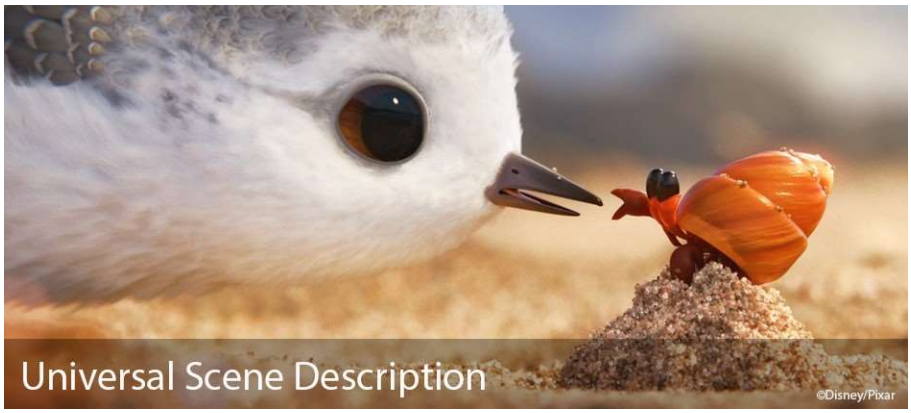
TUTORIAL: SETTING UP AND BUILDING PIXAR'S USD ON WINDOWS

JANUARY 10, 2019 • 20 COMMENTS



Setting up Pixar's Universal Scene Description (USD) framework can be a daunting task, and documentation tends to be spread around. Here's a quick rundown of what you need to do. (With thanks to ***VFXPro99***'s instructions here)

Installing Required Software

Git Support

You're going to need a way to pull the USD Git repository. If you're new to Git, one of the easiest ways to do this is to use the GitHub Desktop GUI. If you're experienced with Git, of course, use whatever UI you prefer.

- Download and install **GitHub Desktop** or your preferred Git interface.
- While a separate Git install isn't required to use GitHub Desktop, you'll probably find it useful to have it set up on your

system. Get Git here.

7-Zip

- Download the **7Zip** file archiver.
- Install it.
- Add its install location (usually C:\Program Files\7-Zip) to the **System Path** .

Python 2.7

- Download **Python 2.7**. (Python 3 is not yet supported.)
- Run the installer
  - Select the option to add Python 2.7 to the **System Path**
  - Important: you may need to add your **Python Scripts** directory to your System path by hand. (Usually **C:\Python27\Scripts**)
    - Verify that your System Path now includes **C:\Python27** and **C:\Python27\Scripts**, and fix your paths if for some reason they're not present.

CMake

- Download **CMake**.
- Install it and select the option to add CMake to the **System Path** for all users.

NASM

- Download **NASM**.
- Run the installer as an **Administrator**
  - When the warning appears, select **More Info > Run Anyway**
  - Install for anyone using this computer
  - Accept the default options
- Add **C:\Program Files\NASM** to the **System Path**

Microsoft Visual Studio

- Download **Visual Studio Community**.
  - **Important:** As of this writing, the build_usd.py script we'll be running to build USD does not support VS 2019. You'll need **VS 2017** for this.
    - You can find older versions here: https://visualstudio.microsoft.com/vs/older-

downloads/

- Install it with the option to **develop for desktop C++** selected.

Microsoft Visual Studio Code

- Download and install **Visual Studio Code**.
- Install the Python plugin for VSCode and reload it
- Create a simple Python file (any file with extension .py)
  - Contents can be simple, for example:

```
msg = “Foo”
Print msg
```

  - Saving the file with a .py extension should prompt VSCode to install **PyLint -** let it.
- For further information on setting up VSCode to work with Python, check here. Be aware though that these directions discuss setting up Python 3, but USD requires Python 2. You can set up both Python versions on the same machine, but life gets more complicated if you do.

Getting the USD source

- Navigate to **Pixar’s USD Github repository**.
- Select **Clone or Download > Open in Desktop** (or again, if you’re already comfortable with Git, clone it by whatever means you prefer.)
- Allow GitHub Desktop to clone the depot to the directory of your choice.

Preparing to Build

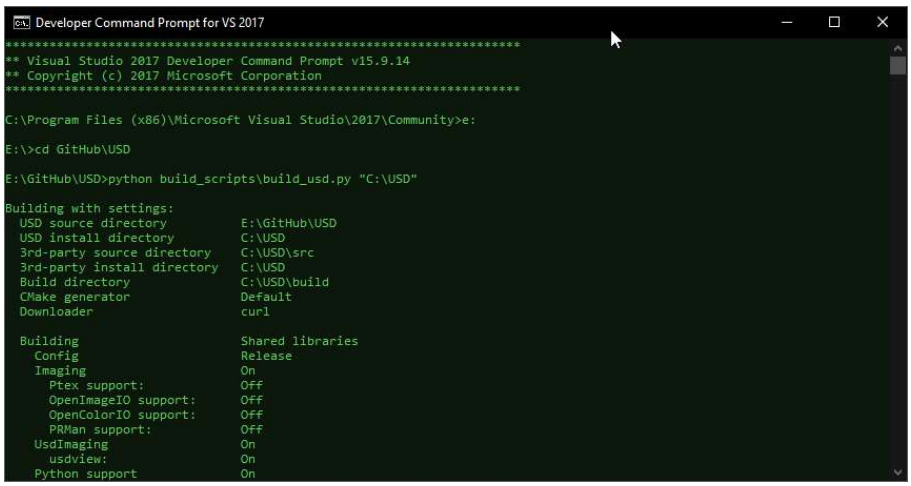
- Run **Start Menu > Visual Studio 2017 > x64 Native Tools Command Prompt for VS 2017**
  - You must use the Native Tools Command Prompt - a standard command prompt won’t allow you to build using MSVC from the command line.
- Verify your tools:
  - Type ‘**cl**’ - you should see the optimizing compiler’s usage notes appear
  - Type ‘**7z**’ - you should see the 7-Zip usage notes appear

- Type '**python**' - Python 2.7 should start. Type exit() to exit it.
  - If you have both Python 2 and Python 3 on your machine, type '**py -2**' instead.
- Type '**cmake**' - CMake usage notes should appear
- Type '**nasm**' - NASM should complain that no input file was specified.

- Install dependencies:
  - Type '**pip install PySide**'
    - For this and the following commands, it may be necessary to specify which pip executable to run if you have multiple versions of Python installed. In this case, you can specify the path directly:
      - C:\Python27\Scripts\pip install PySide
    - Let the install finish.
  - Type '**pip install pyd**'
    - Possibly not required
  - Type '**pip install pyopengl**'
    - Required for the USD viewer
  - Type '**pip install Jinja2**'
    - USD_GENSCHEMA will not be set otherwise. You'll need this if you intend to build Autodesk's (Formerly Animal Logic's) Maya_USD plugin.

Building USD

- In the Native Tools Command Prompt, navigate to the location where you cloned your USD source.



- Type **python build_scripts\build_usd.py "C:\USD"**
  - If you plan to use Autodesk's maya_usd plugin, you need to ensure that your USD build includes the Boost libraries, and does not build Pixar's Maya USD plugin. Add boost arguments to your build command, and use the —no-maya switch, like so:
    - python build_usd.py --build-args boost,"--with-date_time --with-thread --with-system --with-filesystem" --no-maya "C:\USD"
  - Use py -2 rather than python if you need to specify explicitly that you want to run the build script using python 2. The script will not run under python 3.
  - The argument at the end is the location where you'd like to install the USD binaries. You can specify another location here if desired.
  - ***If your build fails:***
    - Verify that you're running from the **Native Tools Command Prompt**, not an ordinary command line.
    - Verify that you have **VS 2017** installed and that you're running the Native Tools Command Prompt associated with it.
      - An error like the following indicates that you're missing the required VS install:

```
cmake -DCMAKE_INSTALL_PREFIX="C:\USD" -DC
```

```
CMake Error at CMakeLists.txt:4 (project)
```

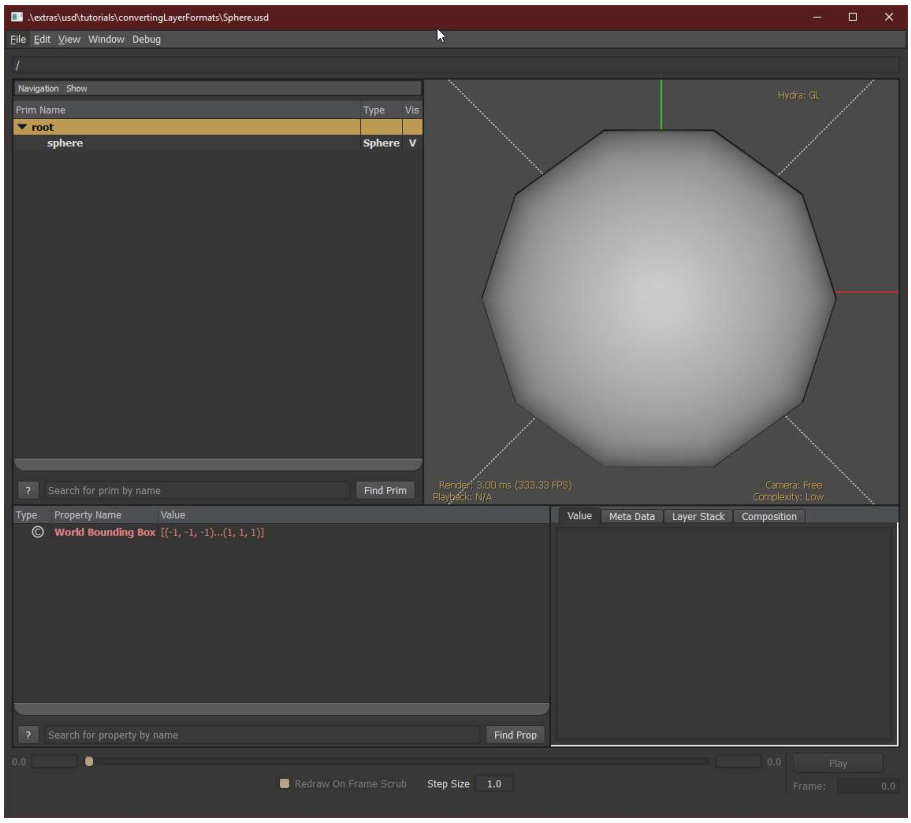
```
  Generator

    Visual Studio 15 2017 Win64

  could not find any instance of Visual S
```

- Allow the build to complete
- Add a **PYTHONPATH** variable to your system variables as instructed at the end of the install output and populate it as instructed.
  - If you installed USD to C:\USD, this path will be **C:\USD\lib\python**
- Add **C:\USD\bin** and **C:\USD\lib** to your **System Path** as instructed.
  - If you installed elsewhere, of course, these paths will be different.

```
    HDF5 Support:             Off
    MaterialX Plugin          Off
    Maya Plugin               Off
    Katana Plugin             Off
    Houdini Plugin            Off

  Dependencies                zlib, boost, TBB, OpenEXR, GLEW, OpenSubdiv
STATUS: Installing zlib...
STATUS: Installing boost...
STATUS: Installing TBB...
STATUS: Installing OpenEXR...
STATUS: Installing GLEW...
STATUS: Installing OpenSubdiv...
STATUS: Installing USD...

Success! To use USD, please ensure that you have:

    The following in your PYTHONPATH environment variable:
    C:\USD\lib\python

    The following in your PATH environment variable:
    C:\USD\bin
    C:\USD\lib


E:\GitHub\USD>
```

Testing the USD Build

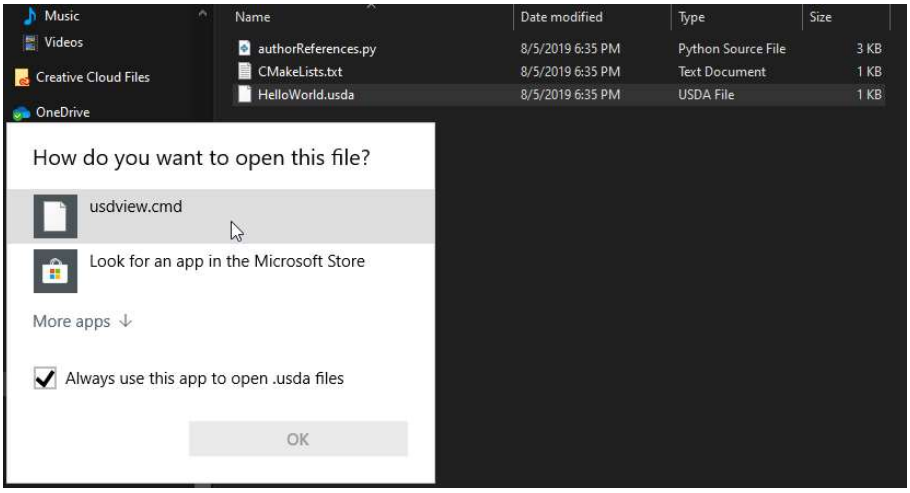
- Open a new command window (it no longer needs to be a Native Tools Command Prompt - Powershell or even the default command line is fine here now.)
- Navigate to the location where you cloned your USD GitHub depot
    - Type: **usdview .\extras\usd\tutorials\convertingLayerFormats\Sphere.usd**
- If you correctly set up your PYTHONPATH and Path variables in the last step, the USD viewer should now open and display a sphere.



USDView

If you've completed these steps and USDView successfully launched, you should be ready to work through the USD tutorials.

(You may also find it useful to set C:\USD\bin\usdview.cmd as the default application for .usd and .usda files in your Windows settings.)



Building the Autodesk (formerly Animal Logic) Maya-USD plugin

If you plan to manage USD stages using Maya, you'll likely want to use the Animal Logic Maya_USD plugin now maintained by Autodesk. Here's how to get it built:

- Pull the source from Autodesk's Maya-USD GitHub depot.
- By now you already have your compiler, CMake and Python set up and verified.
- Download the Ninja build system, extract the executable and put it in your system path. Test it by typing 'ninja -h' from a command line to make sure it's recognized.
- Ensure that you built USD with boost enabled and with the Pixar Maya plugin disabled. If you previously built it without the boost or no-maya build arguments, rebuild it like so:
    - python build_usd.py --build-args boost,"--with-date_time --with-thread --with-system --with-filesystem" --no-maya "C:\USD"
        - Remember that you must build this from the **x64 Native Tools Command Prompt for VS 2017**. CMake will not find your compiler otherwise.
- Navigate to the directory where you pulled the maya-usd source, and invoke the build.py Python script to build it.

- python build.py --generator Ninja -v 3 --maya-location "C:\Program Files\Autodesk\maya2018" --pxrusd-location D:\USD --devkit-location D:\usd\mayaDevkit --install-location D:\usd\usdplugin --build-args="-DBUILD_ADSK_PLUGIN=ON" D:\usd\workspace
  - This too must be done from the **x64 Native Tools Command Prompt for VS 2017**.
- A few of these arguments bear explaining:
  - The first argument specifies that you want to use the **Ninja** build system to build the plugin.
  - Set **maya-location** to the location of your Maya install.
  - Set **pxrusd-location** to the location of your built USD toolset.
  - **devkit-location** may not be needed.
  - Set **install-location** to the location where you want your built plugin to land.
  - **DBUILD_ADSK_PLUGIN** defaults to ON so it may not be necessary to specify this here.
  - The final argument is the location of a workspace directory where you'd like your generated intermediate files to land.
- NOTE: As of 13 Jan 2020, the Master branch does not build correctly. This has been corrected in this branch:
  - https://github.com/Autodesk/maya-usd/blob/fix_import_module_macosx/doc/build.md
  - If your build fails claiming that:
    - "set_tests_properties called with incorrect number of arguments"
    - Then use this alternate branch to build.

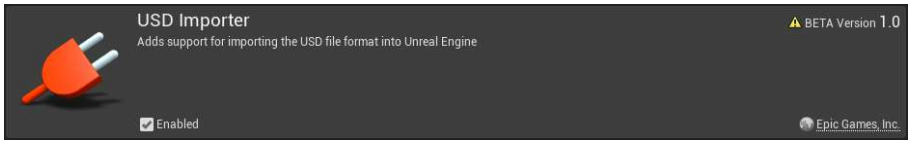
Find further information about the Maya-USD plugin here.

OUTDATED - Performing a simple Export/Import test using Unreal 4.22

Next, let's set up Unreal Engine 4 to import and export USD. This functionality is experimental as of 4.22 and requires a bit of manual handling, but it's enough to begin experimenting and see how the system works.

First, create a new project in Unreal 4.22.

Once the project has been created, open Settings | Plugins, and enable the USD Importer plugin and the Python Editor Script Plugin.



Allow the editor to restart.

Now for a simple test:

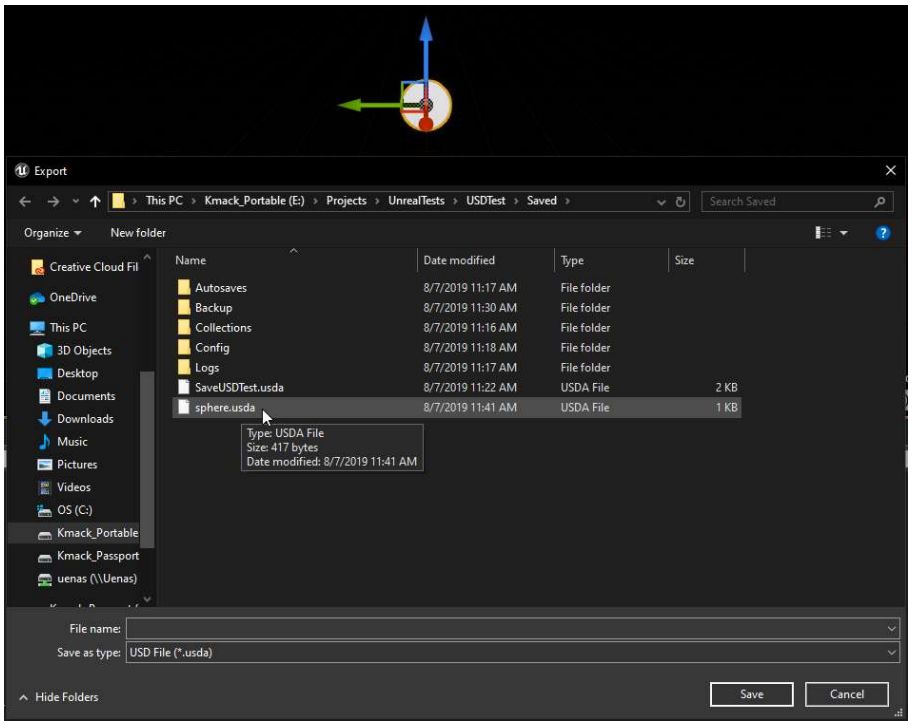
Create a new empty level.

Place a basic primitive, like a Sphere, into the level.

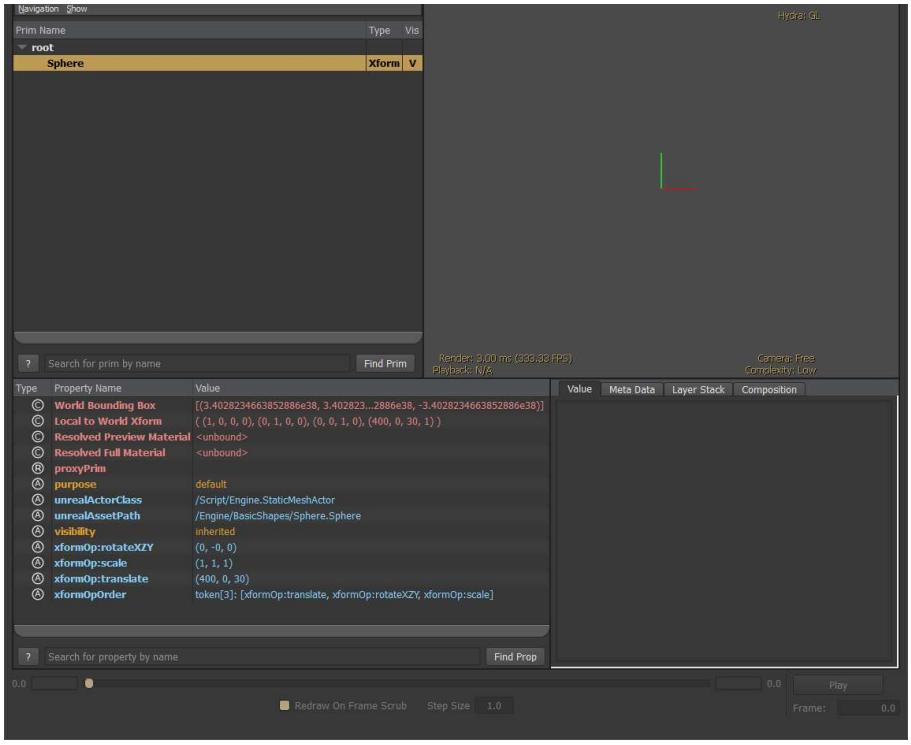
Select the sphere and hit File | Export Selected.

Set its Save As Type to .usda (USD supports binary .usd files and ASCII .usda files. When debugging, you'll find it handy to work with .usda files so you can view their contents.)

Save it in a reasonable location.



If you now try opening the exported file in USDView, you'll see that the geometry does not appear in the viewport, but its data appears valid.



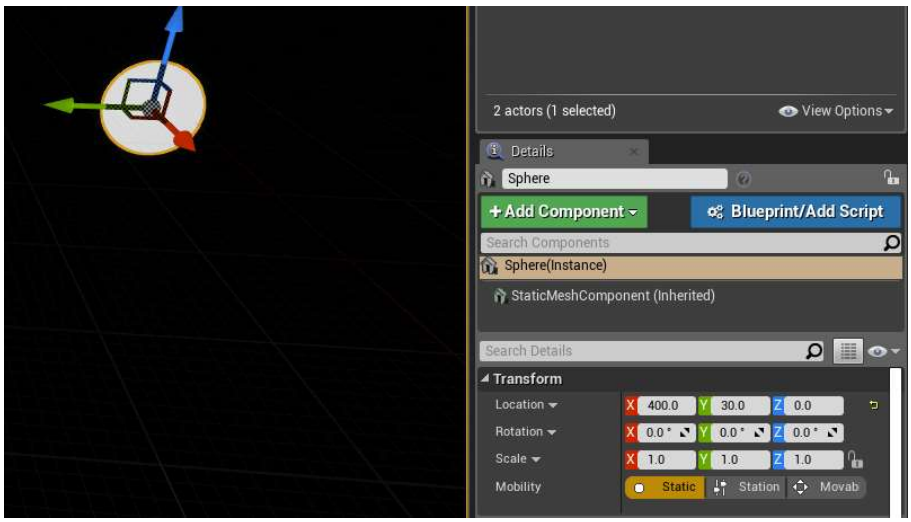
This is because the geometry has been saved as a reference to an unrealActorClass, but we haven't made any information about this class available to USD.

```
#usda 1.0
(
    upAxis = "Y"
)

def Xform "Sphere"
{
    custom string unrealActorClass = "/Script/Engine.StaticMeshActor"
    custom string unrealAssetPath = "/Engine/BasicShapes/Sphere.Sphere"
    float3 xformOp:rotateXZY = (0, -0, 0)
    float3 xformOp:scale = (1, 1, 1)
    double3 xformOp:translate = (400, 0, 30)
    uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXZY", "xformOp:scale"]
}
```

If we view the file in a text editor (If you're using Visual Studio Code, I recommend enabling Animal Logic's USD Language extension), we can see this as well - the actor's unrealActorClass and unrealAssetPath have been specified along with its transform, but no other information is contained within the file.

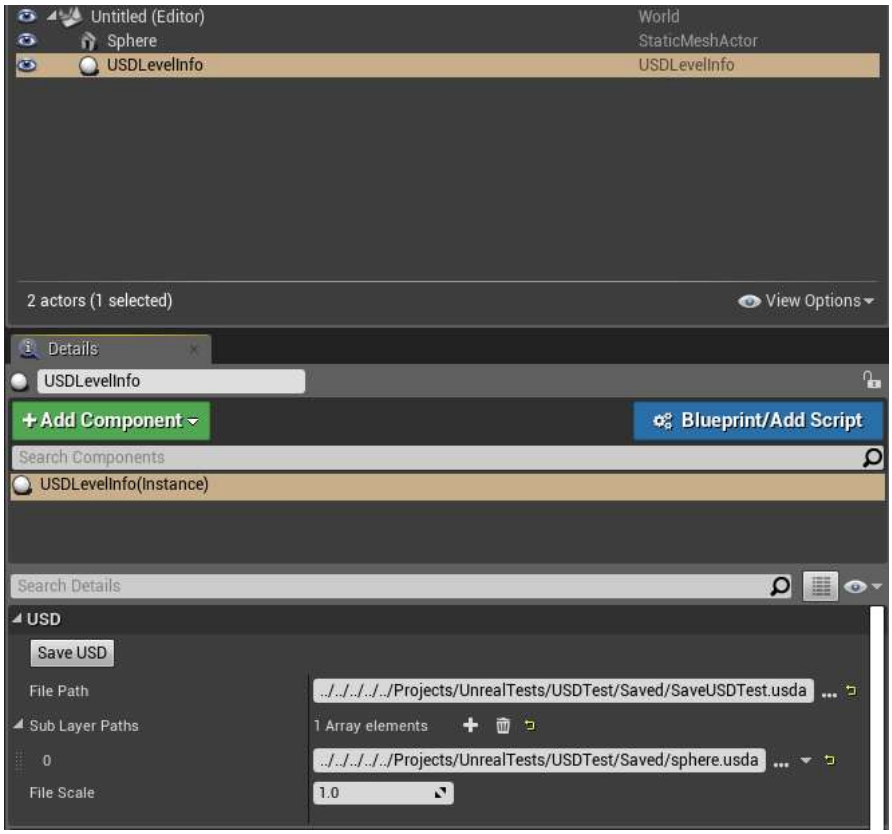
For this simple test, this is fine.

Now let's create another empty map in Unreal, select File | Import Into Level, and select the .usda file we just exported. You should see it appear in your level at the specified transform.



There's also an interesting additional object in our level: a USDLevelInfo actor has been created.

If we select this actor, we'll see that it gives us the option to specify a save file path, and that our imported sphere has automatically been added as a USD sub-layer. Let's specify a destination path and hit Save USD.



The resulting USD file is even simpler than the previous - it simply references the Sphere sublayer we'd previously exported.

```
#usda 1.0
(
    subLayers = [
        @./sphere.usda@
    ]
    upAxis = "Y"
)
```

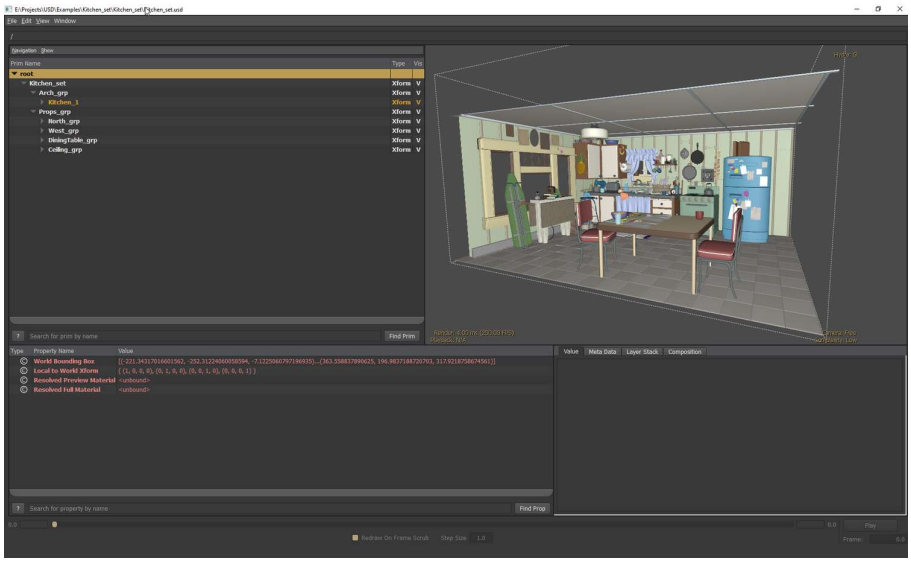
If we now create an empty Unreal level again and import the level .usda we just imported, we'll see our geometry appear and if we look at our USDLevelInfo object, we'll see that the imported level .usda is now identified as a sub-layer.

This is a major aspect of USD's power - it can describe extremely large projects by referencing assets and entire scenes as sub-layers. We'll talk about this more in another post.

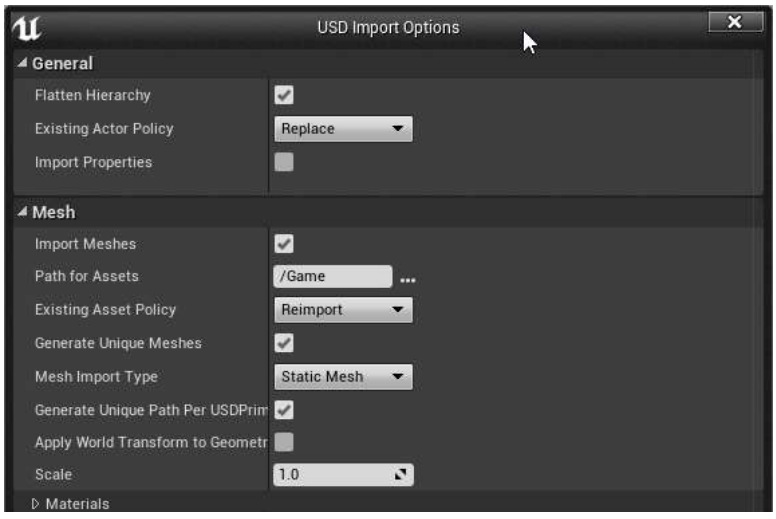
What we've done here though should begin to give you a framework from which to explore USD, learn how it describes information, and learn how Unreal can generate and read the format.

Importing a USD Scene to Unreal

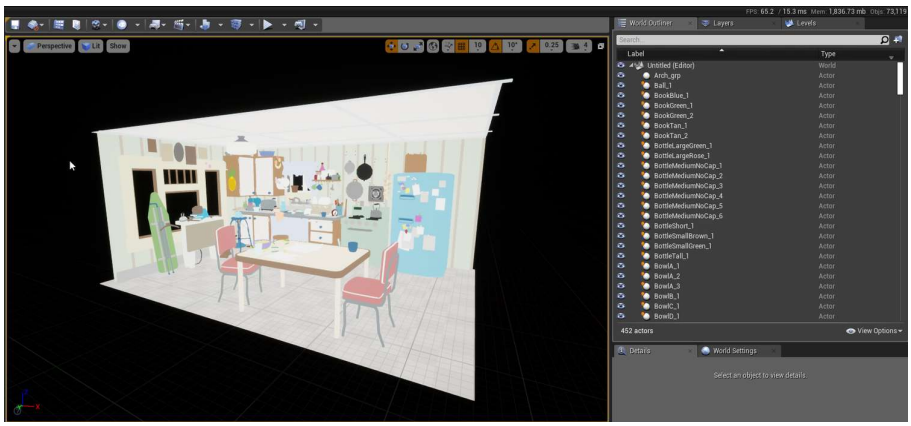
Now let's try importing a full scene. Head over to http://graphics.pixar.com/usd/downloads.html and download the Kitchen Set scene. Extract it and verify it in USDView.



Now, in UE4, create a new empty level and select File | Import Into Level. Point the importer at the Kitchen_set.usd you just checked. The default import options are fine.



After a few moments, the scene should be properly constructed in your Unreal level.



Additional Resources

- Pixar's official USD documentation
- Colin Kennedy's excellent USD Cookbook
- The USD Interest Google group
- AnimalLogic's USD Maya plugin
- ILM's MaterialX specification, while not part of USD, is relevant.

Comments (20)

Newest First

Preview POST COMMENT...

Francois 4 months ago · 0 Likes

This is an amazing source. Whether you get stuck along the way or not, this still gets you much further than one would've gotten to by yourself. So thanks! It worked 100% perfectly for me btw, except that the --no-maya part doesn't seem to work anymore. Doesn't seem like the --no-

maya argument even exists anymore? Anyways, the rest worked fine!

weirdmonkey2807 7 months ago · 0 Likes

Nvidia now provides pre-built binaries that make this a lot easier:
https://developer.nvidia.com/usd

symbz 7 months ago · 0 Likes

ive never seen such messy installs to get something working. It amazes me that in 2020 this "new standard" of doing pipeline is such a mess. Far from user friendly or having clear instructions across the community. Frustrating process to say the least. Thanks for the help. But its a total mess.

Gleb 9 months ago · 0 Likes

Hi, problems with boost.

ERROR: Failed to extract archive boost_1_70_0.tar.gz: unrecognized archive file type

but if i change extension to .zip in build_usd.py i got:

ERROR: [WinError 32] The process cannot access the file because it is being used by another process: 'boost_1_70_0.zip'

Jorge del Valle 9 months ago · 0 Likes

Thank you for writing this guide. After following all the steps I'm running into a problem I'm hoping you can help with.

When I try to run usdview I get this error:
Traceback (most recent call last):
File "C:\USD\bin\usdview", line 28, in <module>
import pxr.Usdviewq as Usdviewq
File "C:\USD\lib\python\pxr\Usdviewq__init__.py", line 31, in <module>
from .appController import AppController
File "C:\USD\lib\python\pxr\Usdviewq\appController.py", line 38, in <module>
from pxr import Usd, UsdGeom, UsdShade, UsdUtils

UsdImagingGL, Glf, Sdf, Tf, Ar

File "C:\USD\lib\python\pxr\UsdImagingGL__init__.py", line 24, in <module>
from . import _usdImagingGL
ImportError: DLL load failed: The specified procedure could not be found.

Looks like it's tripping on trying to import UsdImagingGL, as I get the same thing if I run "from pxr import UsdImagingGL" in python.

I'd appreciate any ideas you may have on how fix this issue.

Pankaj Baviskar 10 months ago · 0 Likes

It's really a good guide to install USD, however I'm still getting below error after following all the steps mentioned above,
"2020-05-18 07:56
bootstrap.bat --prefix="C:\Program Files\USD"
Building Boost.Build engine
=C:\Program was unexpected at this time.

ERROR: Failed to run 'bootstrap.bat --prefix="C:\Program Files\USD"'
See C:\Program Files\USD\src\boost_1_70_0\log.txt for more details."
Also the log file has the same error message.

Pankaj Baviskar 9 months ago · 0 Likes

I have solved the issue mentioned above, I think installation path should not have space in folder name.

John Pope 10 months ago · 0 Likes

I'd add a big DEPRECATED to this page down - and point people here instead - https://github.com/vfxpro99/usd-build-club

Mike Pelton 10 months ago · 0 Likes

Not sure I agree John - the instructions here are very clear - I found the build club page less digestible.

Sushant 11 months ago · 0 Likes

Hello
I got this error

"boost_filesystem" is
considered to be NOT FOUND. Reason given by package:

No suitable build variant has been found.
Could some one help me???

Dani Ramdani A year ago · 0 Likes

Hi, I followed the tutorial, and I got this error:
STATUS: Installing boost...
ERROR: [Error 2] The system cannot find the file specified
Do you know what's the problem?
Thank you for the tutorial

james A year ago · 0 Likes

Thanks for all the info!,

This install was going well for me until core USD build.
Using the command:

python build_usd.py --build-args boost,"--with-date_time --with-thread --with-system --with-filesystem" --no-maya "C:\USD"

I got an error reporting there was no --no-maya flag:
build_usd.py: error: unrecognized arguments: --no-maya

Is this no longer needed because it has been removed?

https://github.com/PixarAnimationStudios/USD/commit/702873ecea7fed0d0a43f2

Thanks

James

Volodymyr Haivoronskyi A year ago · 0 Likes

Hi! Thank You for great tutorial!

It works fantastically on my machine. Only for some reason I can run usdview only under VS2015 x64 Native Command Prompt.
Any Ideas how to fix that?

Pankaj Nailwal 2 years ago · 0 Likes

Hi
This is one and only great blog I found 3 days which is

I have stuck at "Type python build_scripts\build_usd.py "C:\USD"" this step. My native 64 command prompt is throwing an error - ERROR: pyside-uic not found -- please install PySide and adjust your PATH. (Note that this program may be named pyside-uic or python2-pyside-uic or pyside-uic-2.7 depending on your platform)
I have checked this file from browser and ran it from window and command prompt both. on running from command prompt it is throwing error - Error: one input ui-file must be specified.
Can you please help me out how can i come out from this error.

Thanks

brad 2 years ago · 0 Likes

This site is a huge help.Made it very easy to setup and start compile. But currently running into memory issues. Even though I have 4GB free space, the compiler errors and stops with the error"
c1xx : error C3859: Failed to create virtual memory for PCH" anyone have memory issues like this?

Kevin Mack 2 years ago · 0 Likes

Hi Brad,
Really happy you found it useful! I haven't run into the issue you're describing but it looks like there's a good discussion of the topic here:
https://devblogs.microsoft.com/cppblog/precompiled-header-pch-issues-and-recommendations/

Harry 2 years ago · 0 Likes

Thanks for the great tutorial for building USD on windows. Do you think that USD for windows is stable enough to use for production work without a lot of bugs?
We see a lot of potential in USD but are afraid to spend time on it if its not ready for production level work
We will be using it as a plugin for Maya.

Kevin Mack 2 years ago · 0 Likes

Hi Harry!
I would say the right way to think about it now is that it's a tech that's coming fast, but isn't yet a plug-and-play solution. I think it's worth it to start exploring and see where it fits your needs and where it doesn't, but I

wouldn't yet make the decision to jump fully in without doing some tests. You'll probably still need some engineering support to keep everything running right now, but if you start experimenting and testing with it, you'll be in good shape as everything stabilizes and becomes more user-friendly. I think the right way to think about USD now is as a standard that's rapidly arriving, so it's not a waste of time to learn it by any means, but not as a standard you could drop in without expecting to step on a few rakes.

Tom 2 years ago · 0 Likes

Hi, thanks for the tutorial!
I'm running into a problem when building USD, however... I followed all the steps, and when I get to `py -2 build_scripts/build_usd.py &quot;C:\USD&quot;`, it fails when it tries to build `zlib` with the message: No CMAKE_C_COMPILER could be found. In CMakeError.log it complains with `fatal error LNK1104: cannot open file 'MSVCRTD.lib'`.
This MSVCRTD.lib is in my Visual Studio/Tools/.../lib folder.

Any ideas or suggestions about how to get around this?

Kevin Mack 2 years ago · 1 Like

Hi Tom!
Happy you found it helpful! My first question would be are you initiating the build from the Native Tools Command Prompt? This automatically sets up the required environment for VS to execute the build - it won't easily be made to work from a standard command line.

PREVIOUS
Merging Meshes in Unreal and Why It Matters