Rendering Everything Everywhere
Creating a Flexible Rendering Pipeline

George Maestri
Staff Author – LinkedIn Learning
George Maestri
Staff Author, CAD, 3D, Animation
Goal:

Render Everything Everywhere

Create a pipeline that allows for images to be created at any point.
Outline

Introduction / Definition
Current Technology
Possible Pipelines
Materials
Lights & Cameras
Recommendations
Futures
Introduction
Visualization
The Challenge
The Challenge

1 – We have to render for a lot of outputs
The Challenge

1 – We have to render for a lot of outputs

Stills
The Challenge

1 – We have to render for a **lot** of outputs
The Challenge

1 – We have to render for a lot of outputs

Stills

Motion

Interactive/VR
The Challenge

1 – We have to render for a lot of outputs
The Challenge

1 – We have to render for a lot of outputs

2 – Those outputs often require different technologies
The Challenge

Stills

Motion

Interactive/VR
The Challenge

Stills

Motion

Interactive/VR

Classic Rendering
Arnold
V-Ray
…etc

Real Time Rendering
Unity
Unreal
…etc
The Challenge

1 – We have to render for a lot of outputs

2 – Sometimes those outputs require different technologies

3 – Building an efficient pipeline is challenging
The Challenge

1 – We have to render for a lot of outputs

2 – Sometimes those outputs require different technologies

3 – Building an efficient pipeline is challenging

4 – Understand What Works, What Doesn’t
It’s Kind of a Mess

1 – Traditional 3D Creation Tools (Maya, 3ds Max, Revit, etc…)

2 – Renderers Galore (Arnold, V-Ray, Redshift, Octane, etc)

3 – Translation tools (that don’t always work)

4 – Real-Time Engines (different workflows / requirements)
3D / Modeling
(Max, Maya, Revit, etc)

Interactive/Real Time
(Unreal, Unity, etc...)
Obstacles

**Familiarity with Tools** – Artists often go with what they know.

**Archival Assets** – All that stuff you created may need to change.

**Outside Assets** – Clients and freelancers also need to be on board.

**Changing Landscape** – Which technology will prevail?
Changing Landscape

3D / Modeling
(Max, Maya, Revit, etc)

Interactive/Real Time
(Unreal, Unity, etc...)
Changing Landscape

3D / Modeling
(Max, Maya, Revit, etc)

Interactive/Real Time
(Unreal, Unity, etc...)

Changing Landscape
Paradigm Shift

The move towards Real-Time rendering and interactivity has created a paradigm shift in the way we create images.

Workflows are changing quickly, creating obstacles.

This will eventually get better, but how do you survive the shift?
Technology
Rendering Technology

PARADIGM SHIFT
Rendering technology is increasingly moving towards real-time creation and playback.

FASTER HARDWARE
Fast graphics cards allow for much higher quality and more interactive rendering.

SOFTWARE EVOLUTION
Traditional Tools embracing the GPU as well as prioritizing interactivity.
“Classic” Rendering

“CLASSIC” WORKFLOW
The classic workflow is tightly integrated into the 3D creation tools, creating a seamless workflow.
Ability to do Post-Processing/Compositing for added quality.

RENDERING TAKES AS LONG AS NEEDED
Image quality is king.
High-end features/effects can be added, but at a speed cost.
Interactivity takes a back seat.
“Classic” Rendering

“CLASSIC” PROS
No compromise on image quality.
Familiar workflows
Tight integration with 3D apps – easy to create an image.
Can do finishing/compositing on output.

“CLASSIC” CONS
Not truly interactive
No “VR”
Some things just take longer
“Real-Time” Rendering

“REAL-TIME” WORKFLOW
Emphasis is on speed/real-time first.
True interactivity with scene contents
VR/AR and other advanced visualizations

RENDERING HAPPENS NOW
Every image takes 1/30 second
Very quick iterations when authoring
“Real-Time” Rendering

“REAL-TIME” PROS
True Interactivity
Instant Results / Quick iterations
VR/AR

“REAL-TIME” CONS
Speed sometimes wins over Quality
Unfamiliar workflows
Integration with 3D Tools can be challenging.
CLASSIC IS GAINING SPEED
Interactive viewports
Better GPU support/integration
Fewer iterations

REAL TIME IS GAINING QUALITY
Raytracing with RTX/Nvidia
Advanced Features
Better quality
Classic Getting Faster
Real-Time Getting Better
We’re Not Quite There (yet)

THE LINES ARE BLURRING

EACH STILL HAS ADVANTAGES
Pipelines
Typical Pipeline

MODELING

UV/TEXTURES

MATERIALS

LIGHTS/CAMERAS

RENDER
Typical Pipeline

MODELING  UV/TEXTURES  MATERIALS  LIGHTS/CAMERAS  RENDER

3D / Modeling
(Max, Maya, Revit, etc)
Typical Pipeline

**MODELING**
3D / Modeling
(Max, Maya, Revit, etc)

**UV/TEXTURES**

**MATERIALS**

**LIGHTS/CAMERAS**

**RENDER**

Classic
(Arnold, V-Ray, etc)

Interactive/Real Time
(Unity, Unreal, etc)
Typical Pipeline

3D / Modeling
(Max, Maya, Revit, etc)

Classic
(Arnold, V-Ray, etc)
Typical Pipeline

3D / Modeling
(Max, Maya, Revit, etc)

Interactive/Real Time
(Unity, Unreal, etc)

GREY AREA
Typical Pipeline

MODELING

UV/TEXTURES

MATERIALS

LIGHTS/CAMERAS

RENDER

3D / Modeling
(Max, Maya, Revit, etc)

Interactive/Real Time
Possible Pipelines

Render Classic Only

Render Real-Time Only

Move Between the Two
Possible Pipeline: Render 100% Classic
Possible Pipeline: Render 100% Classic

3D / Modeling (Max, Maya, Revit, etc) Classic (Arnold, V-Ray, etc)
Possible Pipeline: Render 100% Classic

Render Everything “Classic”

Interactivity is limited to real-time in viewports
Possible Pipeline : Render 100% Classic
Possible Pipeline: Render 100% Real-Time
Possible Pipeline: Render 100% Real-Time

MODELING

UV/TEXTURES

MATERIALS

LIGHTS/CAMERAS

RENDER

3D / Modeling
(Max, Maya, Revit, etc)

Interactive/Real Time
Possible Pipeline: Render 100% Real-Time

Gain Latest Features

May Compromise Some Quality

May Compromise Integration
Possible Pipeline: Render 100% Real-Time

Pro

Cons
Possible Pipeline: Use Both / Hybrid
Possible Pipeline: Use Both / Hybrid

- **MODELING**
  - 3D / Modeling
    - (Max, Maya, Revit, etc)

- **UV/TEXTURES**
  - Classic
    - (Arnold, V-Ray, etc)

- **MATERIALS**
  - Interactive/Real Time
    - (Unity, Unreal, etc)

- **LIGHTS/CAMERAS**
  - Render
Possible Pipeline: Use Both / Hybrid

MODELING

UV/TEXTURES

3D / Modeling
(Max, Maya, Revit, etc)

Classic

MATERIALS

LIGHTS/CAMERAS

RENDER

Interactive/Real Time
Materials
MATERIALS

BIGGEST HURDLE

Different Algorithms (Blinn/Phong/Albedo/PBR)
Lots of Parameters
Custom Features
Different “Looks”
Different Shading Models

STANDARD
Phong / Blinn / Lambert

PHYSICAL
Physically Based Rendering (PBR)
Lots of Parameters
Different Models

**STANDARD**
- Phong / Blinn / Lambert
  - Color
  - Specularity
  - Specular Falloff
  - Reflectivity
  - Transparency
  - Bump/Normal
  - Refractions

**PHYSICAL**
- Physically Based Rendering (PBR)
  - Base Color
  - Metallic
  - Roughness
  - Opacity
  - Normal
  - Refractions
Different Models

**STANDARD**
- Phong / Blinn / Lambert
  - Color
  - Specularity
  - Specular Falloff
  - Reflectivity
  - Transparency
  - Bump/Normal
  - Refractions

**PHYSICAL**
- Physically Based Rendering (PBR)
  - Base Color
  - Metallic
  - Roughness
  - Opacity
  - Normal
  - Refractions
Different Models

**STANDARD**
- Phong / Blinn / Lambert
- Color
- Specularity
- Specular Falloff
- Reflectivity
- Transparency
- Bump/Normal
- Refractions

**PHYSICAL**
- Physically Based Rendering (PBR)
- Base Color
- Metallic
- Roughness
- Opacity
- Normal
- Refractions
Translation Tools

STANDARD FORMATS

FBX
OSL (Open Shading Language) - future
Material X - future

THIRD PARTY TOOLS

Datasmith (Unreal)
Custom scripts
Others
FBX
FBX

CREATED BY KAYDARA

FBX = “Filmbox”

Later became Motionbuilder

GREAT FOR MESHES / MOTION / RIGGING

Multiple mesh formats (Polygons, NURBS)

Supports animation and multiple takes

Supports skeletons/deformations

NOT IDEAL FOR MATERIALS

Supports “standard” material shading models (“Blinn, Phong, etc…”)

Advanced rendering features not well supported

PBR shading models not really supported
FBX - MATERIALS

**STANDARD**
- Phong / Blinn / Lambert
  - Color
  - Specularity
  - Specular Falloff
  - Reflectivity
  - Transparency
  - Bump/Normal
  - Refractions

**PHYSICAL**
- Physically Based Rendering (PBR)
  - Base Color
  - Metallic
  - Roughness
  - Opacity
  - Normal
  - Refractions
FBX - MATERIALS

**STANDARD**
Phong / Blinn / Lambert
- Color
- Specularity
- Specular Falloff
- Reflectivity
- Transparency
- Bump/Normal
- Refractions

**PHYSICAL**
Physically Based Rendering (PBR)
- Base Color
- Metallic
- Roughness
- Opacity
- Normal
- Refractions
FBX

FBX IS GREAT FOR WHAT IT DOES BEST
- Models
- Animation/Rigging
- Basic Materials / Shading / UVs

YOU MAY NEED TO WORKAROUND
- Advanced Render Features
- Non-trivial shading models
Third-Party Translators

DATASMITH (UNREAL)

- Works well for supported packages (NOTE: Maya currently not supported)
- Translates materials with 80-90% accuracy
- Some advanced features still may not translate
- Works on some third-party renderers (V-Ray, Corona…)

OTHERS

- Third-party tools
- Custom scripts / Plug-ins
EVEN THE BEST TRANSLATION ISN’T 100% ACCURATE

TRANSLATE WHAT YOU CAN, FIX THE REST
Parallel Libraries

**MAYA / 3DS MAX**
- Car_Paint_Red
- Chrome_01
- Rubber_01

**UNITY / UNREAL**
- Car_Paint_Red
- Chrome_01
- Rubber_01

Create Equivalent Materials on Both Sides

Standard Naming Schemes
Parallel Libraries

MAYA / 3DS MAX

Car_Paint_Red  →  Car_Paint_Red
Chrome_01 → Chrome_01
Rubber_01 → Rubber_01

UNITY / UNREAL

Car_Paint_Red
Chrome_01
Rubber_01
Parallel Libraries

MAYA / 3DS MAX
- Car_Paint_Red
- Chrome_01
- Rubber_01

UNITY / UNREAL
- Car_Paint_Red
- Chrome_01
- Rubber_01
Parallel Libraries

MAYA / 3DS MAX

Car_Paint_Red

Chrome_01

Rubber_01

UNITY / UNREAL

Car_Paint_Red

Chrome_01

Rubber_01
Parallel Libraries

PROS
Straightforward to Implement
Enjoy best features of each renderer
No worries about translation errors
Gets easier as you go

CONS
Maintaining dual libraries can be time-consuming
Must enforce standard naming schemes
Kind of annoying
Standardization

STANDARDIZE MATERIALS

Substance

STANDARDIZE RENDERER

V-Ray

Octane
Standardizing Materials

STANDARDIZED MATERIAL FORMATS

- OSL (Open Shading Language)
- PBR Based workflow
- Material X (future)

STANDARDIZE MATERIAL CREATION

- Substance
Standardized Formats

OSL (OPEN SHADING LANGUAGE)
- Lots of features/capabilities
- Supported in Arnold/V-Ray/Octane
- Limited Real-Time Support

PBR-BASED WORKFLOW
- Each renderer’s “PBR” is slightly different (not really a ‘format’).
- Still run into file format / translation issues

MATERIAL X
- Still future - Not developed/released.
Standardizing Material Creation
Standardizing Material Creation
Create materials on each side of the gap

Version control

Testing
CREATES "RENDERER-NEUTRAL" MATERIALS

Materials are created in Substance
These can then be used in the renderer of choice
Supports wide range of packages, including Real-Time engines

CREATE/DOWNLOAD MATERIAL LIBRARIES

Materials can be stored/archived for later use
Substance has a wealth of materials for download / purchase
Third party libraries also available

RESOLUTION-INDEPENDENT

Materials / Textures are generated at render
Lower Res for Real-Time/Interactive
Higher res for quality stills/video
Typical Pipeline

3D / Modeling
(Max, Maya, Revit, etc)

Classic
(Arnold, V-Ray, etc)

Interactive/Real Time
(Unity, Unreal, etc)
Lighting

CLASSIC LIGHTING

Standard Lights (Spot, point, directional, etc…)
Special Lights (Area lights, etc)
Geometry Lights
Environmental Light
Secondary Light (bounce lighting)

REAL-TIME LIGHTING

Standard Lights
Special Lights
Environmental Light
Secondary Light (bounce lighting)
Lighting

CLASSIC LIGHTING

Standard Lights (Spot, point, directional, etc…)
Special Lights (Area lights, etc)
Geometry Lights
Environmental Light
Secondary Light (bounce lighting)

REAL-TIME LIGHTING

Standard Lights
Special Lights
Environmental Light
Secondary Light (bounce lighting)
Real-Time Lighting

DIFFERENT WORKFLOW

Light
Real-Time Lighting

DIFFERENT WORKFLOW

Light

Bake
Real-Time Lighting

DIFFERENT WORKFLOW

Light

Bake

Interact / Render
Real-Time Lighting

DIFFERENT WORKFLOW

Light
Bake
Real-Time
Classic
Interact / Render
**Baking Lights**

**REMOVES LIGHTS FROM FINAL CALCULATIONS**

- Speeds up Real-Time rendering
- Simplifies calculations

**CAN BAKE IN MULTIPLE PLACES**

- Baking in Real-Time engine usually happens automatically
- Can bake externally in some classic renderers to utilize special features
- External baking also captures some **materials**, removing need to translate those.

**NOT ALL LIGHTING CAN BE BAKED**

- Any light that moves or changes.
- Any light casting shadows on moving objects
Baking Lights

BAKING IN REAL-TIME ENGINES

- Lighting is designed in Real-Time engine
- Baking happens automatically (easy to implement)
- Lighting from 3D Applications will need to be translated if rendering everywhere

BAKING IN EXTERNAL (CLASSIC) RENDERER

- Removes the need to translate lights
- Can possibly remove the need to translate some materials
- Workflows can be challenging / not straightforward
Translating Lighting
Translating Lighting

**FBX-SUPPORTED**
- Spotlights
- Point Lights
- Directional

**NOT FBX SUPPORTED**
- Photometric / IES
- Environmental Lighting
- Area Lights, Rect Lights, Tube lights, etc...
- Directional
## Translating Lighting

<table>
<thead>
<tr>
<th>FBX-SUPPORTED</th>
<th>DATASMITH (3DS MAX / REVIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spotlights</td>
<td>Spotlights</td>
</tr>
<tr>
<td>Point Lights</td>
<td>Point Lights</td>
</tr>
<tr>
<td>Directional</td>
<td>Directional</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOT FBX SUPPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photometric / IES</td>
</tr>
<tr>
<td>Environmental Lighting</td>
</tr>
<tr>
<td>Area Lights, Rect Lights, Tube lights, etc…</td>
</tr>
<tr>
<td>Directional</td>
</tr>
<tr>
<td>Photometric / IES</td>
</tr>
<tr>
<td>Environmental Lighting</td>
</tr>
<tr>
<td>V-Ray lights</td>
</tr>
<tr>
<td>Corona Lights</td>
</tr>
</tbody>
</table>
Translating Lighting

FBX TRANSLATION

Only basic lights come across
WILL need to tweak in Real-Time Engine

DATASMITH TRANSLATION

More stuff comes across
Not perfect – will still need adjustment.
Standardize Rendering
Standardized Rendering

ONE RENDERER FOR (ALMOST) EVERYTHING

Get renderer that can plug into both 3D App and Real-Time App

- Lighting is supported
- Materials are also supported
- Other rendering features can come across
Standardized Rendering

ONE RENDERER FOR (ALMOST) EVERYTHING

Get renderer that can plug into both 3D App and Real-Time App

- Lighting is supported
- Materials are also supported
- Other rendering features can come across
V-Ray Workflow

V-Ray for 3ds Max
Get V-Ray for 3ds Max free trial >

V-Ray for Maya
Get V-Ray for Maya free trial >

V-Ray for SketchUp
Get V-Ray for SketchUp free trial >

V-Ray for Rhino
Get V-Ray for Rhino free trial >

V-Ray for Revit
Get V-Ray for Revit free trial >

V-Ray for Modo
Get V-Ray for Modo free trial >

V-Ray for Nuke
Get V-Ray for Nuke free trial >

V-Ray for Katana
Get V-Ray for Katana free trial >

V-Ray for Unreal
Get V-Ray for Unreal free trial >
Standardized Rendering

Geometry / FBX
Standardized Rendering

Materials
Lights
Cameras
Standardized Rendering

Materials
Lights
Cameras
Standardized Rendering

3Max → v-ray → Unreal Engine

Materials
Lights
Cameras

VR headset with virtual environment
Standardized Rendering

3D MAX → v-ray → Unreal Engine
Materials
Lights
Cameras
Standardized Rendering

Materials
Lights
Cameras
Standardized Rendering

3DS MAX → Materials, Lights, Cameras → v-ray → Unreal Engine → VR glasses
Standardized Rendering

3D Studio Max -> v-ray -> Unreal Engine

Materials
Lights
Cameras
Standardized Rendering
A FEW CAVEATS....

FBX does not move V-Ray materials, lights, etc.
Moving V-Ray between scenes requires using .vrscene files
Not all V-Ray features are supported in Unreal
Other Considerations
Cameras

Cameras are straightforward

- A few simple parameters (position, orientation, FOV, etc…)
- They do translate via FBX
- Some render-specific things may not translate (effects, bokeh, etc…)

Usually set up for a specific render

- Most times, we set up the camera for the specific shot
- Recreating a single camera is not that big of a task

Interactive/VR is a separate camera

- You’ll still set up VR in the Real-Time engine, no need to translate
Visual Effects

Can be resource-intensive

- Some effects can be very compute-intensive, not ideal for real time
- Offline or Classic rendering may suit these better
- Much better tools in 3D apps (Bitfrost, Hair, Fur, etc…)

Real-Time can handle some

- Much progress on real-time front.
- Some effects can be cached to save compute cycles
- Unreal - Chaos
Post-Processing

Post-Production

Classic rendering can output multiple passes/layers for post
Photoshop/After Effects/Nuke are great for getting ultimate quality
Can also help with integration (live action / photos)

Real-Time

Great way to add another layer of quality/control.
Many post effects can be duplicated
Upper limit on how much you can do in real time.
Futures

**Faster Machines / More convergence**

- Classic rendering gets fast enough to be real time
- Real time gets enough power to be photoreal

**Better Tools / Translation**

- Better movement of assets to/from Real-Time
- Better translation tools

**Standardization**

- OSL – Already a standard, may become more common
- Material X – Common material definition
Recommendations
Recommendations

Best practices

Whatever pipeline you choose, s
3ds Max to Unreal has slight advantage

Datasmith will preserve a lot more data
Less need to revise/create new assets
Translation
FBX can be used

Will translate geometry, animation
Some materials, lights, etc may not fully translate
Materials
Materials

Substance to Manage Materials
Classic Rendering
Classic Rendering

3D MAX

V-Ray

Octane Render

Unity
Summary