SIGGRAPH 2015

Xroads of Discovery

SIGGRAPH 2015
Xroads of Discovery

The 42nd International Conference and Exhibition on Computer Graphics and Interactive Techniques

Optimizing PBR

Renaldas Zioma
Unity Technologies

Talk Overview

- PBR challenges on Mobile
- What hardware are we optimizing for?
- Faster BRDF
- Linear/Gamma
- Environment Reflections

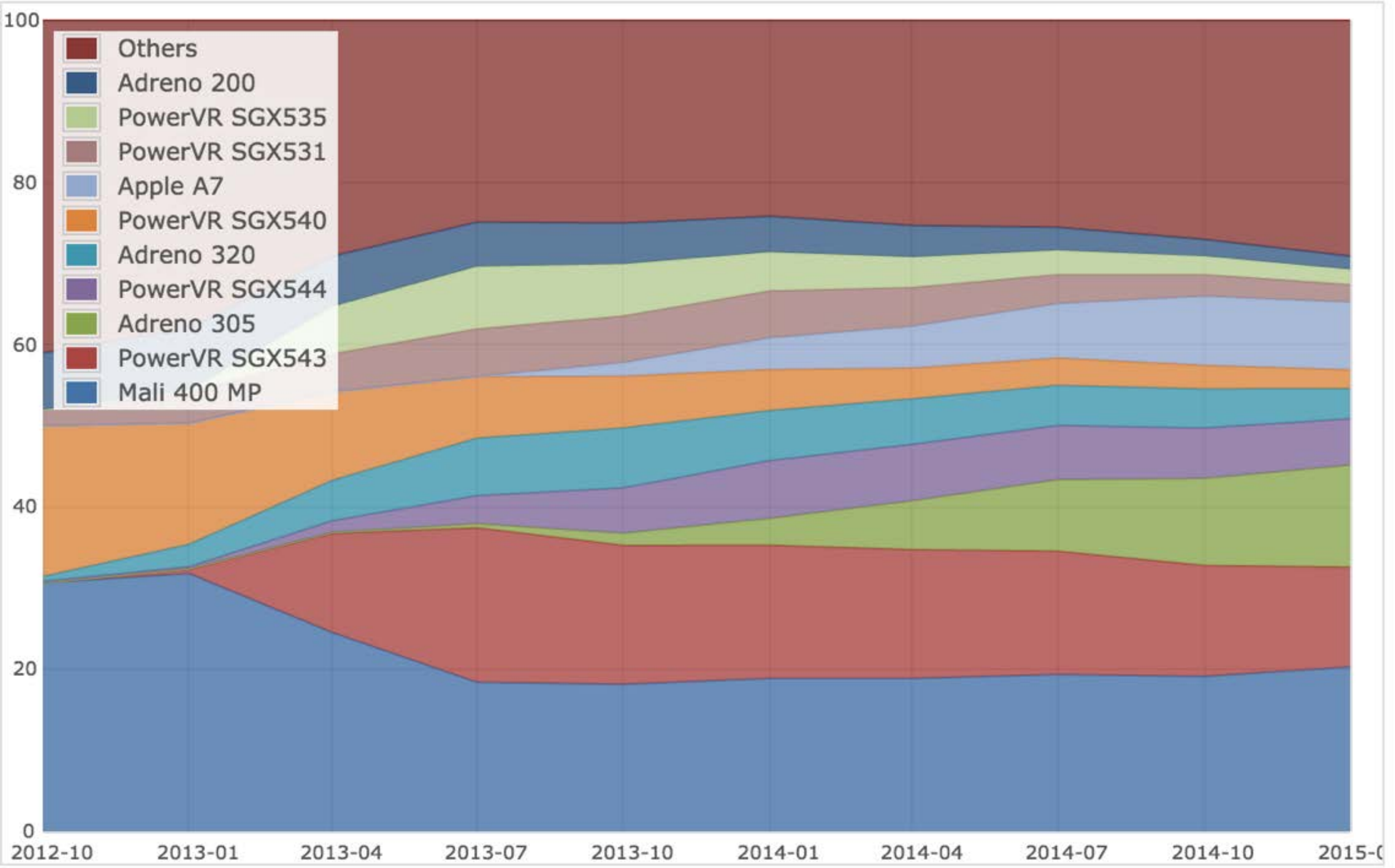
PBR challenges on Mobile

- Performance
- Many GPUs, many architectures, many peculiarities
- Gamma/Linear workflows
- Lack of high quality texture compression formats
  - ASTC - light at the end of the tunnel

PBR challenges on Mobile

- Shader compilers are still not as good as on PC
- Scalar (more recent) vs vector pipeline
- texCUBElod
- FP32 vs FP16 precision
- Lots of shader variations!

Optimization Target



based on # of apps running Unity

Performance

	PowerVR	NVIDIA	Qualcomm	ARM
4 ~ 8 GFlops **0.2 ~ 1 GP/s**	**SGX535** iPad, iPhone4	**Tegra2**	**Adreno2xx**	**Mali400 MPx** SGS3 (I9300) SGS2 (I9100)
16 GFlops **2 ~ 3 GP/s**	**SGX54x** iPad2/3, iPhone4s, iPhone5	**Tegra3**	**Adreno305** SGS4 mini (I9195)	
100 GFlops **4 GP/s**	**G6x30** iPadAir, iPhone5s	**Tegra4**	**Adreno3x0** Nexus 4, Nexus 5	MaliT628
250 GFlops **4 ~ 8 GP/s**	**G6x50** iPadAir2, iPhone6	**K1, X1** Nexus 9, Shield Tablet	**Adreno420**	MaliT760 SGS6

- Huge performance leap in every generation

Market Share

	PowerVR		NVIDIA		Qualcomm		ARM	
4 ~ 8 GFlops 0.2 ~ 1 GP/s	SGX535	3.5%	Tegra2	1.0%	**Adreno2xx**	**9%**	**Mali400 MPx**	**19%**
16 GFlops 2 ~ 3 GP/s	**SGX54x**	**15.4%**	Tegra3	0.9%	**Adreno305**	**7.1%**		
100 GFlops 4 GP/s	**G6x30**	**6.0%**	Tegra4	0.0%	**Adreno3x0**	**10.3%**	MaliT628	0.5%
250 GFlops 4 ~ 8 GP/s	G6x50	0.3%	K1, X1	0.0%	Adreno420	0.1%	MaliT760	0.0%

- Green - GPU with significant market share
- TIP: new devices >10x faster than what most people have in their pocket!

Optimization Tiers

	PowerVR		NVIDIA		Qualcomm		ARM	
Low-end Tier	SGX535	3.5%	Tegra2	1.0%	Adreno2xx	9%		
Mid Tier	SGX54x	15.4%	Tegra3	0.9%	Adreno305	7.1%	Mali400 MPx	19%
	G6x30	6.0%	Tegra4	0.0%	Adreno3x0	10.3%	MaliT628	0.5%
High-end Tier	G6x50	0.3%	K1, X1	0.0%	Adreno420	0.1%	MaliT760	0.0%

- iOS, Android and Windows combined

Important GPU characteristics for PBR

- Ratio between math (ALU) and fetching texture (TEX)
- Scalar or vector architecture
- Precision

	PowerVR		NVIDIA		Qualcomm		ARM	
16 GFlops	SGX54x	**16** FLOPs / 1 TEX **FP16 *** **vector**					Mali400 MPx	**16** FLOPs / 1 TEX **FP16 only** **vector**
70 GFlops	SGX554	**32** FLOPs / 1 TEX **FP16 *** **vector**			Adreno3xx	16? FLOPs / 1 TEX FP32/FP16 scalar	MaliT604	**16*** FLOPs / 1 TEX FP32-FP16 wide **vector**
>100 GFlops	G6x30	**48** FLOPs / 1 TEX FP32-FP16 scalar	K1	**48** FLOPs / 1 TEX FP32 only scalar			MaliT628	**32*** FLOPs / 1 TEX FP32-FP16 wide **vector**
>200 GFlops	G6x50	**64** FLOPs / 1 TEX FP32-FP16 scalar	X1	**64/128** FLOPs / 1 TEX FP32-FP16 scalar	Adreno4x0	**32** FLOPs / 1 TEX FP32-FP16 scalar	MaliT760	**>68** FLOPs / 1 TEX FP32-FP16 wide **vector**

- Unofficial numbers, some based on our measurements. Numbers might be wrong! Numbers are peak values.
- **TEX** - bilinear texture fetch
- **FP32-FP16** - supports both precision, likely to be faster in FP16
- **FP16 *** - definitely faster in FP16, but certain complex operations (EXP, LOG, etc) will be executed in FP32 anyway
- **wide vector** - FP16 are likely to be executed as 8-way vectors

Important GPU characteristics for PBR

- FP16 (“FP16 only” & “FP16 *”)
  - PBS is more prone to artifacts @ low precision
  - Check your epsilons (1e-4 is OK, 1e-5 is not!)
  - Sometimes need additional clamping due to precision overflows
- Vector pipeline might need different optimizations
- ALU/TEX differs a lot for high-end vs low-end GPUs

Optimizing for High-end tier

	PowerVR		NVIDIA		Qualcomm		ARM	
	SGX535	3.5%	Tegra2	1.0%	Adreno2xx	9%	Mali400 MPx	19%
	SGX54x	15.4%	Tegra3	0.9%	Adreno305	7.1%		
High-end Tier	G6x30	6.0%	Tegra4	0.0%	Adreno3x0	10.3%	MaliT628	0.5%
	G6x50	0.3%	K1, X1	0.0%	Adreno420	0.1%	MaliT760	0.0%

Optimizing BRDF for Mobile

$$I_{spec} = \frac{D(N \cdot H, roughness) \cdot G(N \cdot V, N \cdot L, roughness) \cdot F(L \cdot H, specColor)}{4 \cdot (N \cdot V) \cdot (N \cdot L)} \cdot N \cdot L$$

- Specular micro-facet equation
- D: Distribution Term
    - **GGX** vs **Normalized Blinn-Phong** vs **SG** approx.
- V: Visibility term

$$V = \frac{G(N \cdot V, N \cdot L, roughness)}{4 \cdot (N \cdot V) \cdot (N \cdot L)}$$

- F: Fresnel term

GGX vs BlinnPhong

- GGX - more simple ops (ADD, MUL), but only 1 complex (RCP)

$$GGX=\frac{roughness^4}{\pi\cdot\left((N\cdot H)^2\,(roughness^4-1)+1\right)^2}$$

- Normalized Phong - several complex ops (RCP, EXP, LOG)

$$Phong=\frac{1}{\pi\cdot roughness^4}\cdot(N\cdot H)^{\left(\frac{2}{roughness^4}-2\right)}$$

  - even SG approximation (RCP, EXP)

Simple vs Complex op

- PowerVR G6x00 asm
  (Phong example)
- Can do many ops / cycle, but only 1 complex!
- Most other architectures complex op = latency

```
23    : fmad ft0, i0, r22, r9
        fmul ft1, c71, r13
        pck.f32 ft2
        tstgz.f32 ftt, _, ft0
        mov i0.e0.e1.e2.e3, i3, ftt, ft0, ft1

24    : flog i0, i0.abs

25    : fmul ft0, i1, i2
        fmul ft1, i0, i3          cycle
        mov i3, ft0;
        mov i0, ft1;

26    : fexp i0, i0               another cycle

27    : fadd ft0, i3, r23
        fmul ft1, i0, r23
        mov i2, ft0;
        mov i1, ft1;
```

Geometric / Visibility term

- Smith adopted for GGX $V_{Smith} = \frac{1}{((N.L).(1\text{-}k)+k)((N.V).(1\text{-}k)+k)}$
- Kelemen and Szirmay-Kalos (KSK) $V_{SKS} = \frac{1}{(L.H)(L.H)}$
  - does not take roughness into account!
- Fix for KSK (J. Hable)

$$V_{SKSm} = \frac{1}{(L.H)^2.(1\text{-}roughness^2)+roughness^2}$$

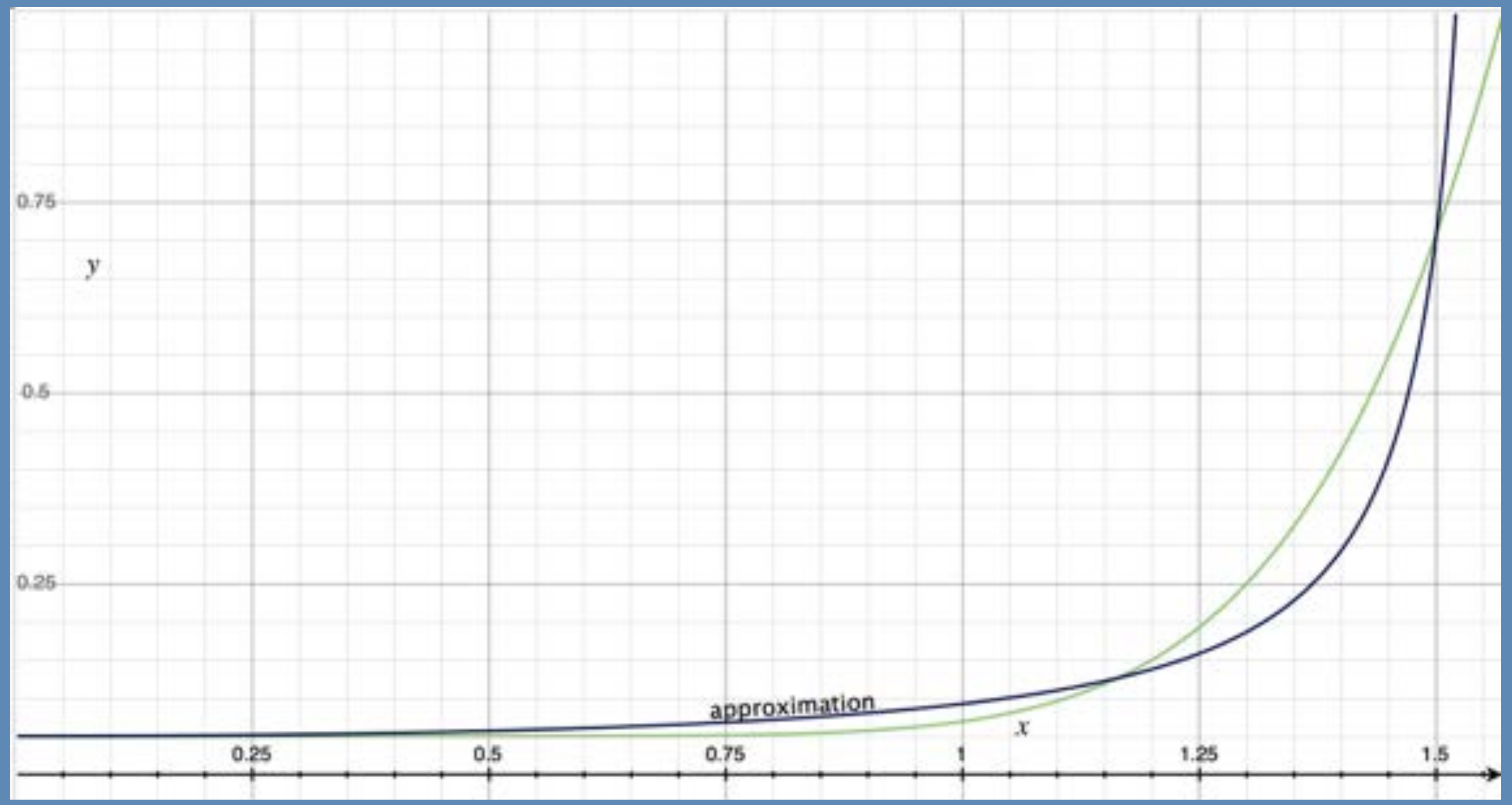
- Dependent only on **L•H** and Roughness!

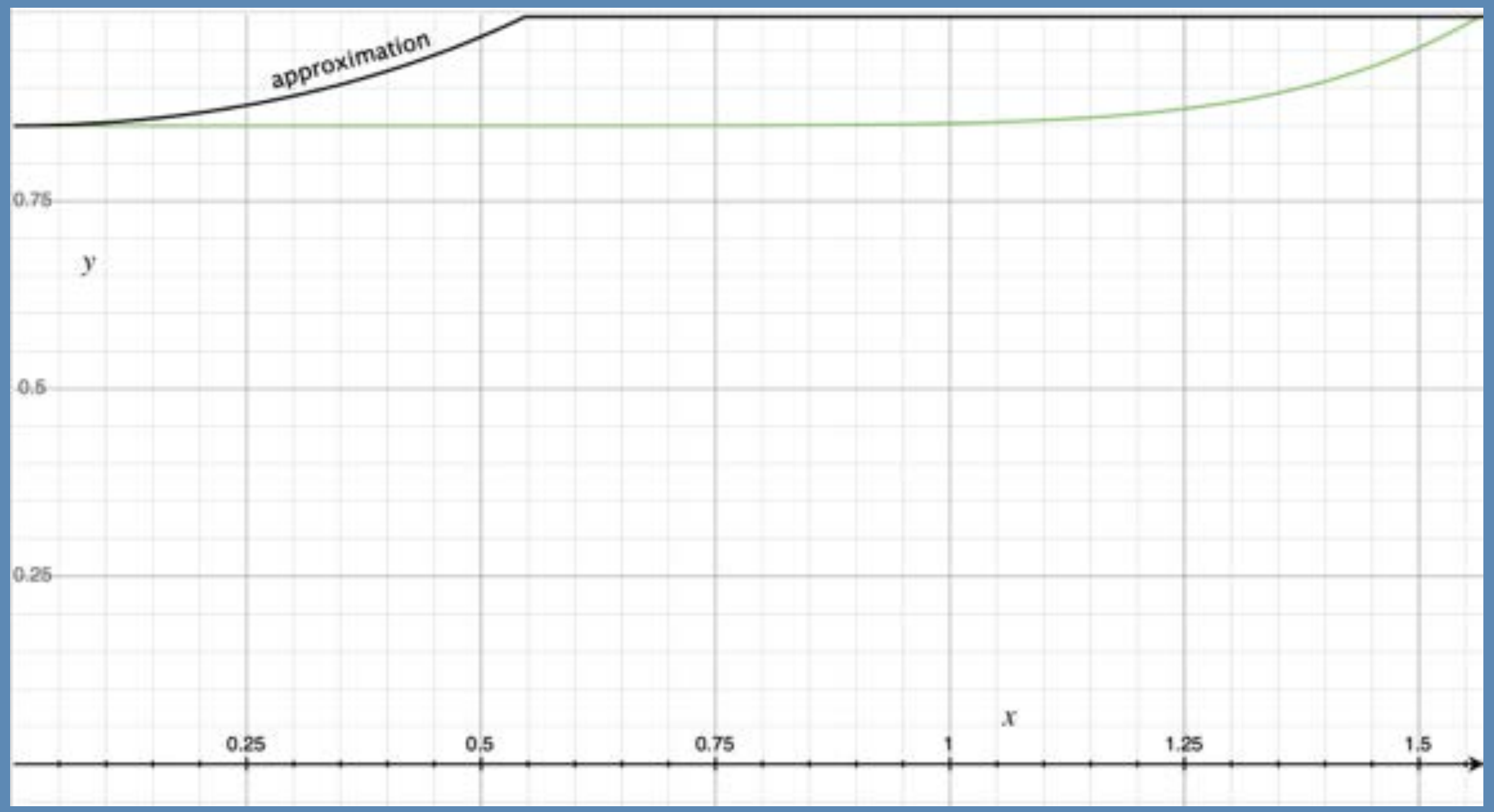
Fresnel term

- Approximation suggested by C. Schüler:

$$F = \frac{specColor}{L \cdot H}$$

- Dielectrics - **OK**
  (reflectance 0.02 ~ 0.15)
- Conductors aka Metals - average value OK
  (reflectance 0.7 ~ 1.0)
  - **has wrong shape**, but Fresnel is almost flat for Metals anyway
- Goes to +Infinity instead of 1





Fresnel term

$$F = \frac{specColor}{L \cdot H}$$

- Will not use Schüler approximation directly
- Just inspiration that specColor can be post multiplied
  - Great for scalar pipeline!

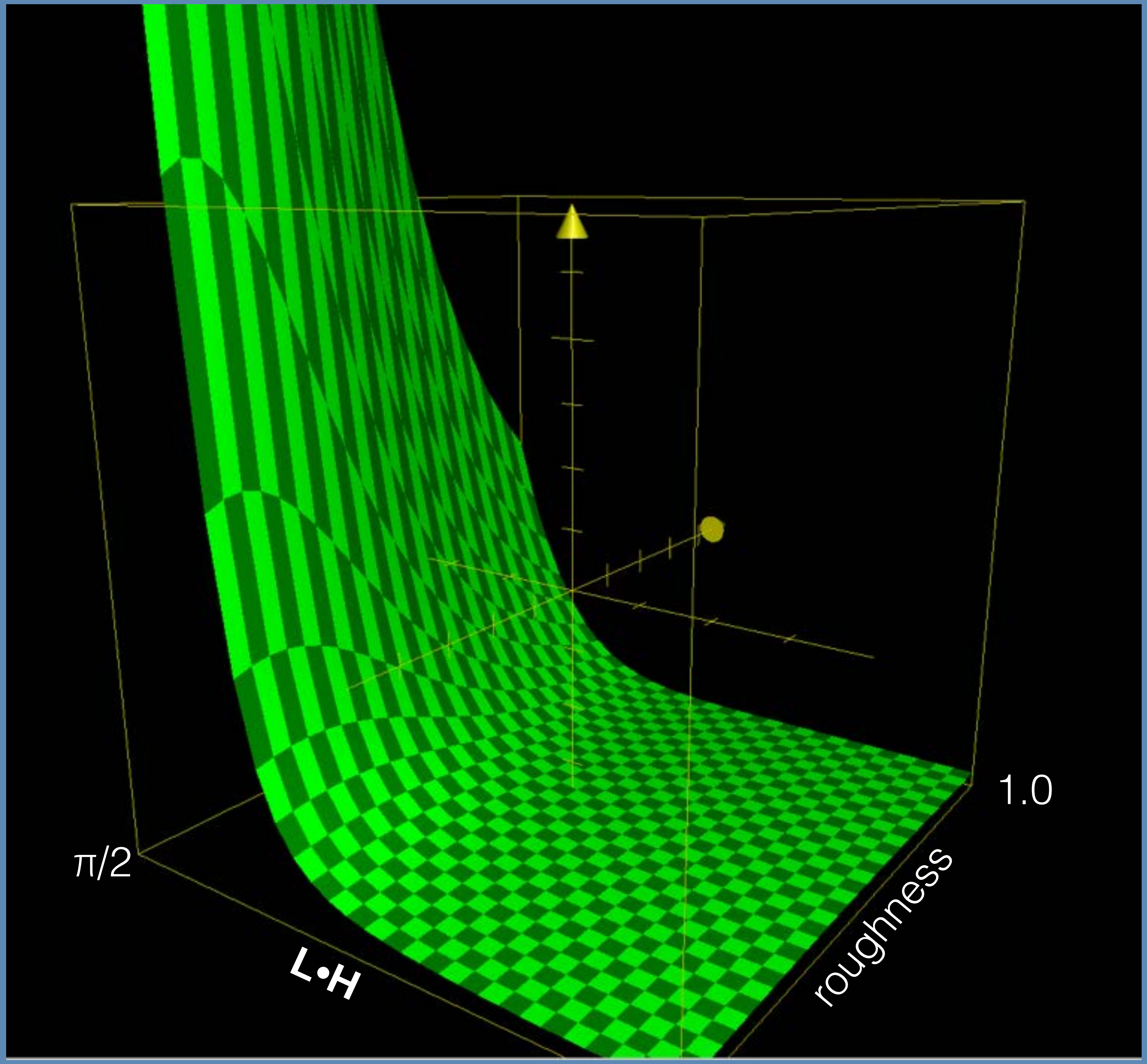
V*F together

- Modified KSK and Schlick Fresnel depend on **L•H**

- Fuse them together

$$V.F = \frac{(1\text{-}L.H)^5}{(L.H)^2.(1\text{-}roughness^2)+roughness^2}$$

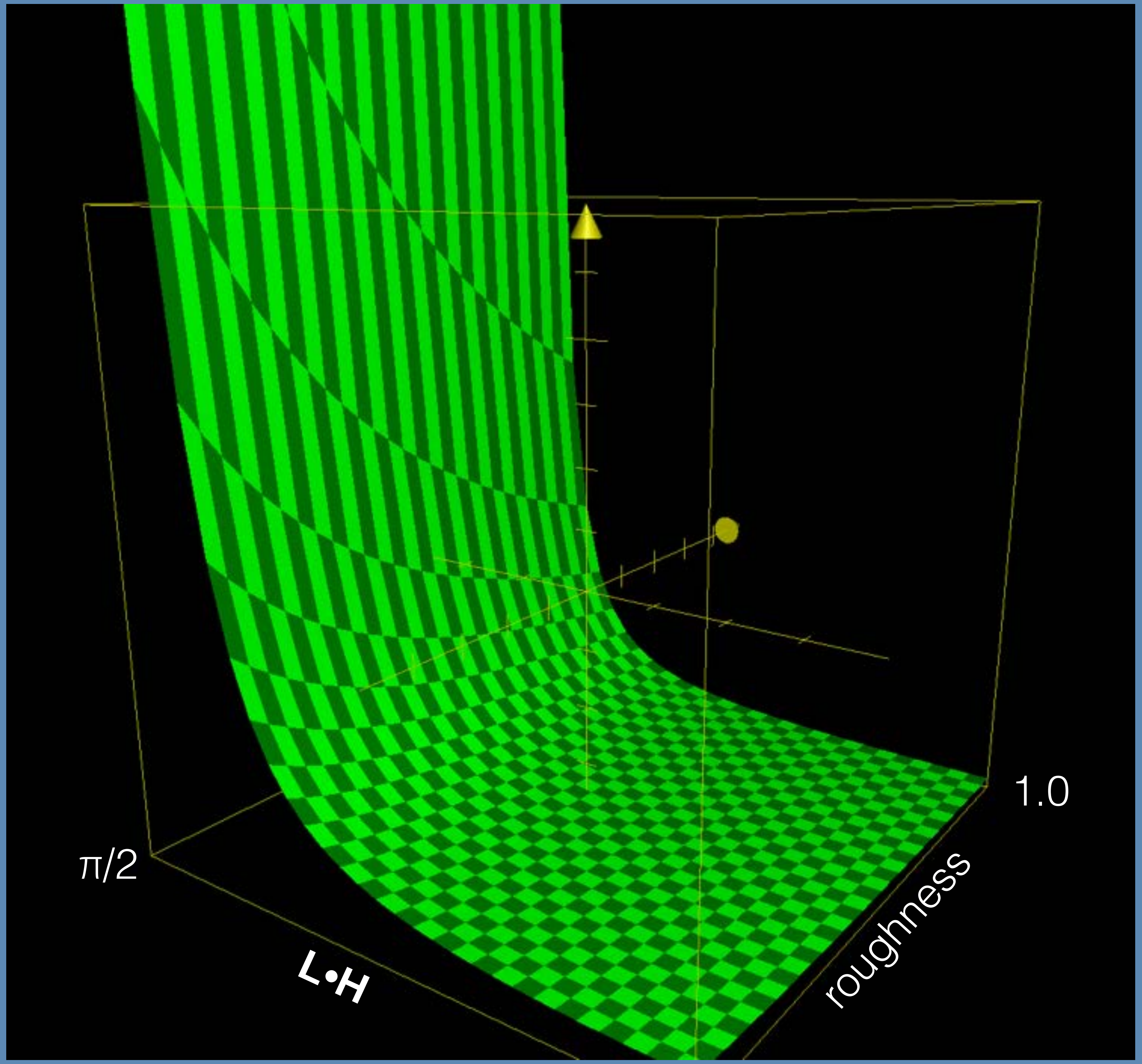
- Cheaper approximation?



Approximate V*F

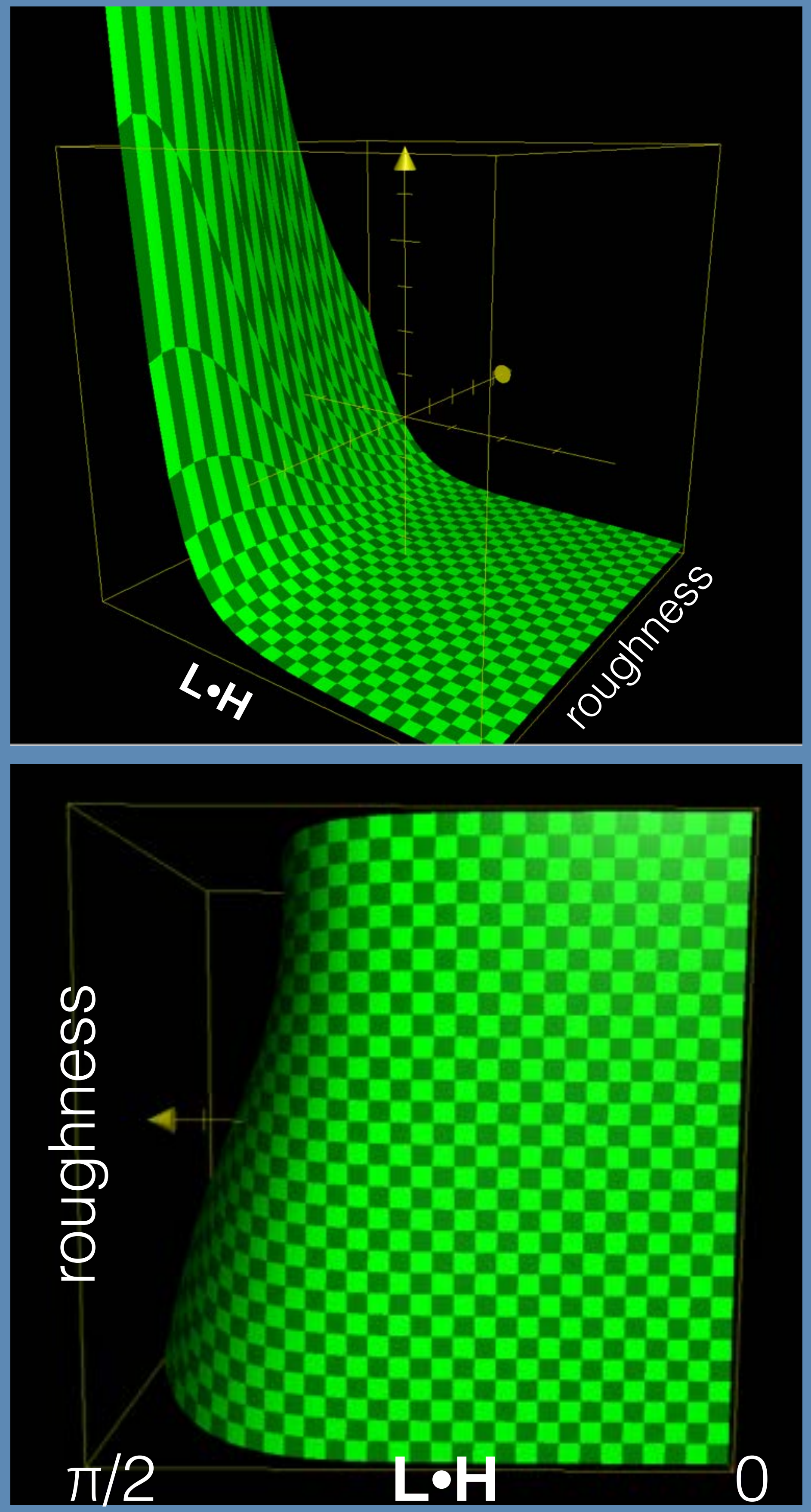
- Not an algebraic simplification
- Fitting similar curve

$$V \cdot F_{approx} = \frac{1}{(L \cdot H)^2 \cdot (roughness + 0.5)} \cdot specColor$$

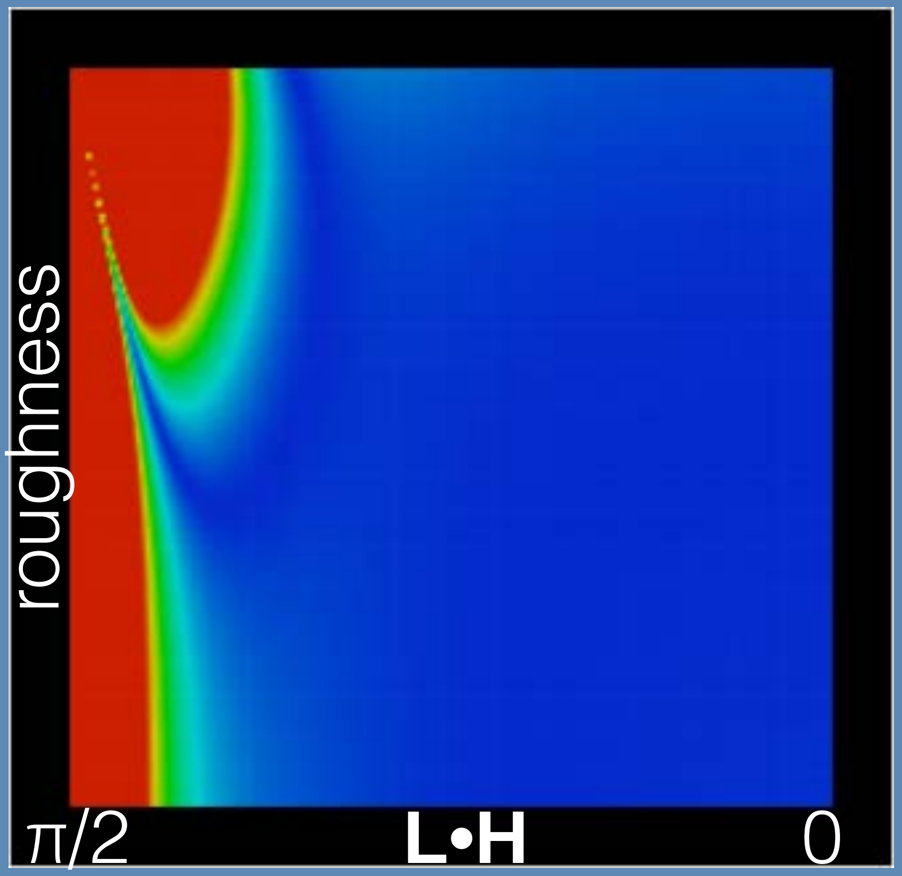


Approximation Results

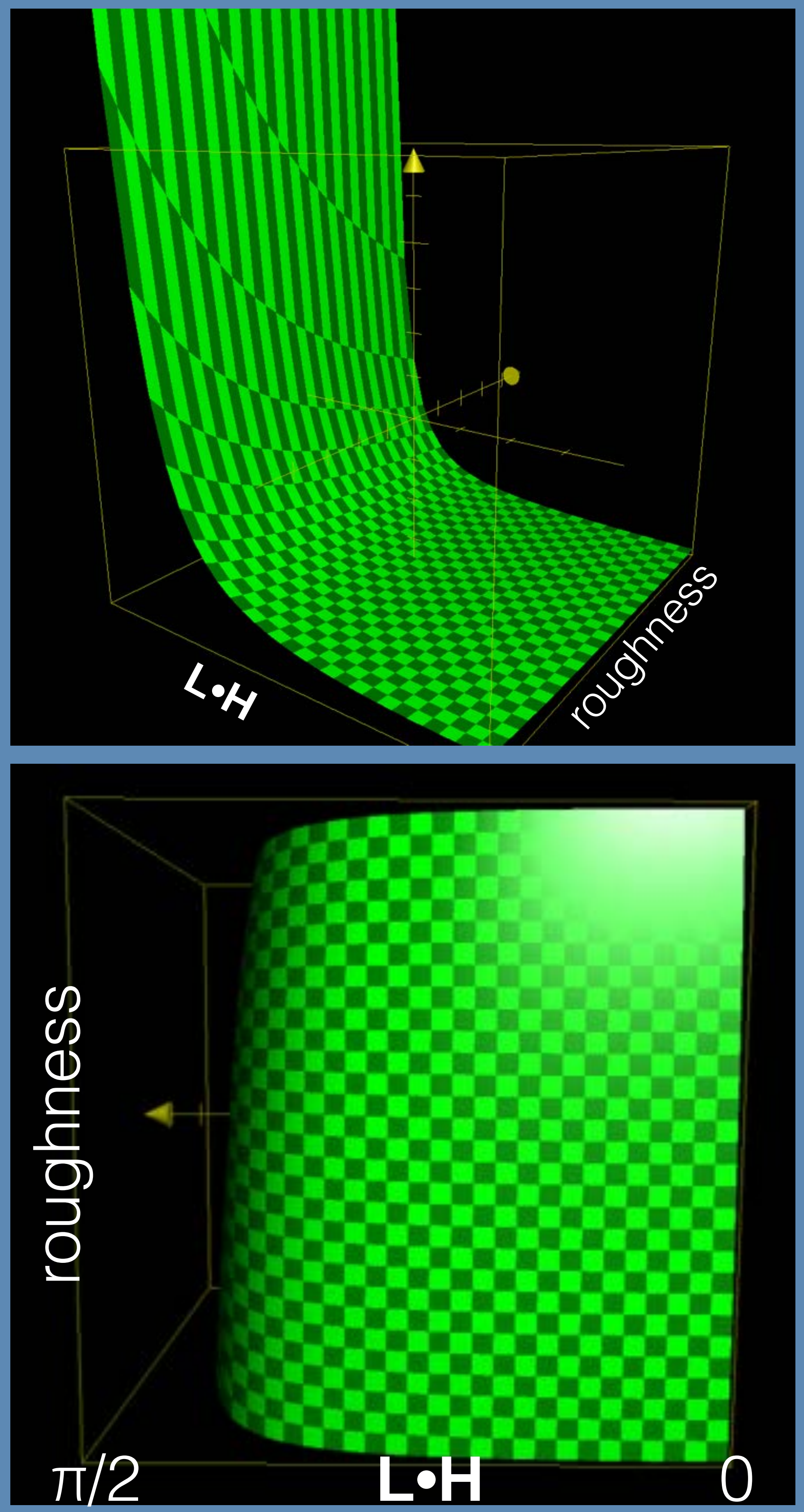
Original (Modified KSK, Fresnel)



Errors



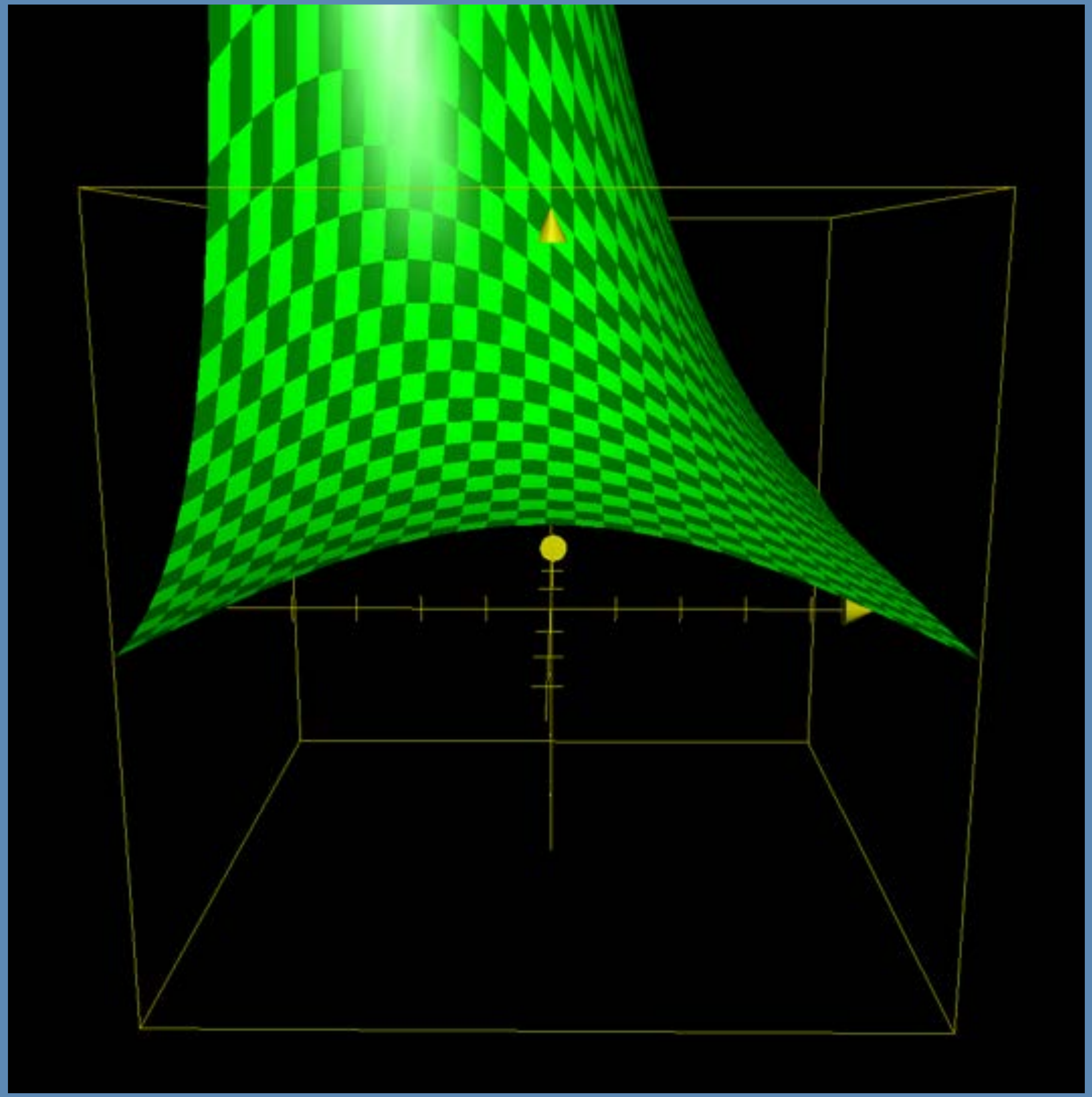
Our approximation



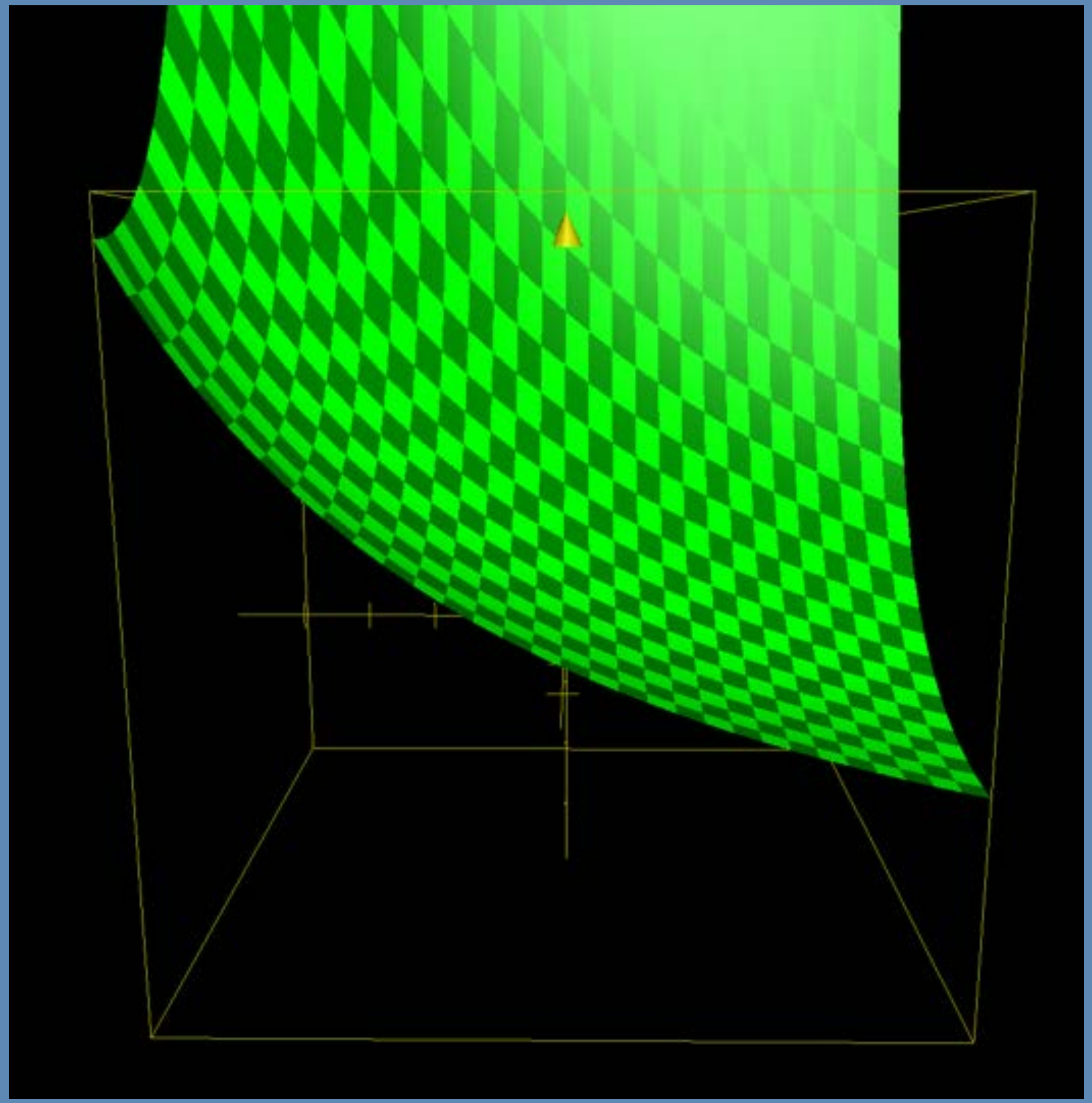
Approximate V*F

- Good for Dielectrics, but diverge for Metals

Original (Modified KSK, Fresnel)



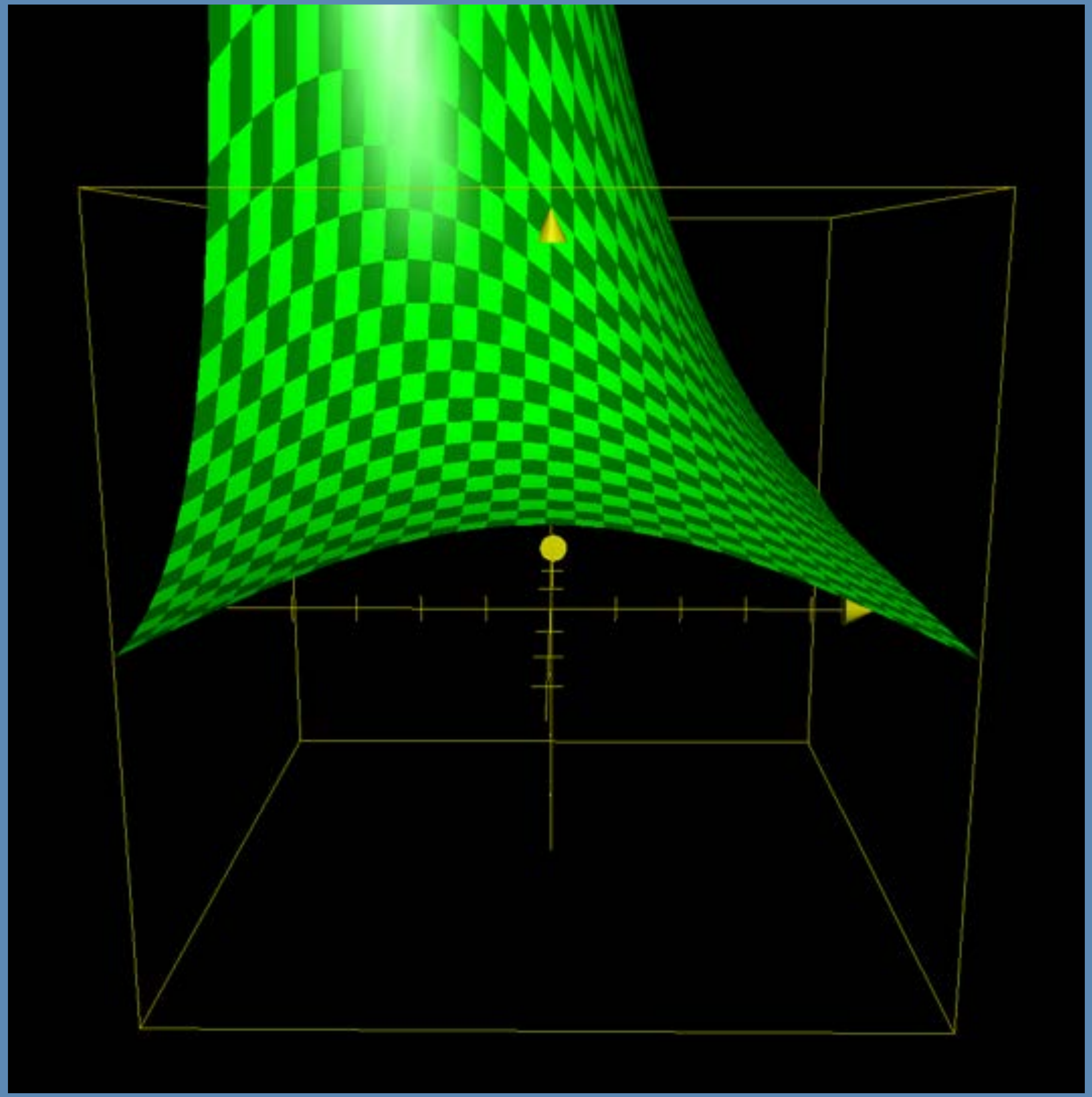
Our approximation



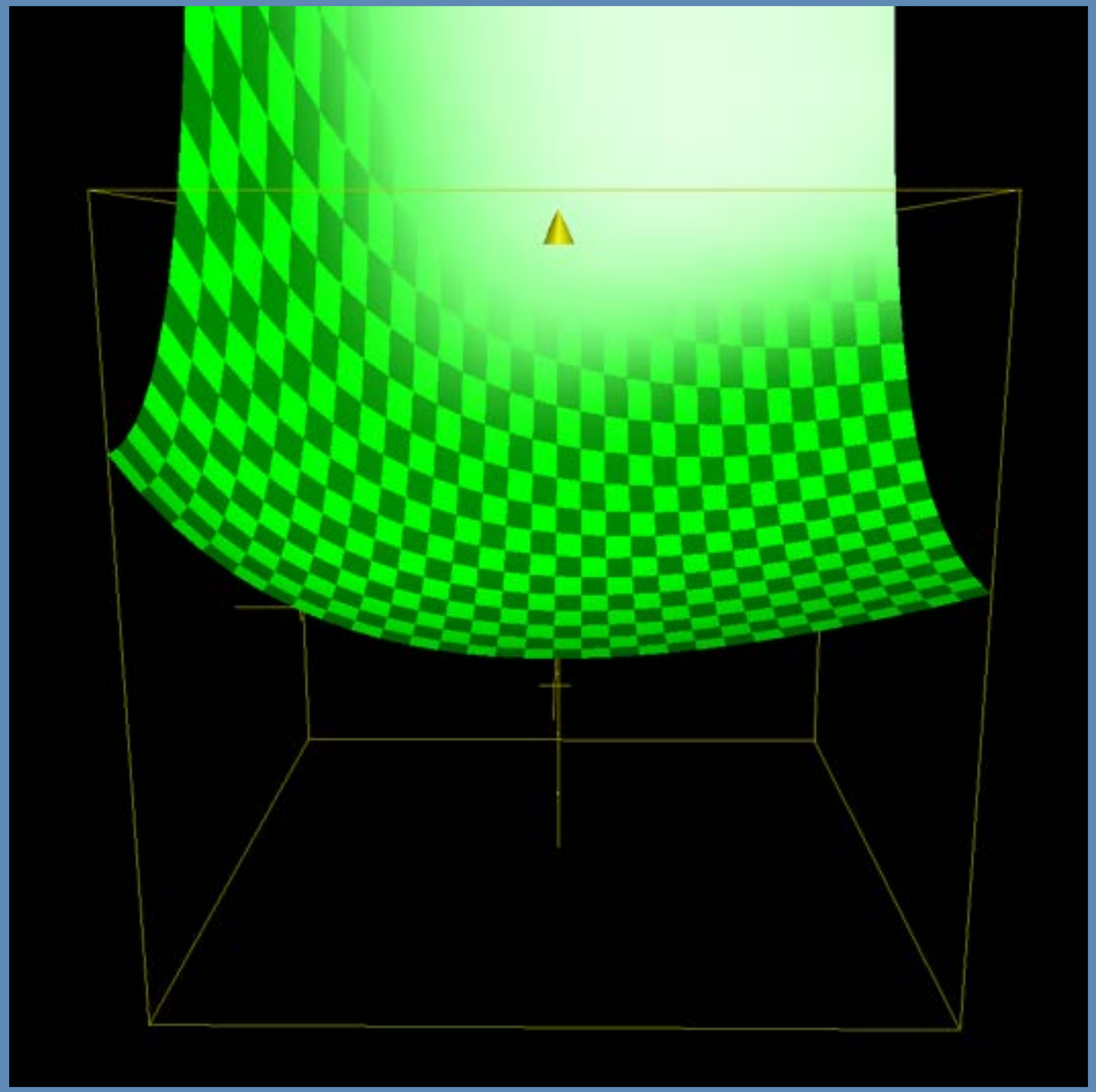
Approximate V*F

- Can be improved with couple more ops, but does not matter in practice

Original (Modified KSK, Fresnel)



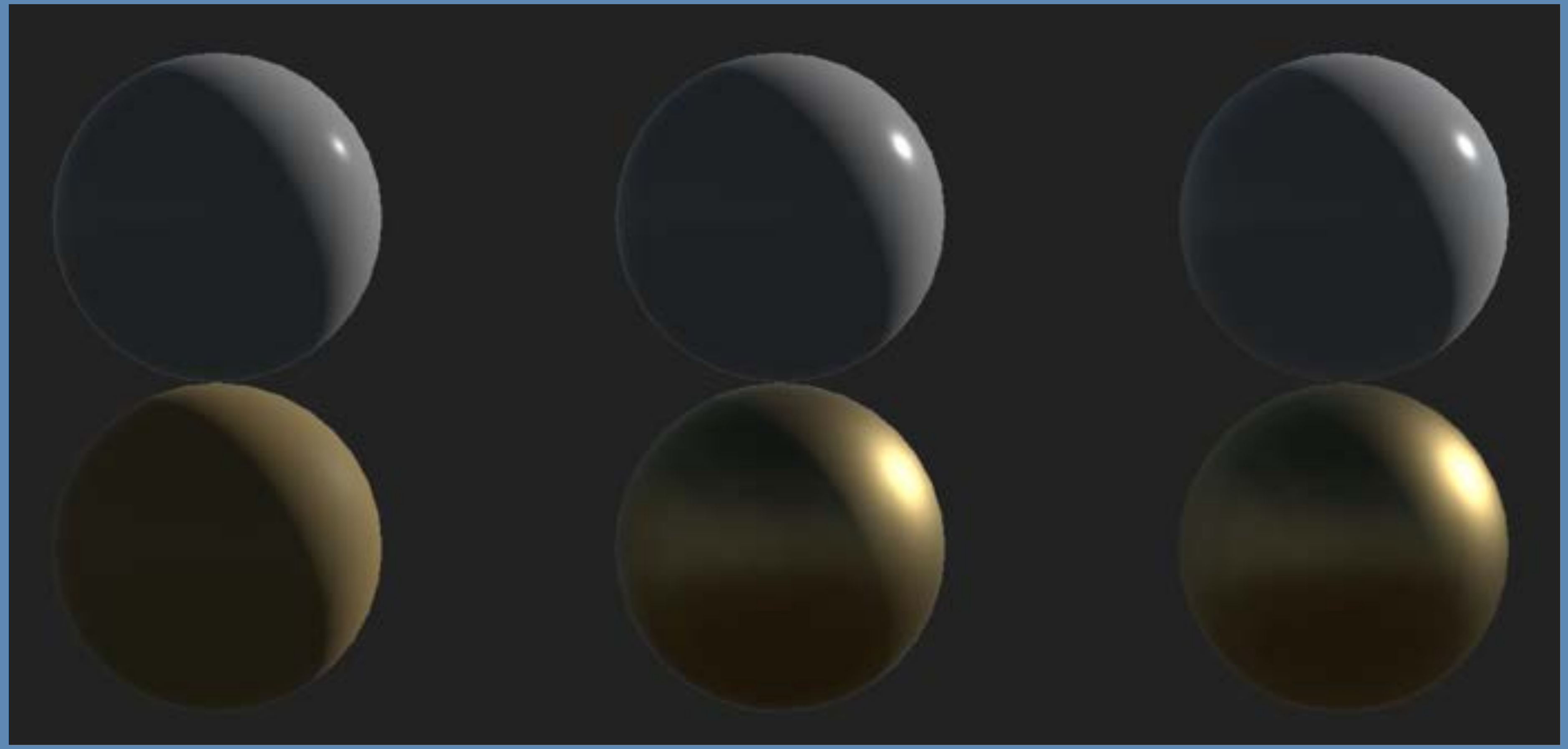
Our approximation*



Comparison of Visibility Terms

Implicit+Fresnel **Smith**+Fresnel **Our Approximation**

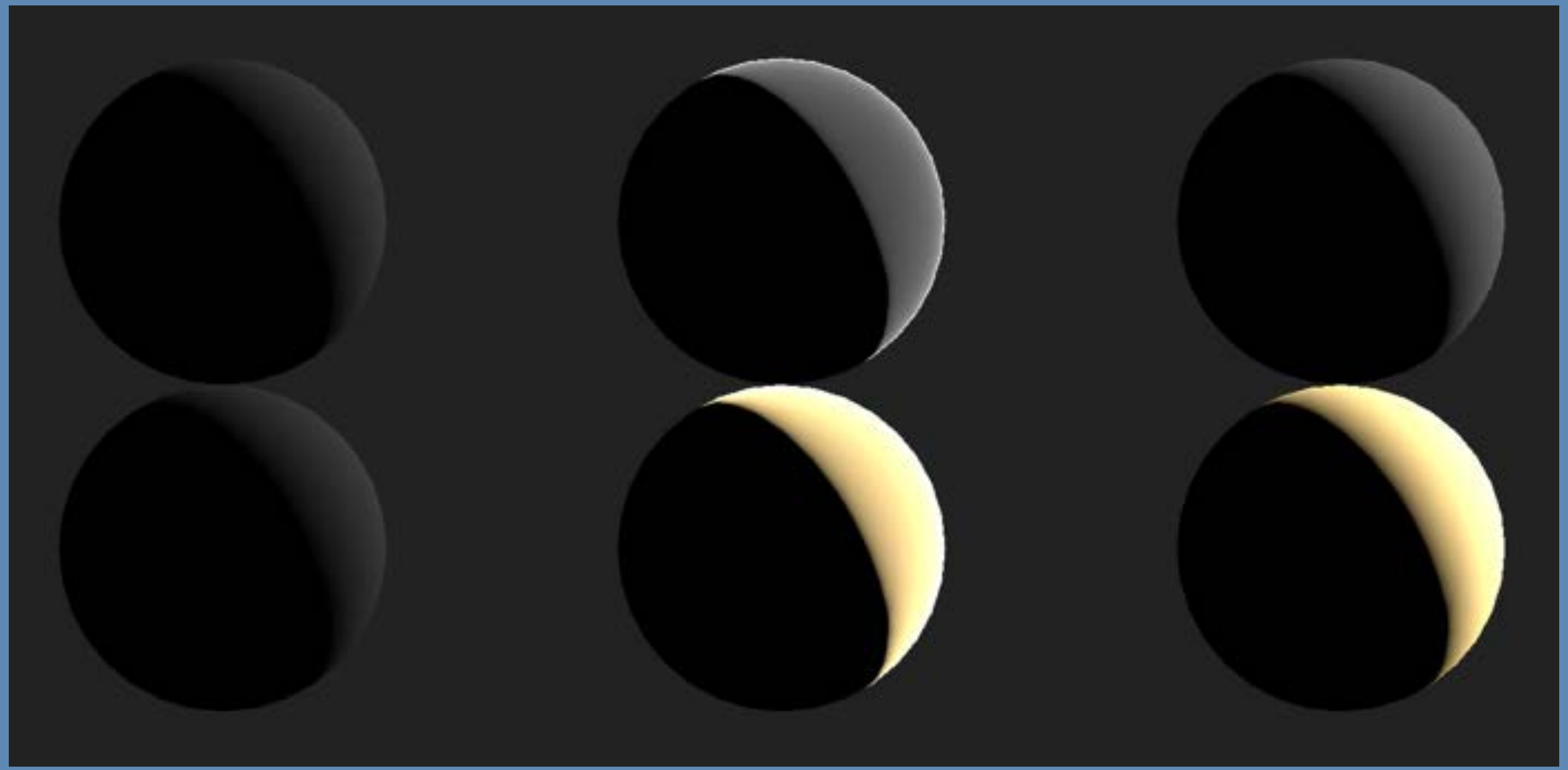
Complete lighting



Plastic

Metal

V*F terms only



Plastic

Metal

Final Specular BRDF

$$BRDF_{spec} = \frac{roughness^4}{4 \cdot \pi \cdot \left((N \cdot H)^2 \left(roughness^4 - 1 \right) + 1 \right)^2 \cdot (L \cdot H)^2 (roughness + 0.5)} \cdot specColor$$

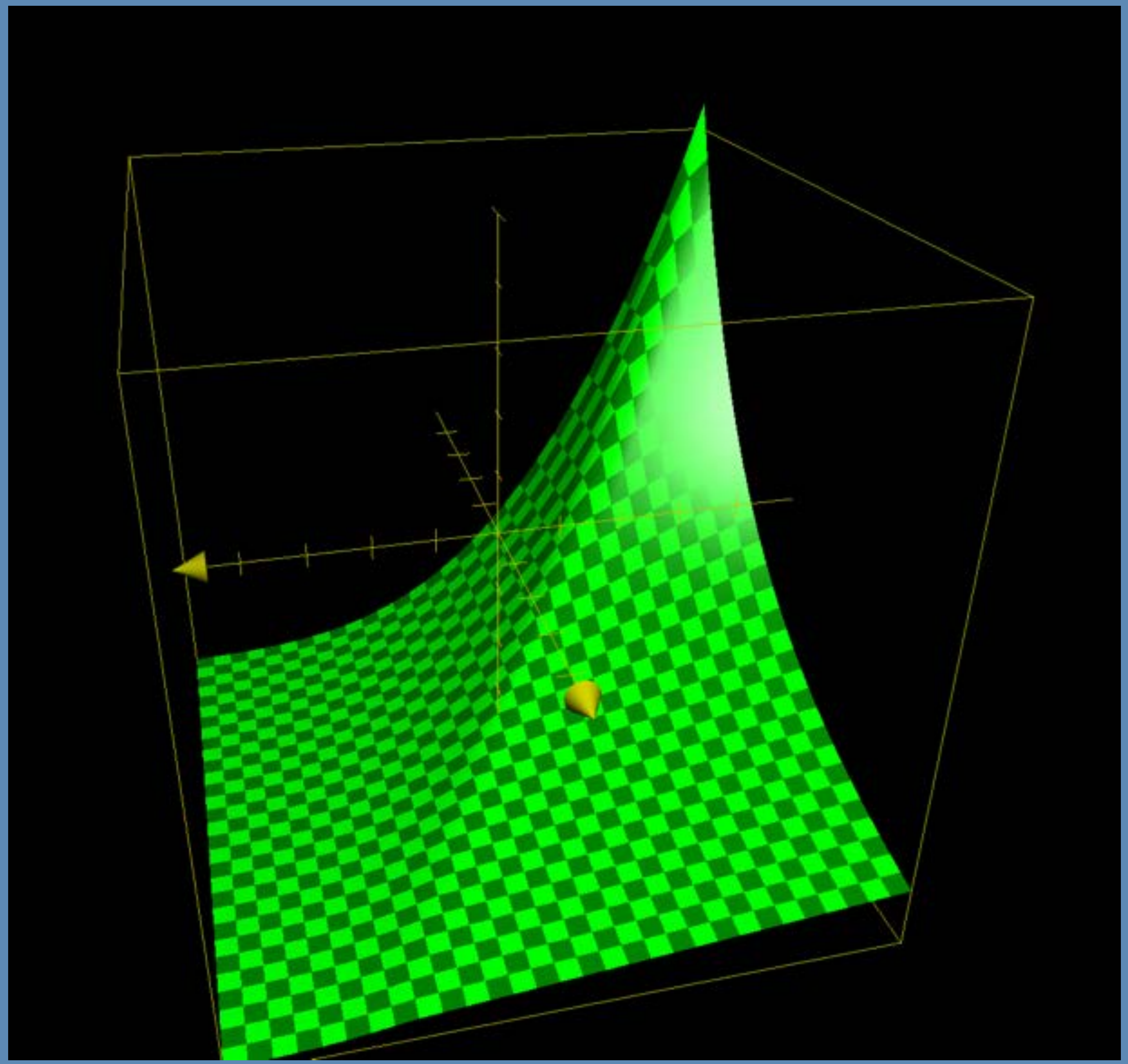
- Just 1 division
- Good for scalar pipeline

Environment BRDF

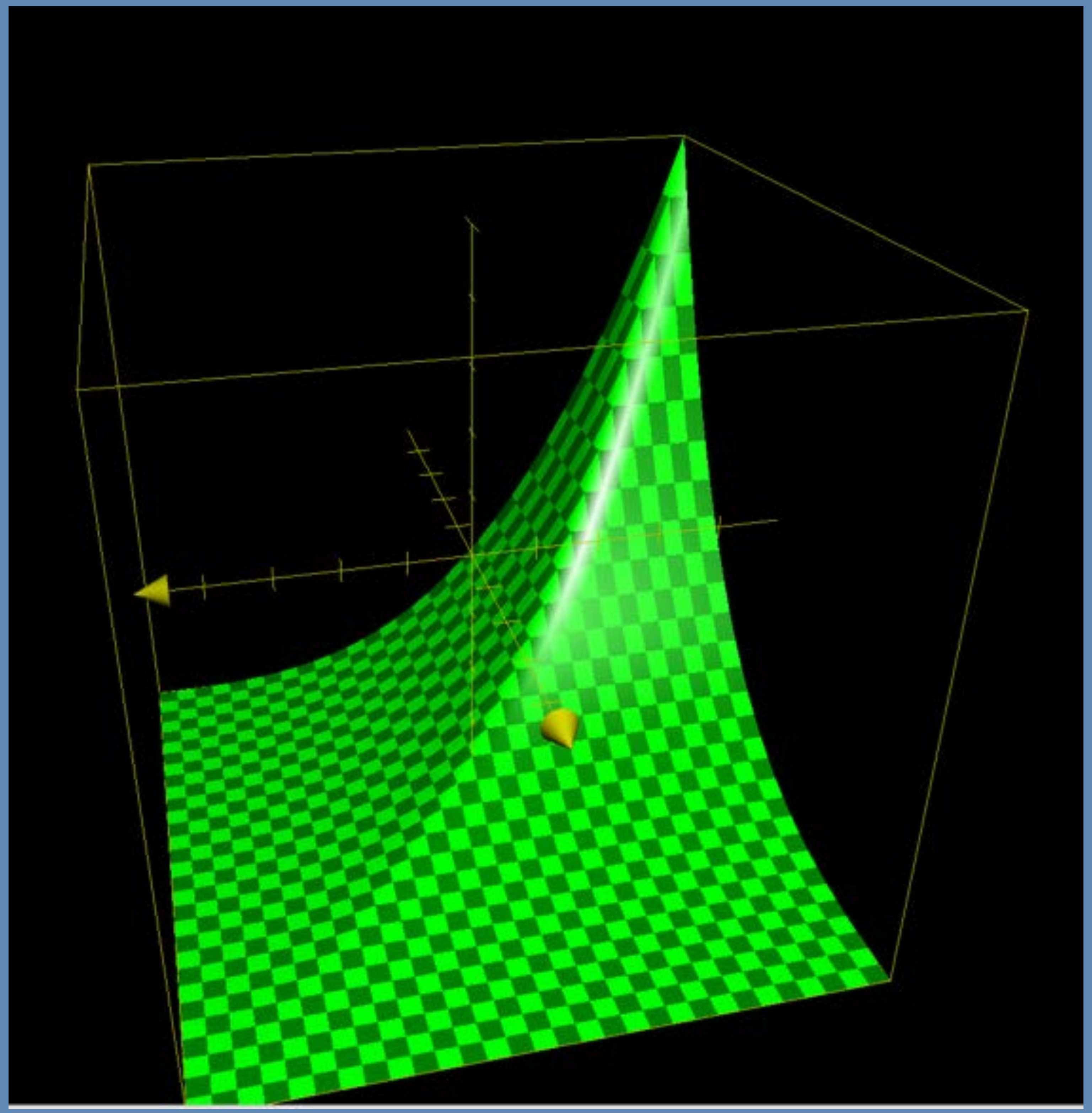
- B. Karis approximation based on D. Lazarov work
- Just refitted with simpler function

$$BRDF_{env} = (1 - \max(roughness, N \cdot V))^3 + spec\mathrm{Color}$$

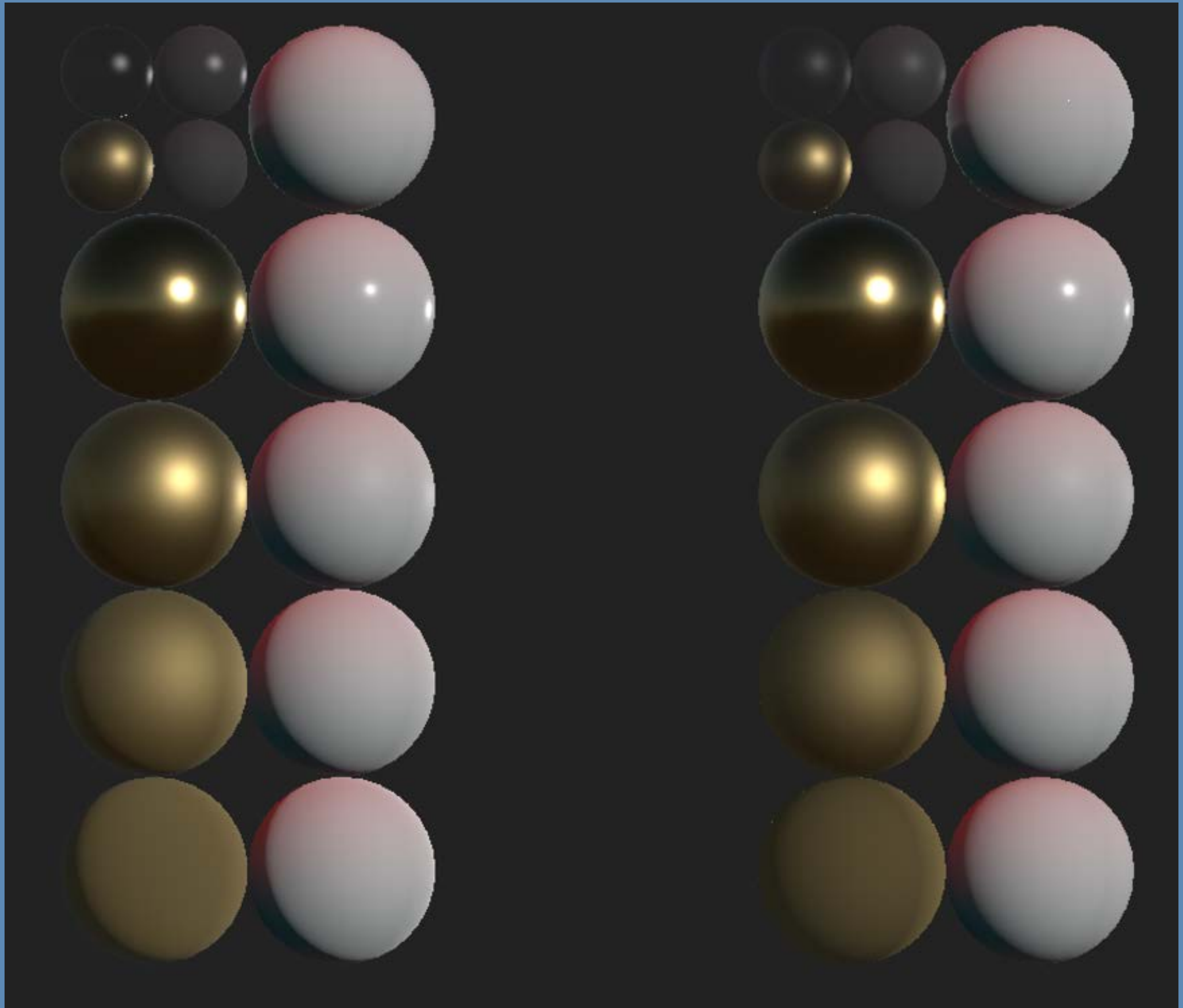
Environment BRDF



$(1\text{-max}(roughness, N{\cdot}V))^3$



Putting everything together



Putting everything together

	ImgTech G6x00 (scalar)	ImgTech SGX554 (vector)	QCOM Adreno305 (scalar)	ARM MaliT760 (vector)	ARM Mali400MP4 (vector)
old-school-non-PBR unnormalized BinnPhong	141%	172%	154%	140%	
normalized BlinnPhong, Smith **(baseline)**	100%	100%	100%	100%	100%
proposed version GGX	**114%**	**126%**	**118%**	**111%**	**271%**

- Percentages are used to make test runs on different screen resolutions easily comparable.
- Measured with a scene consisting of 50K vertices fully covering screen with >3x overdraw rate.

Optimizing for Mid tier

PowerVR		NVIDIA		Qualcomm		ARM	
SGX535	3.5%	Tegra2	1.0%	Adreno2xx	9%		
SGX54x	15.4%	Tegra3	0.9%	Adreno305	7.1%	Mali400 MPx	19%
G6x30	6.0%	Tegra4	0.0%	Adreno3x0	10.3%	MaliT628	0.5%
G6x50	0.3%	K1, X1	0.0%	Adreno420	0.1%	MaliT760	0.0%

Mid Tier

Per-vertex lighting

- Medium-end hardware:
  - Lower bandwidth, GFLOPs are meh
- Diffuse and ambient per-vertex
- Specular per-pixel
- Environment reflection vector per-vertex
- Specular in Tangent space - saves matrix-vector transformation

Optimizing for Low-end

Low-end Tier

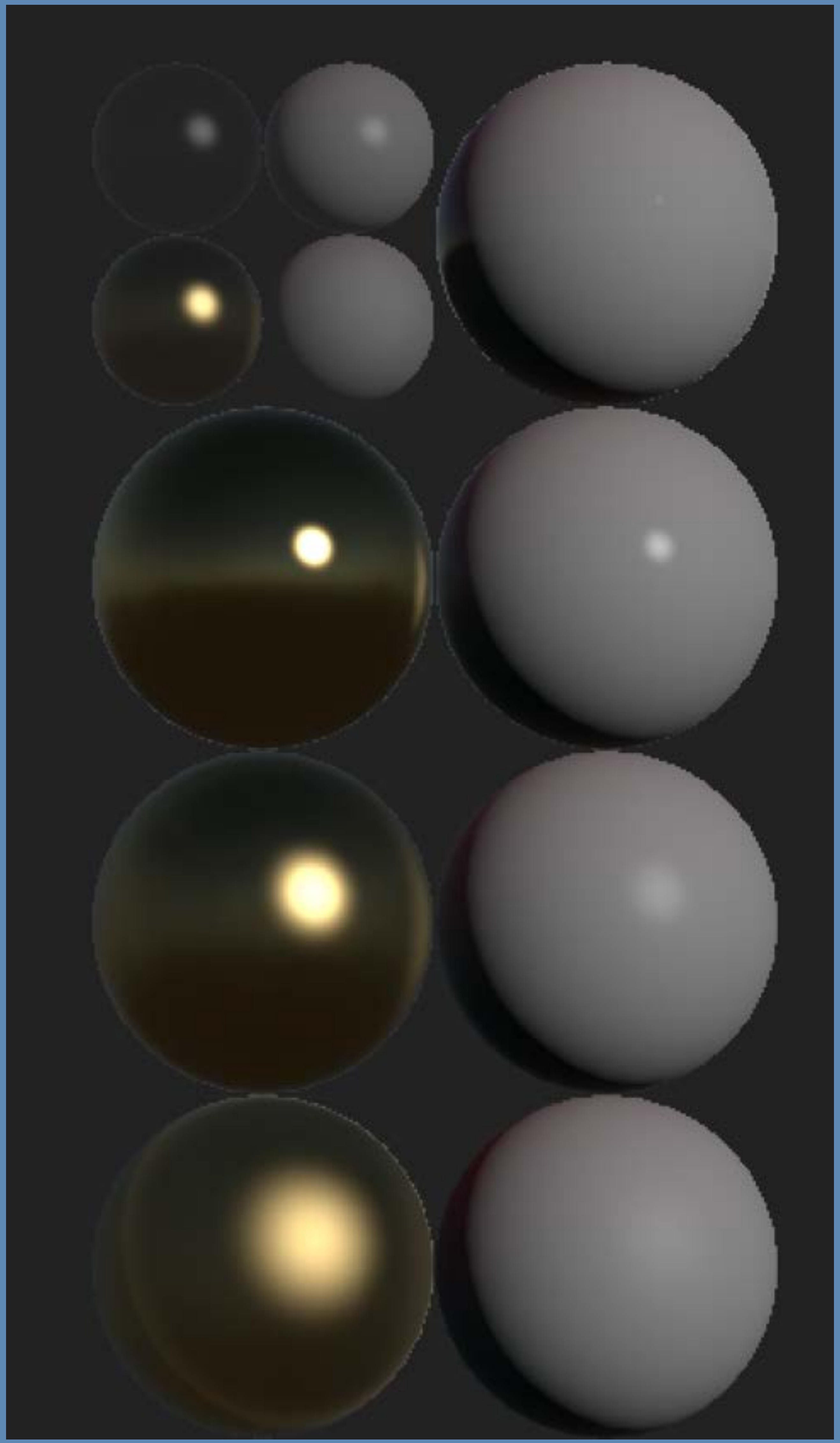
PowerVR		NVIDIA		Qualcomm		ARM	
SGX535	3.5%	Tegra2	1.0%	Adreno2xx	9%		
SGX54x	15.4%	Tegra3	0.9%	Adreno305	7.1%	Mali400 MPx	19%
G6x30	6.0%	Tegra4	0.0%	Adreno3x0	10.3%	MaliT628	0.5%
G6x50	0.3%	K1, X1	0.0%	Adreno420	0.1%	MaliT760	0.0%

LUT

- Low-end hardware:
  - Low ALU/TEX ratio
- Specular intensity in LUT
  - <**N•H**, Roughness>
- Remember implicit Geometric term!
  - I = BRDF * **N•L**
- **N•H** is cosine - highlights are really crammed

LUT specular

- Store 1/16 intensity in LUT
- **R•L** instead on **N•H** saves couple of ops
  - suggested by B.Karis
- Warp LUT /w **R•L⁴** to get more space for highlights



- PBR challenges on Mobile
- What hardware are we optimizing for?
- Faster BRDF
- **Linear/Gamma**
- **Environment Reflections**

Linear/Gamma

- Linear lighting
  - hard on older GPUs
  - has additional cost
- Gamma and Linear will never look the same, but we can aim for:
  - consistent base light intensity
  - consistent highlight size

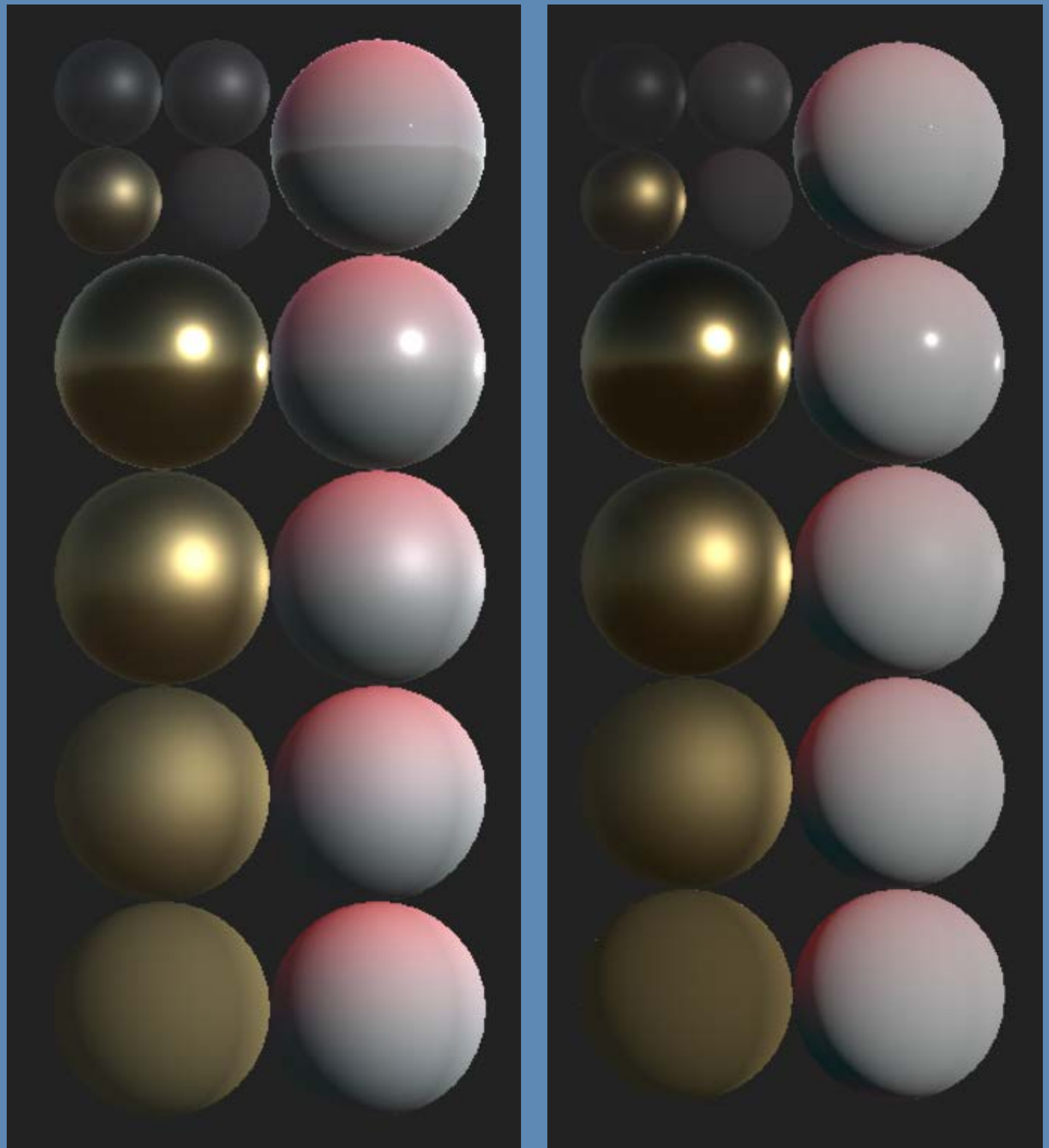
Hack for Gamma to “match” Linear

- Approximate gamma with 2.0
- “Fixup” just **specular intensity**:
  - Keep parameters (Roughness) for specular part of equation in Linear
  - Evaluate specular intensity as in Linear space
  - Convert resulting specular intensity to sRGB space before applying colors:
    = *sqrt (specIntensity_Linear) * specColor_sRGB*

Pros of Gamma hack

- No need to uncompress colors/textures from sRGB to Linear
- Roughness is Linear already
  - usually stored in Alpha channel
- Potentially long latency op (INVSQRT) is NOT at the end of the shader
  - cost can be hidden by other ops

Gamma vs Linear



Environment reflections

- texCUBElod can be really expensive sometimes
  - G6xx0 - high-end mobile GPU!
  - optional extension on ES2.0
- G6xx0: use dynamic branches to pick 2 closest mips and lerp
  - slightly faster!

texCUBElod

- Lerp 2 extreme mips
  - ugly, but fast
- 3-way lerp:
  hardcoded highest mip#
  middle mip#
  2nd order SH
  - for middle you can cut mip levels (/w extension) and hardcode to a very large number

Thanks

John Hable

Morten Mikkelsen

Florian Penzkofer

Alexey Orlov

Dominykas Kiauleikis

Sakari Pitkänen

References

1. Morten Mikkelsen, “Microfacet Based Bidirectional Reflectance Distribution Function”, 2009
2. John Hable, “Optimizing GGX Shaders with dot(L,H)”, 2014, online
3. Christian Schüler, “An Efficient and Physically Plausible Real-Time Shading Model.” ShaderX 7, Chapter 2.5, pp. 175 – 187
4. Brian Karis, “Physically Based Shading on Mobile”, 2014, online
5. Sébastien Lagarde, “Spherical Gaussian approximation for Blinn-Phong, Phong and Fresnel”, 2012, online
6. Kelemen and Szirmay- Kalos, “A Microfacet Based Coupled Specular- Matte BRDF Model with Importance Sampling”, Eurographics 2001
7. Robert Cook and Kenneth Torrance, “A reflectance model for computer graphics”

Bonus Slides

OpenGL ES3.0

	PowerVR	NVIDIA	Qualcomm	ARM
4 ~ 8 GFlops 0.2 ~ 1 GP/s	SGX535 iPad, iPhone4	Tegra2	Adreno2xx	Mali400 MPx SGS3 (I9300) SGS2 (I9100)
16 GFlops 2 ~ 3 GP/s	SGX54x iPad2/3, iPhone4s, iPhone5	Tegra3	**Adreno305** SGS4 mini (I9195)	
100 GFlops 4 GP/s	**G6x30** iPadAir, iPhone5s	Tegra4	**Adreno3x0** Nexus 4, Nexus 5	**MaliT628**
250 GFlops 4 ~ 8 GP/s	**G6x50** iPadAir2, iPhone6	**K1, X1** Nexus 9, Shield Tablet	**Adreno420**	**MaliT760** SGS6

- Green - GPU with ES3.0 support
- TIP: you can’t just use ES2.0 / ES3.0 to determine performance of GPU

Low-end with large share

	PowerVR		NVIDIA		Qualcomm		ARM	
4 ~ 8 GFlops **0.2 ~ 1 GP/s**	SGX535	3.5%	Tegra2	1.0%	**Adreno2xx**	**9%**	**Mali400 MPx**	**19%**
16 GFlops 2 ~ 3 GP/s	**SGX54x**	**15.4%**	Tegra3	0.9%	**Adreno305**	**7.1%**		
100 GFlops 4 GP/s	**G6x30**	**6.0%**	Tegra4	0.0%	**Adreno3x0**	**10.3%**	MaliT628	0.5%
250 GFlops 4 ~ 8 GP/s	G6x50	0.3%	K1, X1	0.0%	Adreno420	0.1%	MaliT760	0.0%

- Yellow - Low-end with large share, but most in APAC and Latin America
- And you still need to support iPhone4

Textures

- Lack of uncorrelated 4 channel compression
  - Consider Roughness in a separate texture
  - Pairing Roughness with Specular/Metal instead of Albedo or Normals since former is low frequency & low variance data

Textures

- Lack of HDR compression
    - IBL, Lightmaps: RGB*2 instead of RGBm/HDR
    - IBL: uncompressed HDR cubemaps
    - An awful tradeoff :(