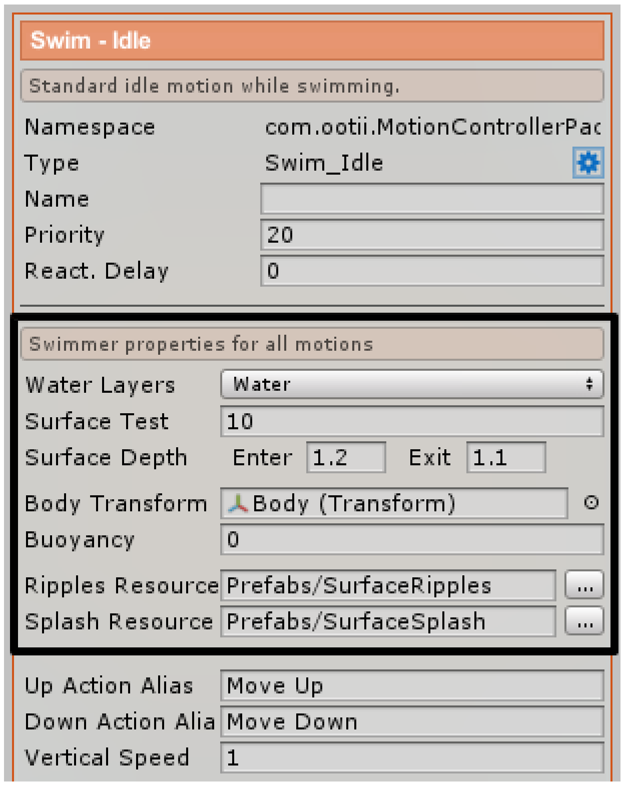
Swim – Dive: Used to dive into the water vs. simply walking in.

Swim – Exit: Used to climb out using an edge vs. simply walking out. This is great for pools, climbing onto rafts, etc.

Swim – Strafe: Movement motion while swimming. This motion actually blends between an upright slow swim and a forward facing "frog kick". I use these because those are the animations Mixamo gives away for free. You can replace these animations if you have something better.

Swimmer Info

In order to centralize information about the swimmer, I've created a "SwimmerInfo" object that stays with the character. It's actually part of the "Swim – Idle" motion and you can use the motion properties to set its values:



Swim - Idle

Standard idle motion while swimming.

Namespace	com.ootii.MotionControllerPac
Type	Swim_Idle
Name	
Priority	20
React. Delay	0

Swimmer properties for all motions

Water Layers	Water
Surface Test	10
Surface Depth	Enter 1.2 Exit 1.1
Body Transform	Body (Transform)
Buoyancy	0
Ripples Resource	Prefabs/SurfaceRipples
Splash Resource	Prefabs/SurfaceSplash
Up Action Alias	Move Up
Down Action Alia	Move Down
Vertical Speed	1

By putting the swimmer properties here, we can define these values once and not have to set them for each motion.

In addition to the properties, you can also use SwimmerInfo functions to get things like the actor's current depth and to test if the character is in shallow water.

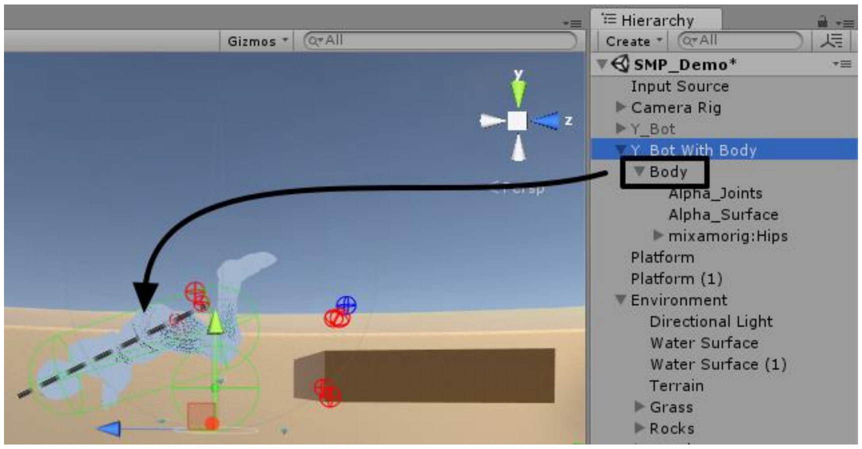
To get the SwimmerInfo object from outside the motion, you can do this:

```
SwimmerInfo lInfo =
SwimmerInfo.GetSwimmerInfo(gameObject.transform);
```

Each of the properties has a tool-tip that is self-explanatory. So, I won't go into them here. The one exception is "Body Transform"

Body Transform

When swimming up or down, we don't actually want to tilt the whole character controller because that can throw cameras off and the controls can become unwieldy. So, this version of swimming keeps the character controller upright and provides a way to rotate a separate "body" transform.



While not required, if you use a transform to hold the skeleton and body mesh, I'll rotate the transform as you're swimming up and down.

Notice in the image that the whole character is still lined with the world's up axis. It's the "Body" child transform that is pitched down since we're moving down.

In order for this to work, the "body" must be the parent of the skeleton and the body mesh. Otherwise, the skeleton's animation will counter my rotation.

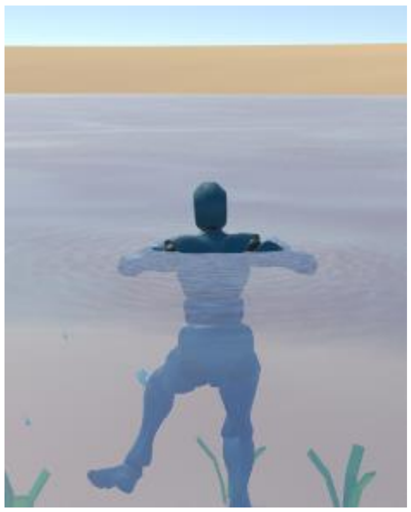
"Cores" Introduction

A "core" represents the heart-beat of a character or object. Typically, this is the MonoBehaviour whose Update() function controls the object. For this pack, we have one specific core:

Particle Core – This core will be a component on your particle effects and it is used to auto-destroy the effect once it's done playing.

This works great for splashes and ripples because we can instantiate a splash object and forget about it. Once the splash instance finishes the effect, the core will remove itself.

A particle core is used on the "SurfaceRipples", "SurfaceSplash", and "Underwater" prefabs. The first is used for rippling the water while on the surface. The second is for the actual splash.

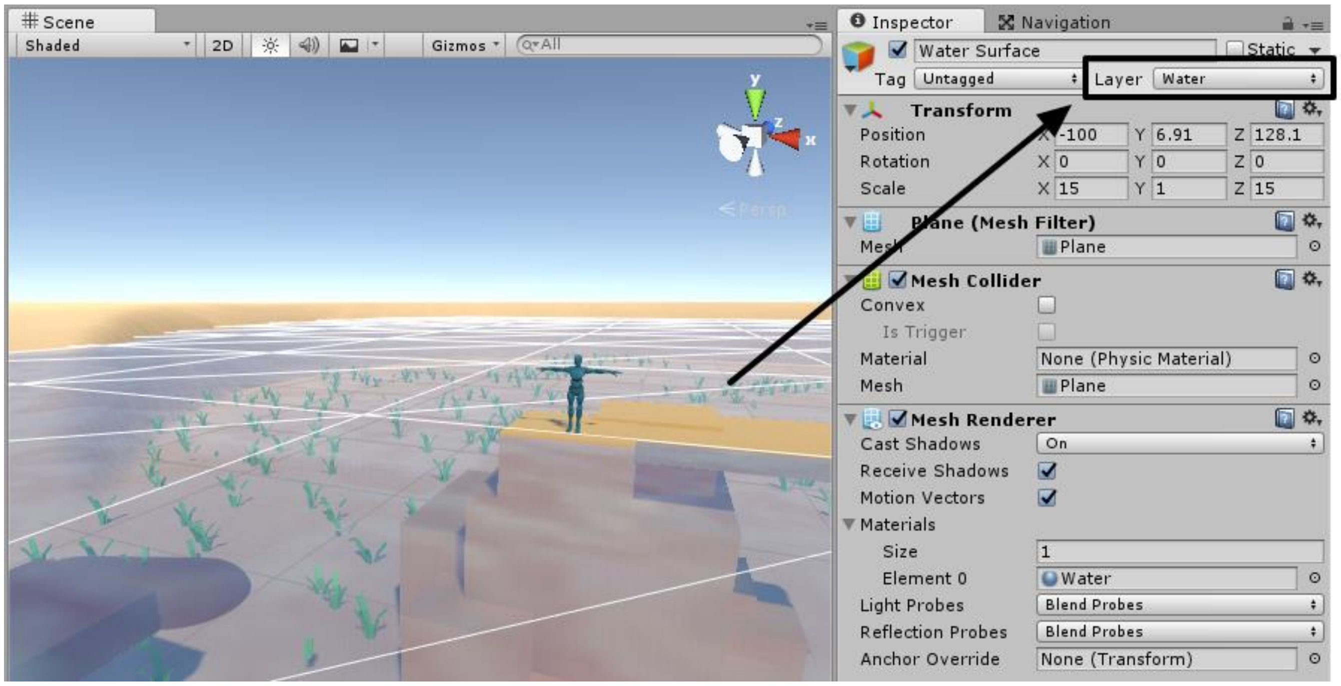


In addition to the particle effects, some of the cores include sound clips. You can modify the core textures, materials, and/or sound clips.

Prefabs:	Assets\ootii\MotionControllerPacks\Swimming\Content\Resources\Prefabs
Textures:	Assets\ootii\MotionControllerPacks\Swimming\Content\Textures
Materials:	Assets\ootii\MotionControllerPacks\Swimming\Content\Materials
Sound Clips:	Assets\ootii\MotionControllerPacks\Swimming\Content\Audio

Water Plane

The water plane is just that… a simple Unity Plane that you add to your scene.


Water Surface
Tag Untagged
Layer Water

Transform
Position X -100 Y 6.91 Z 128.1
Rotation X 0 Y 0 Z 0
Scale X 15 Y 1 Z 15

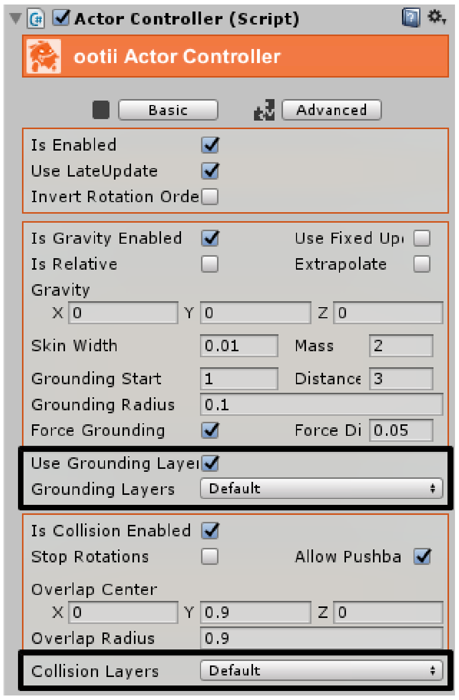
Plane (Mesh Filter)
Mesh Plane

Mesh Collider
Convex ☐
Is Trigger ☐
Material None (Physic Material)
Mesh Plane

Mesh Renderer
Cast Shadows On
Receive Shadows ✓
Motion Vectors ✓
Materials
Size 1
Element 0 Water
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)

There's nothing special about this plane other than setting its "Layer" property to "Water". This will match the "Water Layers" property that you set on the SwimmerInfo object.

While the plane does need a collider, you can set any material you want.


Actor Controller (Script)
ootii Actor Controller
Basic Advanced

Is Enabled ✓
Use LateUpdate ✓
Invert Rotation Orde ☐

Is Gravity Enabled ✓ Use Fixed Up ☐
Is Relative ☐ Extrapolate ☐
Gravity X 0 Y 0 Z 0
Skin Width 0.01 Mass 2
Grounding Start 1 Distance 3
Grounding Radius 0.1
Force Grounding ✓ Force Di 0.05
Use Grounding Layer ✓
Grounding Layers Default

Is Collision Enabled ✓
Stop Rotations ☐ Allow Pushba ✓
Overlap Center X 0 Y 0.9 Z 0
Overlap Radius 0.9
Collision Layers Default

Ensure that your Actor Controller has "Use Ground Layers" checked and you are NOT using the "Water" layer for Grounding Layers or Collision Layers.

This will allow your character to recognize that he's breaking the surface of the water and still pass through it.

Customizations & Code

As with all motions, you can access properties through code in order to change how the MC works on the fly. You can also get notified of specific events (i.e. when you enter and exit swimming).

Notifications

To get notified when your character enters and exits swimming, you can either assign a function handler or have the Event System – Dispatcher asset send a message. The second option is only valid if you won the Event System – Dispatcher.

Function Handler

The SwimmerInfo object has two handlers you can tap into: OnSwimEnter and OnSwimExit

```
using UnityEngine;
using com.ootii.MotionControllerPacks;

public class dev_SwimmingDemo_Code : MonoBehaviour
{
    void Start()
    {
        SwimmerInfo lInfo = SwimmerInfo.GetSwimmerInfo(gameObject.transform);
        lInfo.OnSwimEnter = OnSwimEnter;
        lInfo.OnSwimExit = OnSwimExit;
    }

    void OnSwimEnter(SwimmerInfo rSwimmer)
    {
        // Do stuff when the swimmer enters the water to swim
    }

    void OnSwimExit(SwimmerInfo rSwimmer)
    {
        // Do stuff when the swimmer exits the water from swimming
    }
}
```

Event System – Dispatcher

If you own the Event System – Dispatcher, I can send message through it. Simply open the SwimmerInfo.cs file and uncomment the first line. You're changing the code from this:

```
//#define USE_MESSAGE_DISPATCHER
```

to this:

```
#define USE_MESSAGE_DISPATCHER
```

Once done, the SwimmerInfo object will send "SWIM_ENTER" and "SWIM_EXIT" message through the dispatcher to whoever is listening.

Code

Here are some common things you may want to do through code in order to control how your swimmer behaves.

Get SwimmerInfo

```
SwimmerInfo lInfo = SwimmerInfo.GetSwimmerInfo(gameObject.transform);
```

Determine if your character is swimming and how

```
SwimmerInfo lInfo = SwimmerInfo.GetSwimmerInfo(gameObject.transform);
if (lInfo.IsInWater)
{
    // Your code here
}

if (lInfo.IsInShallowWater)
{
    // Your code here
}

if (lInfo.IsAtWaterSurface)
{
    // Your code here
}
```

Determine depth

```
SwimmerInfo lInfo = SwimmerInfo.GetSwimmerInfo(gameObject.transform);
float lDepth = lInfo.GetDepth();
```

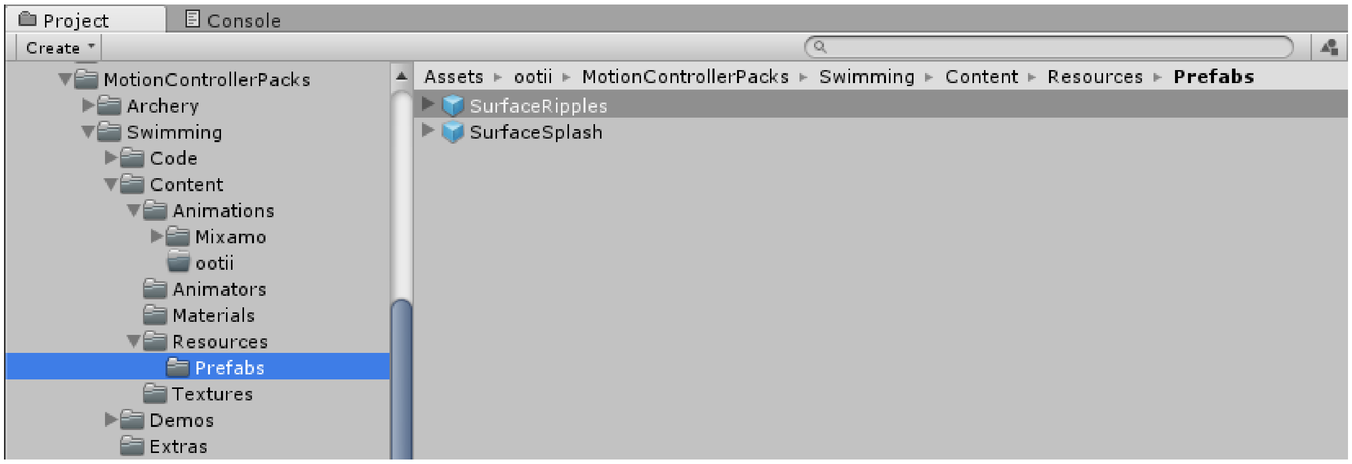
Force the character to swim (or stop swimming)

```
SwimmerInfo lInfo = SwimmerInfo.GetSwimmerInfo(gameObject.transform);
lInfo.EnterWater();
lInfo.ExitWater();
```

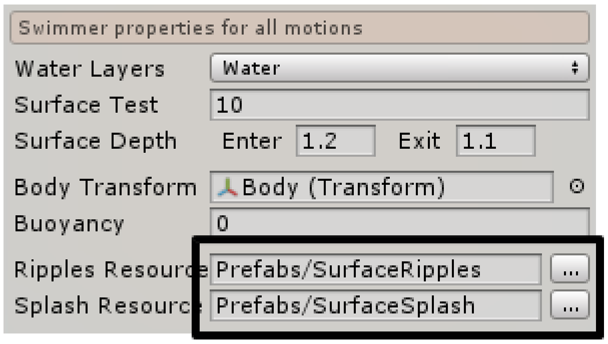
Particles

I mentioned earlier that this pack uses several types of particles: ripples, splashes, and bubbles. You can modify these particle effects as you want or create totally different ones.

Simply open the prefabs and modify the Unity Particle Systems as needed.



You'll notice that I reference these prefabs as "resources" in the SwimmerInfo properties on the Swim – Idle motion:


Swimmer properties for all motions

Water Layers	Water
Surface Test	10
Surface Depth	Enter 1.2 Exit 1.1
Body Transform	Body (Transform)
Buoyancy	0
Ripples Resource	Prefabs/SurfaceRipples
Splash Resource	Prefabs/SurfaceSplash

The reason I do this as resources is so you can swap out the effects on the fly or create your own effects and use them instead. It also makes it easier to instance the prefab at run-time.

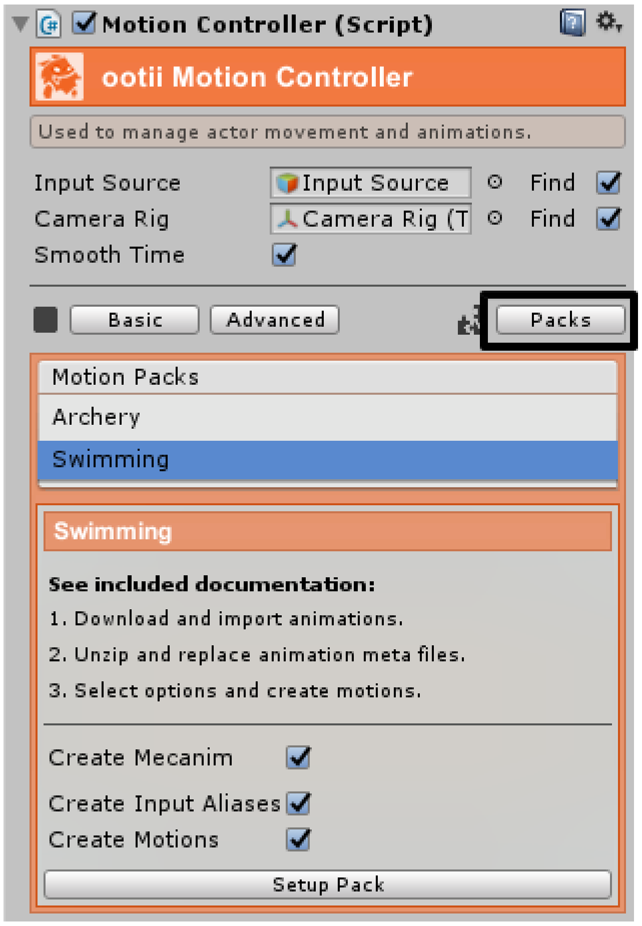
Motion Packs

The Motion Controller UI has changed slightly and now includes a button for “Packs”.

When the “Packs” button is pressed, the Motion Packs imported into the project will be listed. Selecting a pack will provide detailed information about the pack.

In this case, we select the “Swimming” pack and the Swimming Motion Pack details are listed.

Each pack may show different options. For the Swimming Motion Pack, I’ll go through each option. When the “Setup Pack” button at the bottom is clicked, the checked options are run.



See included documentation:
1. Download and import animations.
2. Unzip and replace animation meta files.
3. Select options and create motions.

Create Mecanim	✓
Create Input Aliases	✓
Create Motions	✓

Setup Pack

Create Mecanim – This option is used to create/recreate the Mecanim sub-state machines that are used by the pack’s motions. This is exactly what we do with normal motions. I just create a short cut here.

Create Input Aliases – When checked, all the required input aliases used by the motions are created in Unity’s Input Manager. This only needs to be done once for the whole project.

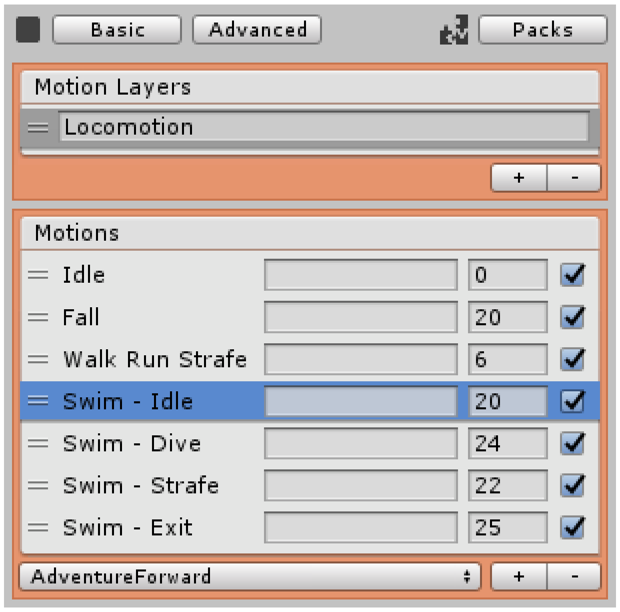
Create Motions – This is the object that actually creates the swim motions.

Typically, you’ll just leave all these options checked before pressing “Setup Pack”. However, as you customize your character and include other assets you may no longer need to re-create the Mecanim sub-state machines.

The following pages show what is created by the pack (assuming all options are checked):

Motion List

With the motions created, if you go back to the "Advanced" tab, you'll see the new swim motions listed and setup based on the options you've checked.



Locomotion

Motions

Idle		0	✓
Fall		20	✓
Walk Run Strafe		6	✓
Swim - Idle		20	✓
Swim - Dive		24	✓
Swim - Strafe		22	✓
Swim - Exit		25	✓

AdventureForward

Once setup, you can treat them like any other motion. You can disable them, remove them, activate them using AI, etc. They are motions just like any other motion.

Earlier, I gave a quick description of each motion. Here I'll go into more detail about the motion properties.

Swim – Idle

Simple idle animation while in water.

Up Action Alias – Key to swim up.

Down Action Alias – Key to swim down.

Vertical Speed – Speed that the character moves up and down.

Rotate With Input – Determines if the character rotates as the mouse (right stick) moves.

Rotate With Camera – Determines if the character rotates to match the direction of the camera.

Rotation Speed – Degrees per second to rotate.

Swim – Dive

Animation that has the character enter water by diving.

Action Alias – Input entry that starts the dive.

Test Distance – Forward distance to test if there's water we can dive into.

Move Test Distance – Forward distance to test if there's water we can dive into. This is used when we're moving and typically has us test further out than 'Test Distance'.

Swim – Strafe

Movement motion while in the water. This movement style is more "Shooter" style and always has the character facing forward. It supports strafing.

Default to Run – Determines if the character runs or walks by default (Action Alias inverts it).

Run Action Alias – Used to trigger the character to swim fast (or slow) based on the Default to Run option.

Walk Speed – When greater than 0, overrides root-motion and creates a constant velocity (units per second).

Run Speed – When greater than 0, overrides root-motion and creates a constant velocity (units per second).

Up Action Alias – Key to swim up.

Down Action Alias – Key to swim down.

Dive With Camera – Determines if the pitch of the camera has us swim up or down.

Vertical Speed – Speed that the character moves up and down.

Tilt With Dive – Determines if we tilt the 'Body Transform' as we dive and ascend.

Max Body Pitch – If using the 'Body Transform', amount to tilt when going up or down.

Rotate With Input – Determines if the character rotates as the mouse (right stick) moves.

Rotate With Camera – Determines if the character rotates to match the direction of the camera.

Rotation Speed – Degrees per second to rotate.

Swim – Exit

This is used when the swimmer exits the water using an edge.

Grab Alias – Used to trigger the character to start the motion.

Climbing Layers – Layers determine what can be used to climb out of the water.

Body Scale – Edge detection is built for a character that is ~1.8 meters tall. If your character is much smaller or taller, I'll use the scale you set here to help better find the edge.

For example: If you're character is 3.6 meters tall, your Body Scale would be 2.0 (3.6 / 1.8 = 2.0).

Body Radius – Determines how thick your character is. This helps determine how far from their center to check for an edge.

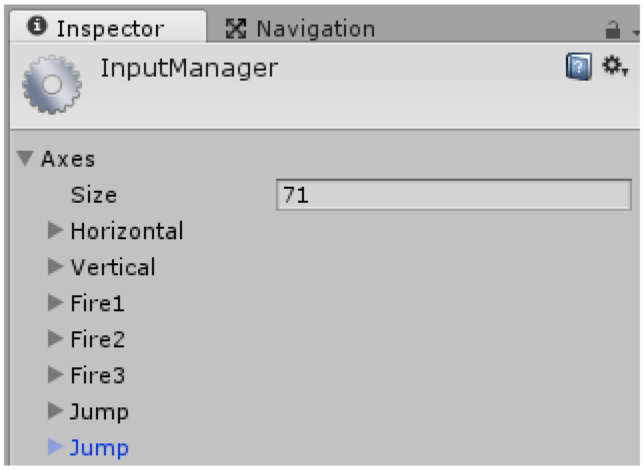
Distance (min/max) – Determines how far away (horizontally) the edge can be from the character.

Height (min/max) – Determines how far away (vertically) the edge can be from the character.

Depth (max) – Max depth from the surface the character can be in order to grab an edge.

Inputs

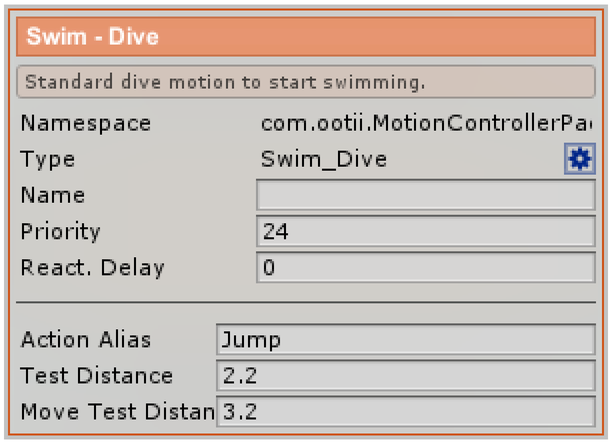
For the most part, I use Unity's standard input for moving and controlling the character. However, some new input entries are added to support things like diving, swimming up/down, and exiting.



Size 71
Horizontal
Vertical
Fire1
Fire2
Fire3
Jump
Jump

For all these extra entries, there is a 'keyboard' entry and an Xbox controller entry.

Following my Motion Controller and Input Source standards, each motion that requires a special key (like Swim – Dive), I allow you to change the input entry. I call these Action Aliases and they trigger the motion. For example, Swim – Dive uses the "Jump" entry you see above.



Swim - Dive

Standard dive motion to start swimming.

Namespace	com.ootii.MotionControllerPa
Type	Swim_Dive
Name	
Priority	24
React. Delay	0
Action Alias	Jump
Test Distance	2.2
Move Test Distan	3.2

Using this approach, you can change the input entry the motion uses by setting the "**Action Alias**" or changing the key or button that the alias represents.

How you change the key or button depends on the input solution you're using. For standard Unity Input Manager, you'd just select the 'Edit | Project Settings... | Input' menu item in Unity.

Frequently Asked Questions

Below is a list of how-to and responses to questions that I get (or expect to get). Hopefully they help provide some quick insight and tutorials.

Pre-Purchase

Can I use this with Rewired or other non-ootii assets?

Yes. I've built this (and the MC) to be very modular. You can simply use a different "input source" to read from any input solution you want. See this YouTube video or the MC documentation for more info.

Can I use ootii's Event System - Dispatcher with the Swimming Motion Pack?

Absolutely. See the section on events.

Swimming

How do I make the movement speed faster?

If you want to use the root-motion, you can increase the speed of the animations in the Animator (specifically Swim_Strafe-SM).

If you don't want to use root-motion, just change the "Walk Speed" and "Run Speed" properties of the motion.

Why does my character get stuck in dive mode at the surface?

This is probably because your water doesn't have a collider or doesn't have its layer isn't set to 'Water'.

If you are using a water solution like Suimono or AQUAS, you may have to double-check that a collider exists. If it doesn't, add as needed.

Camera

Can I use the Adventure Camera & Rig?

Absolutely.

Can I use my own camera system?

Absolutely.

This asset doesn't control the camera at all. It does use the camera's direction to help rotate, but that is simply a transform that I access.

You'll have to play around with the motion properties to get the kind of behavior that you're looking for.

Mixamo Animations

This asset requires several animations from Mixamo.com. Currently Mixamo gives these animations away for free, but I'm not allowed to distribute them. So, you have to get them yourself.

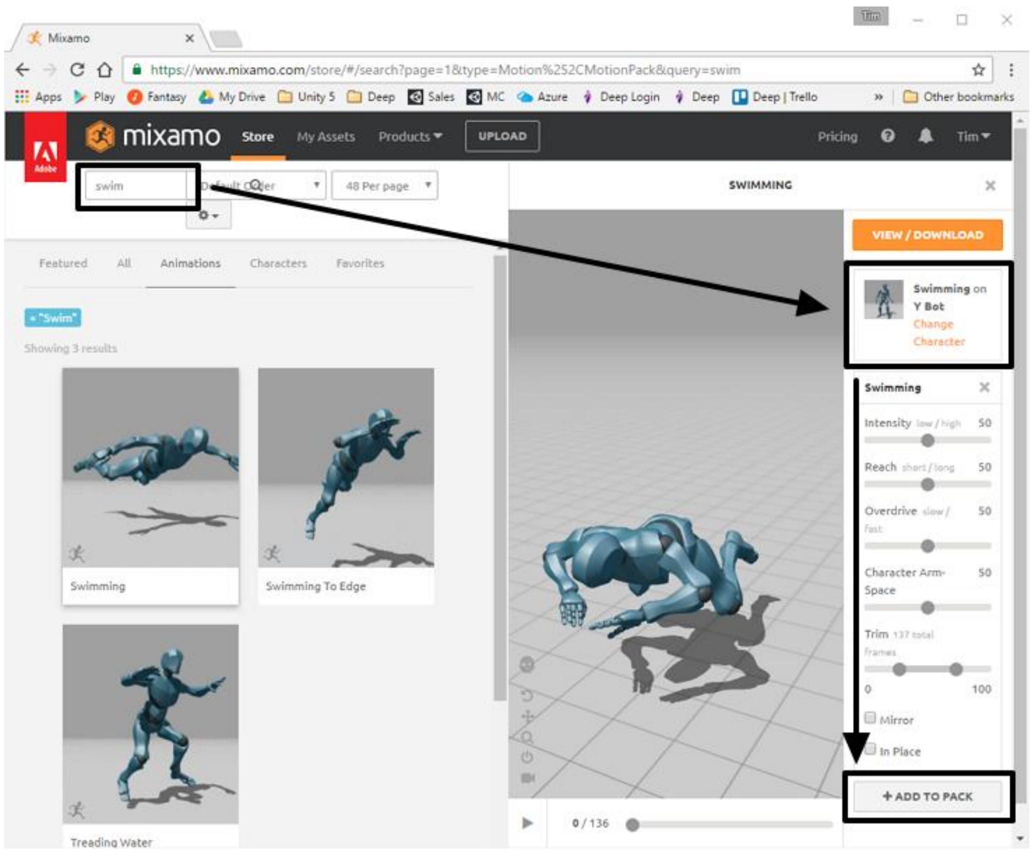
As you get them, you want to grab them for the "Y-bot" character.

1. Go to Mixamo.com and sign in.

2. Under "Store", search for the following animations:

- Swimming
- Swimming To Edge
- Treading Water
- Dive roll
- Run To Dive
- Crouched To Standing
- Braced Hang To Crouch

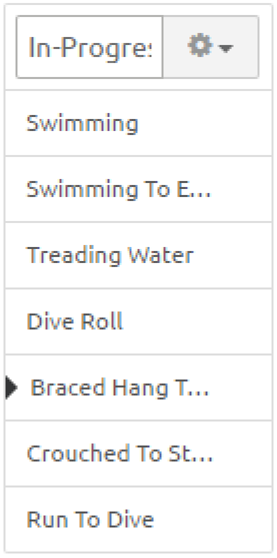
As you find each animation, Mixamo will look like this:



Intensity low / high 50
Reach short / long 50
Overdrive slow / fast 50
Character Arm-Space 50
Trim 137 total frames 0 100

Ensure you're using the "Y Bot" character (at the top right).

The easiest thing to do is add each of the animations to a pack (at the bottom right).



In-Progre:

Swimming

Swimming To E…

Treading Water

Dive Roll

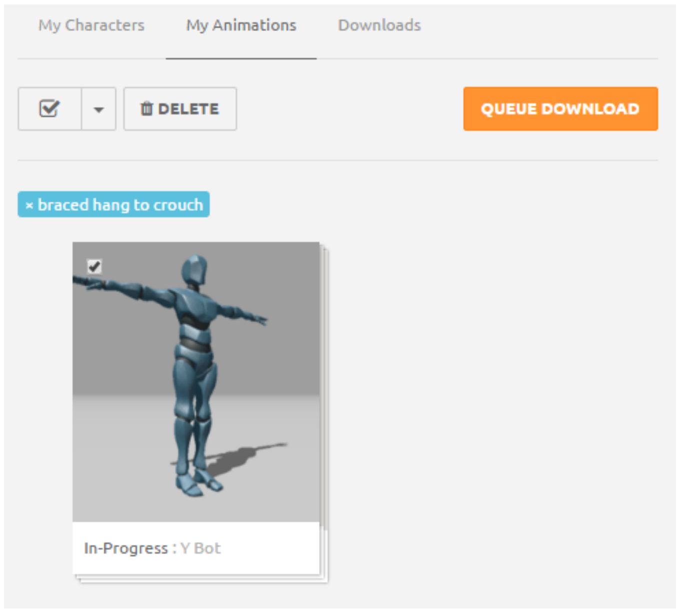
Braced Hang T…

Crouched To St…

Run To Dive

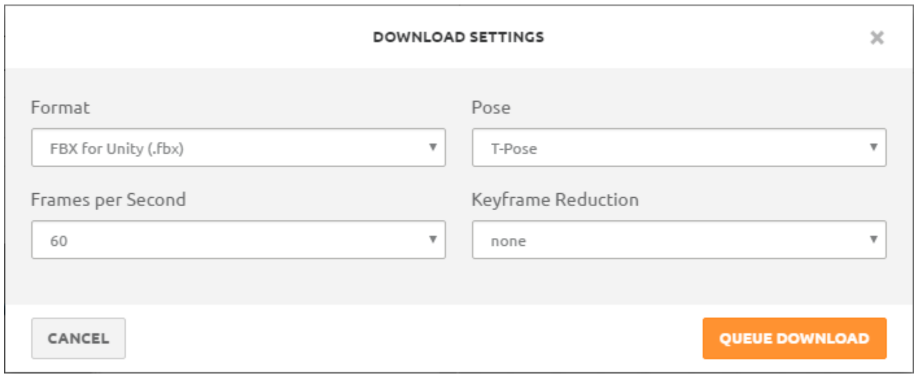
Once the seven animations are added to the pack, you can press the big orange "View / Download" button at the top right.

3. The page will update to now show your animation pack.



Press the big orange "Queue Download" button at the top middle of the page.

4. Choose these options and press the “Queue Download” button


DOWNLOAD SETTINGS

Format: FBX for Unity (.fbx)

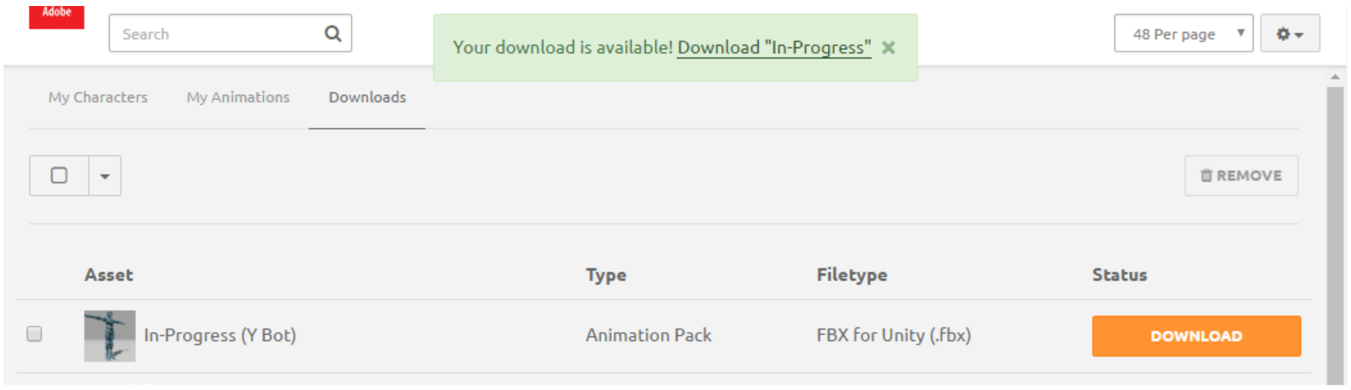
Pose: T-Pose

Frames per Second: 60

Keyframe Reduction: none

Once the animations finish processing, the page will update.

5. With the processing done, press the big orange “Download” button



Your download is available! Download "In-Progress"

Asset	Type	Filetype	Status
In-Progress (Y Bot)	Animation Pack	FBX for Unity (.fbx)	DOWNLOAD

Save the zip file anywhere on your computer, but not in the Unity project.

6. Unzip the animations to the Unity project’s “Mixamo” folder found here:

```
Assets\ootii\MotionControllerPacks\Swimming\Content\Animations\Mixamo
```

7. Finally, don’t forget to unzip the meta data files from this asset’s “Extras” folder to the same folder in step #6.