VRKitchen2.0-Tutorial

Release 0.0.4

Yizhou Zhao, Wensi Ai

May 14, 2023

GETTING STARTED

VRKitchen 2.0 (06/01/2022) is a toolkit for generate indoor scened and tasks in Nvidia Omniverse This tutorial helpes you generate photo-realistic and physics-reliable indoor environment to help the researches in generate AI.

This documentation also offers a *simple* and *intuitive* API despite the official documentation

CHAPTER

ONE

SOURCE CODE

The source code is available on Github

Note: This project is under active development. Update(06/01/2022): Initialize documentation.

1.1 Overview

To facilitate the study of **indoor scene building** methods and their potential applications for `robotics`, `animation`, and `Embodied AI`, we introduce VRKITCHEN2.0: a toolkit built by NVIDIA OMNIVERSE that provides flexible pipelines for indoor scene building, scene randomizing, and robotic controls.

Besides, by combining Python coding in the animation software VRKITCHEN2.0 can assist researches in creating real-time training and control for robotics in the future.

1.1.1 Demo 1: Build customized high-quality indoor scenes

Embodied artificial intelligence (EAI) has attracted significant attention, both in advanced deep learning models and algorithms and the rapid development of simulated platforms. Many open challenges have been proposed to facilitate EAI research. A critical bottleneck in existing simulated platforms is the **limited number of indoor scenes** that support vision-and-language navigation, object interaction, and complex household tasks [ZLJ+21].

In this tutoral: *Tutorial x: Indoor scene bulding*, we shall present how to import tens of thousands of room layouts from the original *3D-Front dataset* into *Omniverse* to give a photo-realistic effect for EAI tasks.

1.1.2 Demo 2: Parsing articulated objects

Articulated objects can be defined as objects composed of more than one rigid parts. In our daily life, humans are constantly interacting with a lot of articulated objects such as *door, keyboard, light switch,* and e.t.c. The rigidbody, softbody, articulated object, and liquid compose a large part of our interaction with the world.

In this tutorial: *Tutorial x: Parsing Articulated Object*, we present how to parse articulated objects in SAPIEN [XQM+20] (a realistic and physics-rich simulated environment) into *Omniverse*, and present their potential applications for dynmaic controls in the virtual environment.

1.1.3 Demo 3: Set up liquid

Research on studies of softbodies and liquid is very prolific and already achieved important success, although with very different technological appliciable level. The interaction with liquid and softbody in Embodied AI is by far the most productive topic.

In this tutorial: *Tutorial x: Set up liquid*, we show how to calculate and build liquid in `Omniverse`.

1.1.4 Demo 4: Set up Robot

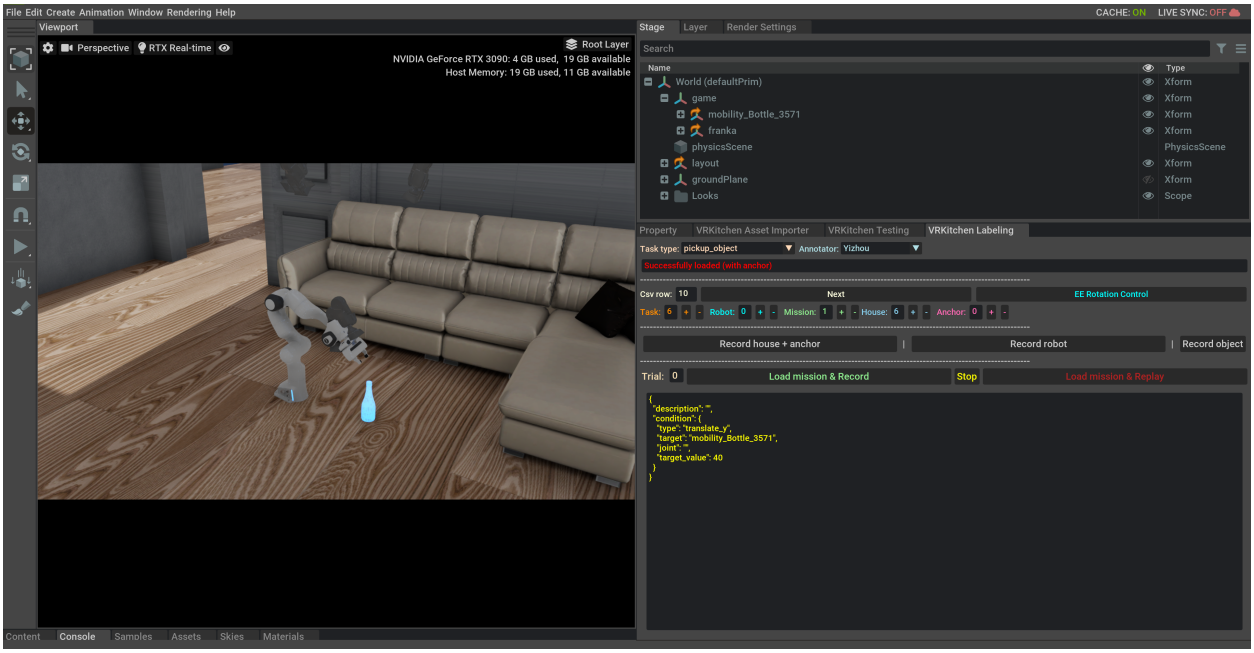
Embodied AI originally refers to AI for virtual robots, which is the field for solving AI problems for virtual robots that can move, see, speak, and interact in the virtual world and with other virtual robots/ Hopefully, the simulated robot solutions are then **transferred to real world robots**.

In this tutorial (*Tutorial x: Set up a Franka-panda robot*), we show how to import a `Franka Emika Panda` robot in `Omniverse`.

1.1.5 Demo 5: (Put together) Data labeling

Machine Learning and Deep Learning have been successful because of the increasing amounts of data and the increasing amounts of computing power (e.g., CPUs, GPUs, TPUs). However, this type of "trainable data" for Embodied AI and robotics are still limited especially when we consider them in virtual environment.

To enrich the dataset for robotics and EAI community, this tutorial (*Tutorial x: Data labeling with scenes, articulated objects, liquid and robots*) teaches how to integrete everything together to label a robot task in a *real* scene with *real* physics with scenes, articulated objects, liquid and robots.



1.2 License

A wide range of software and datasets contributes to this toolkit. To encourage copyright protection, we present the licenses that may be necessary for our *VRKitchen2.0* to show our respect to all original works. Users might need some special conditions that should be clear for future users of our toolkit.

1.2.1 Software

Omniverse

NVIDIA Omniverse is an easily extensible platform for 3D design collaboration and scalable multi-GPU, real-time, true-to-reality simulation. Omniverse revolutionizes the way we create and develop as individuals and work together as teams, bringing more creative possibilities and efficiency to 3D creators, developers and enterprises.

1.2.2 Dataset

3D-Front

3D-Front is large-scale, and comprehensive repository of synthetic indoor scenes highlighted by professionally designed layouts. From layout semantics down to texture details of individual objects, tge dataset is freely available to the academic community and beyond. Currently, 3D-FRONT contains 18,797 rooms diversely furnished by 3D objects. In addition, the 7,302 furniture objects all come with high-quality textures.

SAPIEN Asset

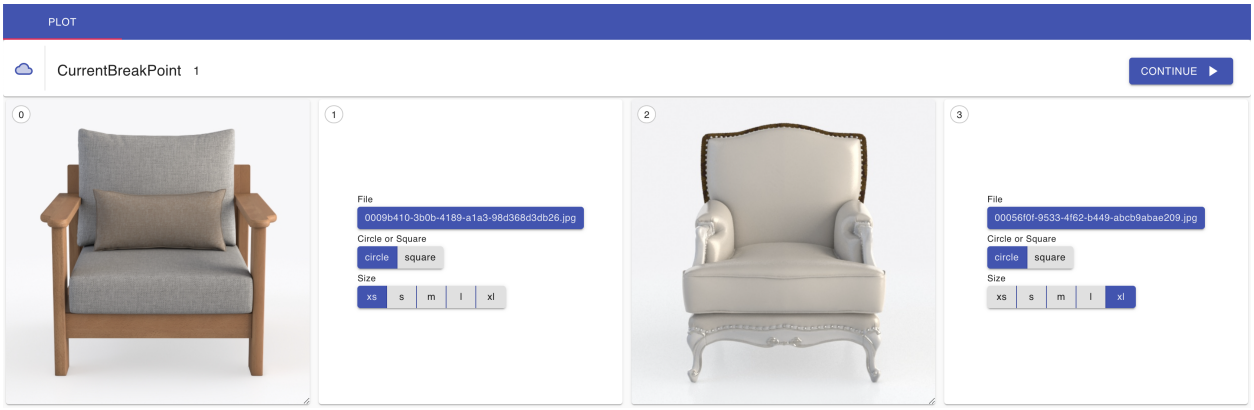
PartNet-Mobility dataset in SAPIEN is a collection of about 2,000 articulated objects with motion annotations and rendernig material. The dataset powers research for generalizable computer vision and manipulation. The dataset is a continuation of ShapeNet and PartNet.

Ai2Thor Asset

Ai2Thor is a near photo-realistic interactable framework for Embodied AI agents. It contains 200+ scenes, 2600+ objects, and many other key features related to robotics and machine learning.

1.2.3 Tool

Trescope



Trescope is a comprehensive 3D machine learning development tool devoted to improve developing experience and speed in 3D field, which helps researchers and developers to label, debug, visualize various 3D data.

GenMotion

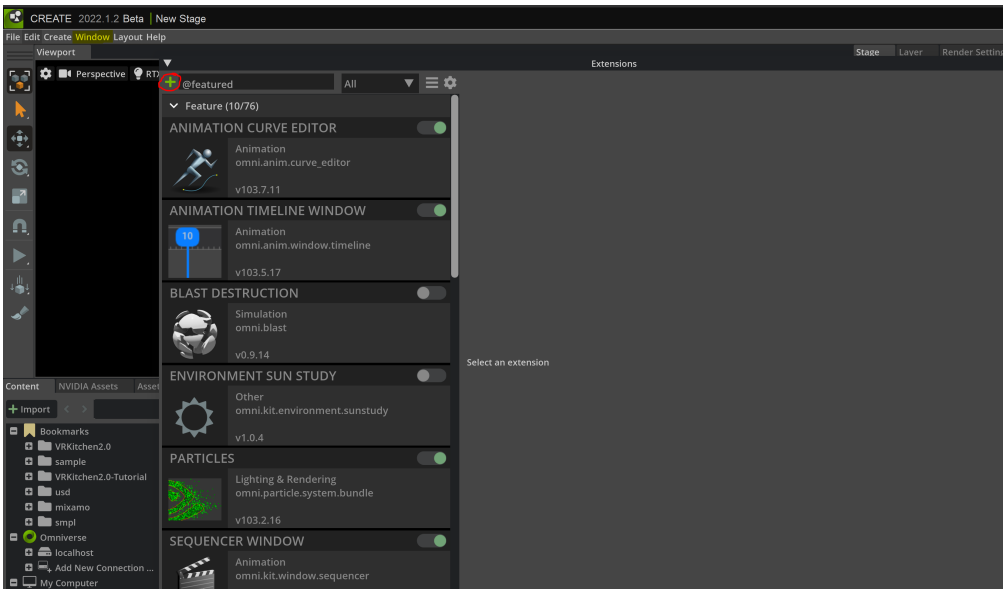
GenMotion is a Python library for making skeletal animations. It enables easy dataset loading and experiment sharing for synthesizing skeleton-Based human animation with the Python API. It also comes with a easy-to-use and industry-compatible API for Autodesk Maya, Maxon Cinema 4D, and Blender.

1.3 B.1 Build your first extension in Omnverse

In this tutorial, we are going to show how to create an extension in Omniverse create. The official documentaion can be found here.

1.3.1 Method one: start an extension in Omniverse

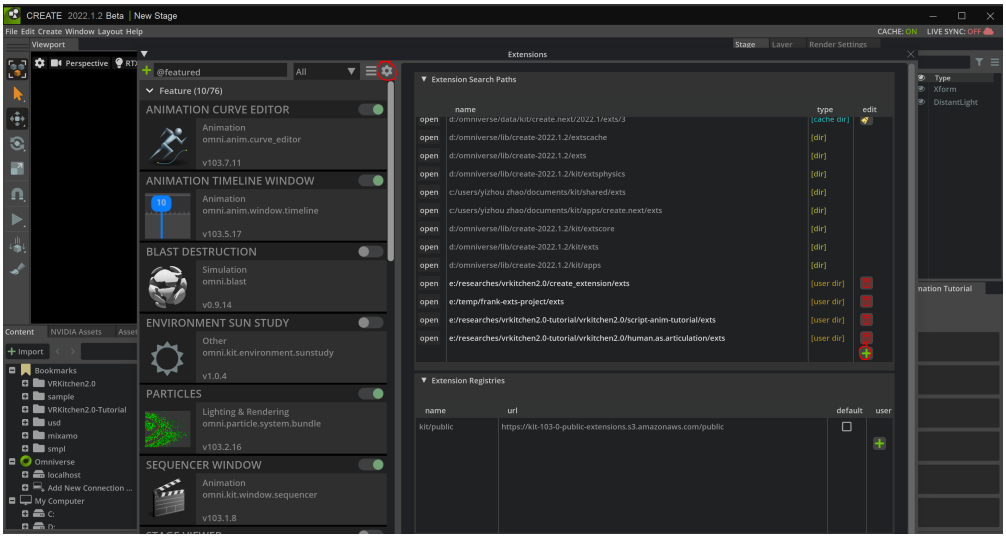
Go to `Menu bar` -> `Window` -> `Extensions`, click the `plus` sign on the top left conner.



Set up the file name and extension name and click `OK`. If on your computer, the Visual Studio code is installed, the script will be open automatically.

Well done!

1.3.2 Method two: add an existing project



Go to `Menu bar` -> `Window` -> `Extensions`, click the `setting gear` sign on the top.

Then add your existing project with the path ending with `.exts`.

Well done!

1.4 A.1 Bring Mixamo Animation to Omniverse

In this part, we are going to show how to bring characters and animation clips from Adobe Mixamo into `Omniverse Create`

1.4.1 Method one: use omniverse directly

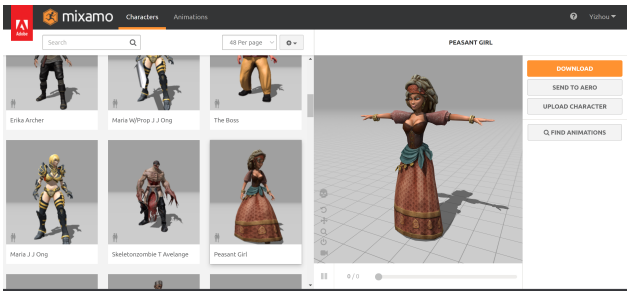
0. Requirements

Warning: Please refer to the licenses (*License*) if necessary.

- Adobe Mixamo
- Nvidia Omniverse

1. Download character & Animation from MIXAMO

Visit Adobe Mixamo, then save the character (e.g. peasant_girl.fbx) and the animation clip (e.g. Silly Dancing.fbx)



Note: The `animation clip` can be saved with skeletal animation only (without skin).

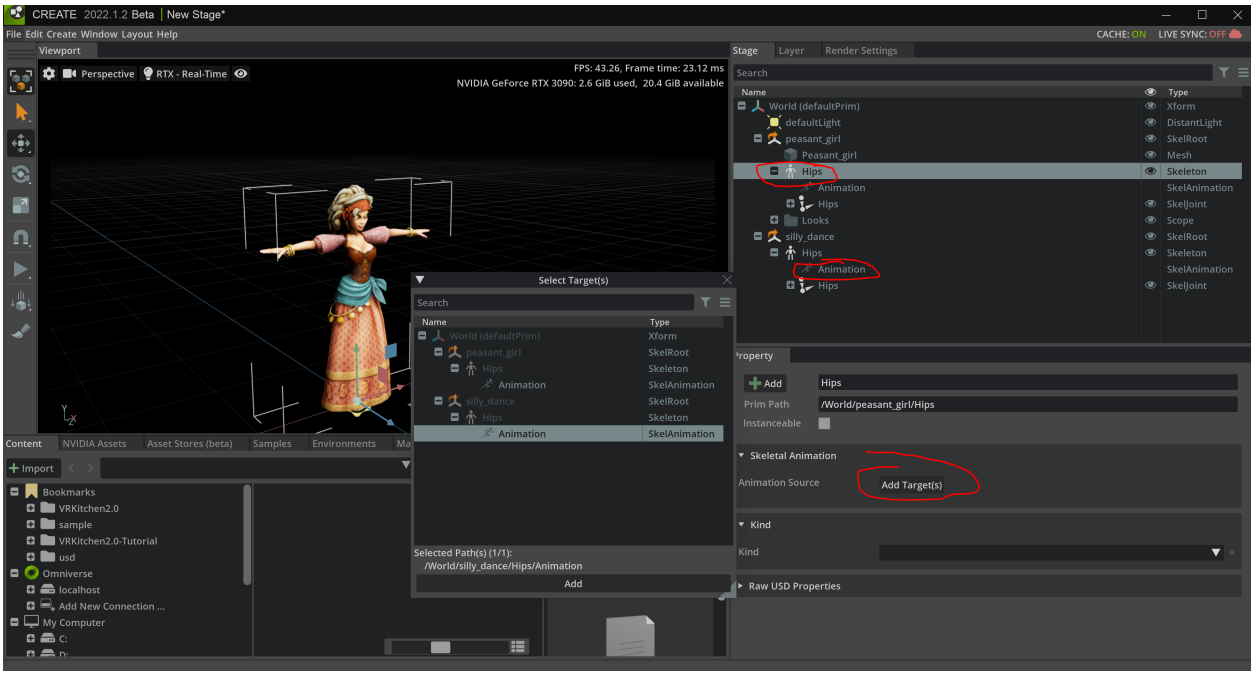
2. Import FBX into Omniverse

(Optional) to convert `FBX` into `USD`, please call the python code:

```
from pathlib import Path

# character
test_data_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/peasant_girl.fbx"

# for animation, the same story applies
# test_data_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/Silly Dancing.fbx"


converter_manager = omni.kit.asset_converter.get_instance()
context = omni.kit.asset_converter.AssetConverterContext()
context.keep_all_materials = True
context.merge_all_meshes = True

# character
output_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/peasant_girl_converted.
↪usd"
# output_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/silly_dancing_
↪converted.usd"

task = converter_manager.create_converter_task(test_data_path, output_path, None,␣
↪context)

success = await task.wait_until_finished()
assert success, "convert not successful"
assert Path(output_path).is_file()
```

3. Load character and animation into Omnvierse

Finally, load character and select your animation clip in Omnvierse.



Note: We can open USD file from script

```
omni.usd.get_context().open_stage_async(output_path)
```

4. Now you can see the mixamo animation:

1.4.2 Method two: use Maya USD converter

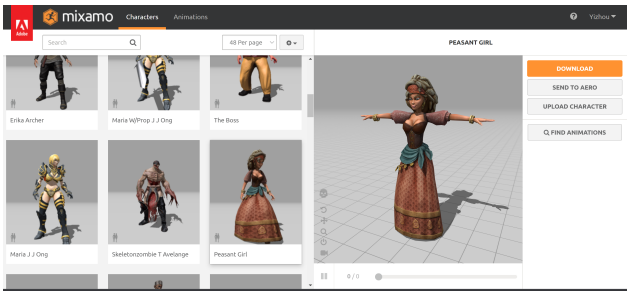
Requirements

Warning: Please refer to the licenses (*License*) if necessary.

- Adobe Mixamo
- Autodesk Maya 2023 (We the Maya version >= 2022) to import `mayaUSD` module.
- Nvidia Omniverse

Download character & Animation from MIXAMO

Visit Adobe Mixamo, then save the character (e.g. peasant_girl.fbx) and the animation clip (e.g. Silly Dancing.fbx)

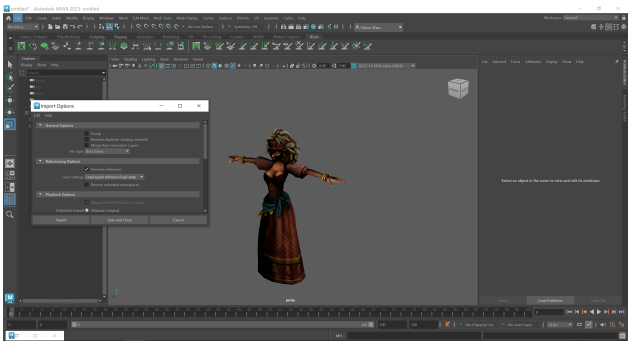


Note: The `animation clip` can be saved with skeletal animation only (without skin).

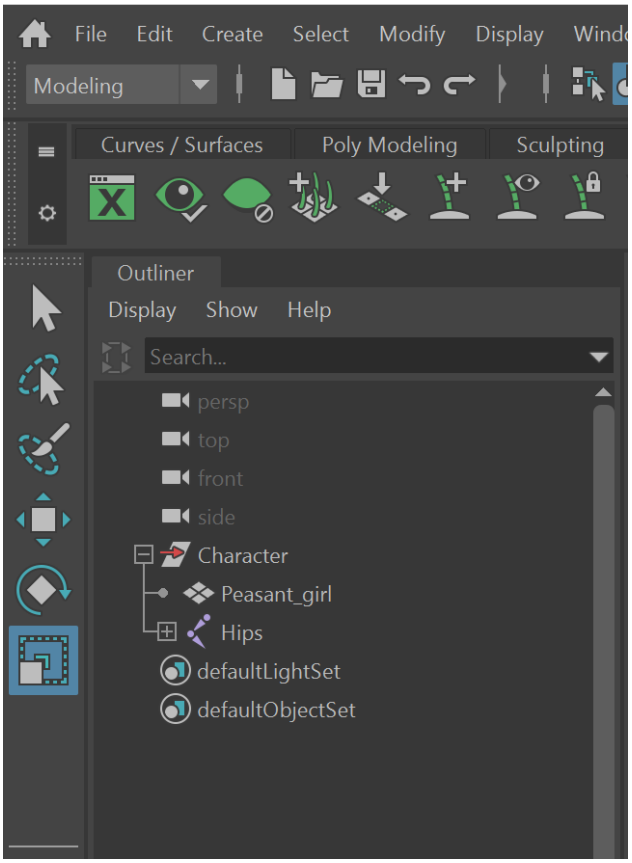
Import FBX into maya

We can also try import with Python code:

```
import maya.cmds as cmds

fbx_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/peasant_girl.fbx"
cmds.file(fbx_path, i=True, type='Fbx')
```

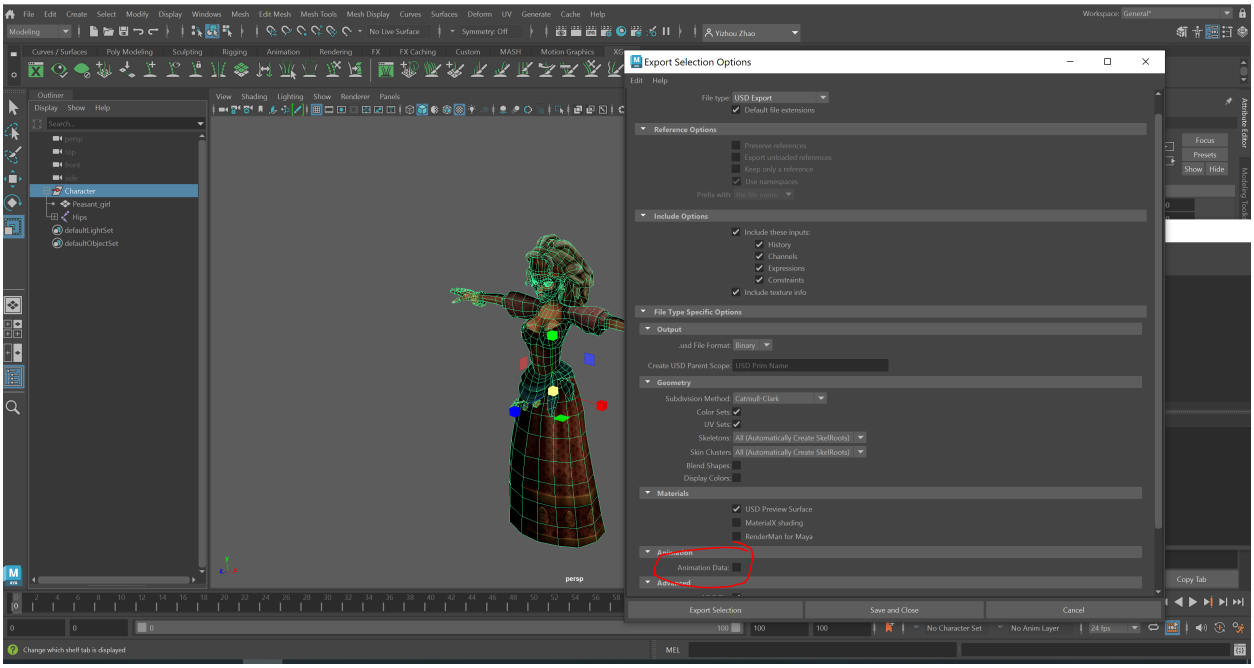


Group character and export




```
cmds.group( 'Peasant_girl', 'Hips', n='Character')

# file -force -options ";exportUVs=1;exportSkels=auto;exportSkin=auto;
↪exportBlendShapes=0;exportDisplayColor=0;exportColorSets=1;
↪defaultMeshScheme=catmullClark;animation=0;eulerFilter=0;staticSingleSample=0;
↪startTime=0;endTime=115;frameStride=1;frameSample=0.0;defaultUSDFormat=usdc;
↪parentScope=;shadingMode=useRegistry;convertMaterialsTo=[UsdPreviewSurface];
↪exportInstances=1;exportVisibility=1;mergeTransformAndShape=1;stripNamespaces=0" -typ
↪"USD Export" -pr -es "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/peasant_girl.usd
↪";
```

(continues on next page)



(continued from previous page)

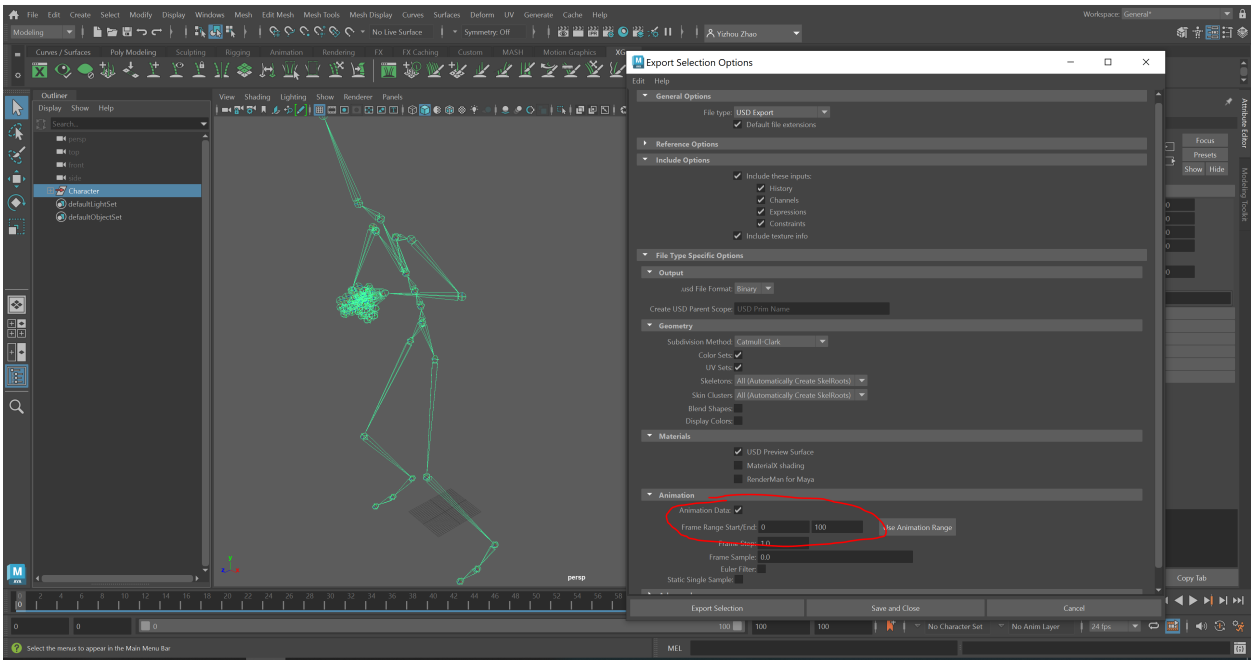
```
options = ";exportUVs=1;exportSkels=auto;exportSkin=auto;exportBlendShapes=0;
↪exportDisplayColor=0;exportColorSets=1;defaultMeshScheme=catmullClark;animation=0;
↪eulerFilter=0;staticSingleSample=0;startTime=0;endTime=115;frameStride=1;frameSample=0.
↪0;defaultUSDFormat=usdc;parentScope=;shadingMode=useRegistry;
↪convertMaterialsTo=[UsdPreviewSurface];exportInstances=1;exportVisibility=1;
↪mergeTransformAndShape=1;stripNamespaces=0"

usd_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/peasant_girl.usd"
cmds.file(usd_path, force = True, options = options, type="USD Export",␣
↪exportSelected=True, preserveReferences=True)
```

Import and export animation

Now we do the same step for the animation clip.

```
# import animation clip
fbx_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/Silly Dancing.fbx"
cmds.file(fbx_path, i=True, type='Fbx')

# group skeleton only
cmds.group('Hips', n='Character')

# output animation
max_time = int(cmds.playbackOptions(maxTime=-1, q=True)) # get timeline max time

# file -force -options ";exportUVs=1;exportSkels=auto;exportSkin=auto;
↪exportBlendShapes=0;exportDisplayColor=0;exportColorSets=1;
↪defaultMeshScheme=catmullClark;animation=1;eulerFilter=0;staticSingleSample=0;
```

(continues on next page)



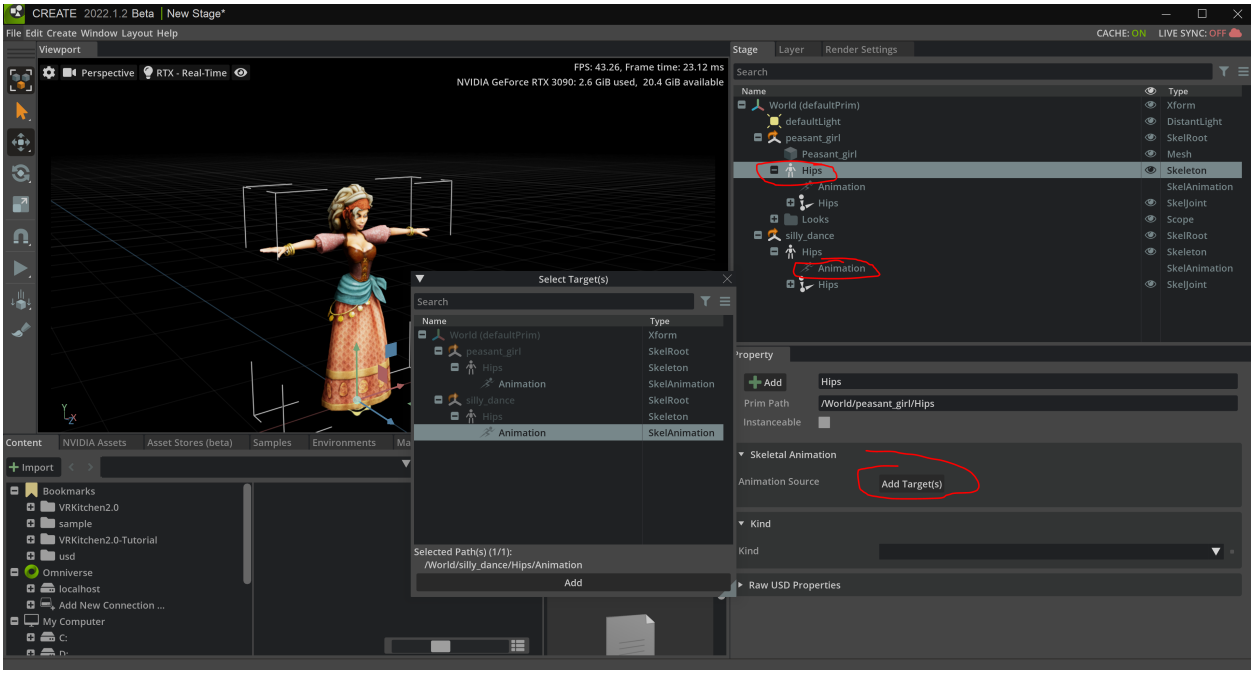
(continued from previous page)

```
↪startTime=0;endTime=100;frameStride=1;frameSample=0.0;defaultUSDFormat=usdc;
↪parentScope=;shadingMode=useRegistry;convertMaterialsTo=[UsdPreviewSurface];
↪exportInstances=1;exportVisibility=1;mergeTransformAndShape=1;stripNamespaces=0" -typ
↪"USD Export" -pr -es "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/silly_dance.usd
↪";

options =f";exportUVs=1;exportSkels=auto;exportSkin=auto;exportBlendShapes=0;
↪exportDisplayColor=0;exportColorSets=1;defaultMeshScheme=catmullClark;animation=1;
↪eulerFilter=0;staticSingleSample=0;startTime=0;endTime={max_time};frameStride=1;
↪frameSample=0.0;defaultUSDFormat=usdc;parentScope=;shadingMode=useRegistry;
↪convertMaterialsTo=[UsdPreviewSurface];exportInstances=1;exportVisibility=1;
↪mergeTransformAndShape=1;stripNamespaces=0"

usd_path = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/silly_dance2.usd"
cmds.file(usd_path, force = True, options = options, type="USD Export",␣
↪exportSelected=True, preserveReferences=True)
```

Load character and animation into Omnvierse

Finally, load character and select your animation clip in Omnvierse.

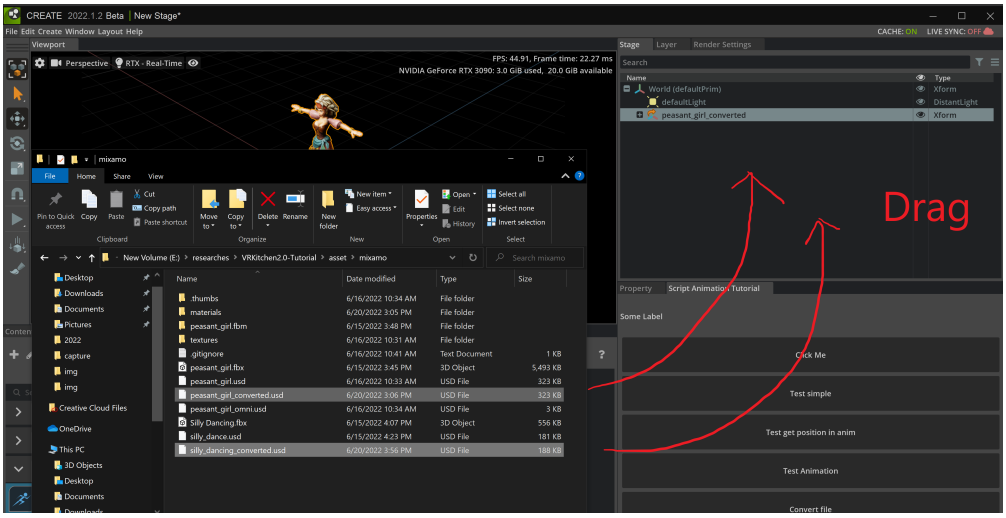
Now you can see the mixamo animation:


CREATE 2022.1.2 Beta | New Stage*
File Edit Create Window Layout Help
CACHE: ON LIVE SYNC: OFF
Viewport
Perspective RTX - Real-Time
FPS: 43.26, Frame time: 23.12 ms
NVIDIA GeForce RTX 3090: 2.6 GiB used, 20.4 GiB available
Stage Layer Render Settings
Search
Name Type
World (defaultPrim) Xform
defaultLight DistantLight
peasant_girl SkelRoot
Peasant_girl Mesh
Hips Skeleton
Animation SkelAnimation
Hips SkelJoint
Looks Scope
silly_dance SkelRoot
Hips Skeleton
Animation SkelAnimation
Hips SkelJoint
Select Target(s)
Search
Name Type
World (defaultPrim) Xform
peasant_girl SkelRoot
Hips Skeleton
Animation SkelAnimation
silly_dance SkelRoot
Hips Skeleton
Animation SkelAnimation
Selected Path(s) (1/1):
/World/silly_dance/Hips/Animation
Add
Property
Add
Hips
Prim Path
/World/peasant_girl/Hips
Instanceable
Skeletal Animation
Animation Source
Add Target(s)
Kind
Kind
Raw USD Properties
Content NVIDIA Assets Asset Stores (beta) Samples Environments
Import
Bookmarks
VRKitchen2.0
sample
VRKitchen2.0-Tutorial
usd
Omniverse
localhost
Add New Connection ...
My Computer
C:
D:

1.5 A.2 Build Animation Graph

In this tutorial, we are going to show how to build your own Animation graph to control character animation.

First, drag character and animation file into Omniverse

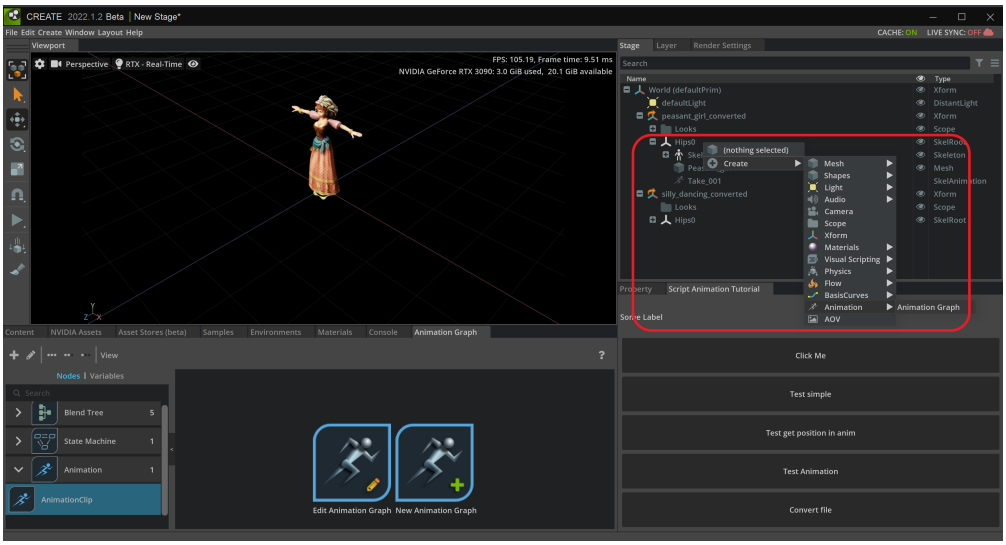


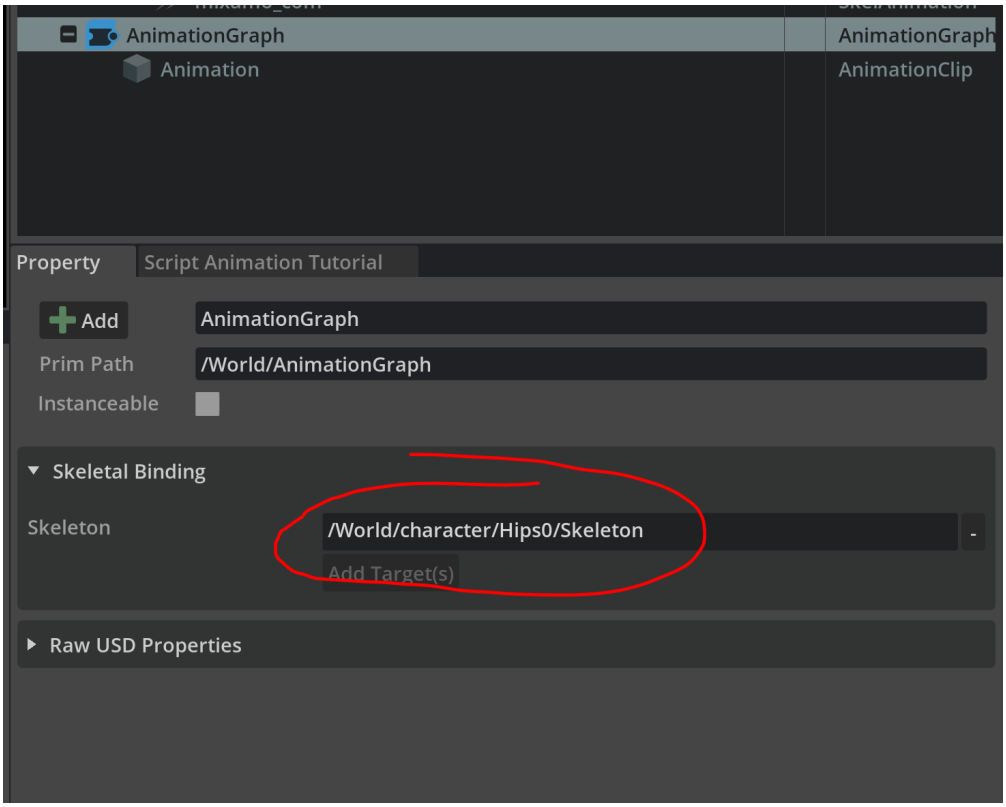
Note: (Optional) We may apply Python code for import *USD* files (see more: *Stage*)

```
character_path = "/World/character"
anim_path = "/World/character_anim_clip"

character_usd = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/peasant_girl_converted.
↪usd"
anim_usd = "E:/researches/VRKitchen2.0-Tutorial/asset/mixamo/silly_dancing_converted.usd"

prim = self.stage.GetPrimAtPath(character_path)
if not prim.IsValid():
    prim = self.stage.DefinePrim(character_path)

success_bool = prim.GetReferences().AddReference(character_usd)
assert success_bool

prim = self.stage.GetPrimAtPath(anim_path)
if not prim.IsValid():
    prim = self.stage.DefinePrim(anim_path)

success_bool = prim.GetReferences().AddReference(anim_usd)
```

Create a new animation graph and assign skeleton:

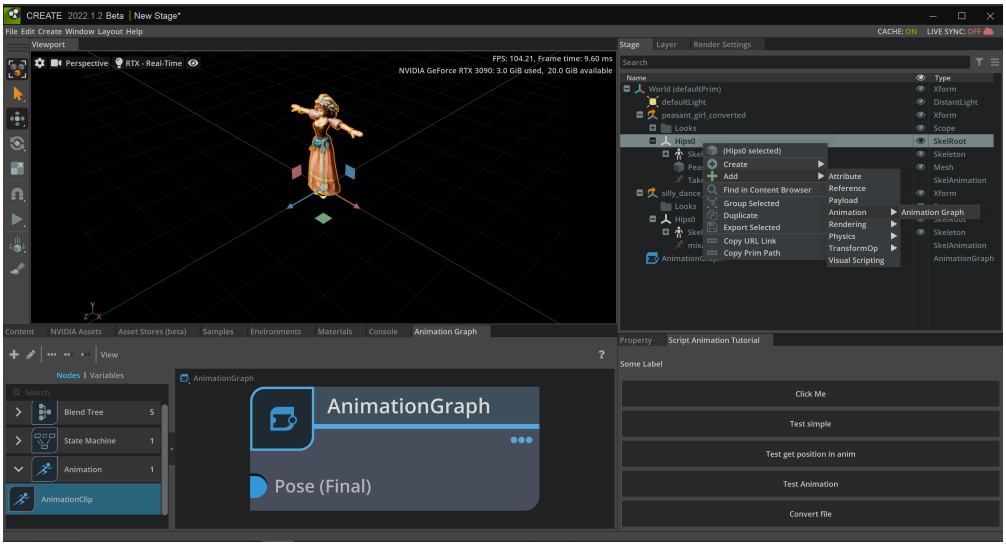
Note: (Optional) We may apply Python code handle animation graph (see more: *Animation Graph*)


CREATE 2022.1.2 Beta | New Stage*
Click Me
Test simple
Test get position in anim
Test Animation
Convert file


AnimationGraph
Animation
AnimationGraph
AnimationClip
Property
Script Animation Tutorial
Add
AnimationGraph
Prim Path
/World/AnimationGraph
Instanceable
Skeletal Binding
Skeleton
/World/character/Hips0/Skeleton
Add Target(s)
Raw USD Properties

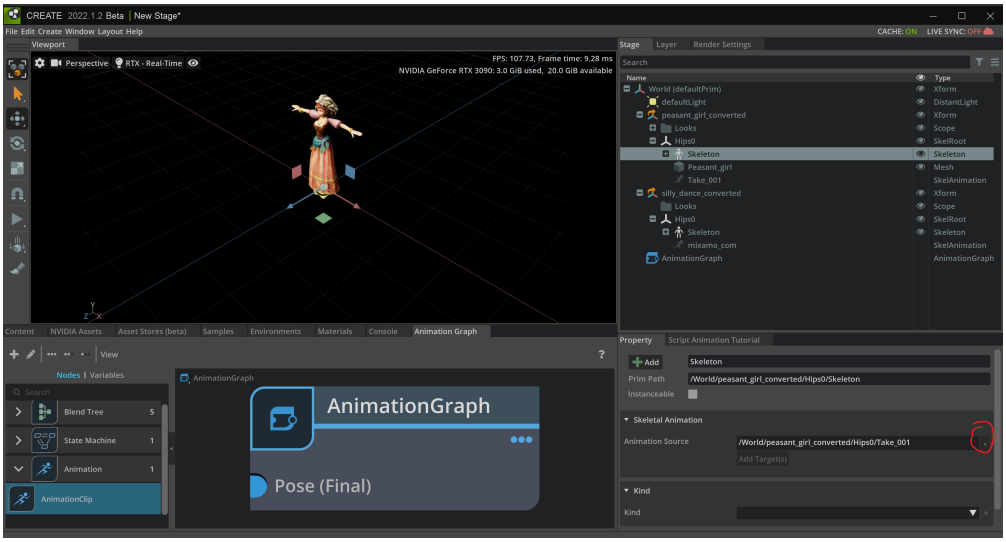
```
anim_graph_path = "/World/AnimationGraph"
skeleton_path = "/World/character/Hips0/Skeleton"

### anim_graph = AnimGraphSchemaTools.createAnimationGraph(stage, Sdf.Path("/World/
↪AnimationGraph"))
omni.kit.commands.execute("CreateAnimationGraphCommand", \
    path=Sdf.Path(anim_graph_path), skeleton_path=Sdf.Path(skeleton_path))
```

Apply the animation to the `skeleton root`:



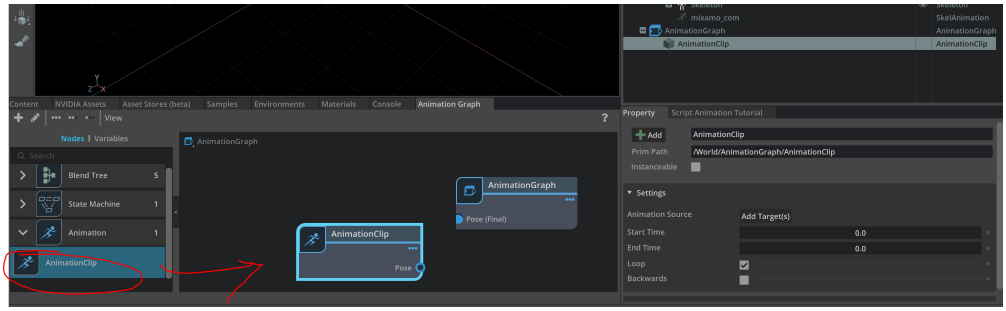

```
skel_root_path = "/World/character/Hips0/" #(e.g. Hips0 for "/World/character/Hips0/
↪Skeleton")
omni.kit.commands.execute("ApplyAnimationGraphAPICommand", \
    paths=[Sdf.Path(skel_root_path)], animation_graph_path=Sdf.Path(anim_graph_path))
```

Possibly need to clean the original clip:



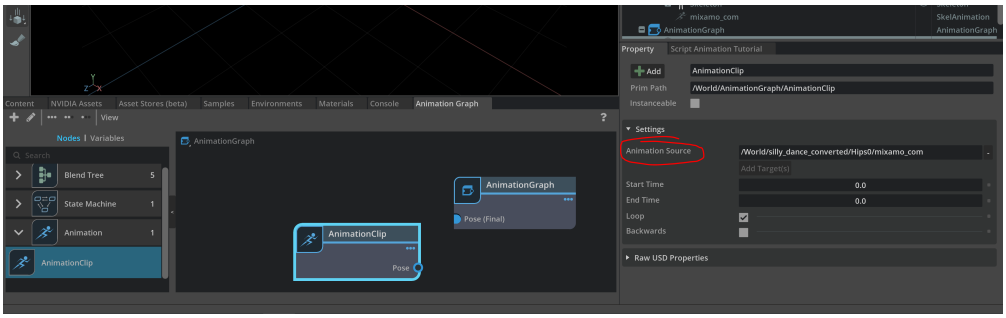
```
skeleton_prim = self.stage.GetPrimAtPath(skeleton_path)
skeleton_bindingAPI = UsdSkel.BindingAPI(skeleton_prim)
skeleton_bindingAPI.GetAnimationSourceRel().SetTargets([])
```

Create animation clip node:



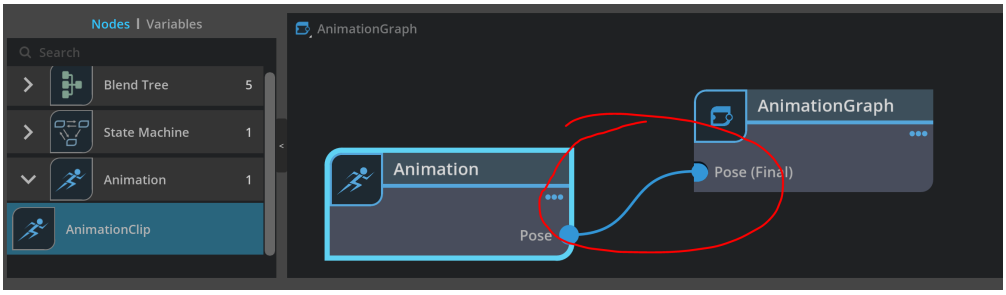

```
# create
omni.kit.commands.execute(
    'CreatePrimCommand',
    prim_type="AnimationClip",
    prim_path="/World/AnimationGraph/Animation",
    select_new_prim=True,
)
```

Add animation clip target source:




```
animclip_prim = self.stage.GetPrimAtPath("/World/AnimationGraph/Animation")
# animclip_bindingAPI = UsdSkel.BindingAPI(animclip_prim)

anim_clip = AnimGraphSchema.AnimationClip(animclip_prim)
source_rel = anim_clip.GetInputsAnimationSourceRel()

omni.kit.commands.execute(
    'omni.anim.graph.ui.scripts.command.SetRelationshipTargetsCommand',
    relationship=source_rel,
    targets=[Sdf.Path("/World/character_anim_clip/Hips0/mixamo_com")]
)
```

Finally, connect connect node:



Warning: The following Python code may cause errors

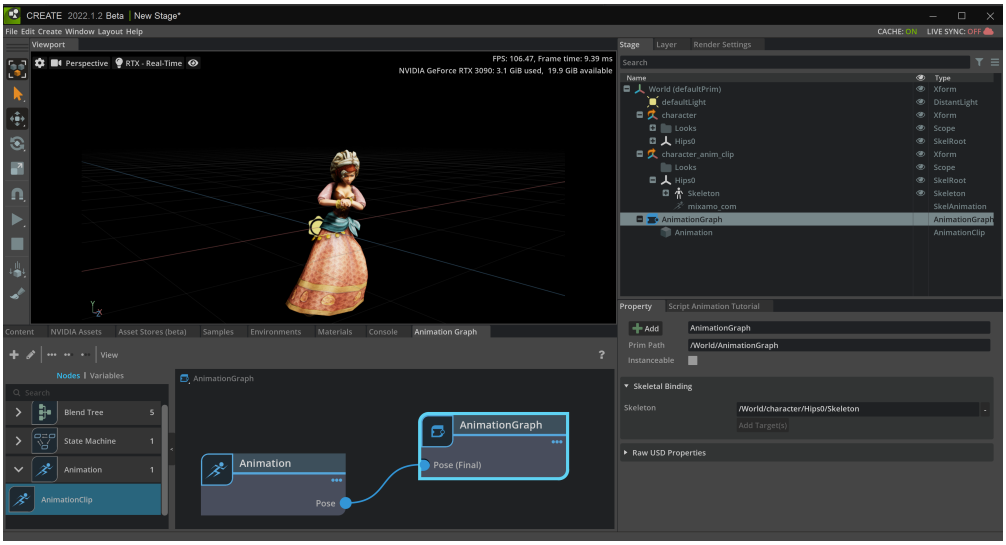
First, modify the original code in `omni.anim.graph.ui.scripts.extension` to make graph manager trackable

```
# from line 13 to 16
class PublicExtension(omni.ext.IExt):
    GRAPH_MANAGER = AnimationGraphManager()
    def on_startup(self):
        self._graph_manager = PublicExtension.GRAPH_MANAGER #AnimationGraphManager()
# ....
```

Then, connect nodes from scripts

```
self._usd_context = omni.usd.get_context()
# omni.usd.get_context().open_stage_async(path)
stage = self._usd_context.get_stage()

from omni.anim.graph.ui.scripts.extension import PublicExtension

graph_manager = PublicExtension.GRAPH_MANAGER
# print("graph manager dict", graph_manager._node_graph_dict)

from pxr import Sdf
node_graph = graph_manager.get_node_graph(Sdf.Path("/World/AnimationGraph"))
# print("node_graph: ", node_graph._path_to_node)
# node_graph._on_create_node("/World/AnimationGraph/Animation")

anim_clip_node = node_graph._path_to_node.get(Sdf.Path("/World/AnimationGraph/
↪Animation"))
root_node = node_graph._path_to_node.get(Sdf.Path("/World/AnimationGraph"))

root_input_port = root_node.ports[0]
#print("root ports", root_input_port.kind, root_input_port.rel)

output_port = anim_clip_node.output
# print("anim port", output_port)

node_graph.create_connection(output_port,root_input_port)
```

Now, the animaion is here:



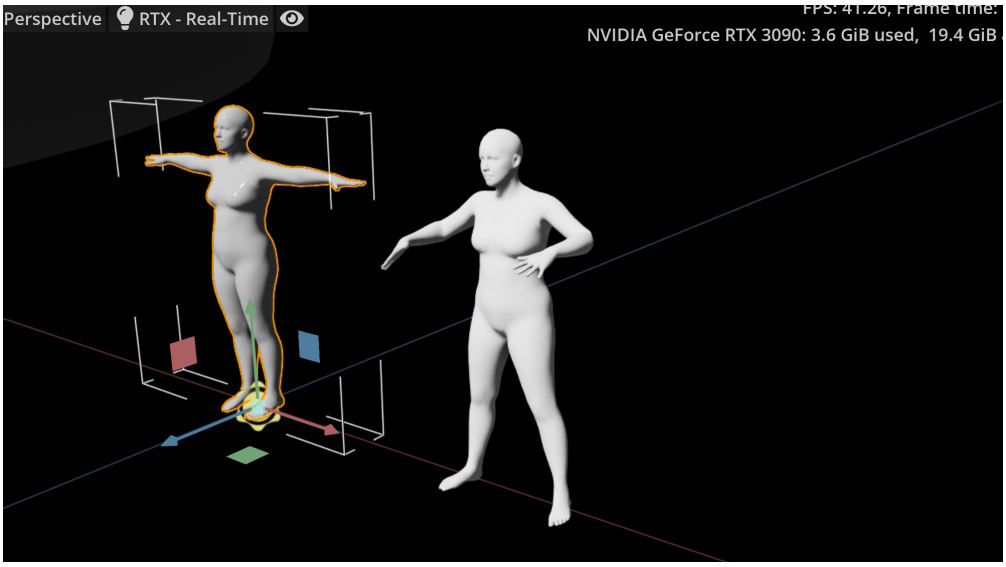
1.6 A.3 Bring arbitrary animation in SMPL format

In this tutorial, we are going to show how to bring animation of SMPL format into Omniverse.

For more information about SMPL, see

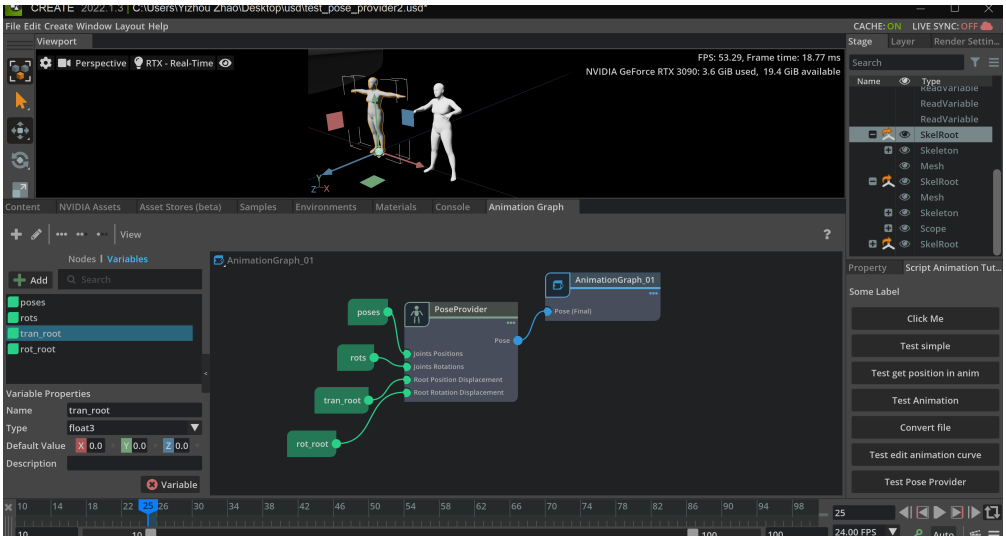
- SMPL-X
- VPoser (human_body_prior)
- SMPL Blender addon

Bring animation into the stage:



Set up an animation graph like this:

Get animation information from script:




```
animprim = stage.GetPrimAtPath("/World/chicken2/f_avg_root/Animation")

anim = UsdSkel.Animation(animprim) # from pxr import UsdSkel
trans = anim.GetTranslationsAttr().Get(current_frame_code) # e.g. current_frame_code = 0
quats = anim.GetRotationsAttr().Get(current_frame_code)
```

Set animation information from script:

```
trans = [carb.Float3(e[0],e[1],e[2]) for e in trans]
quats =  [carb.Float4(e.imaginary[0], e.imaginary[1], e.imaginary[2], e.real) for e in␣
↪quats ]

character.set_variable("poses", trans)
character.set_variable("rots", quats)
```

1.7 Build your first Omniverse Extension

1.8 Tutorial x: Indoor scene bulding

1.9 Tutorial x: Parsing Articulated Object

1.10 Tutorial x: Set up liquid

1.11 Tutorial x: Set up a Franka-panda robot

1.12 Tutorial x: Data labeling with scenes, articulated objects, liquid and robots

1.13 Extented Omniverse Kit User Manual

We strongly recommend users to take a look at the official API documentation for

- Omniverse Kit API
- Pixar API & Tutorial
- Pixar USD Documentation
- Omniverse Isaac-Sim Documentation
- Omni Graph
- Omni Kit

as we ourselves have learned all the coding lessons from those three documentations.

Get started by importing the `omni`, `carb`, and `pixar` package:

```
import omni
import carb
import pxr
```

1.14 Animation

Note: TODO:

- find `_selected_key_wp` in `curve_editor_curves.py`

1.15 File system

We introduce how to provide a coding solution to the Omniverse file system. The modules listed below are originally derived from `USDContext` module. The original documentation can be found here.

Get started by importing the `omni` package:

```
import omni
```

Note: If you would like to call with *corountine*, please `import asyncio` and run it with

```
import asyncio
asyncio.ensure_future(<your async function>)
```

1.15.1 New Stage

```
await omni.usd.get_context().new_stage_async()
await omni.kit.app.get_app().next_update_async()
```

Or the blocking call:

```
omni.usd.get_context().new_stage()
```

1.15.2 Open stage

Note: The Omniverse uses the scene file with *.usd* format

```
usd_path = "<your usd file>.usd"
success, error = await omni.usd.get_context().open_stage_async(usd_path)
if not success:
    raise Exception(f"Failed to open usd file: {error}")
```

Or the blocking call:

```
omni.usd.get_context().open_stage(usd_path)
```

1.15.3 Close stage

```
(result, err) = await omni.usd.get_context().close_stage_async()
# Assert result == True
```

Or the blocking call:

```
omni.usd.get_context().close_stage()
```

1.15.4 Reopen stage

```
(result, err) = await omni.usd.get_context().reopen_stage_async()
# Assert result == True
```

Or the blocking call:

```
omni.usd.get_context().reopen_stage()
```

1.15.5 Save stage

```
save_file_path = os.path.join(tmpdirname, "<your save path>.usda")
(result, err, saved_layers) = await omni.usd.get_context().save_as_stage_async(save_file_
↪path)
# Assert result == True
```

Or the blocking call:

```
omni.usd.get_context().save_as_stage(save_file_path)
```

1.15.6 Export stage

```
save_file_path = os.path.join(tmpdirname, "<your save path>.usda")
(result, err) = await omni.usd.get_context().export_as_stage_async(save_file_path)
# Assert result == True
```

Or the blocking call:

```
omni.usd.get_context().export_as_stage(save_file_path)
```

1.16 Joint system

This part includes the APIs for setting up joint systems.

1.16.1 Common imports for joints

```
from pxr import UsdPhysics
```

1.16.2 Create joints

```
component = UsdPhysics.FixedJoint.Define(stage, joint_path)
component = UsdPhysics.RevoluteJoint.Define(stage, joint_path)
component = UsdPhysics.PrismaticJoint.Define(stage, joint_path)
```

1.17 Key Frame

This part includes the APIs for setting up keyframe using code

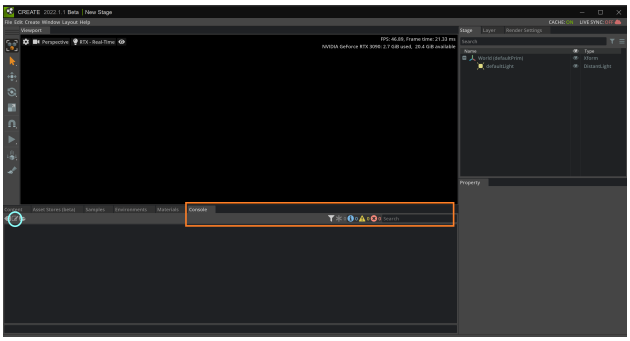
1.17.1 Set up keyframe for a prim attribute

```
(result, err) = omni.kit.commands.execute("SetAnimCurveKeys",
            paths=["/World/Exhibition2/ur3e_nvidia_140/ur3e/shoulder_link/shoulder_lift_
↪joint.drive:angular:physics:targetPosition"],
            value = float(0),
            time=Usd.TimeCode(0)
            )
```

1.18 Log

Omniverse provides a easy-to-use log system. Users can either find logs from

- Command prompt
- Omniverse console window
- Log file



To print a log:

```
import carb

carb.log_error("")
carb.log_warn("")

carb.log_info("")
```

(continues on next page)

(continued from previous page)

```
# which equals
print("")
```

1.19 Mesh

This part works with UsdGeom, UsdMesh, e.t.c.

1.19.1 Get mesh from prim

```
mesh = UsdGeom.Mesh(prim)
```

1.19.2 Get face and point information

```
vertex_points = mesh.GetPointsAttr().Get()
vertex_counts = mesh.GetFaceVertexCountsAttr().Get()
vertex_indices = mesh.GetFaceVertexIndicesAttr().Get()
```

1.20 Particle Systems

This part includes the APIs for setting up particle systems.

1.20.1 Common imports for particle

```
from pxr import UsdPhysics, PhysxSchema, Sdf, Gf
from omni.physx.scripts import physicsUtils, particleUtils
```

1.20.2 Create physics scene

```
self._stage = omni.usd.get_context().get_stage()
self._default_prim_path = Sdf.Path("/World")

# create physics scene
physicsScenePath = self._default_prim_path.AppendChild("physicsScene")
scene = UsdPhysics.Scene.Define(self._stage, physicsScenePath)
physxAPI = PhysxSchema.PhysxSceneAPI.Apply(scene.GetPrim())
physxAPI.CreateSolverTypeAttr("TGS") # TGS or PGS, TGS is more accurate but slower
```

1.20.3 Create pbd material

```
self._pbd_material_path = self._default_prim_path.AppendChild("PBDMaterial")
particleUtils.add_pbd_particle_material(self._stage, self._pbd_material_path)
self._pbd_material_api = PhysxSchema.PhysxPBDMaterialAPI.Get(self._stage, self._pbd_
↪material_path)
```

1.20.4 Create particle systems

```
self._particle_system_path = self._default_prim_path.AppendChild("particleSystem")
particleUtils.add_physx_particle_system(
    stage=self._stage,
    particle_system_path=self._particle_system_path,
    simulation_owner=scene.GetPath(),
)
self._particle_system = PhysxSchema.PhysxParticleSystem.Get(self._stage, self._particle_
↪system_path)

physicsUtils.add_physics_material_to_prim(self._stage, self._particle_system.GetPrim(),␣
↪self._pbd_material_path)
```

1.20.5 Create particle points

```
# create 1 particle
positions_list = []
velocities_list = []
widths_list = []

positions_list.append(Gf.Vec3f(1.0, 1.0, 1.0))
velocities_list.append(Gf.Vec3f(0.0, 0.0, 0.0))
widths_list.append(0.5)

particlePointsPath = Sdf.Path("/World/particles")
particleSet = particleUtils.add_physx_particleset_points(
    self._stage,
    particlePointsPath,
    positions_list,
    velocities_list,
    widths_list,
    self._particle_system_path,
    self_collision = True,
    fluid = True,
    particle_group = 0,
    particle_mass = 1.0,
    density = 0.02,
)
```

1.21 Physics

This part includes the APIs for setting up physics in the stage , read more about USDPhysics

1.21.1 Rigid body & Collision

```
from pxr import UsdPhysics

# check rigidbody api
prim.HasAPI(UsdPhysics.RigidBodyAPI)
prim.HasAPI(pxr.UsdPhysics.CollisionAPI)

# get api
UsdPhysics.RigidBodyAPI.Get(stage, path)
UsdPhysics.CollisionAPI.Get(stage, path)

# apply/add api

UsdPhysics.RigidBodyAPI.Apply(stage, path)
UsdPhysics.CollisionAPI.Apply(stage, path)
```

1.21.2 Set up GPU for physics

```
from omni.physx import acquire_physx_interface

physx = acquire_physx_interface()

physx.overwrite_gpu_setting(1) # 1 means using GPU
```

1.21.3 Set mass

```
mass = 10
massAPI = UsdPhysics.MassAPI.Apply(prim)
massAPI.GetMassAttr().Set(mass)
```

1.21.4 Set/Get Gravity value

```
# FIXME: correct gravity api
# Assume _my_world is of type World
# self._my_world ._physics_context.get_gravity()
# meters_per_unit = get_stage_units()
# self._my_world ._physics_context.set_gravity(value=-9.81 / meters_per_unit)
```

1.21.5 Set up collision

```
from omni.physx.scripts import utils, physicsUtils

utils.setPhysics(prim=cup_shape_prim, kinematic=False)

utils.setCollider(prim=cup_shape_prim, approximationShape="convexDecomposition")
# other approximationShape: none, triangulate, convexHull, e.t.c.
```

1.21.6 Set up physical material

```
def _setup_physics_material(self, path: Sdf.Path):
    from pxr import UsdGeom, UsdLux, Gf, Vt, UsdPhysics, PhysxSchema, Usd, UsdShade, Sdf

    if self._physicsMaterialPath is None:
        self._physicsMaterialPath = self._stage.GetDefaultPrim().GetPath().AppendChild(
↪"physicsMaterial")
        UsdShade.Material.Define(self._stage, self._physicsMaterialPath)
        material = UsdPhysics.MaterialAPI.Apply(self._stage.GetPrimAtPath(self._
↪physicsMaterialPath))
        material.CreateStaticFrictionAttr().Set(self._material_static_friction)
        material.CreateDynamicFrictionAttr().Set(self._material_dynamic_friction)
        material.CreateRestitutionAttr().Set(self._material_restitution)

    collisionAPI = UsdPhysics.CollisionAPI.Get(self._stage, path)
    prim = self._stage.GetPrimAtPath(path)
    if not collisionAPI:
        collisionAPI = UsdPhysics.CollisionAPI.Apply(prim)

    # apply material
    physicsUtils.add_physics_material_to_prim(self._stage, prim, self._
↪physicsMaterialPath)
```

1.21.7 Set up force

The *forceAPI* is will a Xform. Pushing the parent *Xform*. e.g. .../Xform_ball/ballForce is pushing .../Xform_ball

```
import omni.timeline
from pxr import PhysxSchema, UsdGeom, Gf
stage = omni.usd.get_context().get_stage()

# changing timeline end time to avoid looping
omni.timeline.get_timeline_interface().set_end_time(10000/24)

xform = UsdGeom.Xform.Define(stage, "/World/ball1_04/Xform_01/ballForce")
forceApi = PhysxSchema.PhysxForceAPI.Apply(xform.GetPrim())

forceAttr = forceApi.GetForceAttr()
forceAttr.Set(time=0, value=Gf.Vec3f(0.0, 0, 300.0))
forceAttr.Set(time=20, value=Gf.Vec3f(0.0, 0, 300.0))
```

(continues on next page)

(continued from previous page)

```
forceEnabledAttr = forceApi.GetForceEnabledAttr()
forceEnabledAttr.Set(time=0, value=True)
forceEnabledAttr.Set(time=20, value=False)

xformable = UsdGeom.Xformable(xform.GetPrim())
translateOp = xformable.AddTranslateOp()
translateOp.Set(time=0, value = Gf.Vec3d(0.0, 0.0, 0.0))
translateOp.Set(time=50, value = Gf.Vec3d(0.0, -1.0, 0.0))
```

1.22 Settings

The original documentation can be found here

1.22.1 Get app title

```
import carb
app_name = carb.settings.get_settings().get("/app/window/title")
# Isaac Sim, Create, Code, e.t.c.
```

1.22.2 Get app version

```
import carb
app_version = str(carb.settings.get_settings().get("/app/version"))
# 2022.1.1, e.t.c.
```

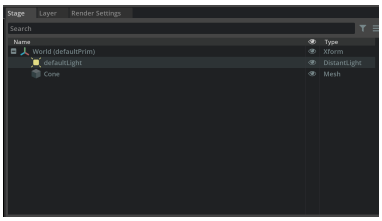
1.22.3 Persist preferences changes

Change the setting permanently

1.23 Stage

The stage window allows you to get all the assets in your current USD Scene. Listed in a hierarchical (parent/child) order the Stage offers convenient access and is typically used to navigate large scenes.

Learn more about Omniverse Stage



1.23.1 Get current stage

```
stage = omni.usd.get_context().get_stage()
```

1.23.2 Get layer

```
layer = stage.GetRootLayer()
# get layer path: layer.realpath
```

1.23.3 Get session layer

```
session_layer = stage.GetSessionLayer()
```

1.23.4 Get default prim

```
default_prim = stage.GetDefaultPrim()
```

1.23.5 Set default prim

```
stage.SetDefaultPrim(stage.GetPrimAtPath("/World"))
```

1.23.6 Traverse all prims

```
prim_list = stage.TraverseAll()
```

1.23.7 Get prim path and name

```
prim = stage.GetPrimAtPath("/World/Cube")
prim_path = prim.GetPath()
prim_name = prim_path.pathString
```

1.23.8 Create empty prim (xform)

```
from omni.physx.scripts import physicsUtils
physicsUtils.add_xform(stage, "/xform", pxr.Gf.Vec3f(0.0, 0, 0.0))

# or
path = omni.usd.get_stage_next_free_path(stage, "/xform", True)
xform_geom = pxr.UsdGeom.Xform.Define(stage, path)
```

1.23.9 Set up-axis and units

```
# set up axis to z
pxr.UsdGeom.SetStageUpAxis(stage, pxr.UsdGeom.Tokens.z)
pxr.UsdGeom.SetStageMetersPerUnit(stage, 0.01) # 0.01 meter
```

1.23.10 Move/Rename prim

```
old_prim_name = prim.GetPath().pathString
new_prim_name = prim.GetPath().GetParentPath()
new_prim_name = new_prim_name.AppendChild("Door1")
new_prim_name = omni.usd.get_stage_next_free_path(self.stage, new_prim_name.pathString,␣
↪False)
print("new_prim_name: ", new_prim_name)

move_dict = {old_prim_name: new_prim_name}
if pxr.Sdf.Path.IsValidPathString(new_prim_name):
    move_dict = {old_prim_name: new_prim_name}
    omni.kit.commands.execute("MovePrims", paths_to_move=move_dict,  on_move_fn=None)
else:
    carb.log_error(f"Cannot rename {old_prim_name} to {new_prim_name} as its not a valid␣
↪USD path")
```

1.23.11 Get/Set Prim Attribute

```
test_prim = self.stage.GetPrimAtPath("/World/Looks/component_45146_solid_001_wire1/
↪component_45146_solid_001_wire1")
attr = test_prim.GetAttribute("inputs:diffuse_texture").Get()
new_asset_path = str(attr).replace(":","_").replace("@","")
test_prim.CreateAttribute("inputs:diffuse_texture", pxr.Sdf.ValueTypeNames.String,␣
↪False).Set(new_asset_path)
```

1.24 Stream

This part includes the APIs for setting up physics in the event,

read more about event streams

1.24.1 App update

```
self._update_sub = omni.kit.app.get_app().get_update_event_stream().create_subscription_
↪to_pop(
            self._on_update, name="omni.physx demo update"
        )

def _on_update(self, e):
    # dt = e.payload["dt"]
```

1.24.2 Physics update

```
self._physics_update_sub = omni.physx.get_physx_interface().subscribe_physics_step_
↪events(self._on_physics_step)

def _on_physics_step(self, dt):
    # pass
```

1.24.3 Timeline event

```
stream = omni.timeline.get_timeline_interface().get_timeline_event_stream()
self._timeline_sub = stream.create_subscription_to_pop(self._on_timeline_event)

def _on_timeline_event(self, e):
    if e.type == int(omni.timeline.TimelineEventType.STOP):
        # pass
```

1.24.4 Stage event

```
stage_event_stream = self._usd_context.get_stage_event_stream()
        self._stage_event_sub = stage_event_stream.create_subscription_to_pop(self._on_
↪stage_event)


def _on_stage_event(self, event):
    if event.type is int(omni.usd.StageEventType.CLOSING):
        # pass
```

1.25 Tensor system

Package name `omni.physics.tensors` or `omni.physx.tensors`

1.25.1 Pure tensor view

Create tensor view:

```
sim = omni.physics.tensors.create_simulation_view("numpy")
sim.set_subspace_roots("/World/envs/*")
self.robots = sim.create_articulation_view("/World/envs/*/jetbot")
```

Obtain tensor value:

```
# get position
self.robots.get_dof_positions()
```

Set tensor value:

```
# set joint position
self.robots.set_dof_positions(self.robot_original_position, self.robot_indices)
self.robots.set_dof_position_targets(self.robot_original_position, self.robot_indices)

# set root position
self.robots.set_root_transforms(pos, self.robot_indices)
```

1.25.2 Robot

Set up robots

```
self.robots = RobotView("/World/envs/*/jetbot")
self.robot_indices = np.arange(self.robots.count, dtype=np.int32)
self.robots.initialize()
```

Get robot information

```
# self._backend_utils = np_utils  # import omni.isaac.core.utils.numpy as np_utils
self.robot_original_position = self._backend_utils.clone_tensor(self.robots._physics_
↪view.get_dof_positions())
self.xform_original_transform = self.robots.get_world_poses()
```

Apply action / Set position

```
action = np.random.uniform(-1, 1, (len(self.robot_indices), 2)) * 20
self.robots.set_joint_velocity_targets(action, self.robot_indices)
pos = [[0.5 * i, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0] for i in range(len(self.robot_indices))]
self.robots._physics_view.set_root_transforms(pos, self.robot_indices)
```

1.26 Timeline & Update

1.26.1 Get timeline

```
from omni.timeline import get_timeline_interface
timeline_iface = get_timeline_interface()
```

1.26.2 Play timeline

```
timeline_iface.play()
```

1.26.3 Pause timeline

```
timeline_iface.pause()
```

1.26.4 Stop timeline

```
timeline_iface.stop()
```

1.26.5 Update timeline

```
timeline_iface.set_auto_update(False)

for i in range(nun_frame):
    timeline_iface.forward_one_frame()
```

1.26.6 Set up time

```
# timeline_iface = get_timeline_interface()
# timeline_iface.play()
# timeline_iface.set_auto_update(False)
timeline_iface.set_current_time(seconds)
```

1.26.7 Set/Get timeline range

```
stage.SetStartTimeCode(10)
# stage.GetStartTimeCode()

stage.SetEndTimeCode(100)
# stage.GetEndTimeCode()
```

1.26.8 Loop/Cancel timeline

```
timeline.set_looping(True) # False
```

1.27 Transform

This part teaches how to get/set the `translate`, `orientation`, and `scale` of an object.

Note: You may get `stage` and object `prim` first,

```
stage = omni.usd.get_context().get_stage()
prim = stage.GetPrimAtPath("/World/game/franka")
```

1.27.1 Get translate/orient/scale (from attribute)

```
translate = prim.GetAttribute("xformOp:translate").Get()
orient = prim.GetAttribute("xformOp:orient").Get()
scale = prim.GetAttribute("xformOp:scale").Get()
```

1.27.2 Get translate/orient/scale (from transform matrix)

```
from omni.usd import get_world_transform_matrix, get_local_transform_matrix

# or new verion
# from omni.usd.utils import get_world_transform_matrix, get_local_transform_matrix

mat = get_world_transform_matrix(prim)

# or
# mat = UsdGeom.Xformable(usd_prim).ComputeLocalToWorldTransform(time)

# or local
mat = get_local_transform_matrix(prim)

translate = mat.ExtractTranslation()
quad = eval(str(mat.ExtractRotation().GetQuat()))
scale = mat.ExtractScale()
```

1.27.3 Set translate/orient/scale (from attribute)

```
prim.GetAttribute("xformOp:translate").Set(pxr.Gf.Vec3f(0,0,0))
prim.GetAttribute("xformOp:orient").Set(pxr.Gf.Quatd(1, 0, 0, 0))
prim.GetAttribute("xformOp:scale").Set(pxr.Gf.Vec3f(1, 2, 1))
```

Note: You may add `Attribute` first if the prim doesn't contain the translate attribute, read more about USDAttribute

```
prim.AddTranslateOp().Set(pxr.Gf.Vec3f(0,0,0))
prim.AddOrientOp().Set()
prim.AddScaleOp().Set(pxr.Gf.Vec3f(1.0, 1.0, 1.0))
```

1.27.4 Set translate/orient/scale (from transform matrix)

```
translate = [0, 0, 0]
scale = [1, 1, 1]
rotation = pxr.Gf.Quatd(1, 0, 0, 0)
xform = pxr.Gf.Matrix4d().SetScale(scale) * pxr.Gf.Matrix4d().SetRotate(rotation) * pxr.
↪Gf.Matrix4d().SetTranslate(translate)

omni.kit.commands.execute(
```

(continues on next page)

(continued from previous page)

```
    "TransformPrimCommand",
    path=prim.GetPath(),
    new_transform_matrix=xform,
)
```

1.28 UI

The UI system allows you to build up your own graphics control.

Learn more about Omniverse omni.ui

Note: Get started by importing the `omni.ui`:

```
import omni.ui as ui
```

1.28.1 Build window

```
self._window = ui.Window("VRKitchen Asset Importer", width=500, height=300)
with self._window.frame:
    with ui.VStack():
        with ui.HStack(height=30):
            ui.Label("Welcome!", width=200)
```

1.28.2 String UI

```
ui.Label("Scene folder: ", width=20)
self.scene_asset_path_ui = omni.ui.StringField(height=20, style={ "margin_height": 4 })
self.scene_asset_path_ui.model.set_value(SCENE_ASSET_PATH)

# get string
self.scene_asset_path_ui.model.get_value_as_string()
```

1.28.3 Integer UI

```
ui.Label("Example id: ", width=20)
self.example_id_ui = omni.ui.IntField(height=20, style={ "margin_height": 8 })

# get int
self.example_id_ui .model.get_value_as_int()
```

1.29 Animation Overview

The way we make animation is to utilize the `OmniGraph`, `Action Graph`, and `Animation Graph` in Omniverse. Along with the graphic expression, we try to manage the animation building by transferring it to coding-based method. Those documentations are really helpful for our work:

- OmniGraph
- Action Graph
- Animation Graph

1.30 Animation Graph

Note: Common import:

```
from pxr import Sdf, Usd
from pxr import AnimGraphSchema, AnimGraphSchemaTools
import omni.anim.graph.core as ag
import carb
```

1.30.1 Add animation graph

```
anim_graph = AnimGraphSchemaTools.createAnimationGraph(stage, Sdf.Path("/World/
↪AnimationGraph"))

# or
omni.kit.commands.execute("CreateAnimationGraphCommand", path=Sdf.Path("/World/
↪AnimationGraph"), skeleton_path=Sdf.Path.emptyPath)
```

Note: we may set/clear original skeletal animation at root joint by

```
skeleton_prim = stage.GetPrimAtPath("/World/character/f_avg_root")
skeleton_bindingAPI = UsdSkel.BindingAPI(skeleton_prim)

# set
skeleton_bindingAPI.GetAnimationSourceRel().SetTargets(["/World/character/f_avg_root/
↪Animation"])

# clear
skeleton_bindingAPI.GetAnimationSourceRel().SetTargets([])
```

1.30.2 Set Skeleton

```
rel = anim_graph.GetSkelSkeletonRel()
rel.SetTargets([_skeleton_path])

# or
omni.kit.commands.execute("CreateAnimationGraphCommand", path=Sdf.Path("/World/
↪AnimationGraph"), skeleton_path=_skeleton_path)
```

1.30.3 Apply animation graph api

```
AnimGraphSchemaTools.applyAnimationGraphAPI(stage, path, self._animation_graph_path)

# or
omni.kit.commands.execute("ApplyAnimationGraphAPICommand", paths=self._apply_prim_paths,␣
↪animation_graph_path=selected_prim.GetPath())
```

1.30.4 Remove animation graph api

```
if stage.GetPrimAtPath(path).HasAPI(AnimGraphSchema.AnimationGraphAPI):
    AnimGraphSchemaTools.removeAnimationGraphAPI(stage, path)

# or
omni.kit.commands.execute("RemoveAnimationGraphAPICommand", paths=selected_paths)
```

1.30.5 Create Animation Clip

```
omni.kit.commands.execute(
            'CreatePrimCommand',
            prim_type="AnimationClip",
            prim_path="/World/AnimationGraph/Animation",
            select_new_prim=True,
        )
```

1.30.6 Get character

```
character = ag.get_character("/World/character")
```

1.30.7 Get character position (run time)

```
t = carb.Float3(0, 0, 0)
q = carb.Float4(0, 0, 0, 1)
character.get_world_transform(t, q)
```

1.30.8 Get joint position (run time)

```
t = carb.Float3(0, 0, 0)
q = carb.Float4(0, 0, 0, 1)
character.get_joint_transform("f_avg_L_Foot", t, q)
```

1.31 Animation Editing

Note: Common import:

```
from pxr import AnimGraphSchema, AnimGraphSchemaTools, UsdSkel
from pxr import Gf, Sdf, Usd
```

Read more about USD SKEL ANIMATION.

1.31.1 Get animation clip from prim

```
anim_clip = AnimGraphSchema.AnimationClip(prim)
has_anim = AnimationSchemaTools.HasAnimation(prim)
```

1.31.2 Get animation source

```
src_rel = anim_clip.GetInputsAnimationSourceRel()
if src_targets:
    src_skel_prim = stage.GetPrimAtPath(src_targets[0])
```

1.31.3 Get skeletal animation

```
src_skel_anim = UsdSkel.Animation(src_skel_prim)
```

1.31.4 Get translation from animation at time

```
timecode = Usd.TimeCode(0.0 * stage.GetTimeCodesPerSecond())
trans = animation.GetTranslationsAttr().Get(timecode)
rots = animation.GetRotationsAttr().Get(timecode)
scales = animation.GetScalesAttr().Get(timecode)
joints = animation.GetJointsAttr().Get()
```

1.31.5 New animation

```
animation_new_path = "/World/AnimationGraph/newAnimation"
animation_new = UsdSkel.Animation.Define(stage, animation_new_path)
# animation_new_prim = animation_new.GetPrim()
```

1.31.6 Set up animation data

```
with Sdf.ChangeBlock():
   animation_new.GetJointsAttr().Set(joints)
   animation_new.GetTranslationsAttr().Set(trans)
   animation_new.GetRotationsAttr().Set(rots)
   animation_new.GetScalesAttr().Set(scales)

   # animation_new.GetRotationsAttr().Clear()
```

1.31.7 Bind animation target

```
skeleton_prim = self.stage.GetPrimAtPath(skeleton_path)
skeleton_bindingAPI = UsdSkel.BindingAPI(skeleton_prim)
skeleton_bindingAPI.GetAnimationSourceRel().SetTargets([animation_new_path])
```

1.32 Autodesk Maya

- Get pip.py
- Install numpy and scipy to Maya

1.32.1 API Documentaion links

- Maya 2023 Python
- Maya 2023 MEL

maya_api.utils.**clear_keys**()

Clear all keyframes for selection.

Parameters

- **a** – The first arg.

- **b** – The second arg.

Returns
Something.

1.33 Pixar USD

http://blog.christianlb.io/building-usd-on-windows-for-python-3

BIBLIOGRAPHY

[XQM+20] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, and others. Sapien: a simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11097–11107. 2020.

[ZLJ+21] Yizhou Zhao, Kaixiang Lin, Zhiwei Jia, Qiaozi Gao, Govind Thattai, Jesse Thomason, and Gaurav S Sukhatme. Luminous: indoor scene generation for embodied ai challenges. *arXiv preprint arXiv:2111.05527*, 2021.

PYTHON MODULE INDEX

m

INDEX

C

M