OpenUSD Notes

1. What is OpenUSD?

OpenUSD (Universal Scene Description) is an open-source framework and file format for describing, composing, and collaborating on complex 3D scenes. **NVIDIA Omniverse** uses OpenUSD.

- Developed by Pixar, now an **industry standard** for VFX, animation, robotics, architecture, and more.
- Not just a file format: it organizes, layers, and exchanges 3D data between tools and teams, it is an API at its core.
- Enables **non-destructive workflows**, **collaboration**, and **extensibility**.

2. Why OpenUSD?

- **Interoperability:** Exchange 3D data between software (Maya, Blender, Omniverse, etc.).
- **Non-destructive workflows:** Layered editing prevents overwriting others' work.
- **Scalability:** Handles massive, complex scenes (entire movie sets, industrial simulations).
- **Extensibility:** Custom schemas, plugins, data types.
- **Collaboration:** Real-time, multi-user workflows in platforms like NVIDIA Omniverse.

3. Key Terms & Concepts

Stage

- Top-level container for a USD scene.
- Holds a **hierarchical scenegraph** of prims, relationships, and attributes.
- Analogy: "The world or project file".

Prim (Primitive)

- The **building block** of a USD scene (nodes/objects).
- Can be geometry, light, camera, group, etc.
- Organized in a **hierarchy** (parent/child).

Types of Prims:

Type	Purpose
Scope	Grouping/organization only (like folders). No transforms.
Xform	Stores transformations (translate, rotate, scale) applied to children.
Geom	Geometry primitives (meshes, curves, points).
Light	Lighting primitives (UsdLux).
Camera	Camera primitives (UsdGeomCamera).

Attributes

- Typed properties of prims: numbers, vectors, strings, etc.
- Can be static or animated.
- Example: position, color, visibility.

Relationships

- Links (paths) to other prims.
- Used to connect objects (assign material, link lights).
- Robust against hierarchy changes.

Primvars

- Short for **primitive variables**.
- Special attributes for **per-vertex, per-face, or per-object data**.
- Examples: UVs, vertex colors, custom metadata.
- Used for interpolation across geometry in shaders.

4. Layers & Composition

- **Layer:** A USD file (.usda, .usdc, .usdz) containing part/all of a scene.
- **Layering:** Stack multiple layers to create a composite scene.
- **Composition arcs:**
    - **Reference:** Insert another USD file as a sub-tree.
    - **SubLayer:** Stack layers for non-destructive edits.
    - **Variant:** Switch between versions/configurations.
    - **Payload:** Lazy-load heavy assets to save memory/performance.

5. Hydra Rendering Architecture

- Decouples scene data from renderers, we can pick what renderer we want (OpenGL/Vulkan).
- Supports multiple rendering backends.
- Used in **NVIDIA Omniverse** for **real-time RTX rendering**.

6. File Formats

Format	Description
.usda	ASCII, human-readable.
.usdc	Binary, efficient for large scenes.
.usd	Generic container (can reference other layers).
.usdz	Package (zip) for AR/VR, contains all assets.

7. Common USD Python Modules

Module	Purpose
Usd	Core API: stages, prims, relationships, composition.

Module	Purpose
Sdf	Scene description foundation: layers, paths, serialization.
Gf	Graphics foundation: math (vectors, matrices, geometry helpers).
UsdGeom	Geometry schemas (mesh, curves, camera, points).
UsdLux	Lighting schemas (DistantLight, RectLight, DomeLight).
UsdShade	Materials and shading schemas.
UsdPhysics	Physics scene description schemas.

8. Typical Python Snippets

```python
from pxr import Usd, UsdGeom, Sdf, Gf, UsdLux

# Create a new stage
stage = Usd.Stage.CreateNew('scene.usda')

# Define a transform prim (Xform)
xform = UsdGeom.Xform.Define(stage, '/World/MyXform')
xform.AddTranslateOp().Set(Gf.Vec3f(1.0, 2.0, 3.0))

# Create a mesh
mesh = UsdGeom.Mesh.Define(stage, '/World/MyMesh')

# Add a relationship (assign material)
rel = mesh.CreateRelationship('material')
rel.AddTarget(Sdf.Path('/World/Materials/Red'))

# Create a primvar (per-vertex color)
primvar_api = UsdGeom.PrimvarsAPI(mesh)
primvar_api.CreatePrimvar('displayColor', Sdf.ValueTypeNames.Color3fArray)
primvar = primvar_api.GetPrimvar('displayColor')
primvar.Set([Gf.Vec3f(1.0, 0.0, 0.0)])  # red color
```

9. Stage Traversal

```
# Traverse all prims
for prim in stage.Traverse():
    print(prim.GetName())

# With predicate filter
from pxr import Usd
predicate = Usd.PrimIsActive & Usd.PrimIsLoaded
for prim in stage.Traverse(predicate=predicate):
    print(prim.GetPath())
```

10. Lighting (UsdLux)

- Directional Light: UsdLux.DistantLight
- Area Lights: UsdLux.RectLight, UsdLux.DiskLight, UsdLux.SphereLight
- Dome Light: Environment lighting for global illumination

11. Transformations (Xform, XformCommonAPI)

- Xform prims store transformations: translate, rotate, scale
- XformCommonAPI provides standard methods for querying and authoring transforms
- Hierarchical: Parent transforms affect all children automatically

12. OpenUSD in Omniverse & Isaac Sim

- **Omniverse:** Real-time, collaborative 3D workflows, RTX rendering, modular asset pipelines
- **Isaac Sim:** Robotics simulation platform, uses OpenUSD for scene composition and synthetic data generation
- **Benefits:** Modular pipelines, real-time updates, scalable simulation, AI/ML integration

13. Quick Reference Table

Term	Description
Stage	Top-level container (scenegraph root)
Prim	Node/object (geometry, light, group, etc.)
Scope	Grouping prim (organization only)
Xform	Prim storing transformations
Attribute	Typed property of a prim
Relationship	Link to other prims
Primvar	Per-vertex/face/object data
Layer	USD file containing part/all of a scene
Reference	Composition arc to insert another USD file
SubLayer	Stack layers for non-destructive edits
Variant	Switch between versions/configurations
Payload	Lazy-load heavy assets
Hydra	Rendering architecture
Usd, Sdf, Gf	Core Python modules
UsdGeom	Geometry schemas
UsdLux	Lighting schemas
UsdShade	Material/Shading schemas
UsdPhysics	Physics schemas

Revision Questions

- What is OpenUSD and why is it important?
- Difference between prim, attribute, relationship
- How layering and composition works
- Role of Hydra in USD
- How OpenUSD supports non-destructive editing
- How Omniverse uses OpenUSD
- What are primvars and why are they useful
- How to traverse a USD stage in Python
- Difference between scope and xform prim

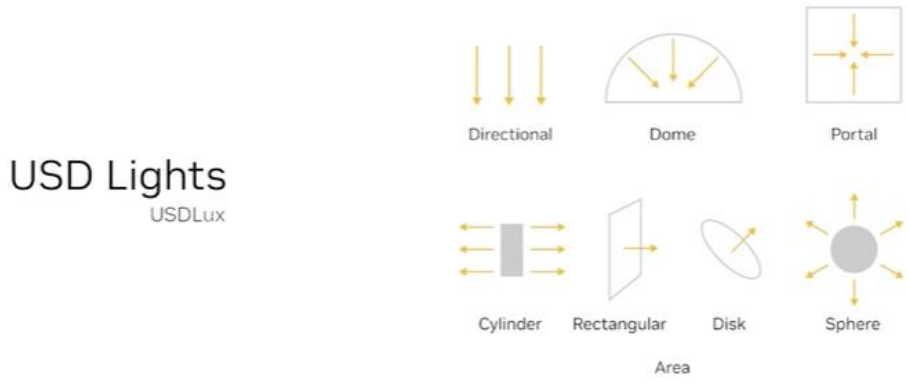
```
# Generic USD API command. Used to define a new prim on a stage at a specified path, and optionally the type
of prim.
stage.DefinePrim(path, prim_type)

# Specific to UsdGeom schema. Used to define a new prim on a USD stage at a specified path of type Xform.
UsdGeom.Xform.Define(stage, path)

# Retrieves the children of a prim. Useful for navigating through the scenegraph.
prim.GetChildren()

# Returns the type of the prim, helping identify what kind of data the prim contains.
prim.GetTypeName()

# Returns all properties of the prim.
prim.GetProperties()
```



```
# Used to define a new scope at a specified path on a given stage
UsdGeom.Scope.Define(stage, path)

# This command is generic, but it's useful to confirm that a prim's type is a scope, ensuring correct usage in
scripts
prim.IsA(UsdGeom.Scope)
```

```
# Used to define a new Xform prim at a specified path on a given stage
UsdGeom.Xform.Define(stage, path)

# Retrieves the order of transformation operations, which is crucial for understanding how multiple
transformations are combined. Different orders can yield different results, so understanding XformOpOrder is
important.
xform.GetXformOpOrderAttr()

# Adds a new transform operation to the xform prim, such as translation or rotation, with specified value
xform.AddXformOp(opType, value)
```

The schema modules are covered more in depth in relevant videos for each domain. Let's have a closer look at a few of the other common USD modules: `Usd`, `Sdf`, and `Gf`. In Python, you can import these modules from the `pxr` namespace:

```
    # Import Usd, Sdf, and Gf libraries from Pixar
    from pxr import Usd, Sdf, Gf
```

`Usd` is the core client-facing module for authoring, composing and reading USD. It provides an interface for creating or opening a Stage and generic interfaces for interacting with prims, properties, metadata, and composition arcs.

`Sdf` (scene description foundation) provides the foundations for serializing scene description to a reference text-based file format and implements scene description layers, (`SdfLayer`) which stores part of the scene description. Most notably, you will commonly see this module used for managing prim and property paths and creating USD layers.

`Gf` is the graphics foundation and contains the foundation classes and functions that contribute graphics, like Linear Algebra, Basic Mathematical Operations and Basic Geometry. This module contains classes for 3D data types that you will use for getting and setting particular USD attributes.

```
  # Import the UsdLux module
  from pxr import UsdLux

  # Create a sphere light primitive
  UsdLux.SphereLight.Define(stage, '/path/to/light')

  # Set the intensity of a light primitive
  light_prim.GetIntensityAttr().Set(500)
```

```
# Get the radius value of sphere_prim that is of type UsdGeom.Sphere
sphere_prim.GetRadiusAttr().Get()

# Set the double-sided property of the prim
sphere_prim.GetDoubleSidedAttr().Set(True)
```