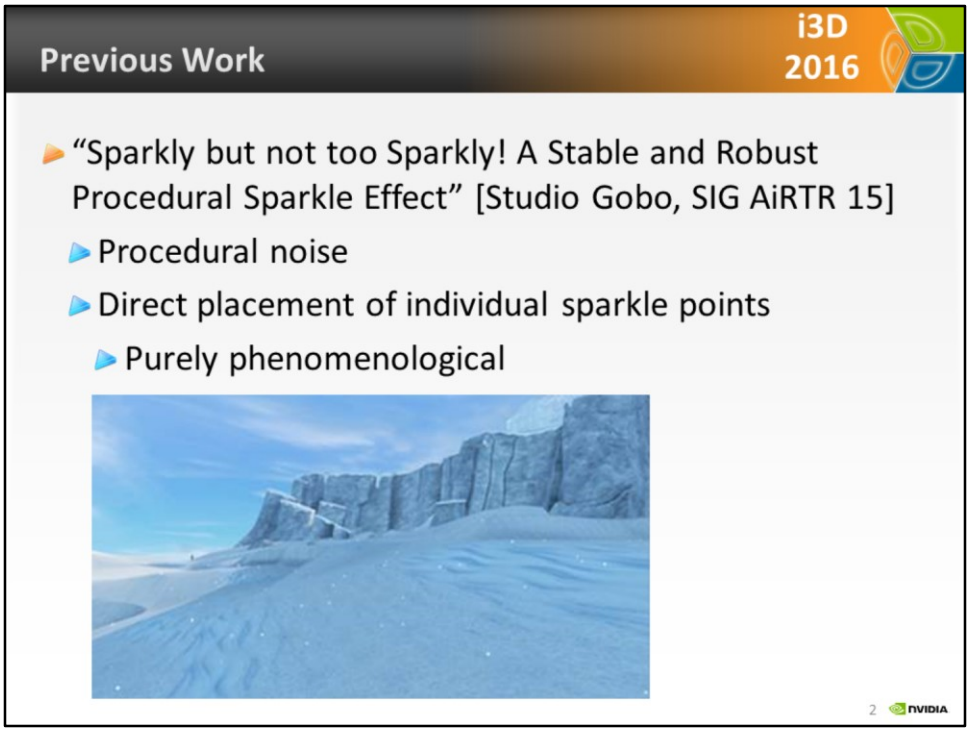
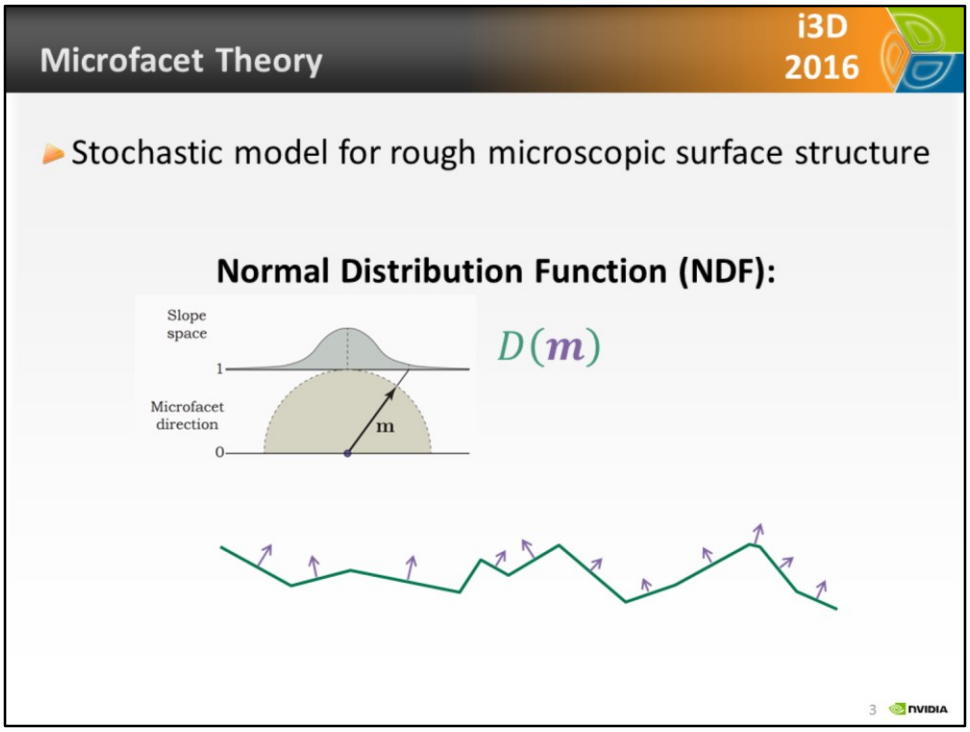
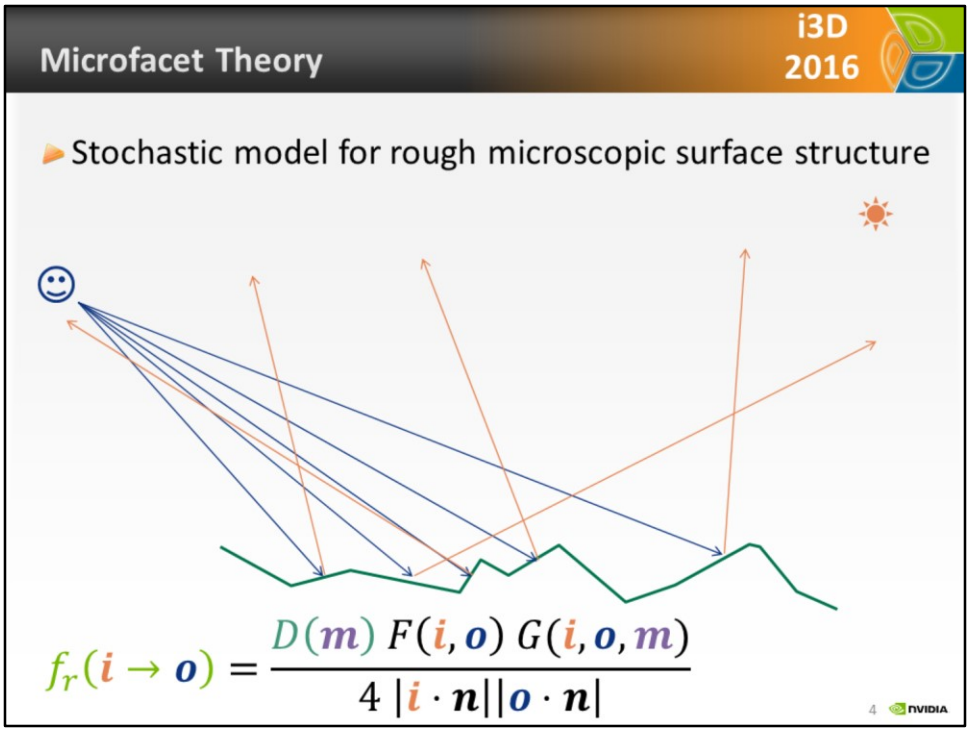
Materials with scale-dependent appearance come up in various contexts and have been subject to intensive research. Seen from a distance, pretty much all structured materials degenerate to some kind of softer appearance. A lot of focus has been put on the proper reproduction of this phenomenon. However, in-between close-up views and wide-angle views, the characteristic appearance of some materials is defined by another phenomenon, which is glinting, caused by a non-uniform, non-smooth distribution of small structural detail across material surfaces. Such materials include snow, glitter and brushed metal, as can be seen on this slide.



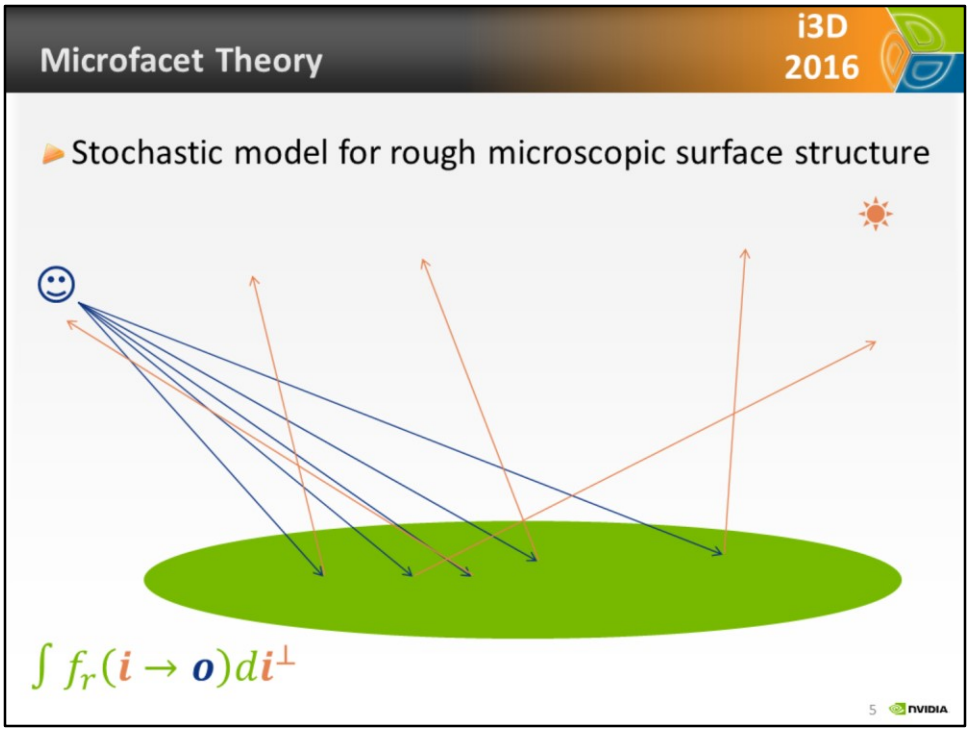
In real-time, previous attempts at glinting materials have used much simpler, non-physically-inspired approaches.
For example, in their ARTR Siggraph talk last year, Studio Gobo presented a simple phenomenological approach that directly places individual sparkle points on the geometry, using scene-encompassing 3D procedural noise grids for seeding.
However, this approach only generates rather sparse glints at predefined scales.
Generally, stable high-density sparkling is challenging, because it consists of many sub-pixel-size specular highlights that need to be placed stably and to be anti-aliased both temporally and spatially to prevent unbearable flickering.
This work tries to fill this gap in a real-time context, by procedurally generating stable, anti-aliased glints caused by such microdetail in a physically-inspired way.



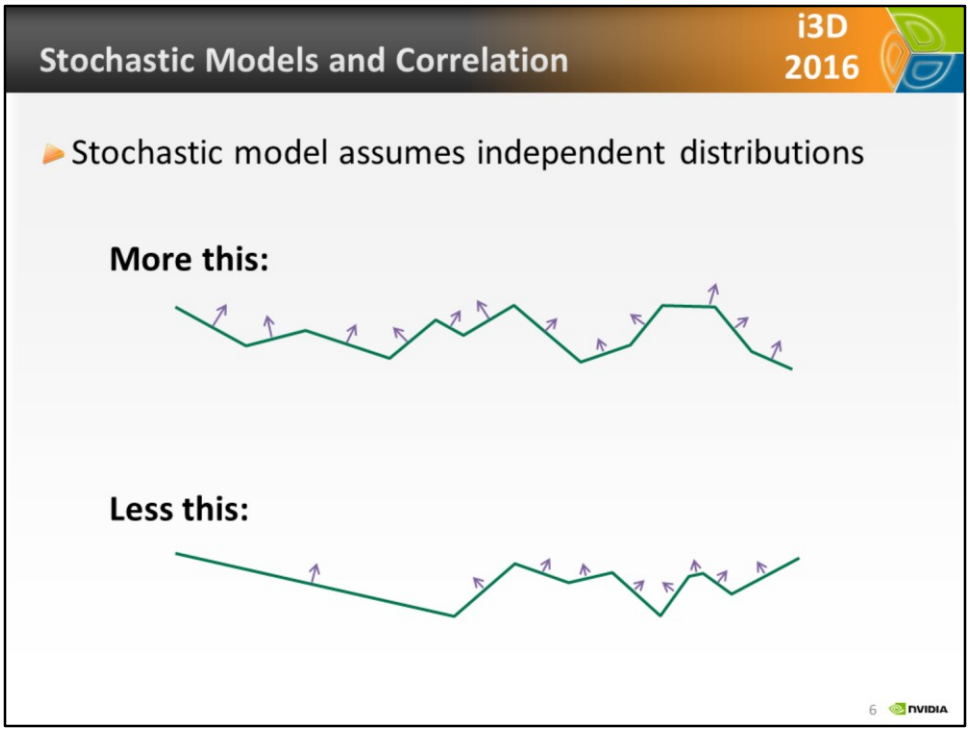
Generally, in physically-based rendering the macroscopic appearance caused by microscopic surface structure is modelled stochastically:
A so-called normal distribution function defines the distribution of microscopic surface orientations.
This distribution basically defines the fraction of the macroscopic surface area occupied by microscopic patches for each possible orientation.
Most common distributions are defined in slope space and then transformed to the space of orientations and corresponding surface area by a change of variables.



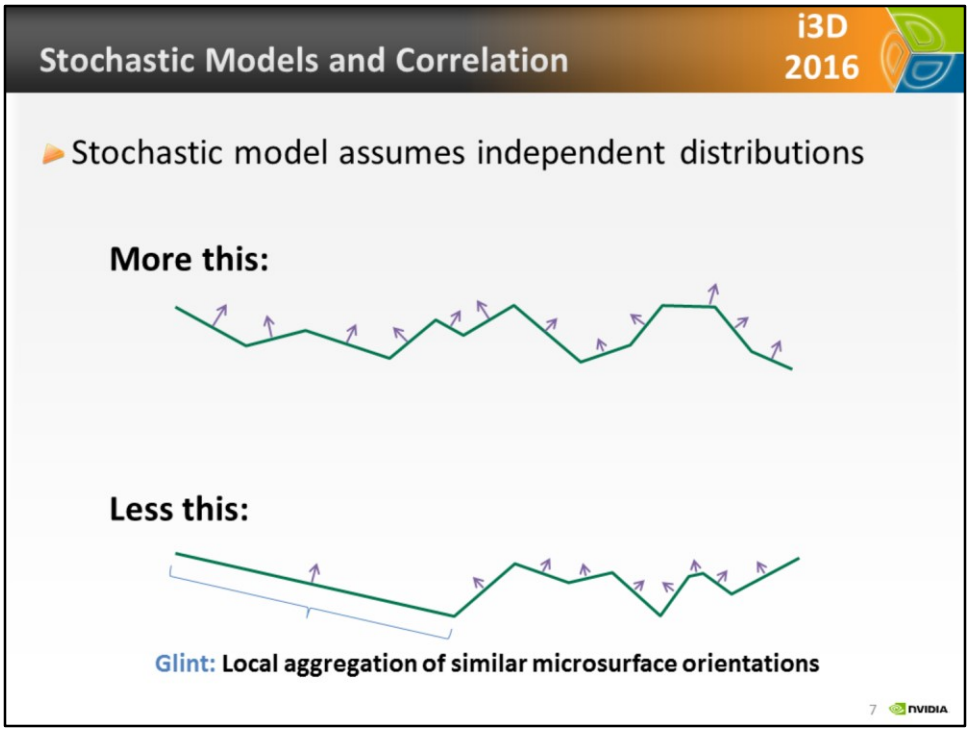
Given a viewing direction, the NDF can be converted to a corresponding distribution of incident light directions that defines how much light from each direction is reflected towards the viewer.
Thus, the NDF defines macroscopic surface appearance.
The corresponding BSDF, often called microfacet BSDF, can be seen at the bottom of the slide.
//
Here, besides the NDF D itself, converted to projected microfacet area,
F is an optional Fresnel term.
G is a microgeometry term that enforces conservation of energy and visible surface area.
The denominator basically computes the visible surface area as seen from the view and light directions, respectively, and includes a partially cancelled-out change of variables from surface orientation to light direction.



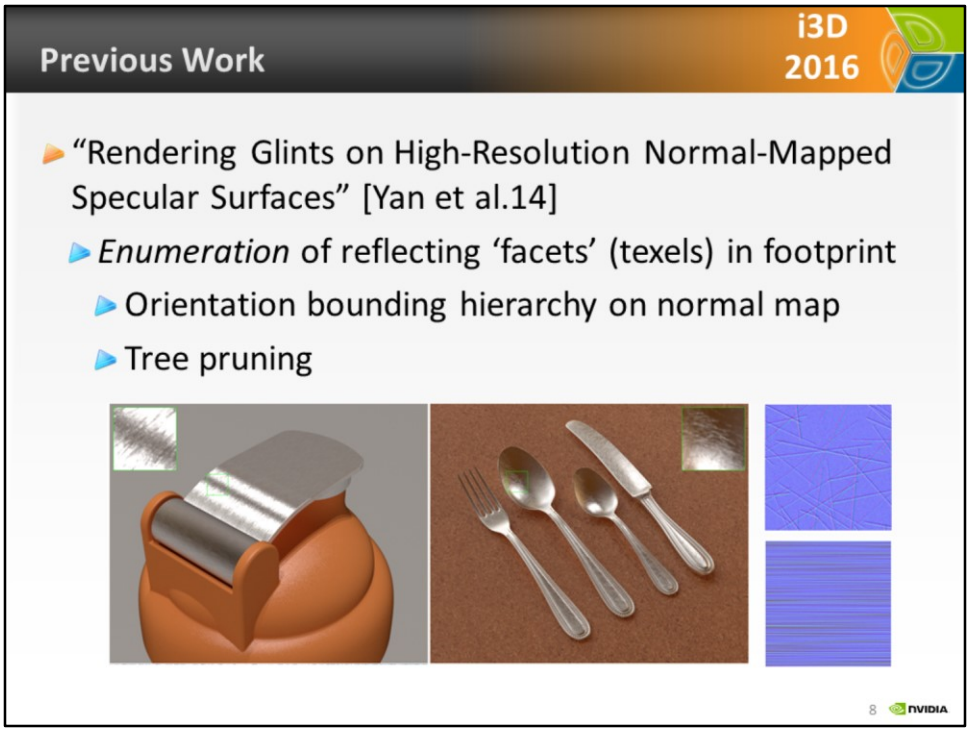
The NDF, and with it the BSDF, basically abstract the precise microsurface away, assuming an independent and continuous distribution of microfacet orientations at all scales across the entire surface.
Given such a distribution, the reflectance integral computes the corresponding average light reflected from each direction for every point on the surface.



Graphically, such stochastic models correspond more to the microsurface structure seen at the top, and less to the microstructure seen at the bottom:
At the top, every portion of the microsurface contains microfacets with many different orientations.
However, realistically, especially for specific materials such as snow, distribution of orientations can vary across different portions of the same material.

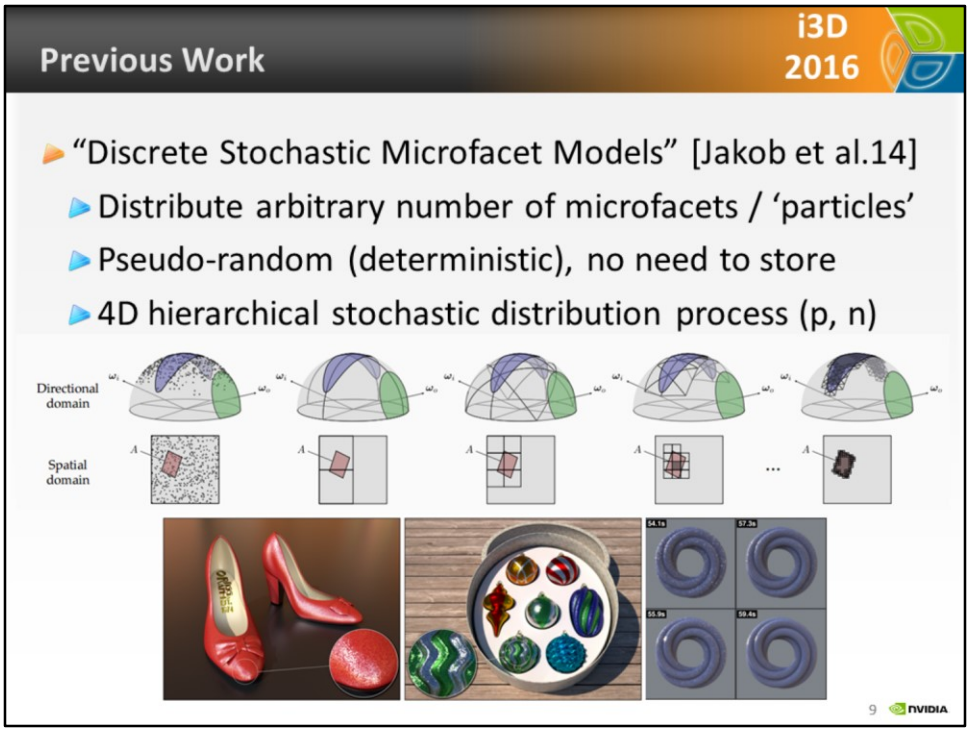


In some places, microfacets of all orientations may be occur equally, whereas in other places, specific orientations might occur much more often than others.
It is this kind of correlation or clustering that is responsible for a glinting appearance, but that is neglected by the stochastic models generally used.



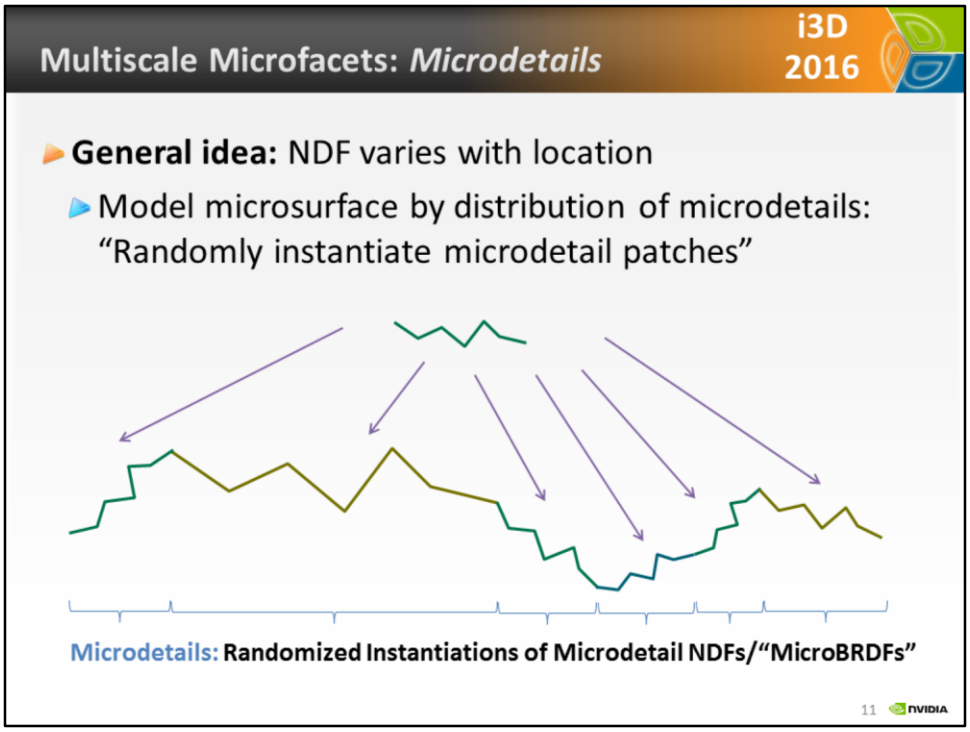
One way of introducing such correlations is by simply defining the entire microsurface explicitly.
Yat et al. do this using high-resolution normal maps that effectively store the orientation of every microfacet.
Inside each pixel footprint, they basically count all relevant microfacets by enumerating all normal map texels covered by each pixel.
They improve efficiency by adding an acceleration data structure and tree pruning.
While this gives high-quality results where such detailed data is available,
- It requires the generation and storage of the entire micro-geometry.
- Moreover, the number of orientations per texel is inherently limited, in this case to one, greatly limiting the glint density for reasonable computation times and memory usage.

In any case, enumerating microflakes at reasonable densities would be way to expensive for real-time applications.



Another way of introducing correlation is by modelling spatially-varying, non-smooth distributions of microfacets stochastically.
Jakob et al. do this by implicitly distributing an arbitrary discrete number of microflakes using a stochastic process.
The exact distribution is defined by a pseudo-random, but deterministically seeded process, which eliminates storage requirements.
The basic idea is to first compute the discrete number of microflakes on a larger surface patch, using a material-specific microflake density.
Then, both the surface patch and the space of orientations is sub-divided hierarchically, in order to precisely match the subspace covered by each pixel.
The sum of the reflecting particles in all subspaces covered by the pixel and potential reflection orientations determines the amount of light reflected for each light direction.
Note that this approach directly allows an arbitrary number of microflakes/glints per footprint, making it a much better fit for highly gliny materials such as snow.

However, doing the 4D hierarchy traversal for every pixel in order to distribute microflakes is also too expensive for real-time.
Therefore, based on the ideas of “Stochastic Microfacet Models”, we will now derive a stochastic biscale microfacet model that allows for a simplified real-time evaluation.

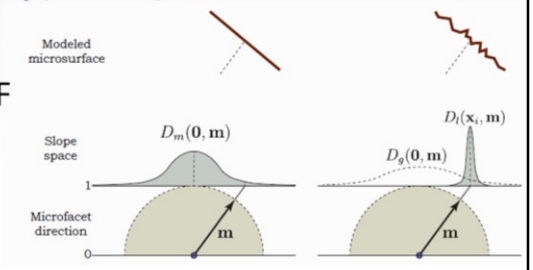


In our model, the basic goal remains the same: In close-ups, the NDF needs to vary with respect to locaton, and with increasing distance, the NDFs need to converge to the global/macroscropic NDF.
However, to get more control over both glint and overall material appearance, we actually compose the global NDF out of local NDFs:
Graphically, we consider macroscopic surfaces to be made up of patches with a certain local NDF. We call these patches microdetails.
We then randomly instantiate microdetail patches using another mesoscale distribution of microdetails (MDDF).
The aggregation of all microdetails forms the global microsurface. Note that microdetail NDFs are generally narrower, i.e. the surface of one microdetail is more flat than in the overall macroscopic surface.
This accounts for the fact that glints are caused by mesoscopically more clustered, locally similar microfacets.

Relationship NDF, MDDF, MNDF

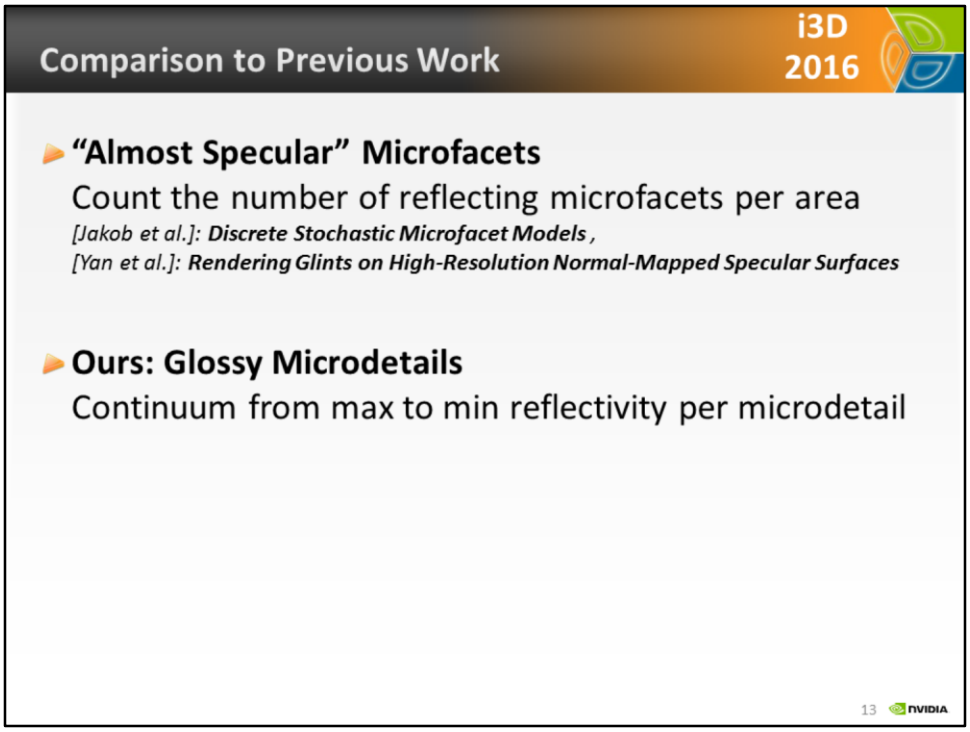
i3D 2016

- Each microdetail represents a *local* microsurface with **Microdetail NDF (MNDF)**: $D_l(\boldsymbol{m})$
- Microdetails $\boldsymbol{x}$ are randomly oriented according to **Microdetail Distribution Function (MDDF)**: $D_\mu(\boldsymbol{x})$
- The global microsurface consists of all microdetails with **NDF**: $D_g(\boldsymbol{m}) = \int D_\mu(\boldsymbol{x}) D_l(\boldsymbol{m} - \boldsymbol{x})\, d\boldsymbol{x}$
- Can decompose *any* NDF into local and global DFs to introduce correlated clusters/glints!

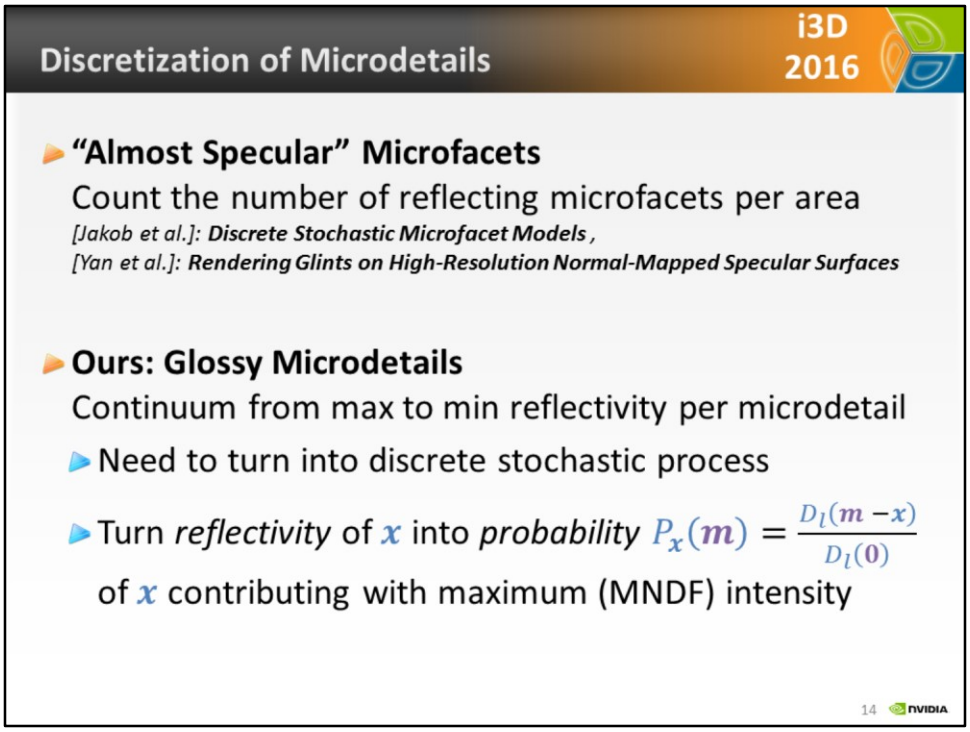


12 NVIDIA

Recap: Our surfaces are made out of randomly instantiated microdetails, which are local microsurface patches described by the microdetail NDF D_l
The randomized instantiation of microdetails follows the microdetail distribution functon D_\mu
[[This means that we can first draw a microdetail orientation from D_mu, and then sample a microfacet direction from the local NDF D_l centered around this orientation.]]
The corresponding global microsurface is then made up of all microdetails, and by design, its NDF D_g is simply the convolution of the two distributions D_\mu and D_l.
Conversely, this means that we can decompose any single-scale macroscopic NDF into a biscale NDF:
-> For example, choosing a microdetail MDF D_l to control the appearance of individual glints, the MDDF is implicitly defined in such a way that the convolution of the two equals the desired original macroscopic NDF and thus appearance.
//
MNDF: Micro-NDF / Micro-Normal Distribution Function
MDDF: Microdetail Distribution Function
(Global) NDF: (Global) Normal Distribution Function



This model slightly adds to the models used by previous work:
Both in Jabob et al.'s and Yan et al.'s paper, there is no notion of microdetails, but rather, both assume such microdetails to be made up of exactly one almost specular microfacet.
//
This is problematic in two ways: For one, their microfacets also cannot be purely specular, since then discretization always nulls the set of reflecting microfacets per area.
Moreover, both solutions counteract this problem by choice of rather arbitrary parameters that effectively add a small amount of roughness to microfacets, changing the macroscopic appearance in a rather uncontrolled way.
Neither of them gives true control over the NDF and thus appearance of individual glints, while also stably maintaining the macroscopic appearance of the material.

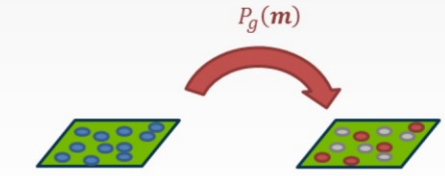


In our work, microdetails can be controlled to be arbitrary glossy without affecting macroscopic appearance.
This makes a discretization of the problem that allows for discrete counting of reflecting microdetails slightly harder:
Every microdetail has a full-fledged NDF with infinite support in the space of slopes, which means that all microdetails always reflect at least some small fraction of the light towards the viewer.
To solve this issue, we choose to convert the continuous reflectivity of each individual microdetail into a discrete probability for each microdetail to either contribute with its maximum reflectivity defined by D_l(0) or not at all.
This probability P_x is straight-forward to compute as D_l of the actual reflection half vector (transformed to the local space of the microdetail), divided by its local peak intensity.
It is clear that the expected value of such a stochastic view still equals the actual local reflectivity: P_x multiplied by D_l(0) equals D_l(m-x)

Discrete Biscale Model Evaluation

i3D 2016

- Probability $P_x(m) = \frac{D_l(m - x)}{D_l(0)}$ of x contributing with max intensity
- Total probability mass $P_g(m) = \int D_\mu(x)\, P_x(m)\, dx = \frac{D_g(m)}{D_l(0)}$
- Expected number of contributing microdetails: $P_g(m)\, N_\mu\, A$



$N_{total} = N_\mu\, A$ $E[N_{contrib}] = P_g(m)\, N_\mu\, A$

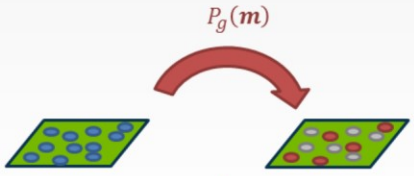
15 NVIDIA

Using this approach, the overall expected probability mass P_g of all reflecting microdetails instantiated by the macroscopic surface can again be computed using a simple convolution,
and results in a nice and simple quotient of the macroscopic NDF, divided by the peak intensity of the microdetail NDF.
This probability mass can be directly used to sample a discrete number of contributing microdetails,
e.g. using a stochastic process that partitions microdetails into contributing and non-contributing ones.
//
The expected number of reflecting microdetails is simply the probability mass P_g times microdetail density N_\mu times surface area A.

Discrete Biscale Model Evaluation

i3D 2016

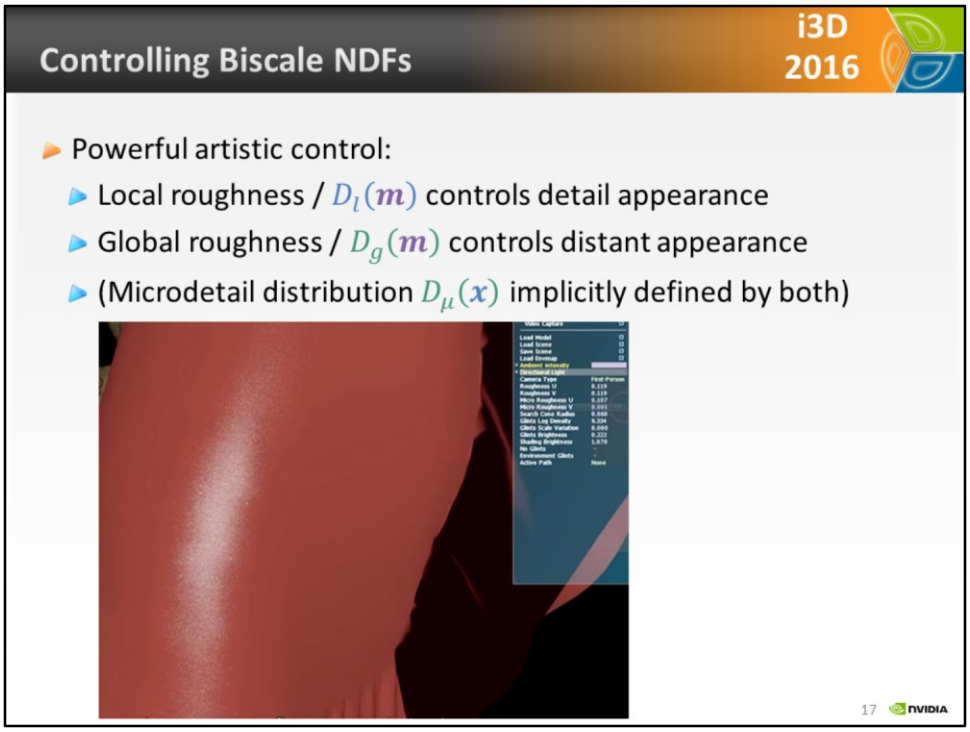
- Probability $P_x(m) = \frac{D_l(m - x)}{D_l(0)}$ of x contributing with max intensity
- Total probability mass $P_g(m) = \int D_\mu(x)\, P_x(m)\, dx = \frac{D_g(m)}{D_l(0)}$
- Expected number of contributing microdetails: $P_g(m)\, N_\mu\, A$



- Multiply contributing fraction $N_{contrib} / E[N_{contrib}]$ by $D_l(0)$:
  - Locally has dynamic range MNDF $D_l(m)$ (zoomed in)
  - Expected value equals global NDF $D_g(m)$ (zoomed out)

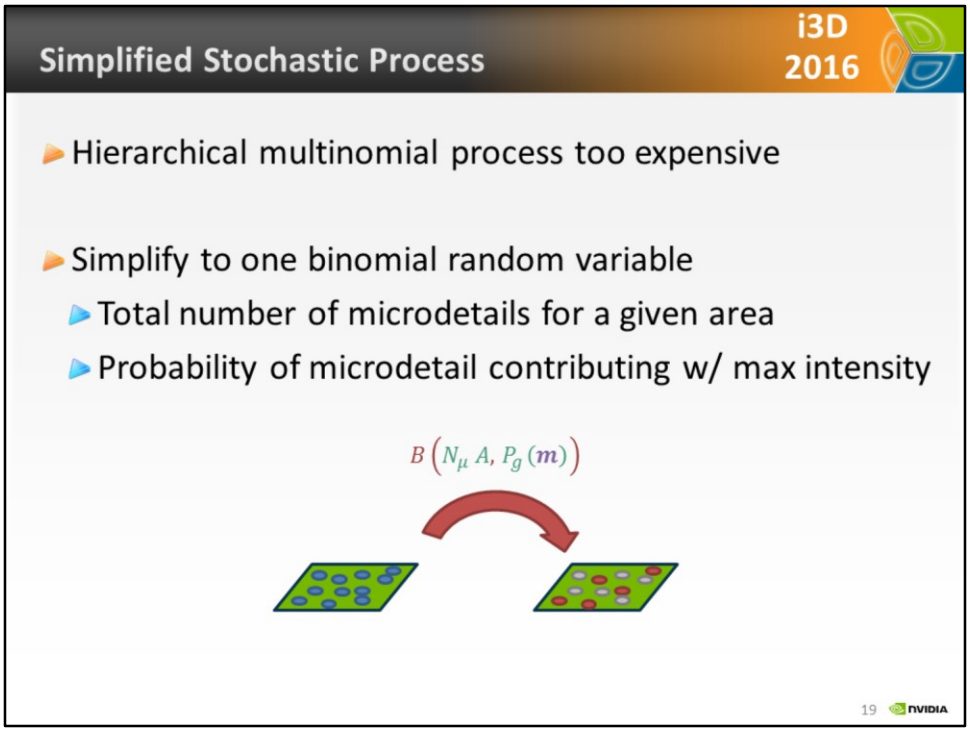
16 NVIDIA

The ratio between the sampled and the expected number of reflecting microdetails can be multiplied by D_l(0) to convert back to actual reflectivity.
It is clear that this locally has the dynamic range of the microdetail NDF D_l.
And, looking at the definition of P_g, it is also clear that, as we zoom out and the counted number of reflecting microfacets approaches the expected value,
the two D_l(0)s cancel out and the NDF converges to the global NDF D_g,
retaintaining macroscopic surface appearance.



This model also makes for good artistic controls: Using Beckmann distributions, a global roughness parameter directly defines distant appearance, whereas a local microdetail roughness appearance directly defines detail appearance, each without affecting the other.
As mentioned before, the microdetail distribution function D_mu does not need to be defined explicitly, since it is fully defined by the convolution equation.

So far, we have looked into the theory behind our approach. In order to make glints practical in real-time, we need apply a few additional simplifcations to the algorithms described by previous work.

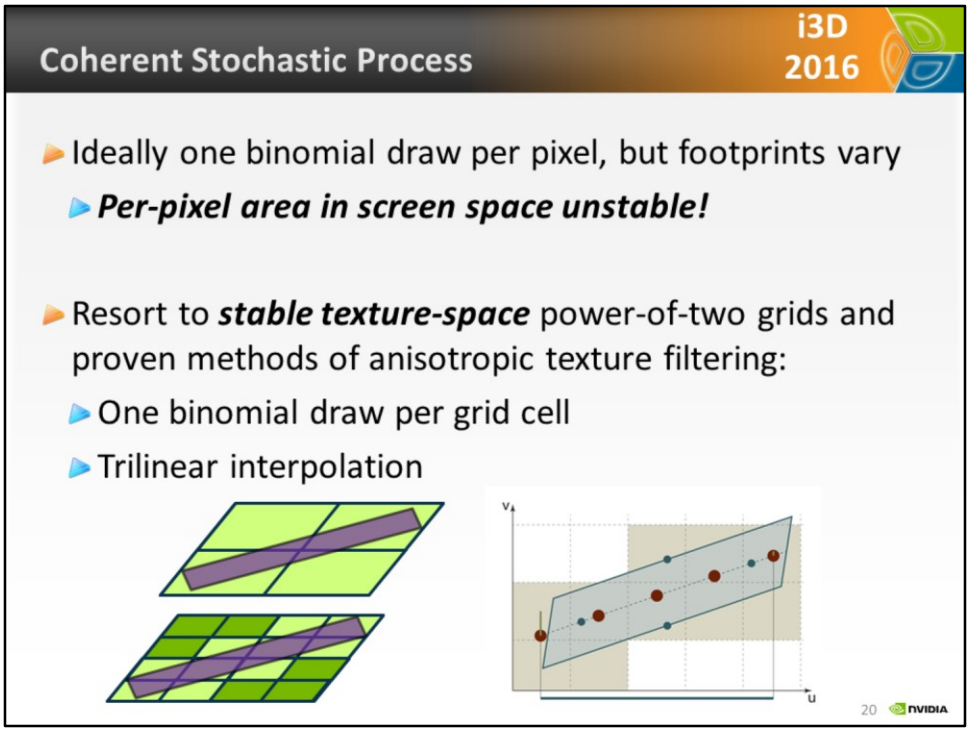


Unfortunately, the hierarchical multinomial particle counting process introduced by Wenzel et al. is way too expansive to perform per pixel in a real-time budget. Therefore, we radically simplify the counting process to one binomial random counting variable:

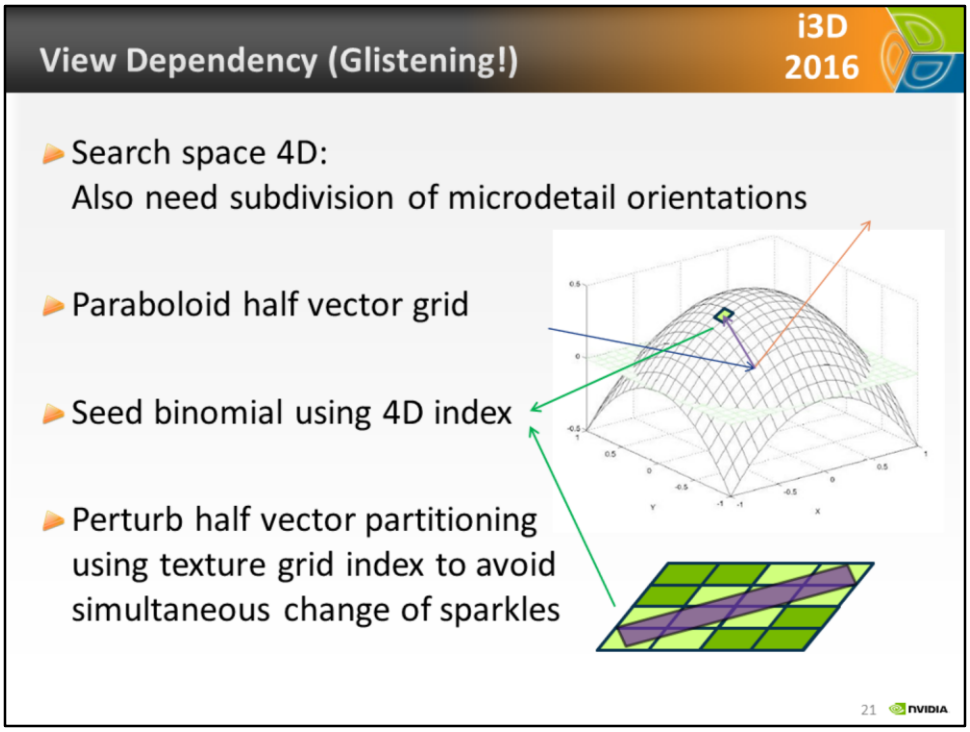
- We note that assuming a locally constant microdetail density, it is easy to compute the total number of microdetails for a given area
- We have also seen how to compute a discrete probability mass for the fraction of microdetails reflecting with their respective maximum intensities

Following these observations, we assume the chance for a microdetail to be reflecting to be equal for every microdetail.

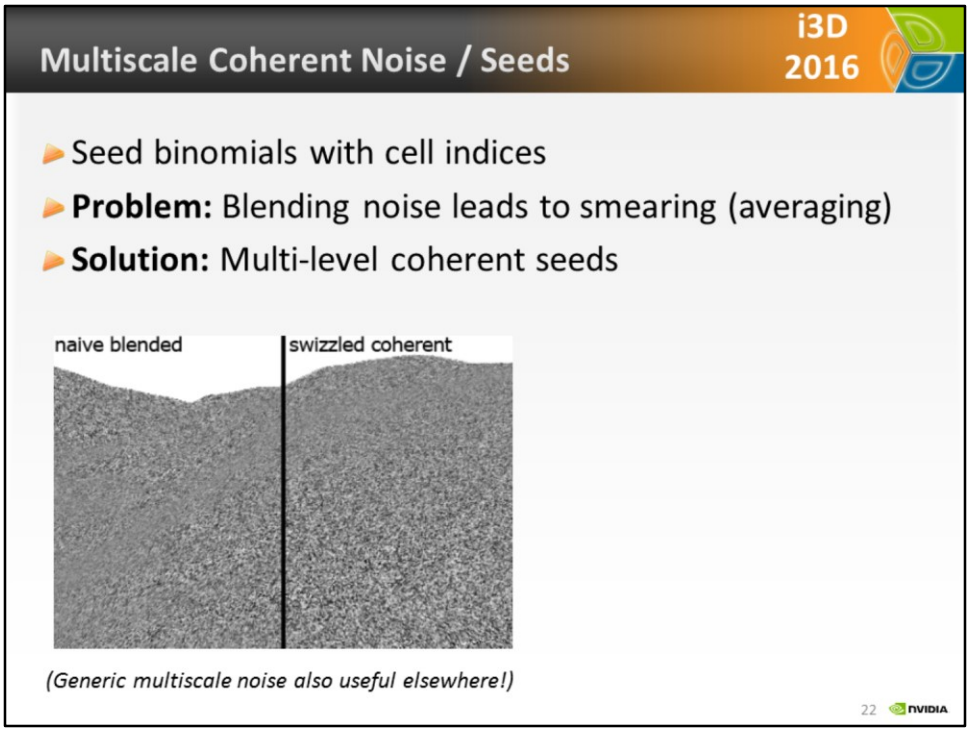
This leaves us with one simple binomial distribution of microdetail counts, that partitions a total number of microdetails per area into reflecting and non-reflecting ones.



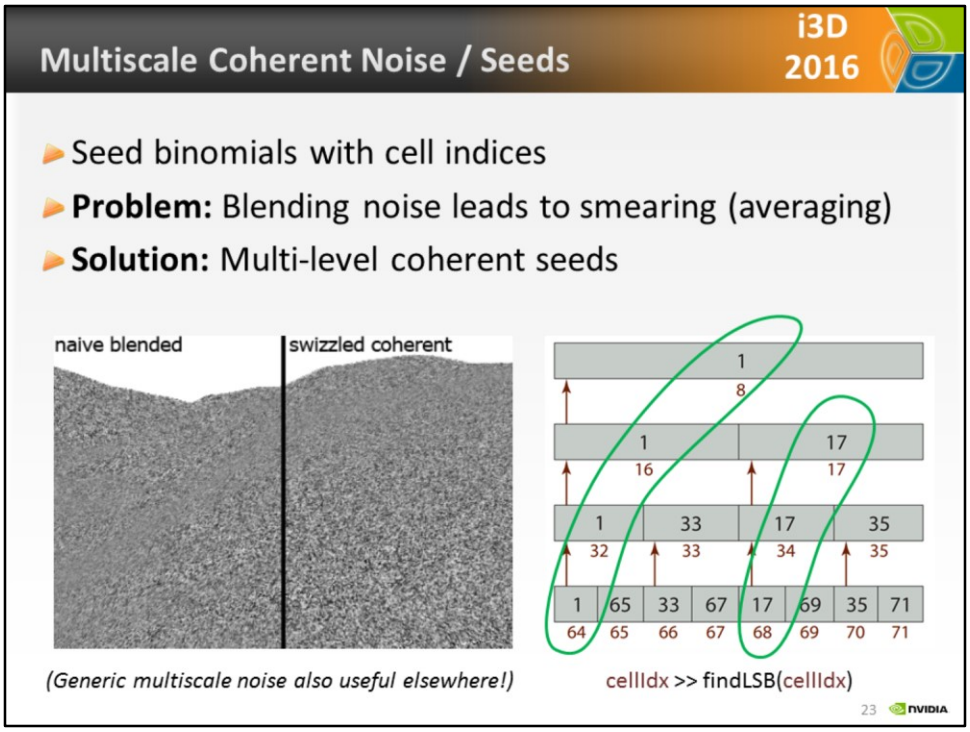
We could directly apply this model to the footprint of each pixel, computing the total area covered by each pixel from its projected area.
However, this would lead to highly unstable results, since the area of each pixel varies greatly in-between frames.
Besides, it is unclear how to stably and coherently seed individual pixels across multiple frames.
We therefore resort to using a stable nested texture-space power-of-2 grid instead, which also enables us to leverage proven methods of anisotropic texture filtering.
For each texture-space grid cell, we can easily compute a total number of contained microdetails and a stable seed value, allowing for a direct binomial draw of reflecting microfacet counts.
We then use anisotropic filtering and trilinear interpolation of the results.
//For each cell inside each pixel footprint and the two nested grid levels matching its extent most closely,



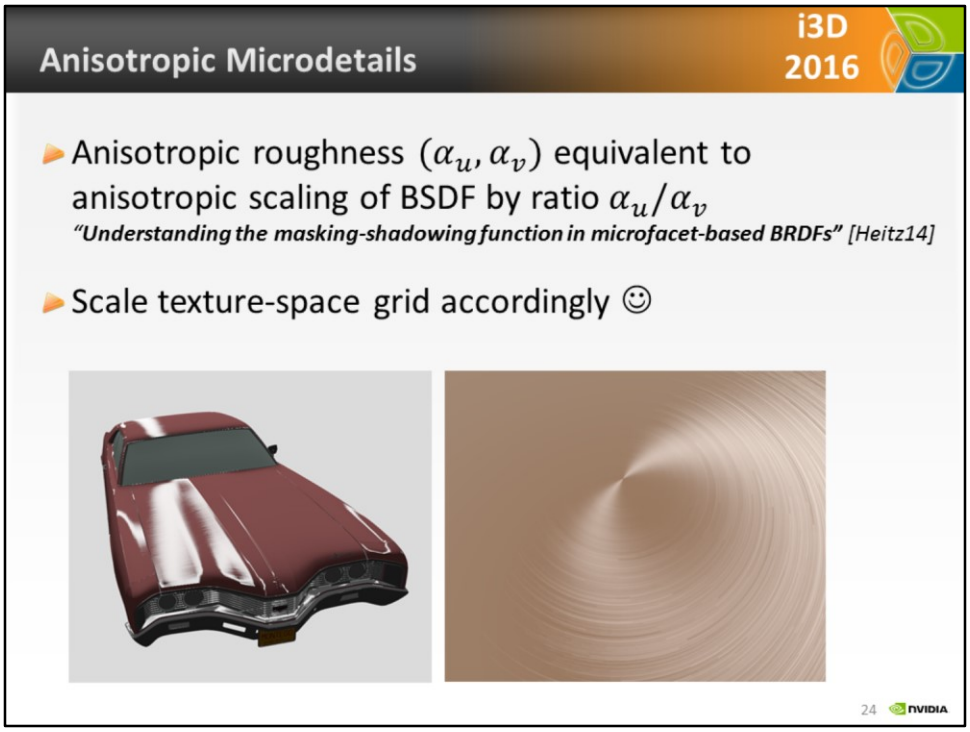
The search space for reflecting microdetails is actually 4D, since the count also varies for different microdetail orientations.
We chose to discretize halfvectors on a 2D regular paraboloid grid, which approximately preserves solid angle.
We then use a seed depending on both the texture grid cell index and the halfvector grid index in order to draw binomial random variables.
One important detail here is to perturb the halfvector partitioning for every texture grid cell and thus pixel: By applying a random fractional offset to the halfvector grid index, we randomize the time at which the halfvector grid index changes and prevent sparkles from changing all at once.



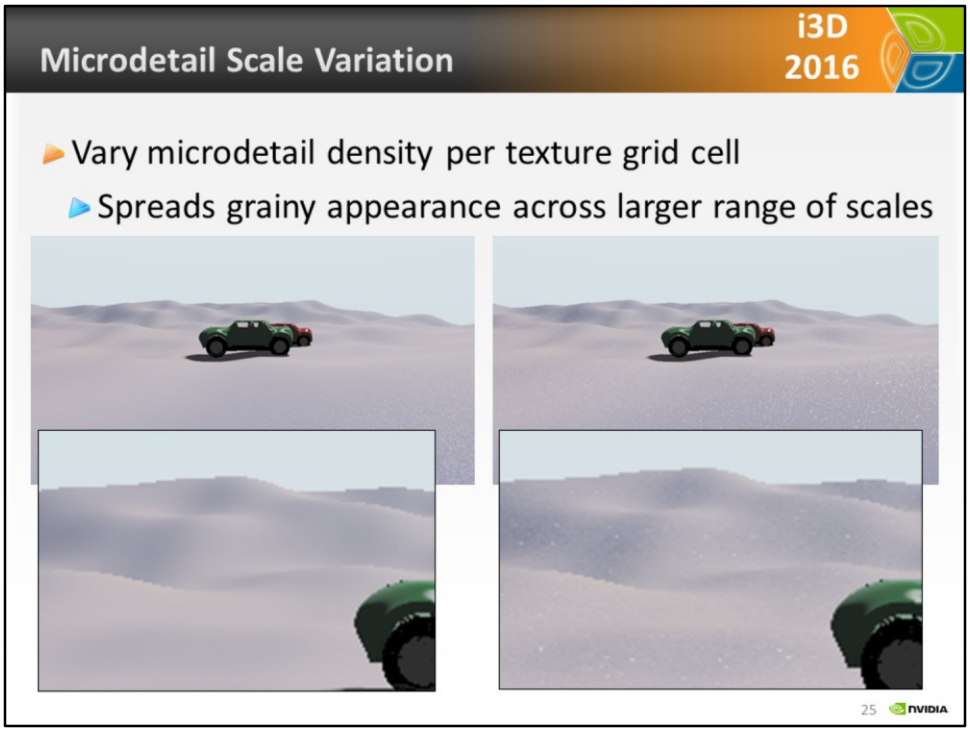
A final important detail is related to the blending between the results of different levels of the nested power-of-two texture grids.
Looking at a simpler test case that simply blends uniform random noise of two grid levels, we can see why:
Blending multiple random variables always brings the result closer to the expected value, resulting in noticeable smearing in transition areas.
We work around this issue by introducing a deliberate correlation between grid levels, ensuring that each coarser grid level always contains a representative of the finer level.



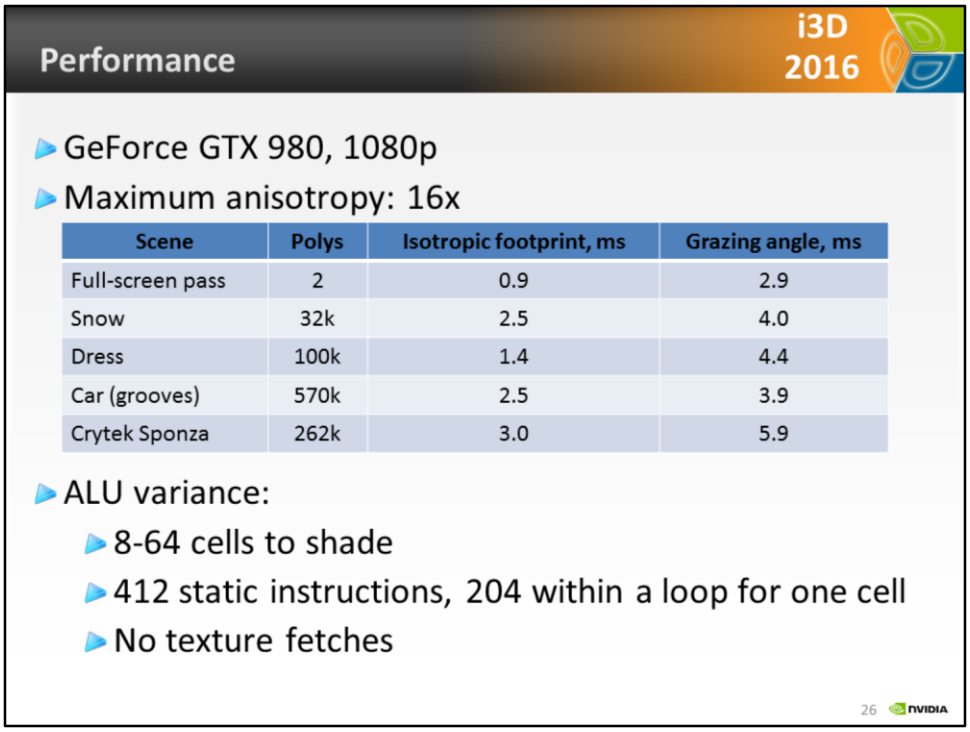
The simplest working implementation that we found was to always shift grid cell indices such that the lowest bit is set.
In a 1D example, we can see that this ensures that the seed of the left-most cell in each tuple always re-appears in the next coarser level.
In the rendered result, we can see that this coherent swizzling nicely hides the smeared-out transition areas.
Of course, this somewhat breaks the independent uniform random noise, but we found results to de-correlate quickly, with a starting offset of around 100.



As pointed out by Eric Heitz in "Understanding the masking-shadowing function in microfacet BRDFs", a BRDF with anisotropic roughness is equivalent to a non-uniformly scaled surface with isotropic roughness.
As it turns out, we can indeed scale the texture-space grid accordingly and thus render anisotropic glints that follow the anisotropic roughness of the microdetail distribution function.
Thus, we can also render anisotropic glints, such as observable in brushed metal and other brushed materials.



One final effective tweak in order to achieve interesting appearance across multiple scales is to also randomize the microdetail density per texture grid cell.
This follows the intuition that in materials such as snow, we can find crystals and cracks of various sizes. Thus, the number of glints decreases more slowly with distance.



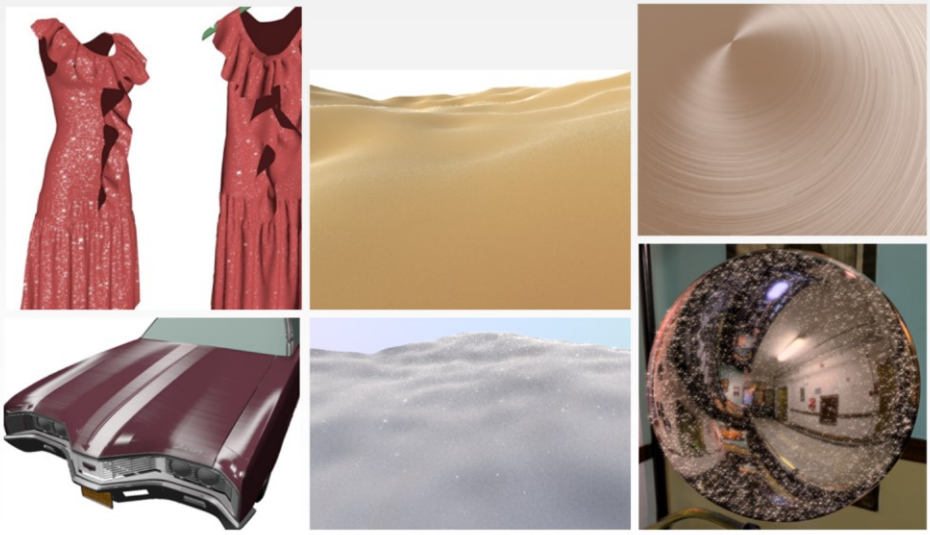
As to performance, we ran a few tests on an NVIDIA Geforce 980 at 1080p. We limited the anisotropy of the pixel footprint to 16x, which gives acceptable results for all viewing angles. As you can see, the number of texture grid cells that need to be iterated for sampling the number of reflecting microdetails greatly affects performance. The ratio between steep and grazing angles is approximately a factor of two.

The computational load varies between 8-64 texture grid cells to shade, while no memory accesses are needed. The compiled shader has about 412 instructions, of which half need to be executed for every texture grid cell.

We have now published a shadertoy source code example that provides more insight into the implementation details and that you can play around with.

Variety of Materials

i3D 2016



Thank you!

i3D 2016

- Questions?
- Contact
  - **Tobias Zirr**
    tobias.zirr@alphanew.net
    Twitter: @alphanew
  - **Anton Kaplanyan**
    kaplanyan@gmail.com
    Twitter: @kaplanyan