Bxdf Evaluation Domain

Bxdfs can help integrators converge more quickly by providing hints about the domain over which they need to be integrated (the full domain being the entire sphere). This is done by the bxdf implementing a `RixBxdf::GetEvaluateDomain()` function that returns the appropriate `RixBXEvaluateDomai` value. This value informs the integrator about the domain on which `RixBxdf::EvaluateSample()` and `RixBxdf::EvaluateSamplesAtIndex()` would return non-zero values (i.e. a non-zero material response).

Note that the bxdf is still able to generate *discrete* samples over the whole spherical domain (through `RixBxdf::GenerateSample()`), independently of the hint returned. However, if the bxdf was generating *non-discrete* samples outside of the one returned by `RixBxdf::GetEvaluateDomain()`), this would mean there is an inconsistency between `RixBxdf::GenerateSample()` and `RixBxdf::EvaluateSample()` or `RixBxdf::EvaluateSamplesAtIndex()`, which is not good for correctness.

The various `RixBXEvaluateDomain` enum values (as defined in `RixBxdf.h`) are detailed below. The entire spherical domain is split into the *outside* and the *inside* hemispheres, defined with respect to the geometric normal of the surface (the geometric normal pointing toward the *outside*). Note that each value is a power of two, which means that the enum acts as a bit field and its values can (should) be OR-ed together.

`k_RixBXEmptyDomain = 0`

This indicates that either:

- the bxdf doesn't scatter light at all, but may still emit some (through `RixBxdf::EmitLocal()`)
- the bxdf only generates *discrete* samples (which by definition can never be evaluated) on the entire domain

`k_RixBXOutsideReflect = 1`

This indicates that the bxdf represents a surface that only reflects light from the outside. That is: `V` and `L` lie in the *outside* hemisphere.

This is useful for closed opaque objects, whose interior will never be seen.

`k_RixBXOutsideTransmit = 2`

This indicates that the bxdf represents a surface that allows light to flow from the *outside* toward the *inside*. That is: `L` lies in the *outside* hemisphere, while `V` lies in the *inside* hemisphere.

Note that if used for a closed object, this would allow light to penetrate inside the object, but not exit it. This would only make sense if the camera lies inside the object.

`k_RixBXInsideReflect = 4`

This indicates that the bxdf represents a surface that only reflects light from the inside. That is: `V` and `L` lie in the *inside* hemisphere.

`k_RixBXInsideTransmit = 8`

This indicates that the bxdf represents a surface that allows light to flow from the inside toward the outside. That is: `L` lies in the *inside* hemisphere, while `V` l ies in the *outside* hemisphere.

Note that if used for a closed object, this would allow light to exit the object, but not go through it. This would only make sense if a light is placed inside the object.

`k_RixBXVolume = 16`

This indicates that the bxdf represents a volumetric scattering event. This is different from using all of the above, since in the volumetric case, there is no surface cosine term to consider during integration.

Common Cases

`RixBXEvaluateDomain` also defines some (composed) values corresponding to the most common cases.

`k_RixBXReflect = (k_RixBXInsideReflect | k_RixBXOutsideReflect)`

This is useful for open surfaces with both the 'outside' and 'inside' being reflective.

`k_RixBXTransmit = (k_RixBXInsideTransmit | k_RixBXOutsideTransmit)`

This is useful for open surfaces with both the 'outside' and 'inside' being transmissive. This is also useful for closed objects that allow light to pass through them (this may require additional indirect bounces)

`k_RixBXBoth = (k_RixBXReflect | k_RixBXTransmit)`

This is the most generic type of surface, allowing light to always be reflected or transmitted independently of the domains.

`k_RixBXOutside = (k_RixBXOutsideReflect | k_RixBXOutsideTransmit)`

This describse surfaces that will only consider light coming from the outside.

`k_RixBXInside = (k_RixBXInsideReflect | k_RixBXInsideTransmit)`

This describes surfaces that will only consider light coming from the inside.

Additional Considerations

Reciprocity

Using only `k_RixBXOutsideReflect` and `k_RixBXInsideReflect` doesn't prevent the bxdf from being reciprocal.

However, if only one of the bits corresponding to `k_RixOutsideTransmit` or `k_RixInsideTransmit` is set, this indicates that the bxdf only allows the light to go through in a particular direction, and as such breaks the reciprocity principle. This may lead to non-intuive, non-plausible renders, and break bidirectional algorithms.

In particular, when a bxdf is used by bidirectional integrators, the meaning for `V` and `L` for all calls to `GenerateSample()`, `EvaluateSample()` and `EvaluateSamplesAtIndex()` may be flipped. For example, when generating photons, `GenerateSample()` will be called with `V` pointing toward the direction of the *light subpath.* In this case, non-reciprocal bxdfs needs to make sure they account for this when computing material response. This case can be detected by checking the value of `RixShadingContext::scTraits.lightPath`. If `lightPath` is `1`, then `V` points towards the light subpath.

How to chose a bxdf `RixBXEvaluateDomain`

First, note that using `k_RixBXBoth` (or `k_RixBXVolume`) is the best way to ensure your bxdf works properly, as a reference. When choosing a different evaluation domain (for evaluation efficiency), you need to make sure the rendered image doesn't change – if it does change, that could mean you are artificially restricting the evaluation domain of your bxdf, and potentially introducing rendering artifacts.

Most of the time, when dealing with closed and non-transmissive (i.e. translucent) objects, you should be able to restrict the evaluation domain (using `k_RixBXOutsideReflect`). This would allow the integrator to skip processing light samples that lie behind the surface (on the other side of the closed object itself for example), reducing the number of bxdf evaluations required, as well as the number of transmission rays that need to be cast.

When dealing with transmissive (i.e. translucent) objects, and only expect translucent scattering to come from lights placed inside the objects, you may be able to use `k_RixBXInsideTransmit`, without `k_RixBXOutsideTransmit`.