MaterialX: An Open Standard for Network-Based CG Object Looks

Presentations

Doug Smythe ILM	MaterialX: What's New in v1.36
Jonathan Stone Lucasfilm ADG	MaterialX in Production at Lucasfilm
	MaterialX Open Source (2018)
George ElKoura Pixar	USD and MaterialX
Davide Pesare Allegorithmic Labs	MaterialX and Substance
Örn Gunnarsson Autodesk	MaterialX @ Autodesk
Henrik Edström Autodesk	MaterialX in Architecture & Design
Discussion and Q & A	

MaterialX Overview

- Schema and File Format used to describe "complete CG object looks":
  - Shading network topology and connections
  - Complex materials, with inheritance and multiple rendering targets
  - Texture assignments
  - Geometric assignments
  - Illumination and shadowing assignments for intrinsic asset lights
- Specific defined behavior for "Standard Nodes"

Features of MaterialX

- Node and connection-based, rather than shader parameter list-based
- Strong data typing
- Fully color managed
- Compatible with (but does not require) other open standards
  - e.g. OpenColorIO, Alembic, USD, OpenEXR, OSL
- "Live, Not Baked": Setups remain editable after re-import
- Extensible: user-defined nodes, node parameters, data types

What's Happened in the Past Year

- Worked with Autodesk as they integrated MaterialX support into their products
- Worked with Pixar's USD team to align USDShade's data model with that of MaterialX
- Significant updates to the MaterialX Specification and library, now at v1.36:
  https://github.com/materialx/MaterialX
- Full details at materialx.org

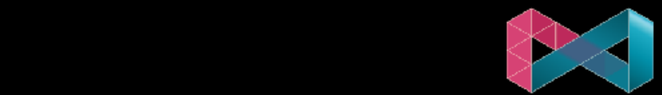
New and Improved in MaterialX 1.36

MaterialX Nodes

- Perform simple operations
  - e.g. add, mix, remap, compare, image read, Perlin noise, current P or N, etc.
- Supplement with application- or studio-specific custom nodes
- Used to create shading networks and define functionality of complex nodes

- Native-language implementations must be ported by hand
- Nodegraph implementations better for portability
  - Can be quite performant (ShaderX)

- Reorganization of node descriptions:
  - Basic Standard Nodes in main document
  - Supplemental Nodes (standard definition via nodegraphs) in new document
    - Native-language implementations allowed



New Nodes

- Trig functions: sin, cos, tan, asin, acos, atan2
- Math functions: sign, ceil, sqrt, ln, exp
- Matrix functions: invert, transpose, determinant
  - Basic math operators extended to 3x3 and 4x4 matrices
- Vector functions: rotate, transformpoint, transformvector, transformnormal
- Array functions: arrayappend
- Adjustment nodes: curveadjust, hsvadjust
- Color conversion: rgbtohsv, hsvtorgb
- Type conversion: convert, extract, separate
- Other new nodes: viewdirection, tiledimage

Element Inheritance

- Key feature of MaterialX: allows creating a base version of something, then inheriting from it as a starting point for specialized versions
  - "metal" material -> copper, bronze, steel, etc.
  - "basic" look of an asset -> wet, damage1/2/3, costume variations, etc.

- Before:

```
<material name="mymtl">
  <materialinherit name="parent_mtl"/>
```

- New in 1.36:

```
<material name="mymtl" inherit="parent_mtl">
```


Node Definition Inheritance

- Inheritance now works for node definitions too:

```
<nodedef name="ND_simplesrf_arnold" target="arnold"
         inherit="ND_simplesrf">
  <parameter name="nsamples" type="integer" value="5"/>
  ...
</nodedef>
```


Node Versioning

- Node capabilities frequently upgraded, not always backward-compatible
- Can now define a **version** attribute for a node definition, and request a specific node version when using a node
- Can declare a node definition to be the default

```
<skinsss_srf name="Nsrf1" type="surfaceshader"
             version="1.3">
 ..parameters..
</skinsss_srf>
```


Variants

- MaterialX v1.35 solution to material variations: MaterialVars
  - Individual values were set using <materialvar> in a Look

- In practice, usually a finite number of specific variations, choose one from a list

- New mechanism:
  - Variants to define all parameter values for a variant
  - Several <variant>s contained within a <variantset> to define groupings
  - Looks choose one specific variant from a variantset

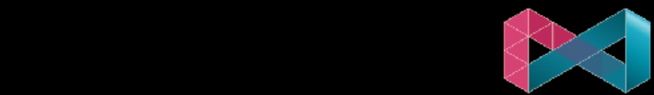
Tokens

- Problem: allowing general geometry attributes to be used in filename string substitutions can be problematic

- Tokens: New mechanism to pass string values to a node graph for substitution into image filenames

```
<parameter name="file" type="filename"
           value="txt/color/asset.color.<txtid>.tif"/>
```

- Simpler data model:
  - Enforces uniform-only string tokens for image filenames
  - General geometry attributes accessible only through <geomattrvalue>



Tokens, cont.

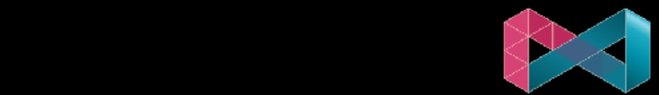
- Geometry-specific "geometry tokens"

```
<geominfo name="gi1" geom="/a/L*engine*">
 <token name="txtid" type="string" value="Lengine"/>
```

- Parameter-like "interface tokens"
  - Name and type defined in node definition, similar to <parameter> definition
  - Value passed from materials using <bindtoken>

```
<material name="Mplastic_wet">
 <shaderref name="sref5" node="simplesrf">
   <bindtoken name="diffmap" type="string"
              value="diffwet"/>
```

- See the "PostShaderComposite.mtlx" example in the MaterialX GitHub



Publicname/Override Removed

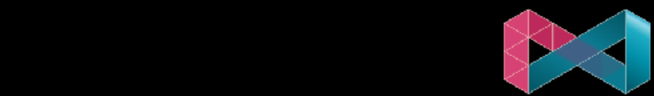
- Old way:
  - Define "publicname" for parameters in a nodegraph
  - "Override" those values from a material

- Wait… those are Global Variables! Global Variables = bad
  - Collisions between publicnames not avoidable
  - No formal interface = not discoverable

- Use existing formal <nodedef> node definition instead

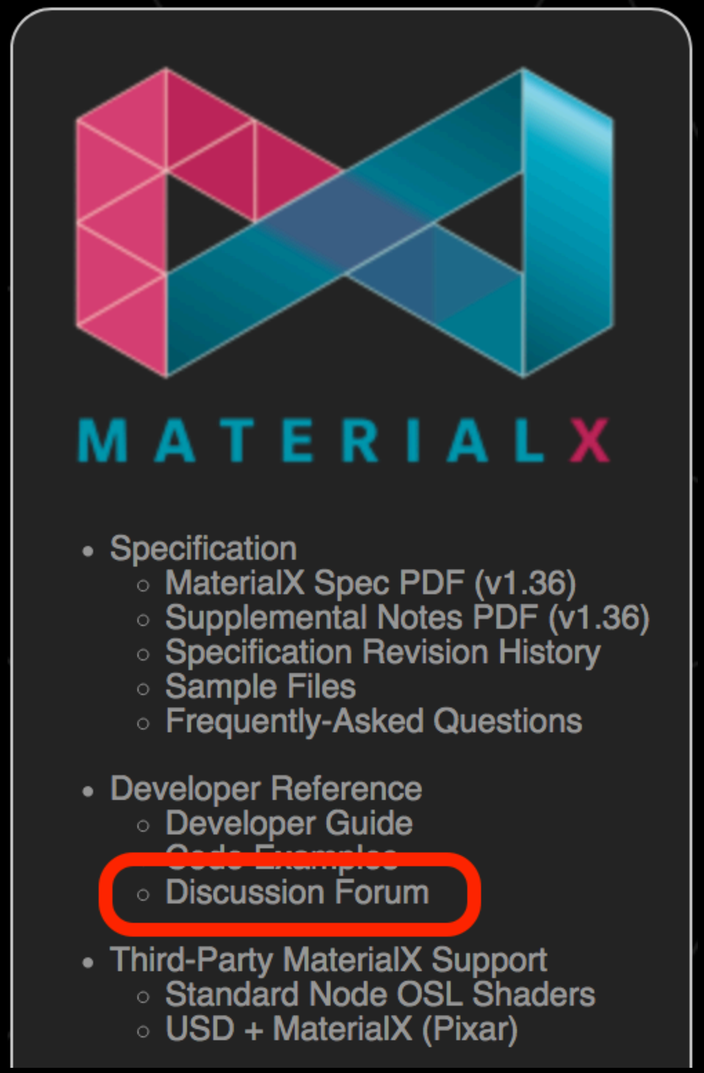
Other Changes

- Collection-definition syntax changed, compatible with USD
- New <attributedef> element to formally declare custom attribute names, types, default values and applicable target applications
- **`require`** declarations have been removed
- Now use XML character entities (e.g. **`&quot;`**) instead of backslashes for special characters in strings
- Removed the "swizzle on input" **`channel`** attribute for node elements
  - use explicit convert/swizzle/extract instead
- Lots more: see the "Changes since v1.35" doc on materialx.org



Future Directions

- More example code!
- More standard nodes
- Standard BxDF nodes, building on the work done for ShaderX
- Join the Discussion Forum and tell us what you'd like!



Thanks!

JONATHAN STONE - LUCASFILM ADG

MATERIALX IN PRODUCTION AT LUCASFILM (2018)

PREVIOUS WORK

- Early development as part of Lucasfilm's Unified Asset Initiative, with tests on *Transformers: Age of Extinction*.
- First mainstream production usage on *The Force Awakens*, representing material presets and deep material archives, and trans-media material transfer for *Trials on Tatooine*.
- Released as open specification in 2016, and open-source codebase in 2017.

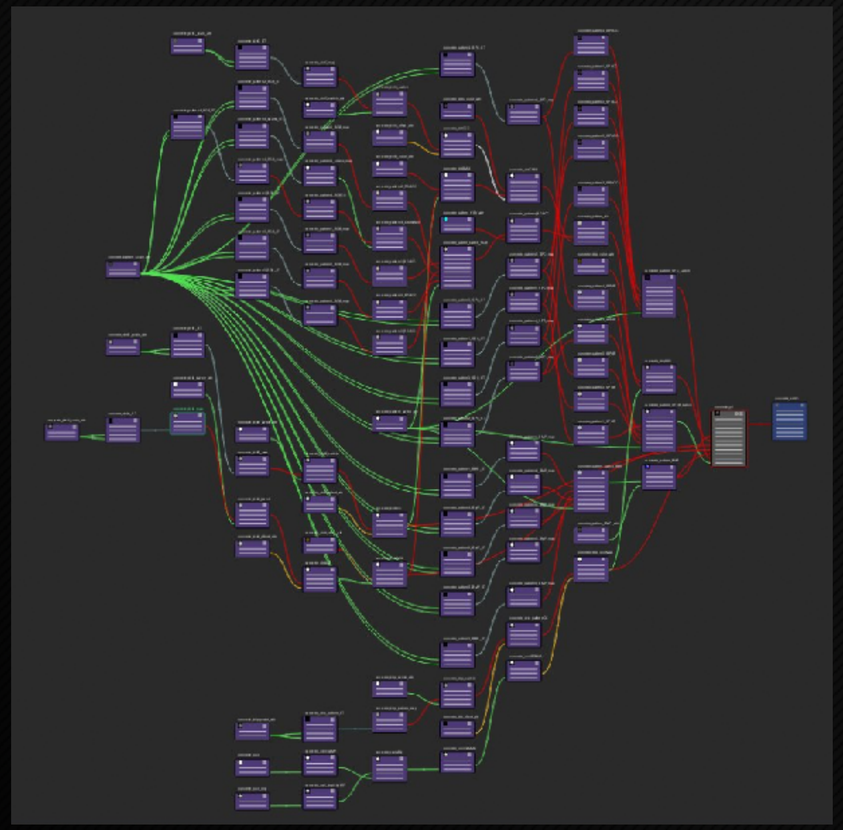
MATERIALX IN PRODUCTION AT INDUSTRIAL LIGHT & MAGIC

READY PLAYER ONE

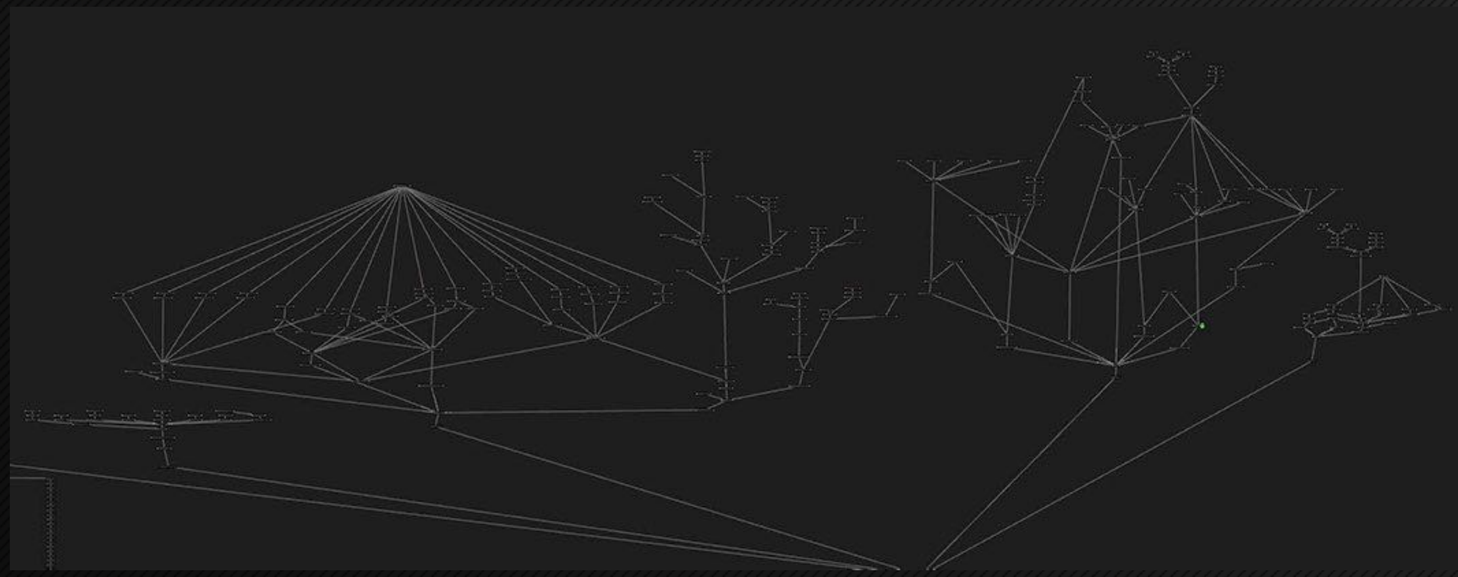
SHARING MATERIAL NETWORKS

- Early in development of Ready Player One, generalist team was interested in sharing environment material networks between Maya/Arnold and Katana/RIS.
- This would allow visual development in Maya to become a first pass of the final rendered look in Katana.
- ILM developer Ben Grimes built a MaterialX pipeline to transfer complete material networks between Maya and Katana.
- For the needs of this show, the pipeline was restricted to the Unified Shader and ILM node set, with plans for future generalization.

NYC ENVIRONMENT - GRAPH COMPARISON



MAYA / ARNOLD



KATANA / RIS

NYC ENVIRONMENT - RENDER COMPARISON

MAYA / ARNOLD

KATANA / RIS

ADDITIONAL DETAILS

- For ILM, this was the first mainstream production usage of deep network transfer using MaterialX.
- Artists found this technique useful for complex material networks such as the NYC buildings and Planet Doom terrain.
- For more details on this show, see the upcoming Production Session titled “Three Keys to Creating the World of 'Ready Player One’”.

MATERIALX IN PRODUCTION AT LUCASFILM

SECRETS OF THE EMPIRE

SECRETS OF THE EMPIRE

- Secrets of the Empire is a real-time immersive experience from ILMxLAB and the VOID.
- The experience is rendered in Kona, a custom build of Unreal supporting MaterialX and Lucasfilm's Unified Shader.
- Hero assets such as K-2SO, Stormtroopers, and the Astromech were authored with standard ILM workflows and transferred to real-time via MaterialX.

K-2SO MATERIAL ASSET

- K-2SO's material was transferred directly from the film asset on *Rogue One: A Star Wars Story*.
- For real-time efficiency, his geometry was simplified and reduced from 72 UDIMs to 5 UDIMs.
- Original material was baked out and transferred to Kona via MaterialX, including dual specular lobes and anisotropy.

K-2SO IN KONA AND RENDERMAN

KONA (2 MILLISECONDS PER FRAME)

RENDERMAN (22 MINUTES PER FRAME)

FUTURE PRODUCTION WORK

- The Secrets of the Empire approach to trans-media material transfer has been extended to a larger asset pipeline for the upcoming Millennium Falcon experience for *Star Wars: Galaxy's Edge* at Disney Parks.
- The Ready Player One approach to environment transfer is being extended to new tools and renderers for upcoming shows.
- Incorporating Autodesk's work on generic MaterialX import/export in Maya and Arnold as it becomes available (see Orn and Henrik's talks for details).

JONATHAN STONE - LUCASFILM ADG

MATERIALX OPEN SOURCE (2018)

CONTINUOUS INTEGRATION

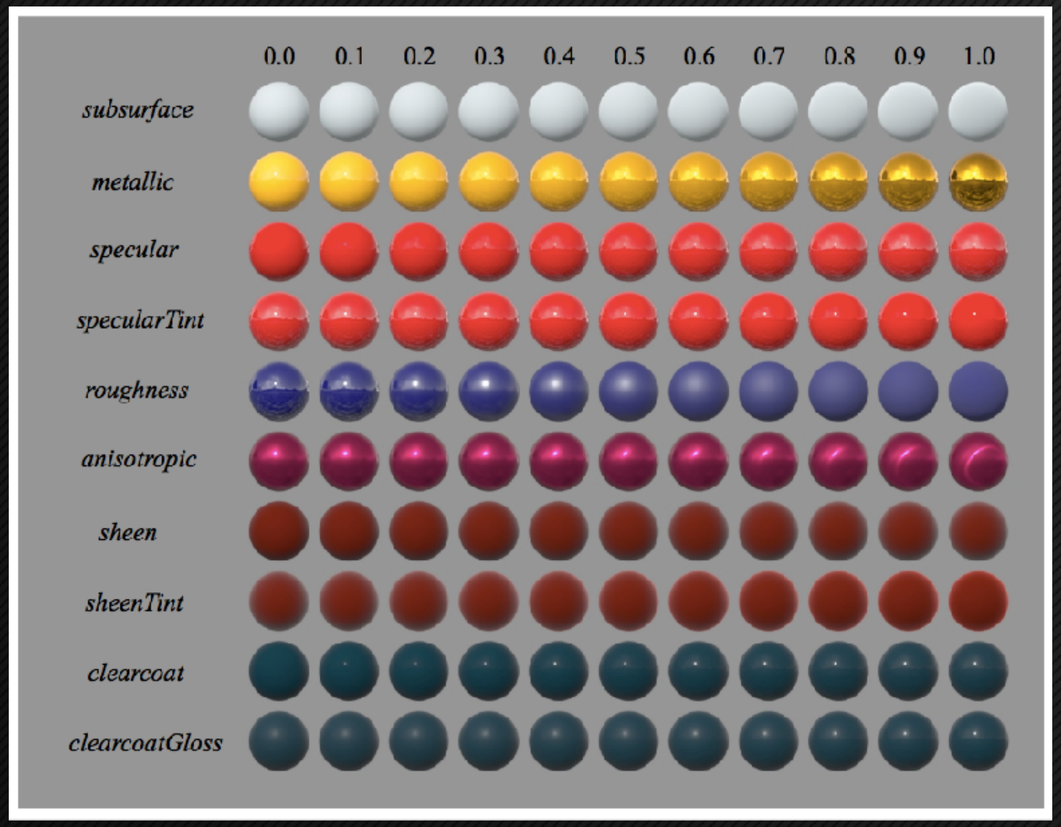
- Cross-platform builds and unit tests run for each pull request and commit.
- Leverages both Travis CI and Appveyor, covering Windows, Linux, and OSX.
- Extra tests for Visual Studio 2017, GCC 7, and Clang 5.
- A powerful tool in vetting changes from the community.

PYTHON 3

- Developers began requesting Python 3 support shortly after the open-source launch.
- Required for Blender, and scheduled to become part of the VFX Reference Platform.
- Python 3 is now supported and integrated into Appveyor unit tests.

SURFACE SHADER EXAMPLES

- The MaterialX repository now hosts example interfaces for the Disney BRDF/BSDF and alSurface.
- Thanks to Brent Burley and Anders Langlands for providing interfaces and online implementations.
- Autodesk's ShaderX project takes this idea further with modular BxDF components (see Orn's upcoming talk for details).



DISNEY BRDF

HIGH LEVEL MATERIAL API

- Requested by ILM developers, allows common operations to be performed with a single C++/Python call.
- Includes new methods such as getPrimaryShaderInputs and getGeometryBindings (see GitHub for details).
- Simplifies many common patterns, and all code examples now use the high-level API when possible.

FUTURE OPEN-SOURCE WORK

- Incorporating Autodesk's work on ShaderX into the main MaterialX repository.
- Developing a standalone material network viewer for MaterialX content.
- Continued collaboration with Pixar and the industry to improve MaterialX/USD parity and asset transfer.
- Ideas from the community!

THANKS!

- To Doug Smythe, Francois Chardavoine, Roger Cordes, and Rob Bredow
- Additional contributors to MaterialX:
  - Eoghan Cunneen, Maggie Oh, Naty Hoffman, Ben Grimes, Masuo Suzuki, Eric Bourque, Niklas Harrysson, Bernard Kwok, Henrik Edström, Örn Gunnarsson, Guido Quaroni, Sebastian Grassia, Chris Schoeneman, Davide Pesare, David Larsson, Brent Burley, Anders Langlands, and Larry Gritz.
- We're hiring!
  - Send resumes and demo reels to siggraph2018recruit@ilm.com.

SIGGRAPH 2018 MaterialX BOF:

USD and MaterialX

George ElKoura

USD + MATERIALX

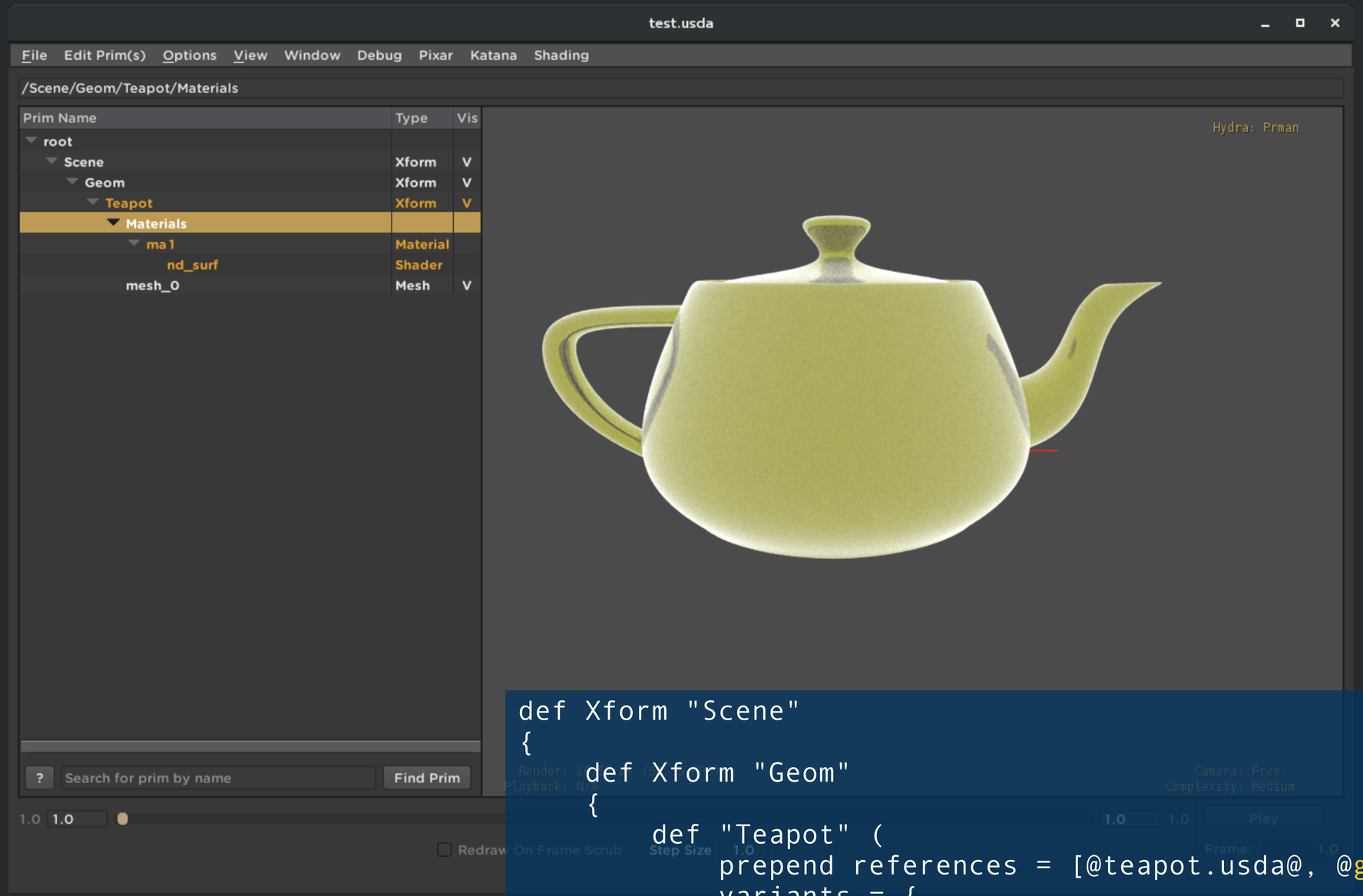
SIGGRAPH 2018 Update

USD Support

- Initial support for latest release MaterialX 1.36
- UsdMtlx plugin provides:
  - File Format plugin reads MaterialX files as UsdShade on the fly
  - Sdr plugin that registers Mtlx Stdlib nodes for consumption from UsdShade

What about UsdMatX?

- Schema functionality now replaced by Sdr registration
- Much more scalable and extensible




```
def Xform "Scene"
{
    def Xform "Geom"
    {
        def "Teapot" (
            prepend references = [@teapot.usda@, @gold.mtlx@</ModelRoot>]
            variants = {
                string LookVariant = "DustyGold"
            }
        )
    }
}
```

USD BOF Tomorrow

Tuesday, 14 August 2018
4pm - 6pm
East Building, Room 1

Thank you!

allegorithmic

MaterialX

And Substance

Davide Pesare – Head of Labs – Allegorithmic

allegorithmic

Davide Pesare

16 years in shading

I care about the topic of material exchanges

10 of those years at Pixar

Solved similar issues:

- RSL Code Generation
- SLIM metadata exported to muddle, Katana

With the advent of RIS, moved to OSL shaders.

We supported two types of portability:

- Metadata maya, katana, flow
- rendering on OSL and CUDA

Got involved with the design of UsdShade library

Worked closely with the ILM team on the convergence with MaterialX

Joined Allegorithmic

Substance Suite is a look development suite

Designer: create materials

Painter: asset lookdev

Alchemist: batch create and mix materials

allegorithmic

Substance Engine

At its core is a 2d compositing graph. Its outputs are textures. Often PBR textures, ready for consumption.

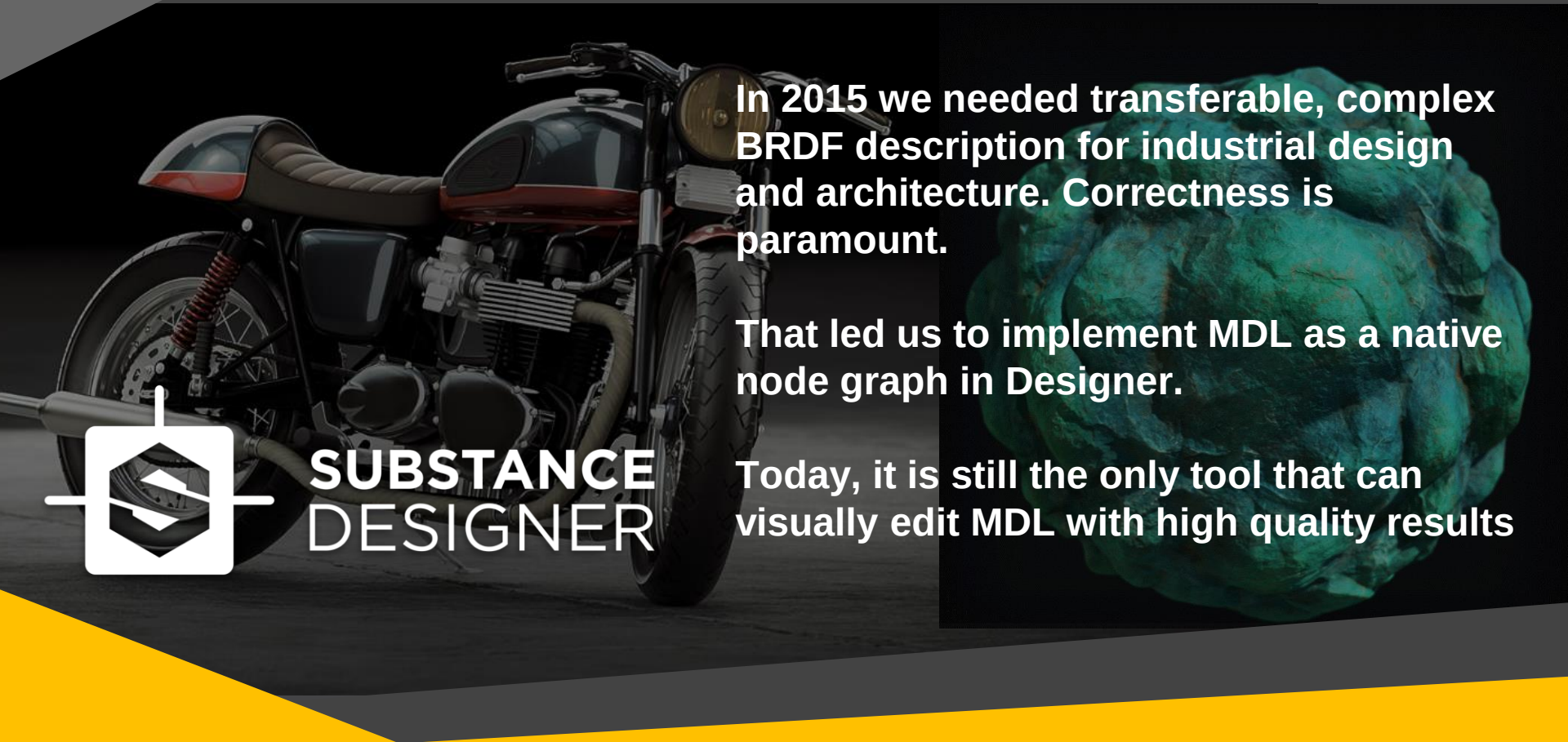
Rendering

Important to consider how are these textures used.

Initially, we focused on UberShaders

Often based on established standards, e.g. Disney 2012

allegorithmic



SUBSTANCE DESIGNER

In 2015 we needed transferable, complex BRDF description for industrial design and architecture. Correctness is paramount.

That led us to implement MDL as a native node graph in Designer.

Today, it is still the only tool that can visually edit MDL with high quality results

allegorithmic

MaterialX and us

We care about portable material descriptions. For the same reasons we are very interested in MaterialX

We are very excited in the direction it is evolving

We are collaborating with Autodesk and ILM

Allegorithmic Labs

In Labs, we build prototypes, that will inform or become future products and integrations

Allegorithmic Labs

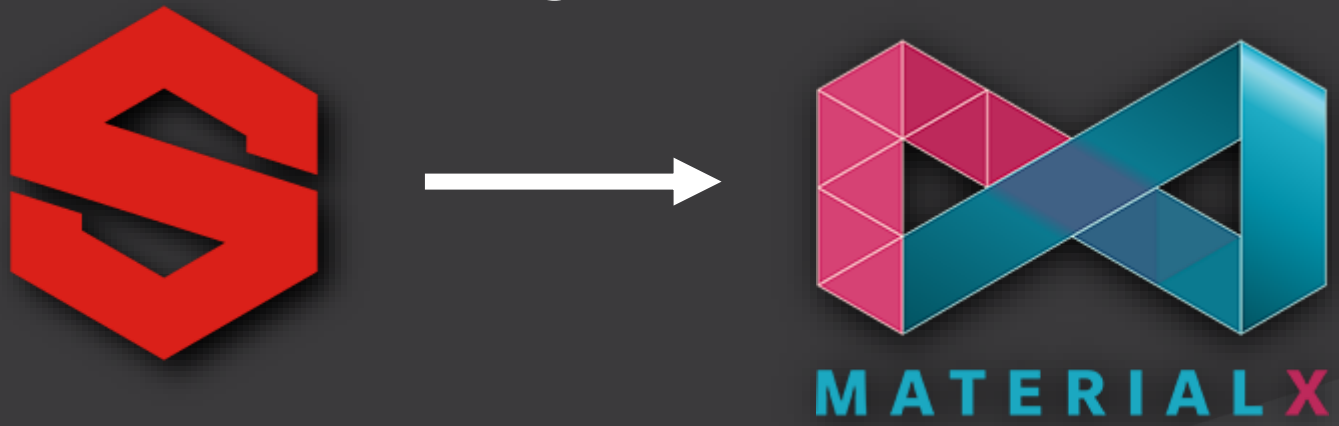
Few tests that are immediately useful

MaterialX prototypes

Import paint materials from MatX into shelf.
Materials can be selectively updated from matX even after painting.

MaterialX prototypes

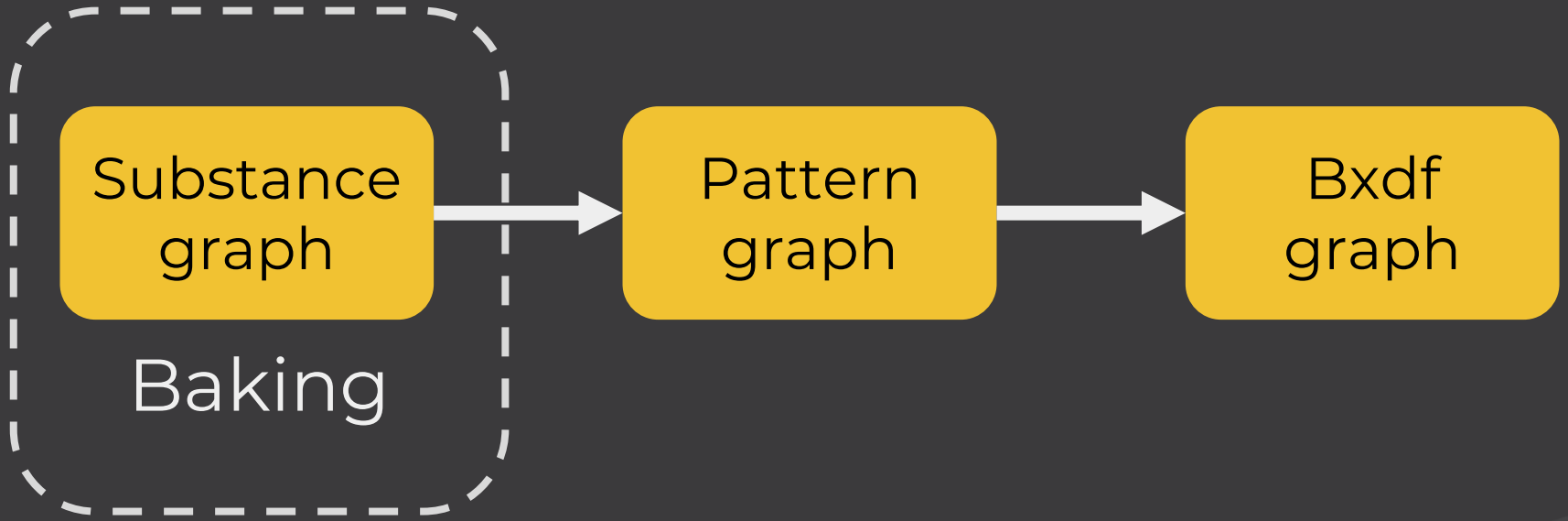
Painter: Export matX side car file.
Describe how textures are plugged.
Define Look bindings.



A lot more can be done...

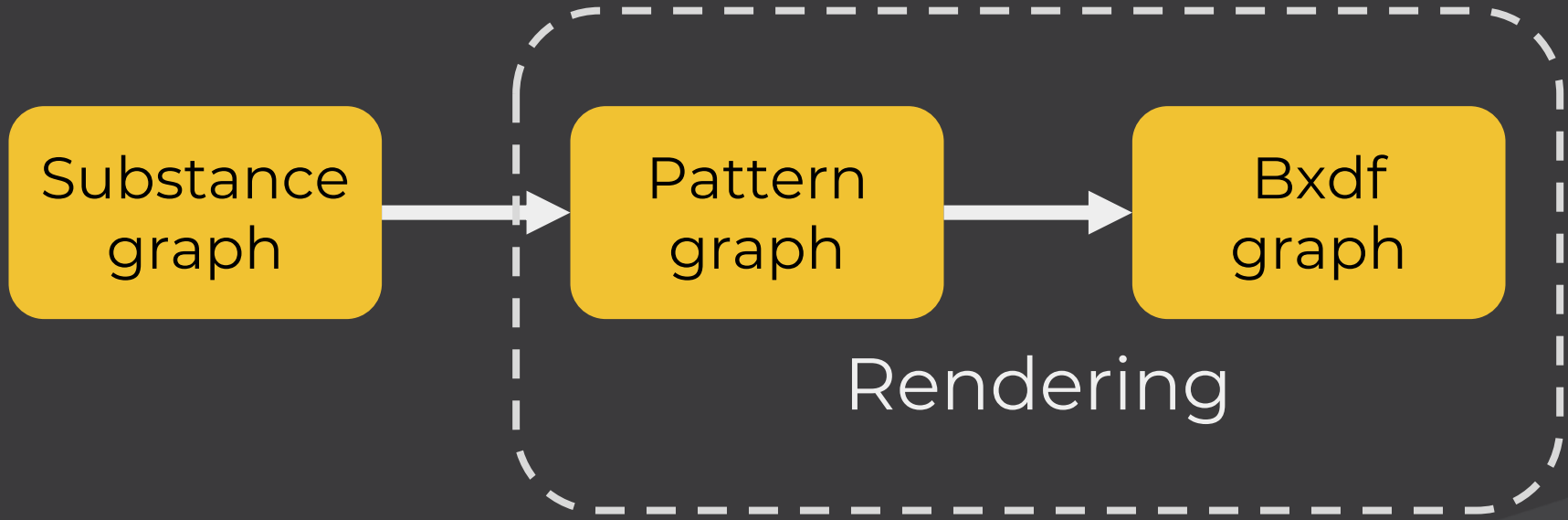
allegorithmic

Perspective



allegorithmic

Perspective



Future plans

We aim to explore in Labs how material fits with the live Rendering part of a material.

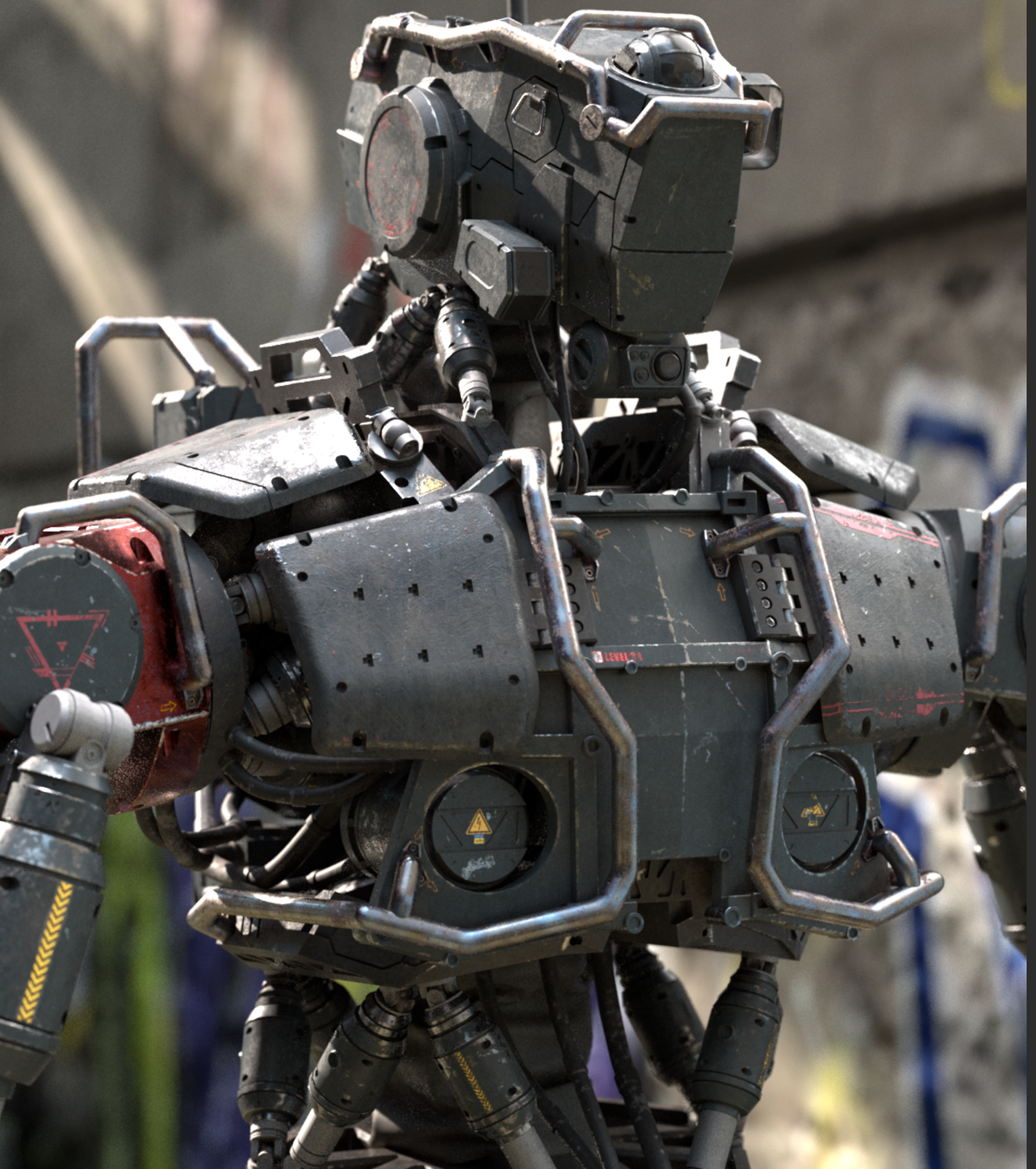
No product announcements today!

allegorithmic

Thanks!

MaterialX And Substance

Davide Pesare - Head of Labs - Allegorithmic



MATERIALX @ AUTODESK.

Örn Gunnarsson
Principal Engineer

Henrik Edström
Software Architect

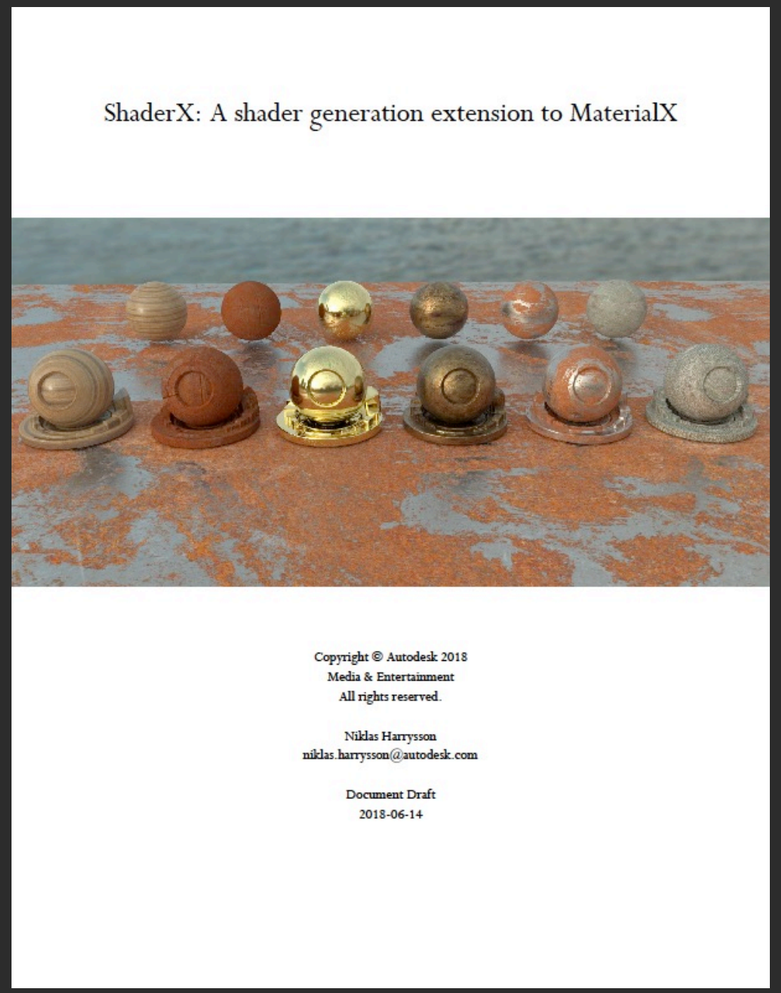
MaterialX @ Autodesk | Overview

- Media & Entertainment
- Architecture & Design

ShaderX | A MaterialX extension

Extending with:

1. An implementation layer
   - Helping applications to transform the agnostic MaterialX descriptions into executable shader code for a specific renderer
2. A node library for PBR shading
   - Data types, nodes and node graphs for layered physically-based shading


ShaderX: A shader generation extension to MaterialX

Copyright © Autodesk 2018
Media & Entertainment
All rights reserved.

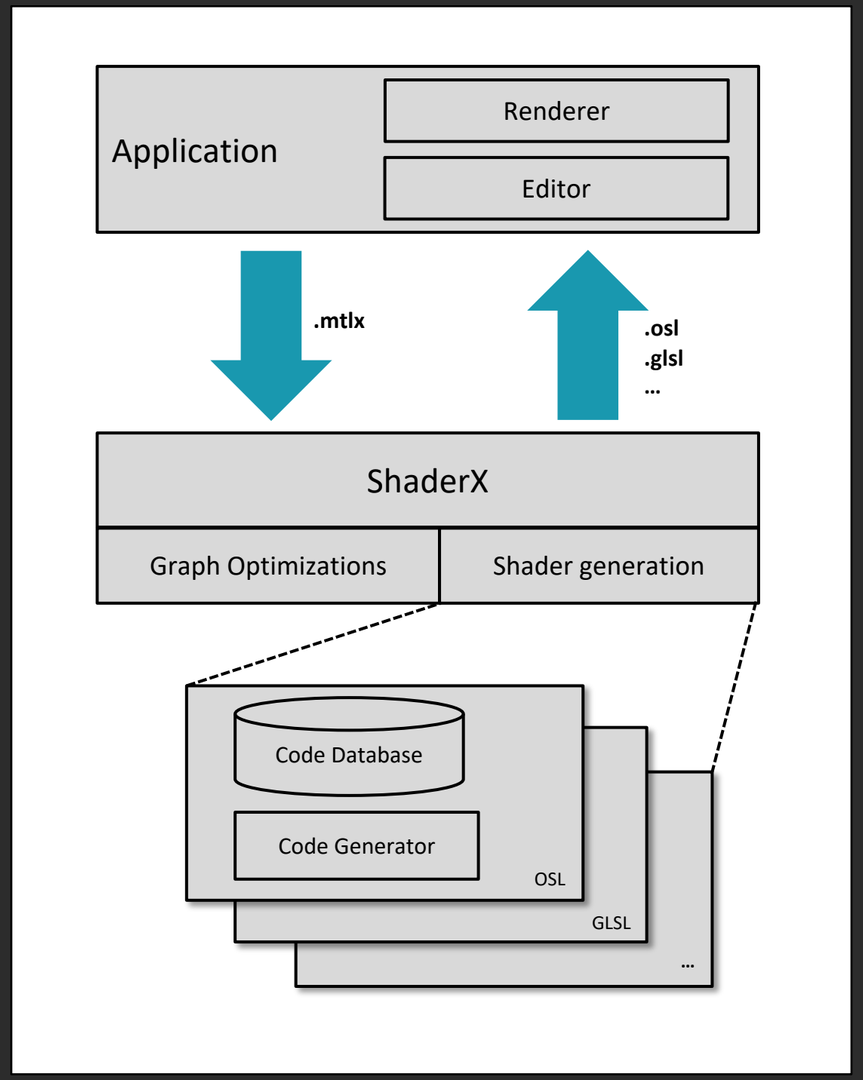
Niklas Harrysson
niklas.harrysson@autodesk.com

Document Draft
2018-06-14

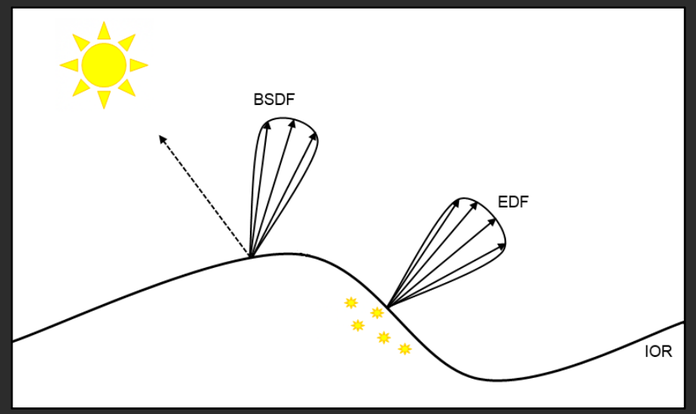
https://github.com/autodesk-forks/MaterialX/blob/adsk_contrib/shaderx/documents/Specification/ShaderX.Draft.pdf

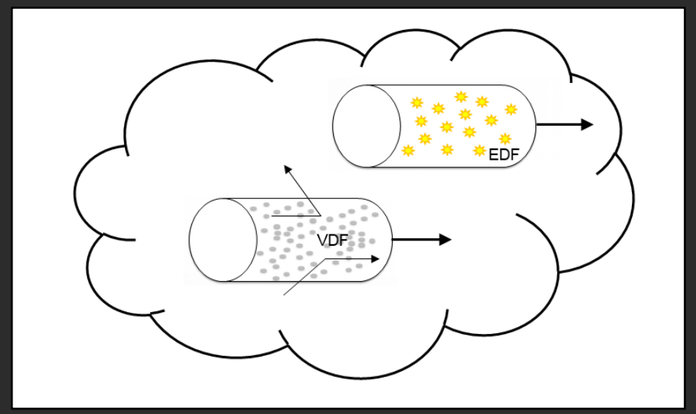
ShaderX | Implementation layer

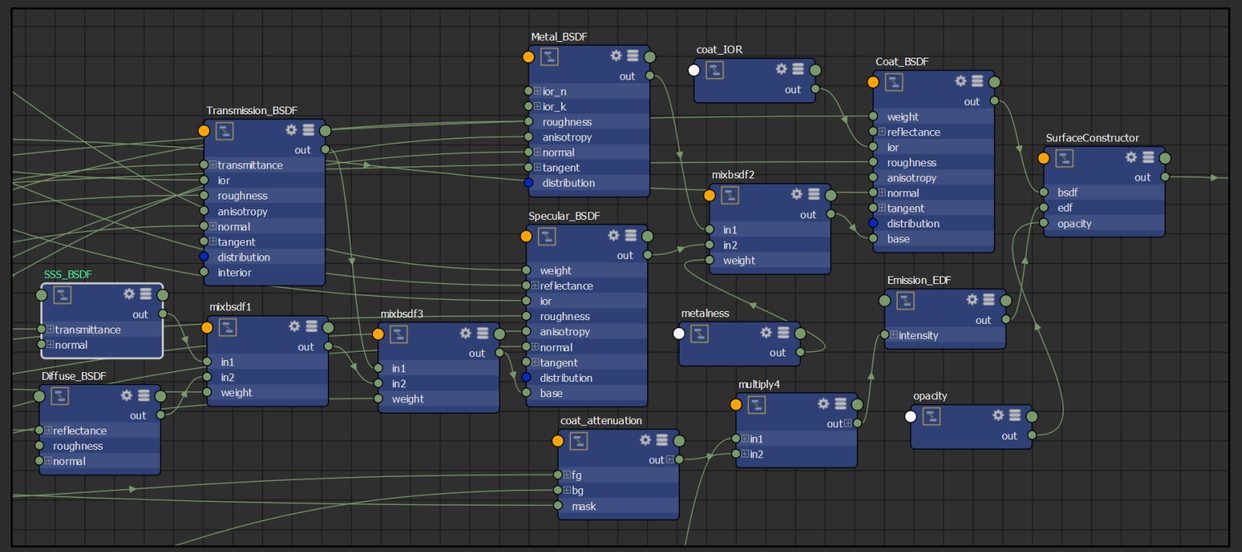
- A framework for shader generation
- Currently supported targets
  - OSL
  - GLSL
  - OGSFX (Autodesk viewports)
- Future targets
  - HLSL
  - GLSL ES (WebGL)
- Extensible to other targets and languages

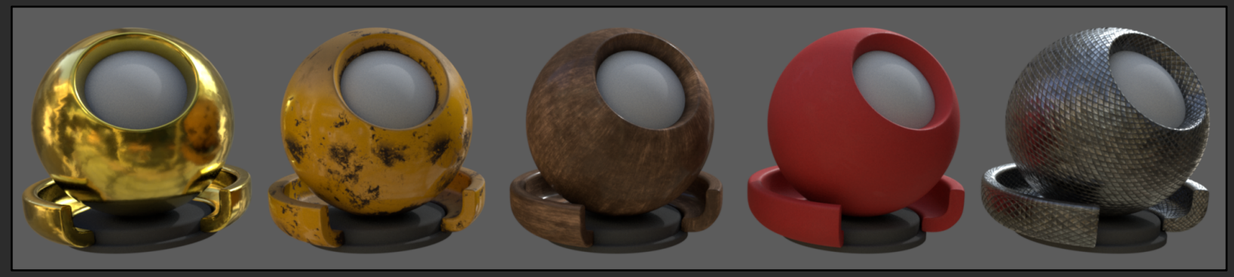


ShaderX | PBR Library



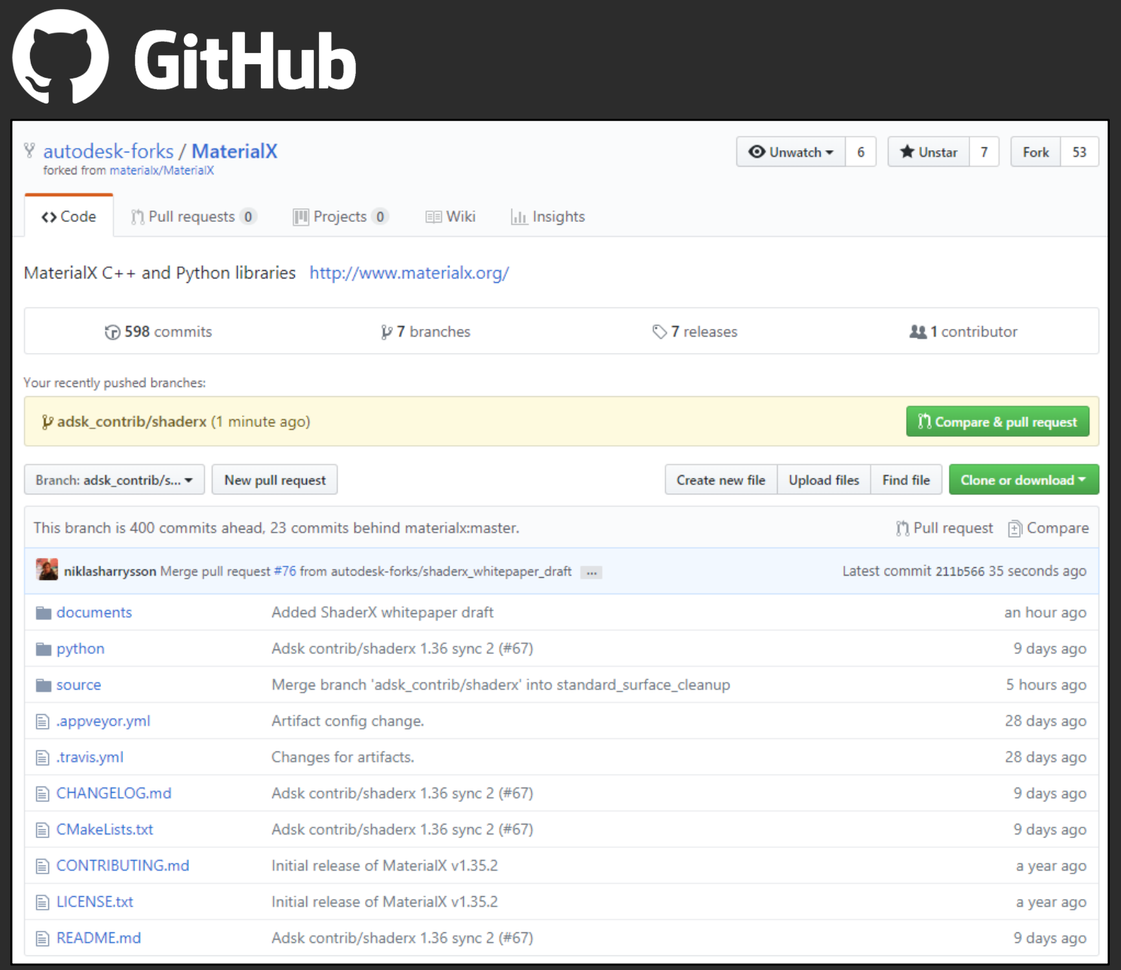






ShaderX | Open Source

- Publicly available now in our fork (pre-release)
- Work initiated to merge all the extensions to MaterialX master
- Goal is to have a first version fully included in MaterialX master by end of the year



https://github.com/autodesk-forks/MaterialX/tree/adsk_contrib/shaderx

ShaderX | Results

Materials exported from Substance Painter
Courtesy of Allegorithmic

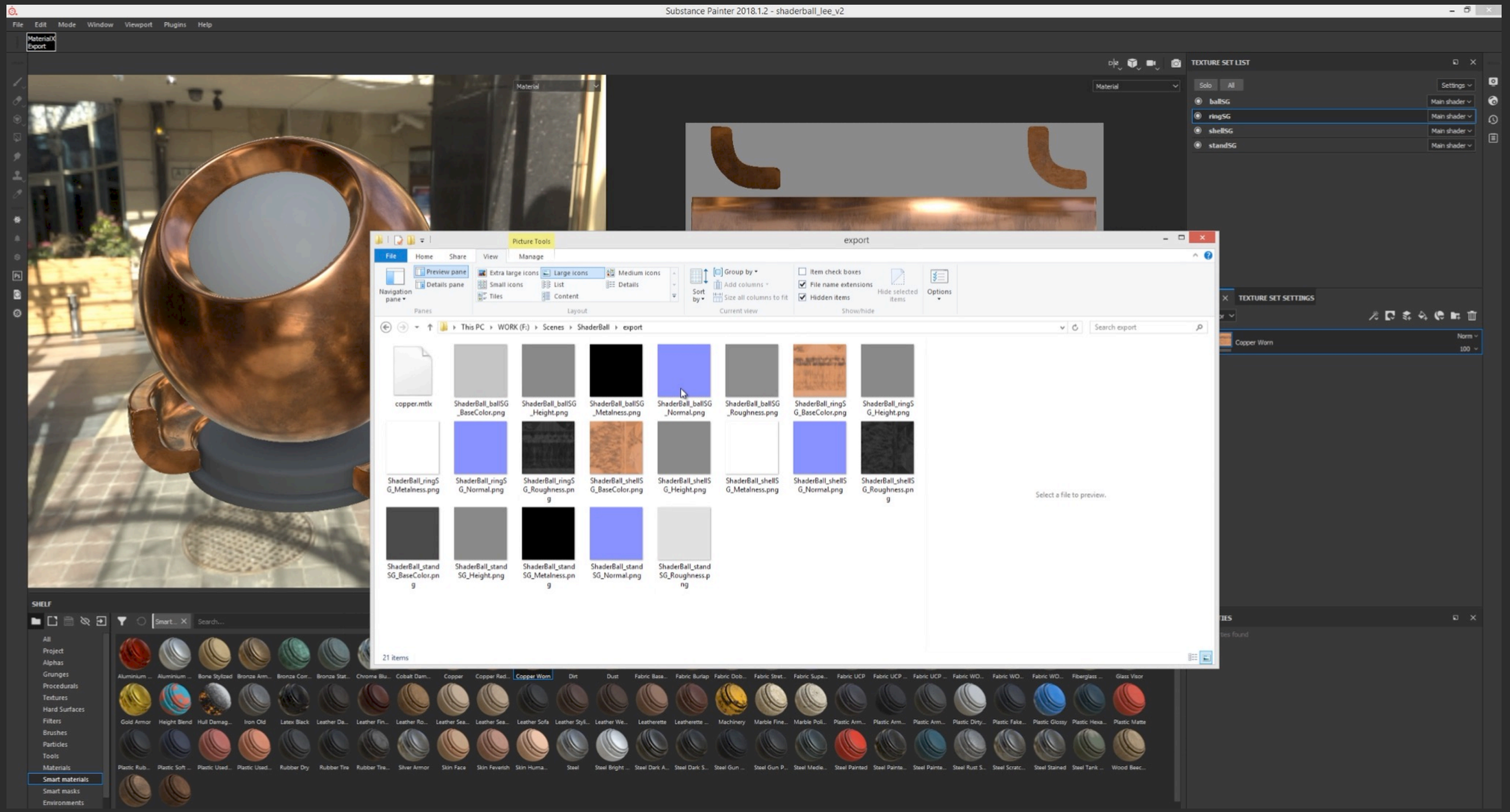
SUBSTANCE PAINTER

ShaderX - GLSL
in Maya's viewport

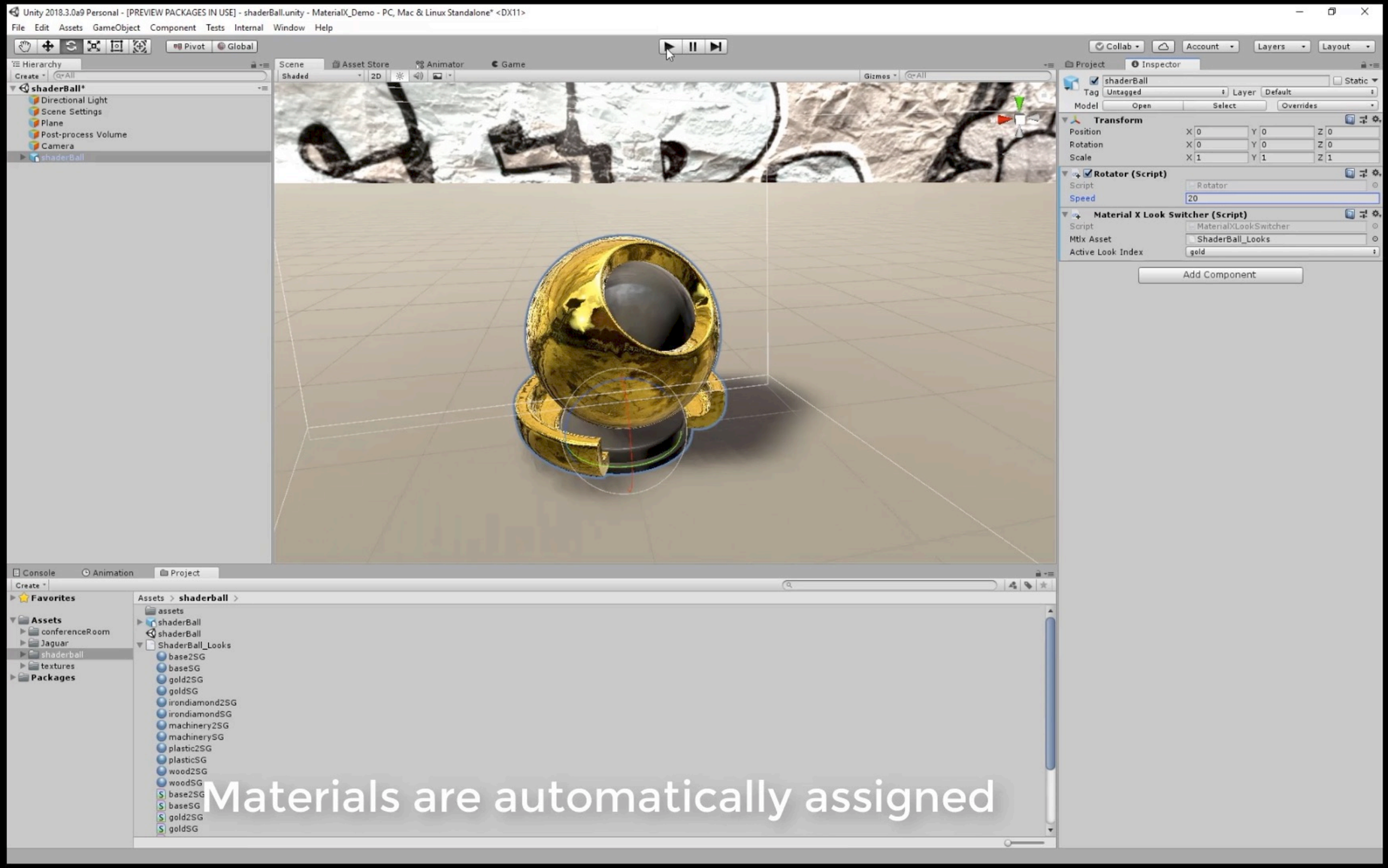
ShaderX - OSL
in Arnold

Reference C++
in Arnold

ShaderX | Substance to Maya/Arnold



ShaderX | Unity

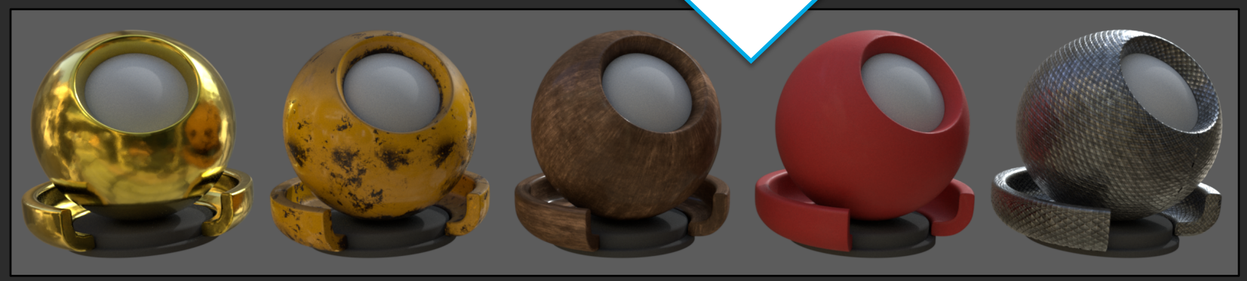


Arnold | ShaderX and MaterialX

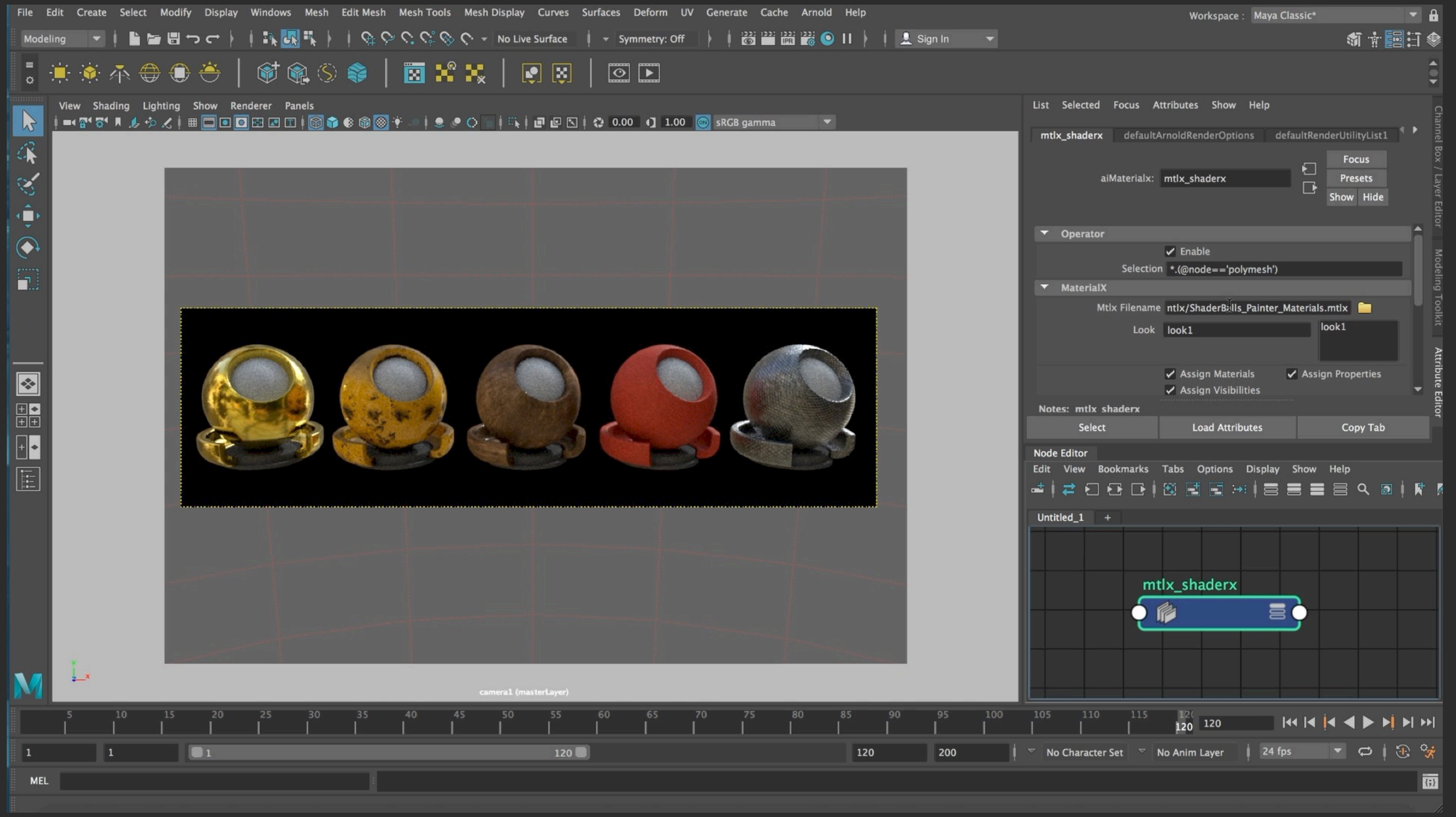
- Goal: First class MaterialX support
- MaterialX Arnold node
- ShaderX prototype
  - Generates OSL code
  - Assigns OSL shaders

MaterialX document

```
<materialx version="1.36">
  <look name="look1">
    <materialassign name="materialassign1" material="aiStandardSurface1SG" collection="collection1" />
    <materialassign name="materialassign2" material="aiStandardSurface3SG" collection="collection2" />
    <materialassign name="materialassign3" material="aiStandardSurface6SG" collection="collection3" />
    <materialassign name="materialassign4" material="aiStandardSurface2SG" collection="collection4" />
    <materialassign name="materialassign5" material="aiStandardSurface4SG" collection="collection5" />
    <materialassign name="materialassign6" material="aiStandardSurface5SG" collection="collection6" />
    <materialassign name="materialassign7" material="aiStandardSurface8SG" collection="collection7" />
    <materialassign name="materialassign8" material="aiStandardSurface10SG" collection="collection8" />
    <materialassign name="materialassign9" material="aiStandardSurface7SG" collection="collection9" />
    <materialassign name="materialassign10" material="aiStandardSurface9SG" collection="collection10" />
    <materialassign name="materialassign11" material="aiStandardSurface11SG" collection="collection11" />
    <materialassign name="materialassign12" material="aiStandardSurface12SG" collection="collection12" />
  </look>
  <material name="aiStandardSurface1SG">
    <shaderref name="base" node="standard_surface">
      <bindinput name="base" type="float" value="0.6" />
      <bindinput name="specular" type="float" value="0" />
      <bindinput name="normal" type="vector3" value="1, 1, 1" />
    </shaderref>
  </material>
  <collection name="collection1" includegeom="/scene/Shaderball3/BALL/ball/ballShape, /scene/Shaderball/Lee_Shaderball:BA
  Lee_Shaderball:ballShape, /scene/Shaderball0/BALL/ball/ballShape, /scene/Shaderball1/BALL/ball/ballShape, /scene/Shader
  />
  <material name="aiStandardSurface3SG">
    <shaderref name="ai_gold" node="standard_surface">
      <bindinput name="base" type="float" value="1" />
      <bindinput name="base_color" type="color3" output="file14_outColor" />
      <bindinput name="specular_roughness" type="float" output="file15_outColorR" />
      <bindinput name="metalness" type="float" value="1" />
      <bindinput name="normal" type="vector3" output="bump2d1_outNormal" />
    </shaderref>
  </material>
  <collection name="collection2" includegeom="/scene/Shaderball1/BALL/ATTACH/shell/shellShape" />
  <output name="file14_outColor" type="color3" nodename="file14_color3" />
  <image name="file14_color3" type="color3">
    <input name="texcoord" type="vector2" nodename="place2dTexture21_vector2" />
    <parameter name="uaddressmode" type="string" value="periodic" />
    <parameter name="vaddressmode" type="string" value="periodic" />
    <parameter name="file" type="filename" value="ShaderBall/gold/ShaderBall_shellSG_BaseColor.png" colorspace="sRGB" />
  </image>
  <texcoord name="place2dTexture21_vector2" type="vector2" />
  <output name="file15_outColorR" type="float"  e15_float" />
  <image name="file15_float" type="float">
```



Arnold | ShaderX – OSL in Arnold



Arnold | MaterialX node

- Design goal: Flexible workflows
- A single document for the whole scene
- A document with look variants for each asset or collection of assets

Arnold | MaterialX node

- In supported DCCs
- Parameters
  - Scene selection
  - Mtlx filename
  - Look variant
  - Search path
  - Assign materials/properties/visibility

Arnold | MaterialX example

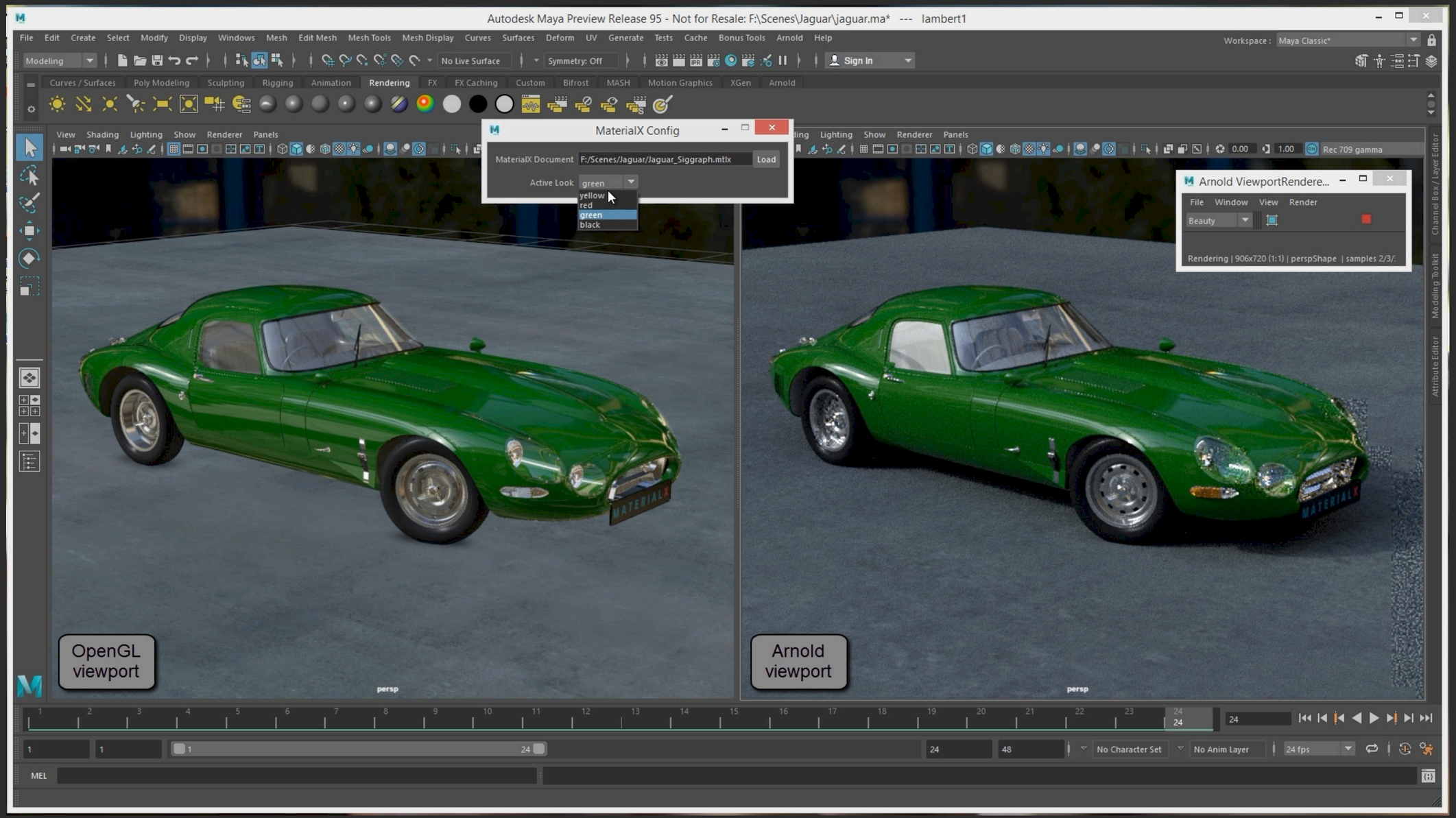
Arnold | MaterialX - In the box

- MaterialX Arnold node (1.36 spec)
- Up-to-date mtlx node definitions for Arnold shaders
- Support for look development with Arnold shaders

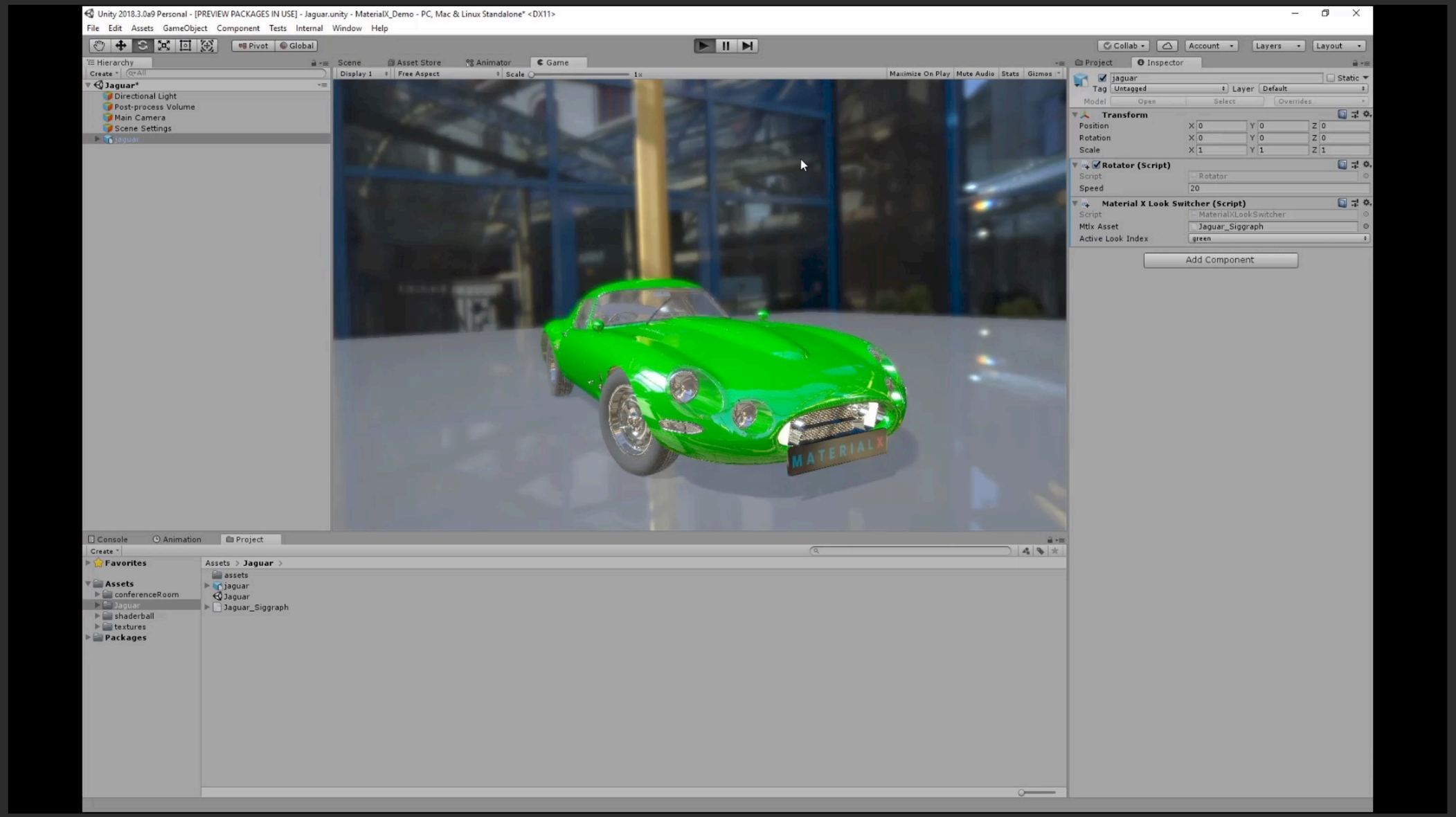
Arnold | Next steps

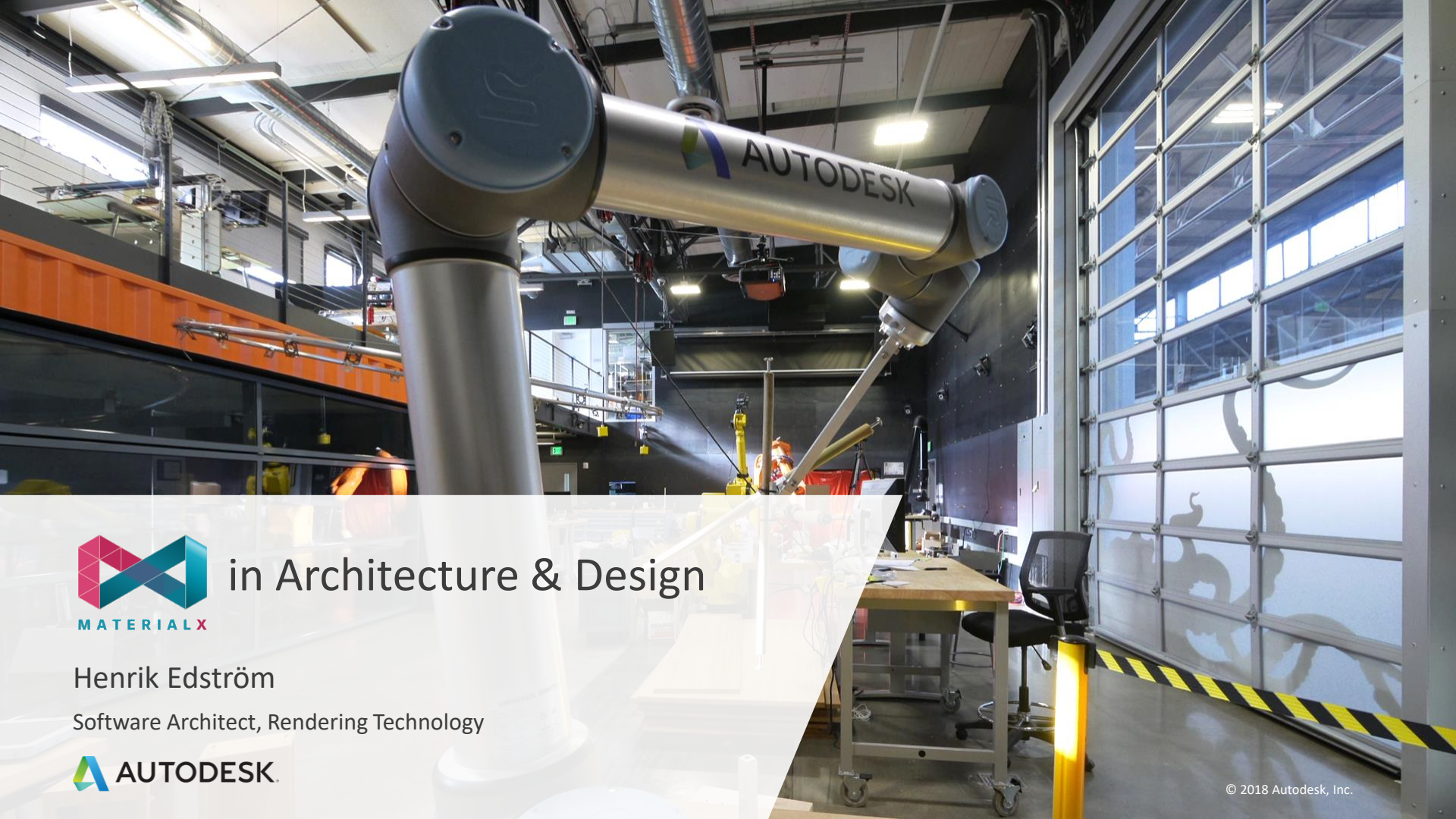
- Add support for missing features in 1.36+
- Support different implementations
  - Arnold native shaders
  - MaterialX standard library
  - ShaderX render-time generation

Material interop | Maya viewport and Arnold



Material interop | Unity





MATERIALX in Architecture & Design

Henrik Edström

Software Architect, Rendering Technology

AUTODESK.

Over 100 products and millions of users…

AUTODESK®

Make anything™

MEDIA & ENTERTAINMENT

Avengers, © 2018 Marvel. Image courtesy of Cinesite.

ARCHITECTURE, ENGINEERING & CONSTRUCTION

Image courtesy of Gensler.

PRODUCT DESIGN & MANUFACTURING

Image courtesy of Tesla Motors, Inc.

Material interop is one of the biggest challenges for our customers when it comes to visualization

“This is our biggest problem at the moment.

We can transfer 3D models over to almost any application, but we can’t move the materials in a good way, with the high fidelity we would like to”

Martin Enthed
Managing Director, IKEA Digital Lab
IKEA Communications AB

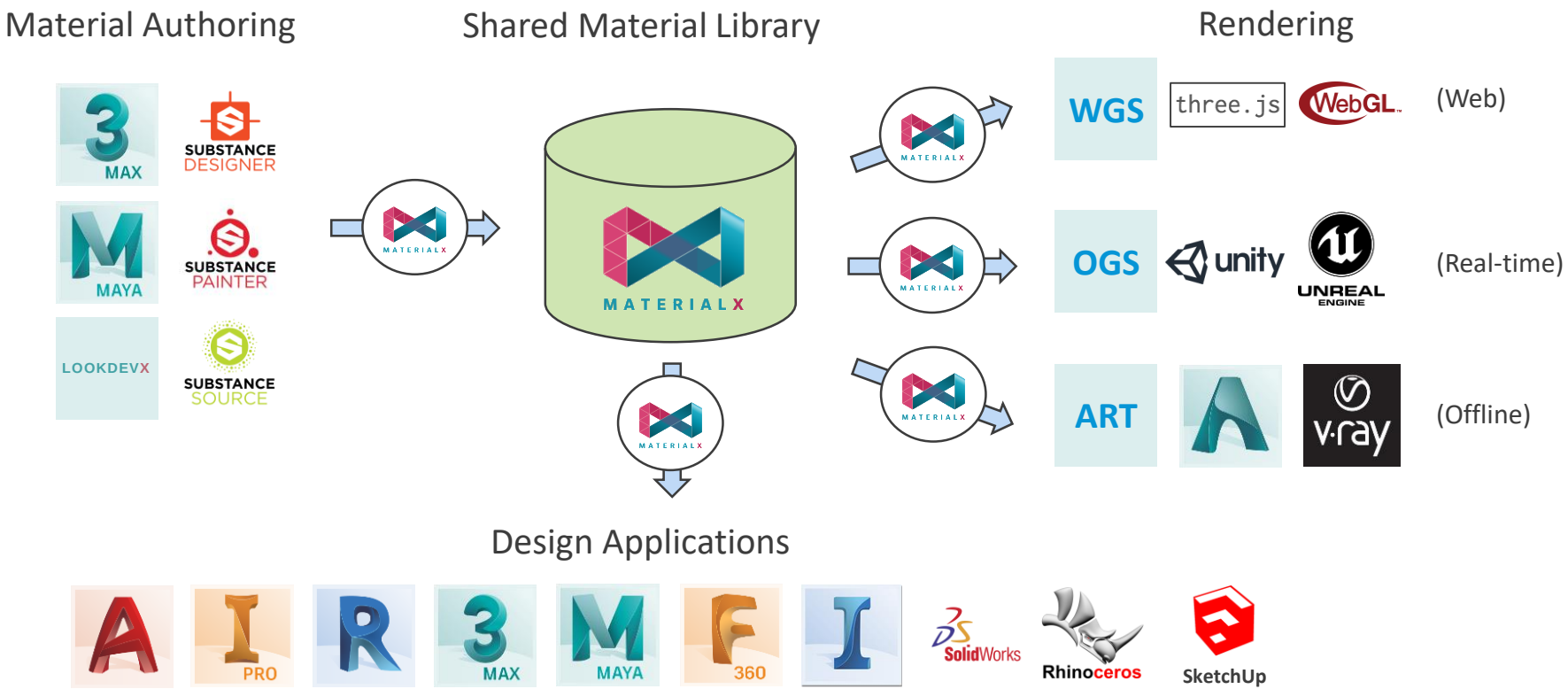
See Martin’s talk from Autodesk University 2017:
https://vimeo.com/243860738

“We see material interop as a fundamental challenge in our workflow.

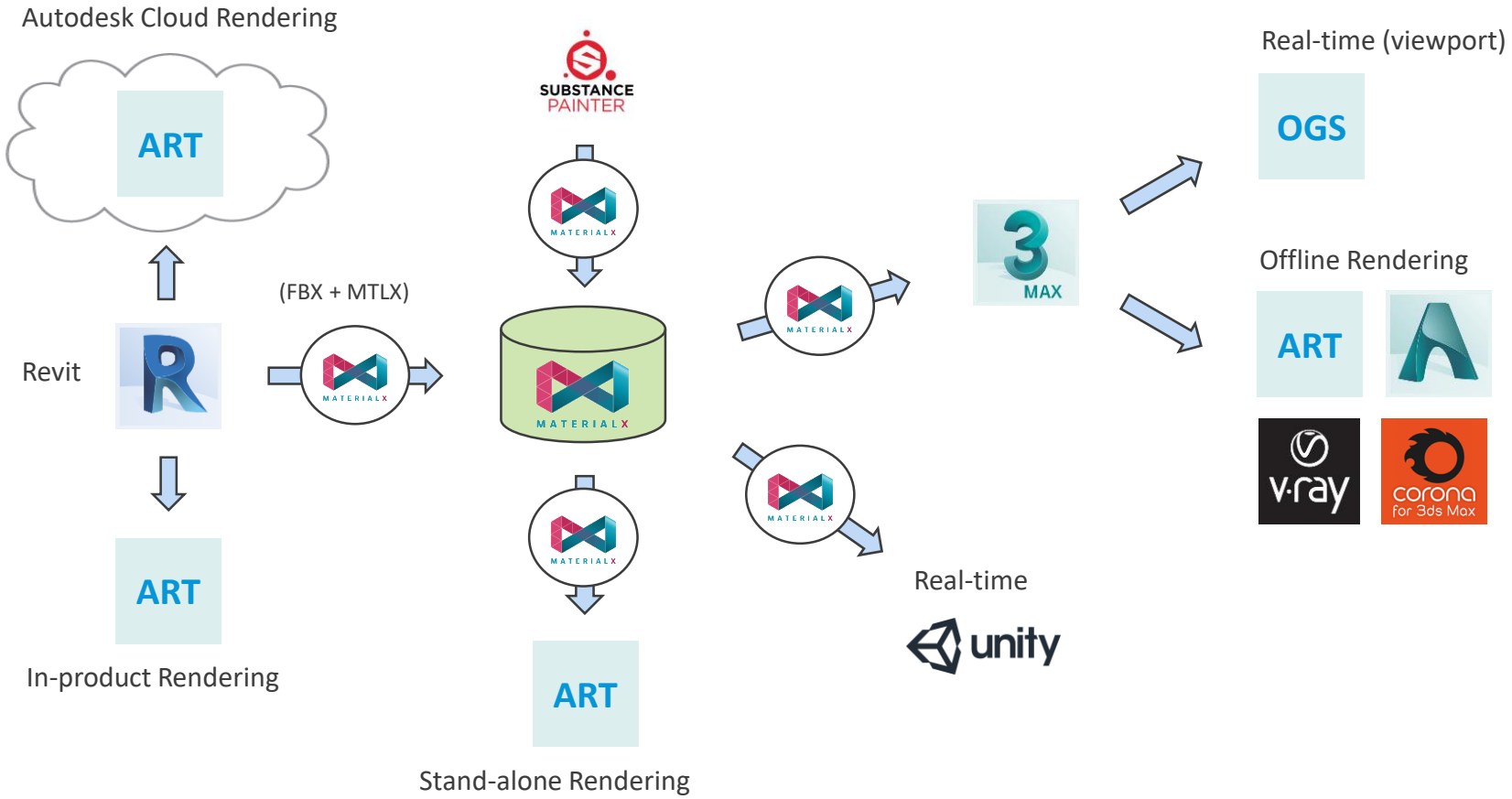
What we need is a render neutral PBR based material, which allows us to exchange content between not only all the applications in our production pipeline, but with all the applications and tools our customers work with”

Mark Kauffman
Technical Lead of Project Visualization
WSP USA

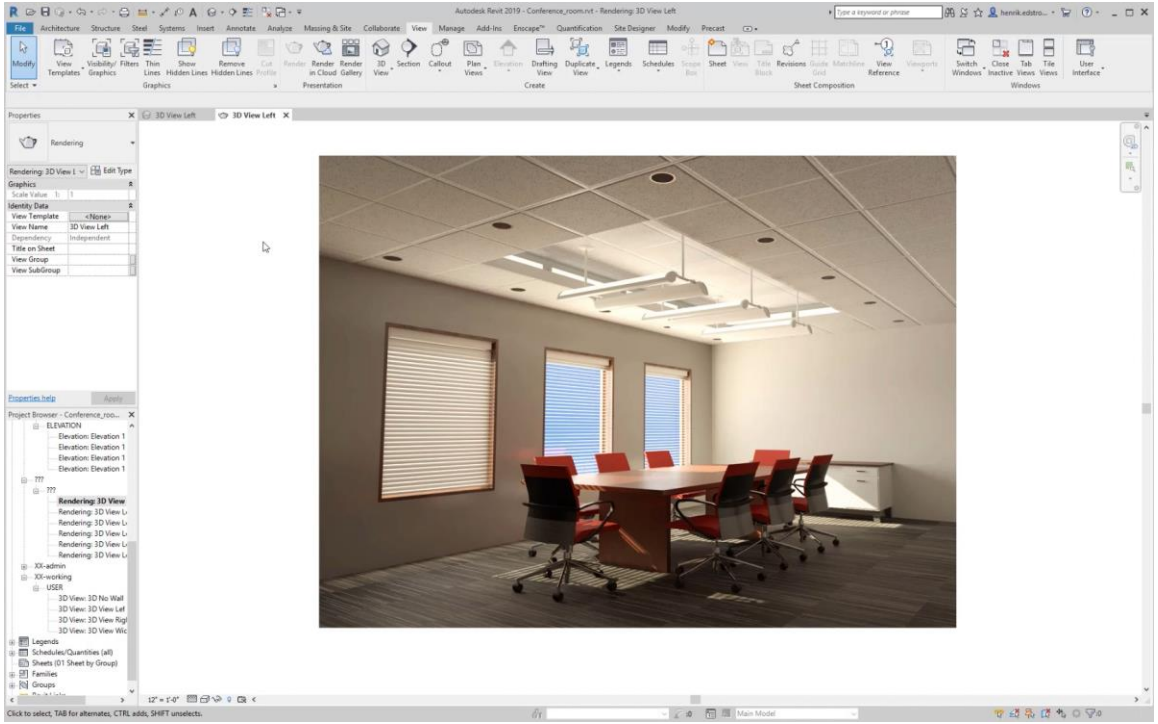
A generic material description that can easily be shared between *any application* and *any renderer*



Proof of Concept Demo – ArchViz Workflow



Video: Proof of Concept Demo – ArchViz Workflow



Acknowledgements

Jonathan Stone
Doug Smythe
Francois Chardavoine
Maggie Oh

David Larsson
Valère Lucquin
Alexis Khouri
Davide Pesare
Wes McDermott

Thomas Chollet
Bastien Humeau
Mark Visser
André Gauthier

Eric Bourque
Chris Vienneau
Gordon Bradley
Niklas Harrysson
Bernard Kwok
Jonathan Feldstein
Roberto Regalino

Zap Andersson
Neil Hazzard
Mike Russo

Örn Gunnarsson
Frederic Servant

Hilmar Koch

Ashwin Bhat
Karl Schmidt
Xiaoqing Zhuang
Susanna Holt
Roberto Ziche
Aradhana Vaidya

Martin Enthed
Mark Kauffman
Scott DeWoody

AUTODESK®

Make anything™