

MaterialX and LookdevX Update

Nikola Milosevic, Bernard Kwok

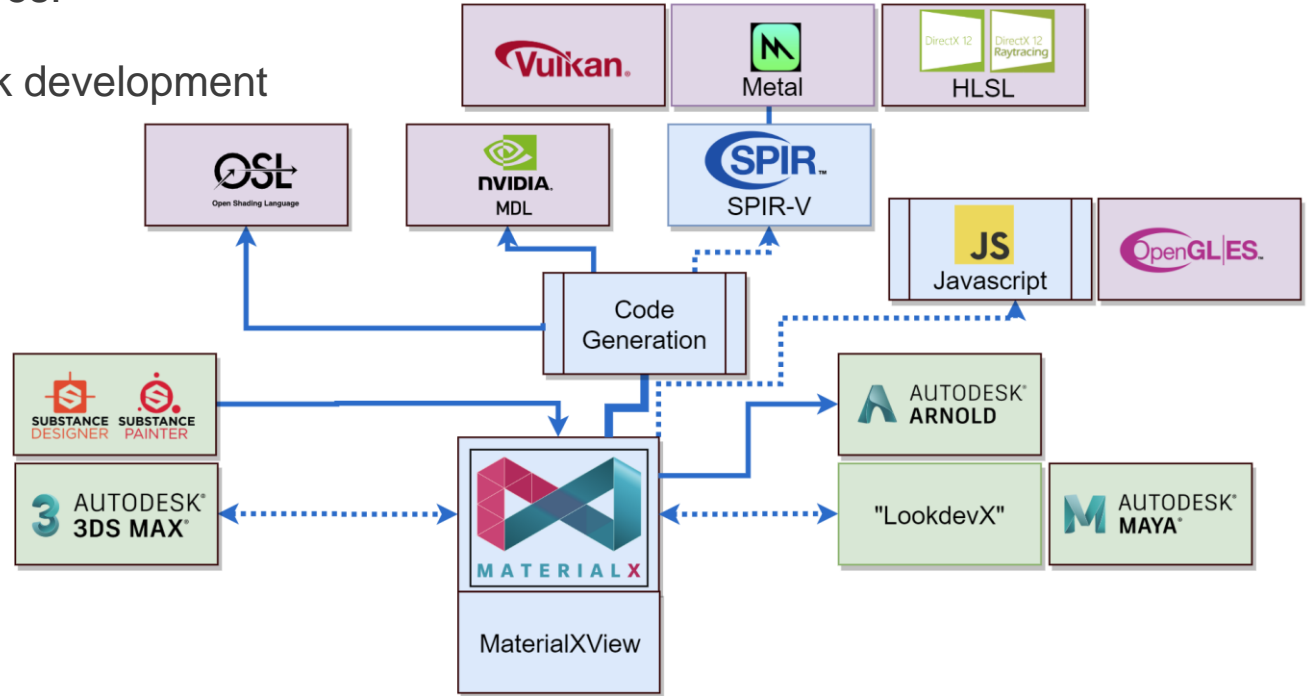
Principal Product Designer, Principal Engineer



Roadmap

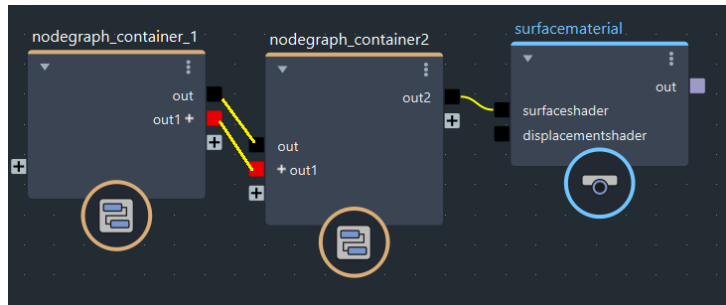
Focus Areas

1. MaterialX Core features.
2. Node editing and look development modules.
3. Interoperability between authoring tools and renderers.
4. MaterialX code generation, shading language and real-time rendering support.

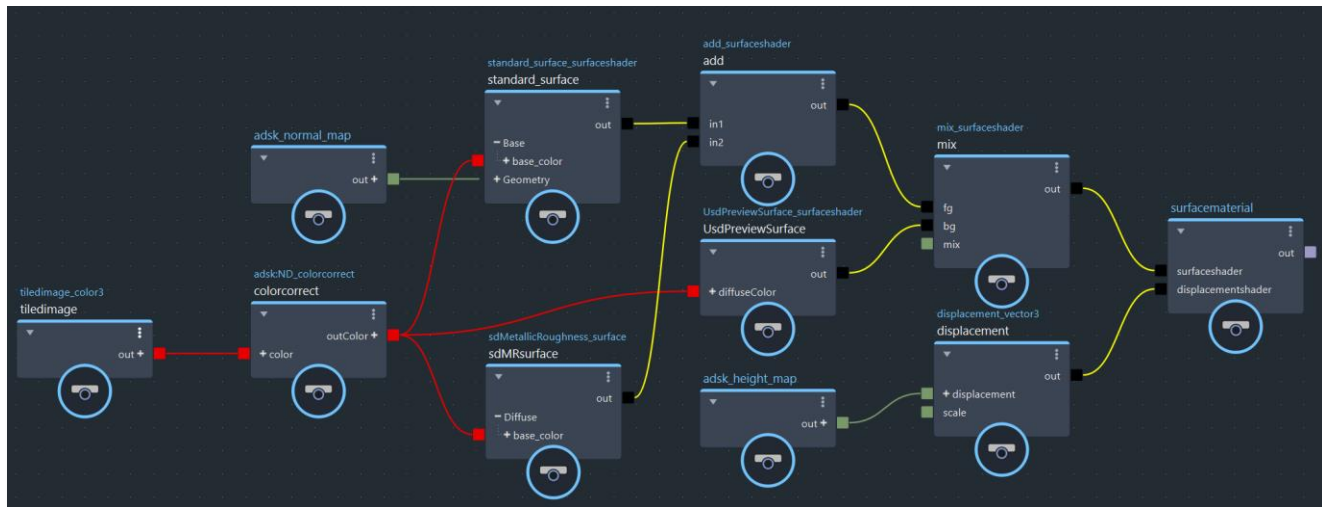


Runtime Data Model

- Key [1.37.1](#) / [1.38](#) updates for runtime shader graph support.
- Material nodes, nodegraphs as containers, <nodedef> publishing, code generation.
- Core libraries simplification:
 - stdlib, pbrlib, bxdx
- Uniform multiple library support:
 - Autodesk (adsk)
 - Others

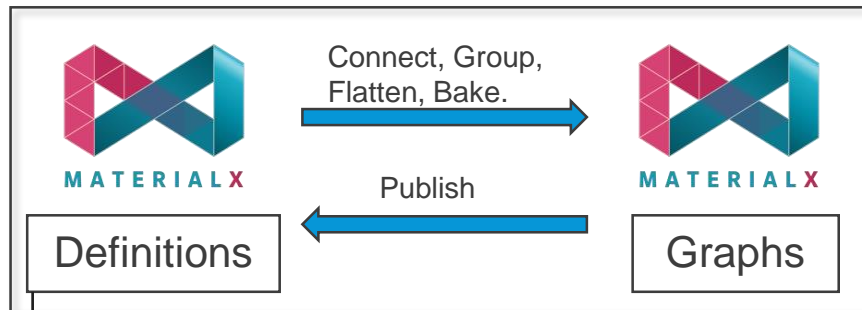


1.38 <nodegraph> pipeline configuration into a final surfacematerial..



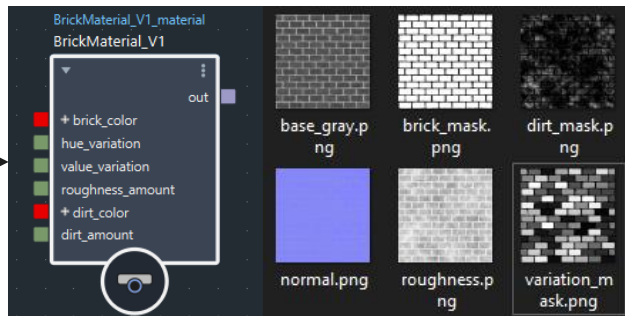
Sample shader graph composed of core, Autodesk, and Substance Designer [MaterialX Plug-In](#) definitions.

Managing Complexity

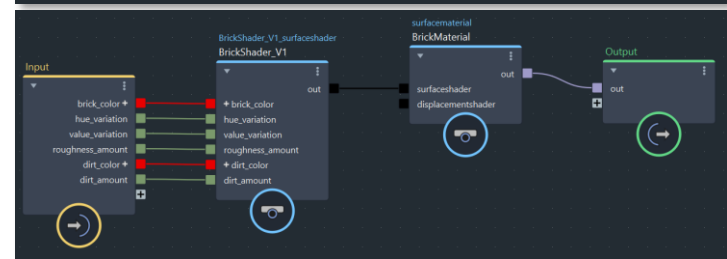
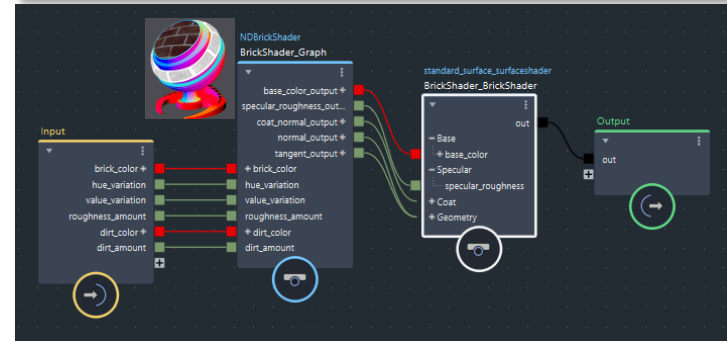
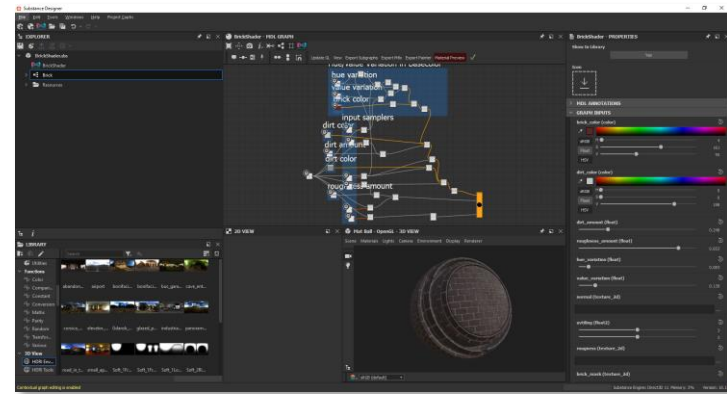


<nodedef>, <nodegraph> characteristics:

- Explicit Interface exposure: <input>, <output>.
- “version” / “namespace” / “target” tags
- Units / color management / user meta-data.



“Published” material instance with associated Substance Designer textures.



Top: MTLX graph, definition, and textures created in [Substance Designer](#). Shader (middle), and material (bottom) definitions created in LookdevX.

Asset Organization

Autodesk Material Definitions	Adobe Material Definitions	Other Material Definitions		
Autodesk Shader Definitions	Adobe Shader Definitions	Other Shader Definitions		
BXDF Library (bxdx) (Standard Surface)		Autodesk Base Definitions	Adobe BXDF Definitions	Other BXDF Definitions
PBR Library (pbrlib)			Adobe Base Definitions	Other Base Definitions
Standard Library (stdlib)				

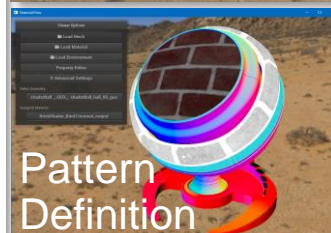
- Plan to Open Source some examples.



Material
Definition



Shader
Definition



Pattern
Definition



BXDF
Definition

```

+---adsklib
| | adsklib_defs.mtlx
| | adsklib_ng.mtlx
|
| +---genosl
| +---genmdl
| \---genosl
|
+---lights
|
+---materials
| |
| | \---brick
| | | \---brick_03
| | | | ND_brick_03_material.mtlx
| | | | NG_brick_03_material.mtlx
|
+---procedurals
| | ND_adsk_colorcorrect.mtlx
|
+---shaders
| |
| | \---brick
| | | \---brick_03
| | | | ND_brick_03_shader.mtlx
| | | | NG_brick_03_shader.mtlx
|
+---textures
| |
| | \---brick_03
| | | \---2k
| | | | brick_03_ao_2k.exr
| | | | brick_03_basecolor_2k.exr
| | | | brick_03_height_2k.exr
| | | | brick_03_normal_2k.exr
| | | | brick_03_roughness_2k.exr

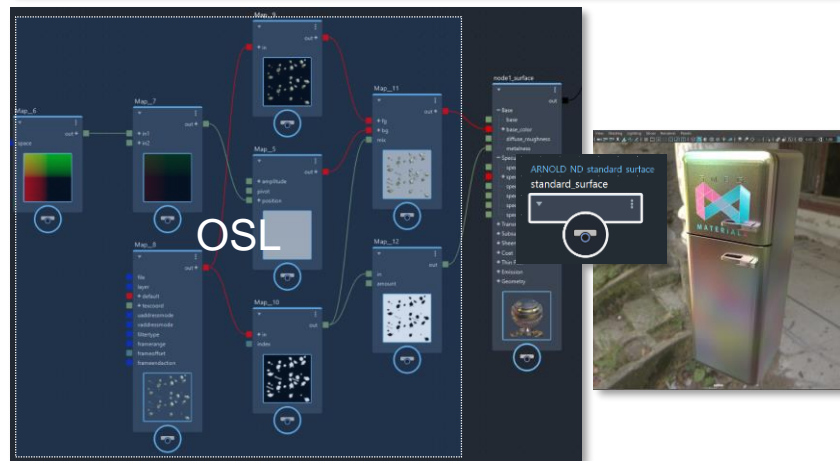
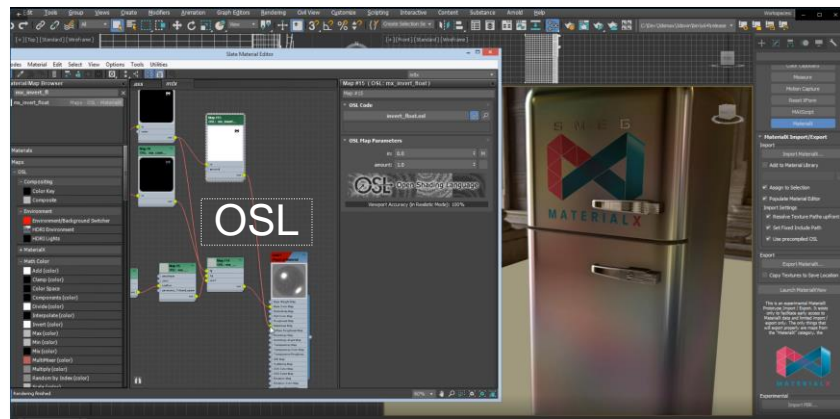
```

Images shown in MaterialXView (top to bottom): Autodesk brick shader, derived Substance Designer brick shader and supporting graph definition, Thin-film Standard Surface example. Shader ball asset created in Maya.

Application Integration

Strategies for consuming assets

- Consistent rendering via common tools:
 - Document merge, flattening / baking
 - Input bindings exposure.
- Examples:
 - 3ds Max:
 - <nodegraph> flattening + codegen OSL reference functions.
 - Arnold:
 - <implementation>: shader / pattern OSL generation or built-in.
 - <nodedef> handling.
 - Swatch: Upstream sub-graph bake.



OSL rendering in 3ds Max (top). OSL rendering using Arnold in Maya. Swatch rendering using MaterialX GLSL renderer. (Fridge shader created in 3ds Max.)

Real Time Viewports

Extending Code Generation

- Make [SPIR-V](#) an official target
- Cross-compile to new targets
 - HLSL for DX12
 - Vulkan
 - MetalSL
 - DXR and Vulkan RT
- Plan to make available as Open Source.



```
void mx_multiply_bsdf_color_indirect(vec3 V, vec3 in1, vec3 in2, vec3 out)
{
    result = in1 * clamp(in2, 0.0, 1.0);
}

void IMPL_standard_surface_surfaceshader(float base, vec3 emission_color, vec3 base_color, float coat_rotation, float coat_roughness, float coat_anisotropy, vec3 coat_attenuation_bg_cm_out, vec3 coat_emission_attenuation_bg_cm_out, vec3 opacity_luminance_out)
{
    vec3 emission_weight_out = emission_color * emission_color;
    vec3 metal_reflectivity_out = base_color * base_color;
    const float coat_tangent_rotate_degree_in2_tmp = 360.0f;
    float coat_tangent_rotate_degree_out = coat_rotation * coat_tangent_rotate_degree_in2_tmp;
    vec2 coat_roughness2_out = vec2(0.0);
    mx_roughness_anisotropy(coat_roughness, coat_anisotropy, coat_roughness2_out);
    vec3 coat_attenuation_bg_cm_out = vec3(0.0);
    mx_acescg_to_linear_color3(vec3(1.000000, 1.000000, 1.000000), coat_attenuation_bg_cm_out);
    vec3 coat_emission_attenuation_bg_cm_out = vec3(0.0);
    mx_acescg_to_linear_color3(vec3(1.000000, 1.000000, 1.000000), coat_emission_attenuation_bg_cm_out);
    vec3 opacity_luminance_out = vec3(0.0);
}

void mx_multiply_bsdf_color_indirect(float3 V, float3 in1, float3 in2, float3 out)
{
    result = in1 * clamp(in2, 0.0f.xxx, 1.0f.xxx);
}

void IMPL_standard_surface_surfaceshader(float base, float3 emission_color, float3 base_color, float coat_rotation, float coat_roughness, float coat_anisotropy, float3 coat_attenuation_bg_cm_out, float3 coat_emission_attenuation_bg_cm_out, float3 opacity_luminance_out)
{
    float3 emission_weight_out = emission_color * emission_color;
    float3 metal_reflectivity_out = base_color * base_color;
    float coat_tangent_rotate_degree_out = coat_rotation * coat_tangent_rotate_degree_in2_tmp;
    float2 coat_roughness2_out = 0.0f.xx;
    float param = coat_roughness;
    float param_1 = coat_anisotropy;
    float2 param_2;
    mx_roughness_anisotropy(param, param_1, param_2);
    float2 coat_roughness2_out = param_2;
    float3 coat_attenuation_bg_cm_out = 0.0f.xxx;
    float3 param_3 = 1.0f.xxx;
    float3 param_4;
```


GLSL - CodeGen

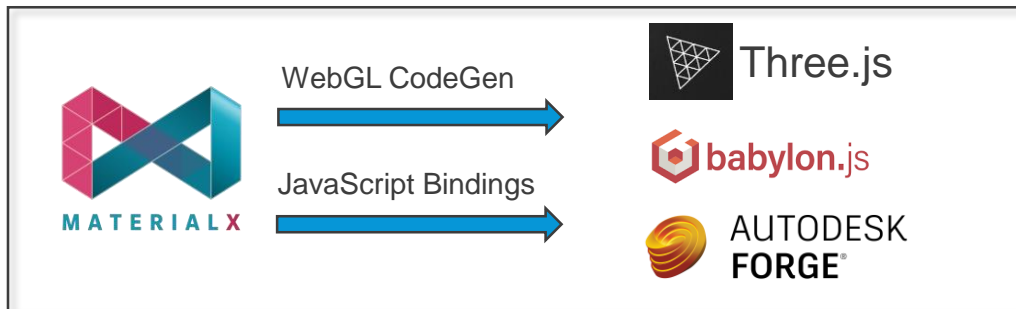
HLSL – via SPIR-V

GLSL to HLSL cross compilation via SPIR-V helps maintain MaterialX node implementation portability.

Web Browser

MaterialX for the Web

- Javascript bindings for MaterialX
 - Goal is to match Python bindings
 - Using Web Assembly ([WASM](#))
 - [Open Source support in progress](#):  **GitHub**
 - *JsMaterialXCore*, *JsMaterialXFormat*
- Autodesk web solution: [Autodesk Forge Viewer](#)

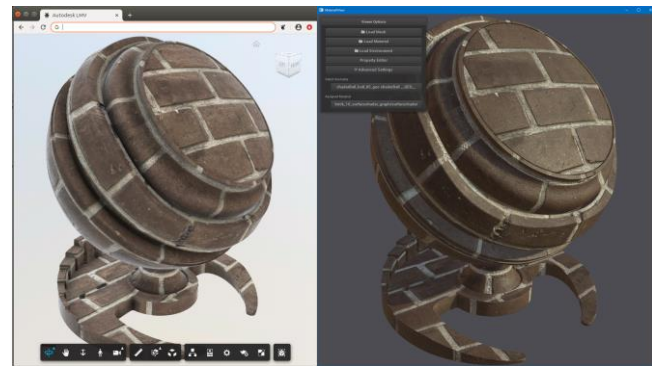


```
import Module from 'JsMaterialX.js';

let mx, doc;
mx = await initMaterialX();
doc = mx.createDocument();

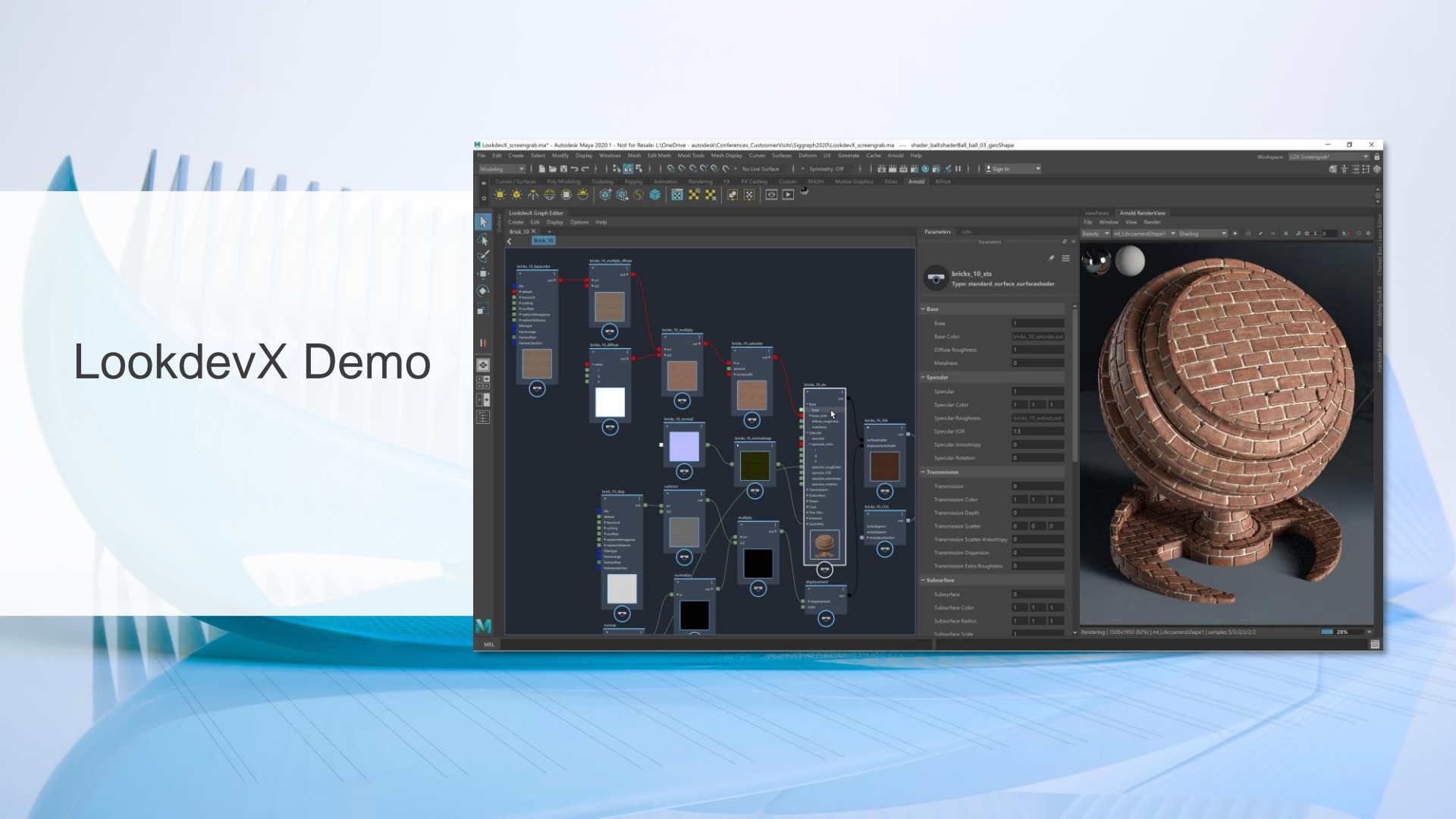
let nodeGraph;
nodeGraph = doc.addNodeGraph();

let output1, output2, constant, image;
constant = nodeGraph.addNode('constant');
image = nodeGraph.addNode('image');
output = nodeGraph.addOutput();
output.setConnectedNode(image);
```



Left: Autodesk Forge Viewer: Standard Surface through [3dsMax "Shared Views"](#).
Right: The same shader rendered in MaterialX view.

LookdevX Demo



Acknowledgements

- MaterialX/ShaderX/LookdevX : Niklas Harrysson, Ashwin Bhat, Jonathan Feldstein, Nicolas Savva, Fedor Nikolayev, Tom Varik, Henrik Edström, Eric Bourque
- 3ds Max: Zap Andersson, Neil Hazzard
- Arnold : Orn Gunnarsson, Krishnan Chunangad Ramachandran
- Substance Designer / Painter: David Larsson
- Lucasfilm: Doug Smythe, Jonathan Stone
- Assets: Arvid Shneider and Dušan Ković (Autodesk), Justin Patton (Adobe), [Matz models](#), [\(Turbosquid\)](#)



AUTODESK®

Make anything™