USD Technical Challenge Notes

Bradley Webster

Concept Idea

I would like to learn about USD's model variants and layers to make an instanced project from one scene.

I've looked around for a few ideas and have decided to create a swarm of spaceships.

This would allow me to create multiple different model variants (ships) and use animation layers to make them move.

The models/shaders would be created in Maya, then exported to USD and brought into Katana for instancing/Lookdev/Lighting/Rendering.

Challenges / Issues faced

1. **Dependencies when building USD**

   I found that when I was building the USD, the Visual Studio build I setup could not download the dependencies for the program.

   Through investigating the USD GitHub issues, I found that other users have had this issue in the past:
   https://github.com/PixarAnimationStudios/USD/issues/841

   Although given not much information how the user solved the problem, I found manually downloading the dependencies and editing the *USD/Build_Scripts/build.usd.py* and searching/modifying the dependency links to a locally downloaded dependency folder, I was able to continue with the installation.

2. **Building Katana for Windows**

   At this current time, Windows Katana is not available with the current release build of USD

   This brings up a major problem, my system is currently Windows and I don't have the space to go and setup a Linux build.

   I did a search of the Pixar USD Github issues area and found a branch with a build for Windows Katana.

   To get Katana for Windows setup, I looked to receiving some support on the situation. I contacted my colleagues and also set up a support request on the Pixar USD page.
   https://github.com/PixarAnimationStudios/USD/issues/854

   After many dependency downloads and Environment Variable changes, I was successful at setting up USD for Katana!

3. **Setting up the USD scene**

   Setting up a scene for the first time is quite interesting in USD.

   One problem with new software, is the sparseness of tutorials outside of the main documentation.

   I decided to follow a few of the tutorials and found them to be complicated but straight forward.

   After learning the basics, I headed into Maya and made a basic model, with a few parts that could be switched around.

I exported the file as a .USD within Maya/

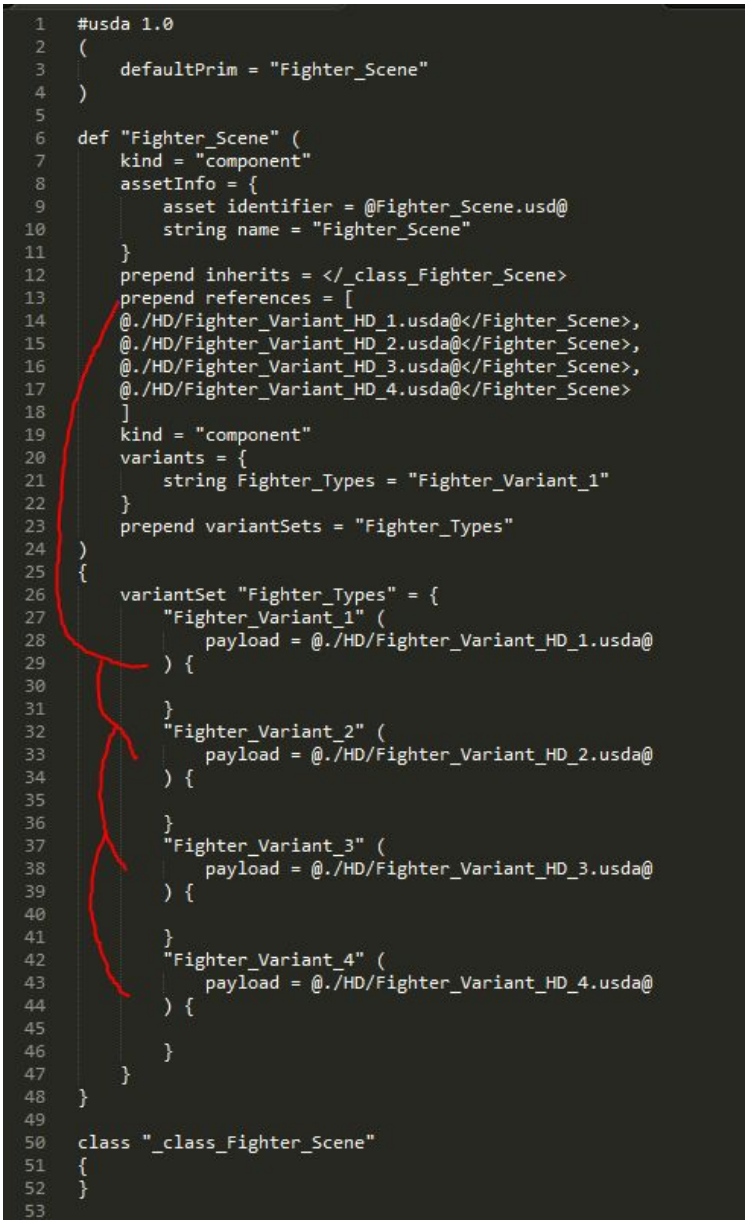
4. **Model Variants**

I wanted to setup two different variants of each fighter I was creating, to have a High-def and Low-def model.

Setting it up like this, would allow me to see how many instances of each I can create, before having framerate issues in the Hydra Viewer.

Using a combination of the help provided by the "Authoring Versions" & "Variants example in Katana" tutorials, I was able to set up my USD variants.

I found that i had a few issues, setting my variants to be prepend payloads, after testing, the way to resolve the issue was to prepend the USD variant references as part of the init to the file.



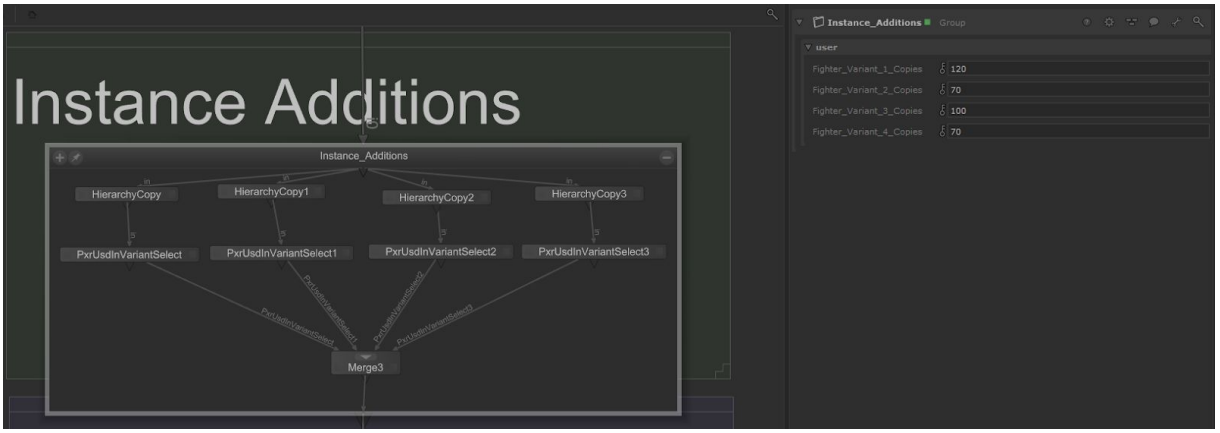
```
#usda 1.0
(
    defaultPrim = "Fighter_Scene"
)

def "Fighter_Scene" (
    kind = "component"
    assetInfo = {
        asset identifier = @Fighter_Scene.usd@
        string name = "Fighter_Scene"
    }
    prepend inherits = </_class_Fighter_Scene>
    prepend references = [
    @./HD/Fighter_Variant_HD_1.usda@</Fighter_Scene>,
    @./HD/Fighter_Variant_HD_2.usda@</Fighter_Scene>,
    @./HD/Fighter_Variant_HD_3.usda@</Fighter_Scene>,
    @./HD/Fighter_Variant_HD_4.usda@</Fighter_Scene>
    ]
    kind = "component"
    variants = {
        string Fighter_Types = "Fighter_Variant_1"
    }
    prepend variantSets = "Fighter_Types"
)
{
    variantSet "Fighter_Types" = {
        "Fighter_Variant_1" (
            payload = @./HD/Fighter_Variant_HD_1.usda@
        ) {

        }
        "Fighter_Variant_2" (
            payload = @./HD/Fighter_Variant_HD_2.usda@
        ) {

        }
        "Fighter_Variant_3" (
            payload = @./HD/Fighter_Variant_HD_3.usda@
        ) {

        }
        "Fighter_Variant_4" (
            payload = @./HD/Fighter_Variant_HD_4.usda@
        ) {

        }
    }
}

class "_class_Fighter_Scene"
{
}
```

5. Instancing the project

I decided to setup the instancing within Katana.

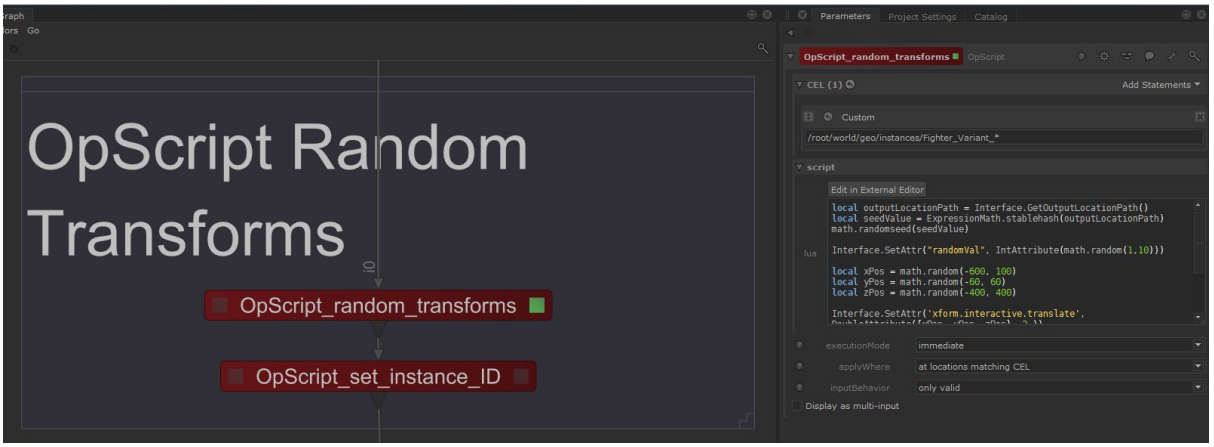
To do this, I used a combination of HierarchyCopy nodes and Pixar's PxrUsdInVariantSelect nodes to setup the amount of instanced objects and the variant of the USD models.

These were set up in a Group node, with the copy parameters exposed for ease of use:



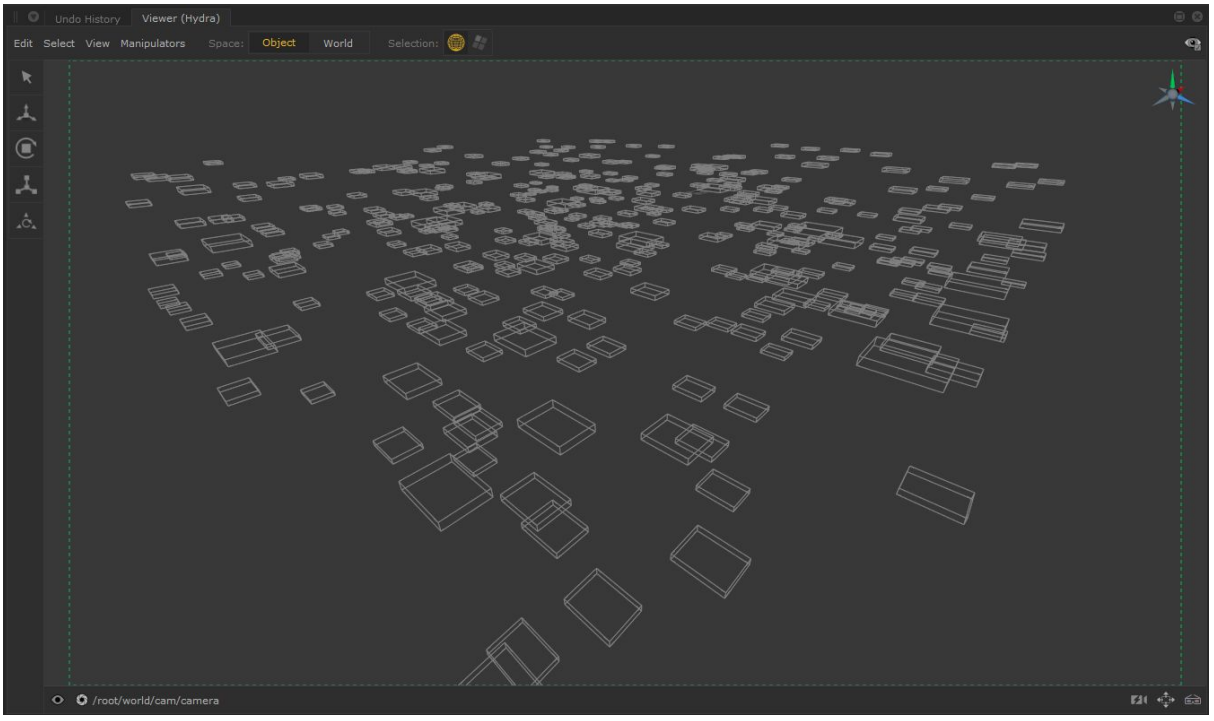
6. Randomizing the model location

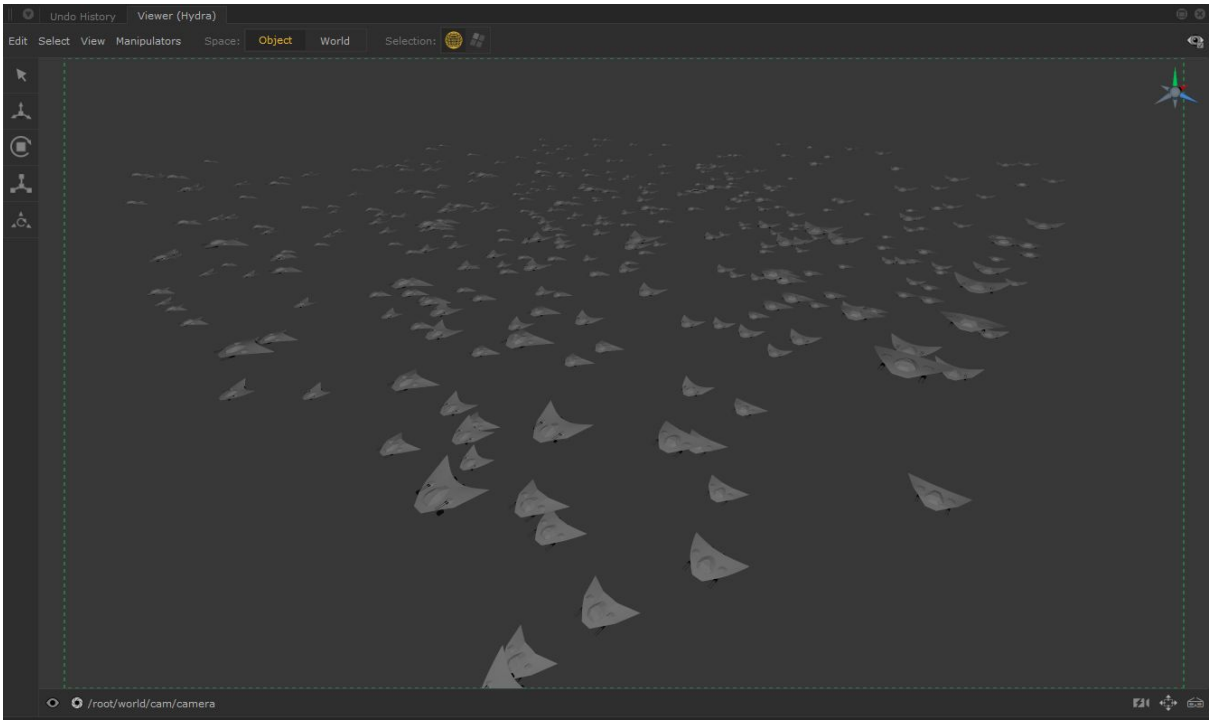
To randomize the fighters, I set up an OpScript, which calls a random value for the xyz positions of the instanced object.



The simple script worked perfectly, each instance is randomly moved and do not seem to overlap.

Instanced Transformed scene:





7. Render Tests / Material Assignment.

I decided to use 3Delight as my chosen renderer for my tests.

To render the scene, I setup a very basic GafferThree with a Sky Light, with a Space HDRI image.
This generally worked for the simple tests I was creating.

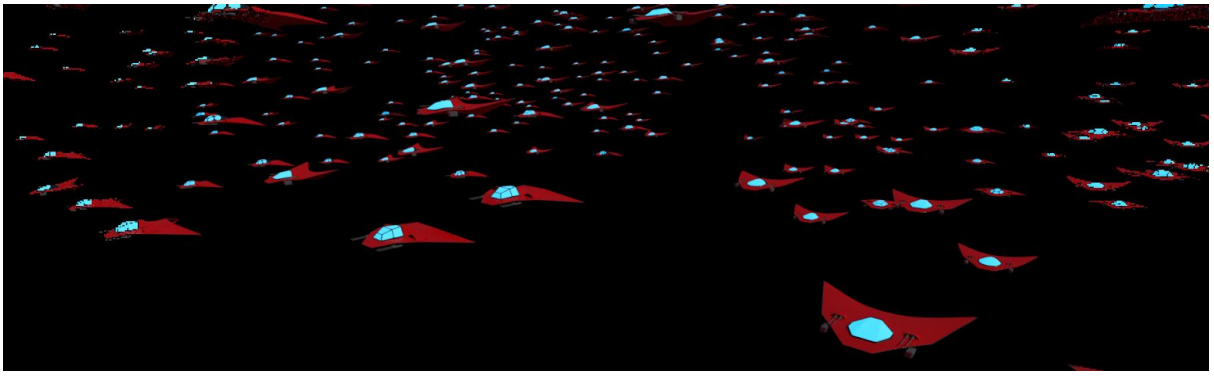
As the whole project is instanced, I needed to setup my material assigns through reference expressions.

While setting this up, I found that I need to update my geometry to include face sets and alternate naming conventions.

I went back to Maya, setup facesets and additionally setup subdivisions for a higher definition model.

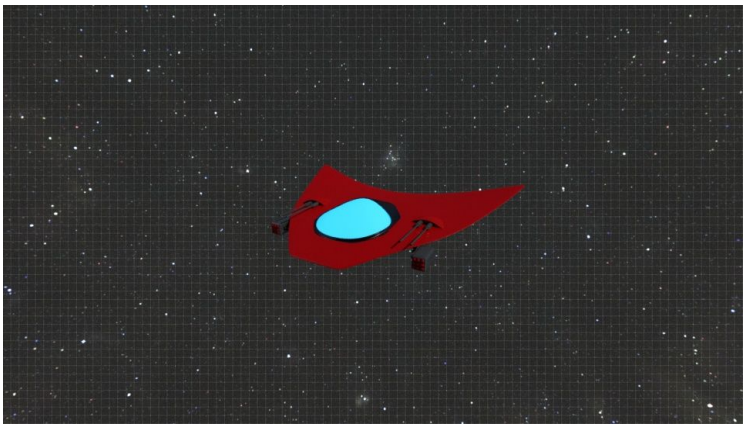
The new models were added as a new USD layer.

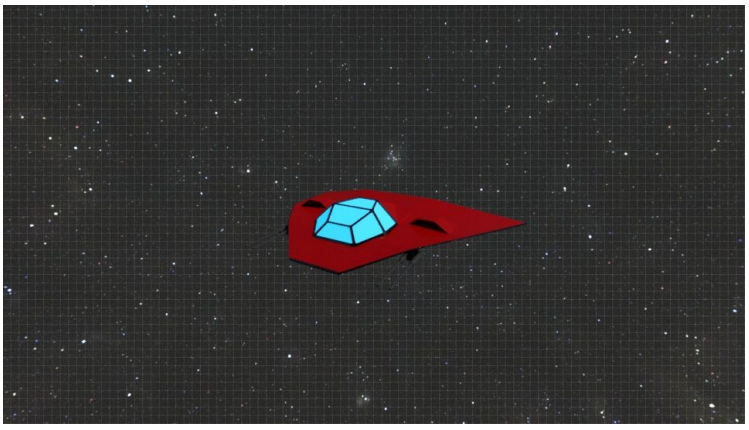
The new USD layer was brought into Katana, the scene rendered nicely:

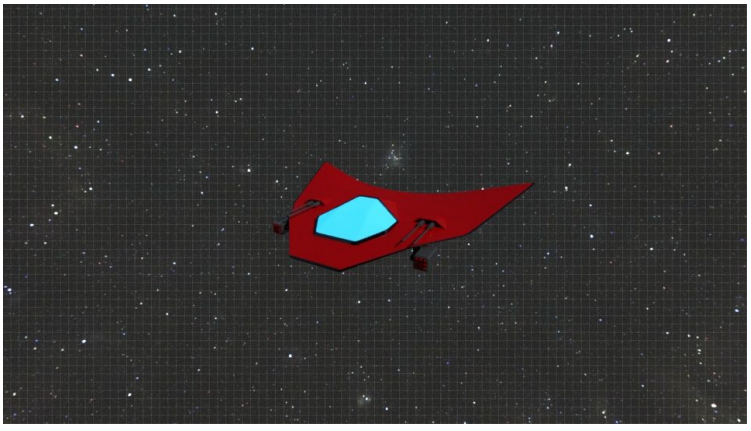


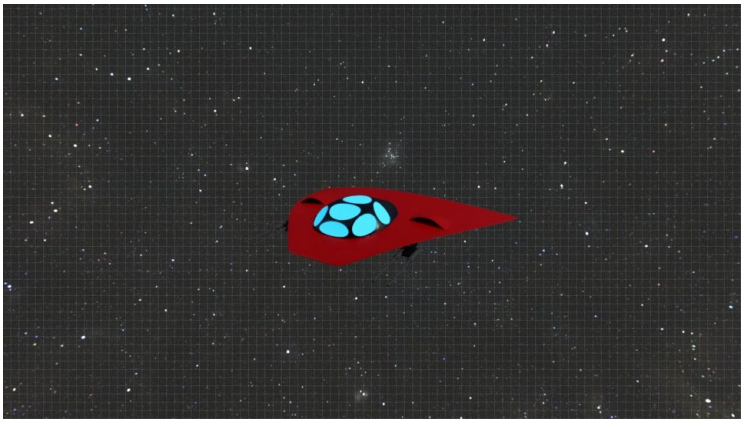
I also had a bit of fun with the Paint material, giving each object a different color through a randomizer:

Model Variants:









8. Layered animation process.

To display a way multiple departments could work on the scene, I decided to create the animations within a layering system.

For this task, I created the animations completely with a Python Script (*project/Assets/Fighters/USD_LAYER_SETUP_ANIMATION.py*)

I split the XYZ transforms into their own separate USD files, before combining them into a singular USD file.

This allowed me to see which values are set on a translation basis.

For reference, I set up a little animation within Maya, and manually added the transforms into the Python document.

After this is setup, the FighterAnim_Combined() function sets up the scene, and references it to the main geometry USD file.

The Geometry file can still be tweaked to an artists liking, the animation is using the main root.

I attempted to create an offset within katana with the PxrUsdInMediaOverride for each instanced ship to slightly change the animation, but so far have not found a way to set this up.

I will look further into this, to see if it is possible.

Conclusions, thoughts, future plans.

It has been a fun and challenging setting up USD and a testing project.

There were many challenges, I found the largest was getting around the installation of the program, and the difficulties I found myself running into.

Setting up USD initially took only an afternoon, but the Katana Plugin took multiple days, with casual chats between Pixar and even the Katana team on potential problems. (This was in the weekday, so didn't have too much time to work on…)

There are still plenty of things I would like to add, I would like to make the animations within Katana offset for each instance created… This would make the ships look like they are animating independently like a Crowd.

Additionally, an effects layer with booster effects and lasers/missiles firing is on the list of things to do.

I have learned quite a lot in the week on USD, and look forward to future projects.