Validating XPU Renders with RIS

XPU produces renders that are predictive of RIS, however, there are features in RIS that don't exist in XPU yet.

Some of these features are the reason why your XPU images may differ slightly from RIS.

If you find yourself in the situation where you need to validate the results you see in XPU using RIS, use the hints below to make your RIS renders achieve that comparison. However, if you tweak these knobs, be aware you may be artificially slowing down RIS, so proceed with caution.

- Sample count - XPU does not yet support adaptive sampling, so you must be sure to disable it
    - minsamples = maxsamples - To ensure that XPU and RIS are sampling at similar rates, you need to make sure that you have disabled adaptive sampling via the easiest mechanism: setting the minsamples parameter value equal to the maxsamples parameter value on PxrPathTracer.

- Ray depths - Make sure your ray depths are equivalent. If you don't do this, you'll be giving unfair advantage to RIS because XPU does not yet support several options that RIS provides to limit ray depths. Limiting ray depths is one of the primary ways in RIS to get better performance.
    - maxIndirectBounces - This is the only PxrPathTracer knob in XPU that controls ray depth.
    - maxdiffusedepth and maxspeculardepth - These are object attributes used to control how many diffuse and specular bounces RenderMan should descend into as it traces. It is one of the knobs that you can use to trade look for performance. XPU does not support these. So for your RIS render, be sure that you have set these object attributes to a number higher than maxIndirectBounces on the Integrator or else you will be giving RIS an unfair advantage. In this way, the control on the Integrator will control the ray depth.

- Other Integrator controls.
    - numLightSamples, numBxdfSamples, numIndirectSamples (and their companions when you're in "manual" mode). XPU does not support these. XPU takes only one sample for each of numBxdfSamples and numIndirectSamples. If your settings are higher than 1 for any of these, you will be giving an unfair advantage to XPU because RIS will be doing extra work. XPU does not have support for numLightSamples either, but has an internal heuristic about how many samples to take. So if you have more than one light in your scene, RIS and XPU will not be doing the same amount of work.
    - rouletteDepth, rouletteThreshold. XPU does not support Russian Roulette yet, so be sure to set the rouletteDepth to a sufficiently high value beyond the maxIndirectBounces so that RIS does not start cutting off ray paths that XPU would be tracing.
    - clampDepth, clampLuminance. XPU does not yet support illumination clamping, so be sure to set clampDepth and clampLuminance to a sufficiently high value so that RIS does not clamp any values that XPU would be splatting to the screen.
    - allowCaustics. RIS defaults this to 'off'. But XPU doesn't yet have the facilities to support the mechanism required to cut off caustics. So set this to 'on' for RIS.

- Light settings.
    - "int traceLightPaths" 1. Under some circumstances, area light sources are not visible in specular transmission/refractions in RIS. In XPU they are always visible. To make RIS similar to XPU (ie. area lights always visible), each light source must be marked with "int traceLightPaths" 1 in RIS.
    - "int visibleInRefractionPath" 1. XPU does not implement the required LPEs necessary to handle visibleInRefractionPath 0, so this should be left at the default setting of 1 in RIS.

- Specular mollification settings.
    - Option "shade" "roughnessmollification" 0.0 (default is 1.0). XPU has not yet fully implemented built-in curvature, so it's specular mollification capabilities are not quite as capable as RIS yet.

- Eliminate any usage of "shot" lighting features, such as light linking, light filters, or the light controls that give more nuance.

- Be aware of the other XPU limitations and eliminate as many of them as possible from your test.