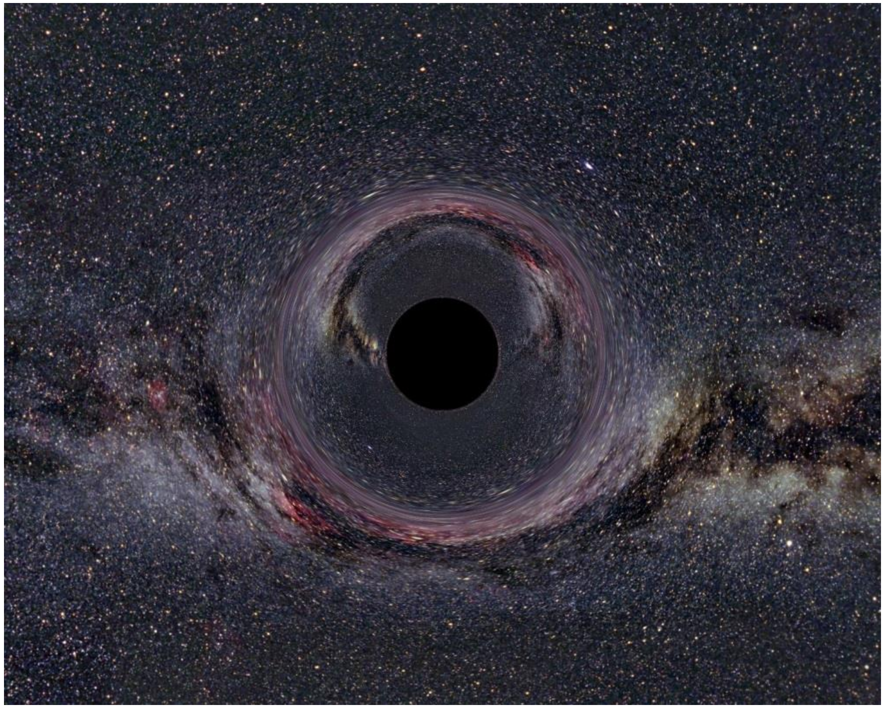
Game Technology

Lecture 7 – 4.12.2017
Physically Based Rendering



Dipl-Inform. Robert Konrad
Polona Caserman, M.Sc.

Prof. Dr.-Ing. Ralf Steinmetz
KOM - Multimedia Communications Lab

Intro

https://www.youtube.com/watch?v=okMUxGFjkpY

Light

- Starts from light source
- Bounces around
  - Looses intensity with each collision
- Eventually reaches the camera

Light Sources

Rendering Lots of Lights

Forward Rendering

- Iterate over all lights in the pixel shader
- Optionally use a pre-depth pass

Deferred Rendering

- Render buffers of depth, normals, materials,…
- Render simple geometry, approximating light distributions
  - Add light inside the geometry using a pixel shader

Forward+

- Create 3D grid, assign most important lights to each grid
- Pull light info from the grid in the pixel shader

Point Lights

Defined by a position and light color/intensity

Spot Lights

TECHNISCHE UNIVERSITÄT DARMSTADT

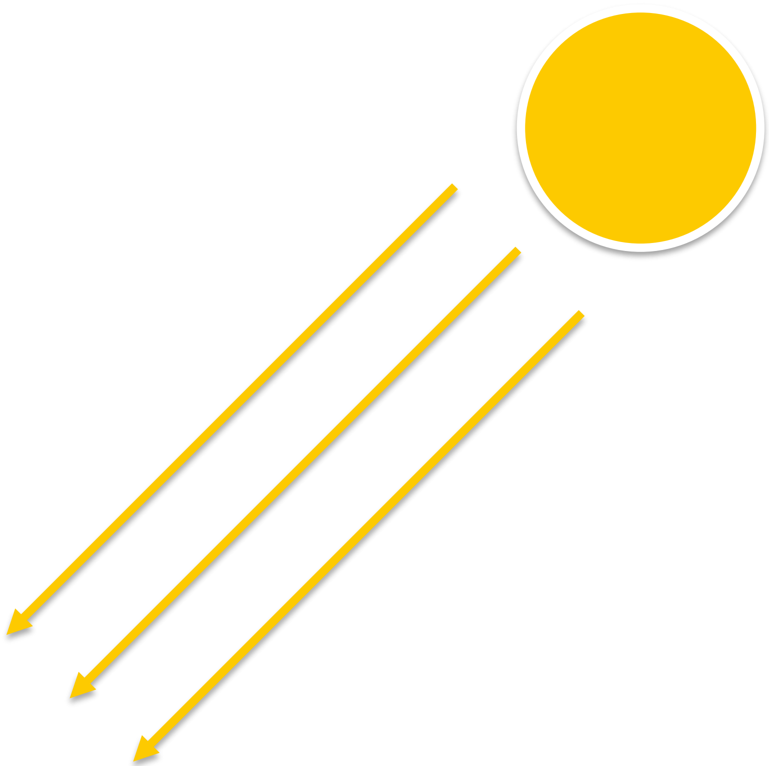
Point light plus an angle

Directional Light

TECHNISCHE UNIVERSITÄT DARMSTADT

Just a direction and light intensity/color

Directional Light

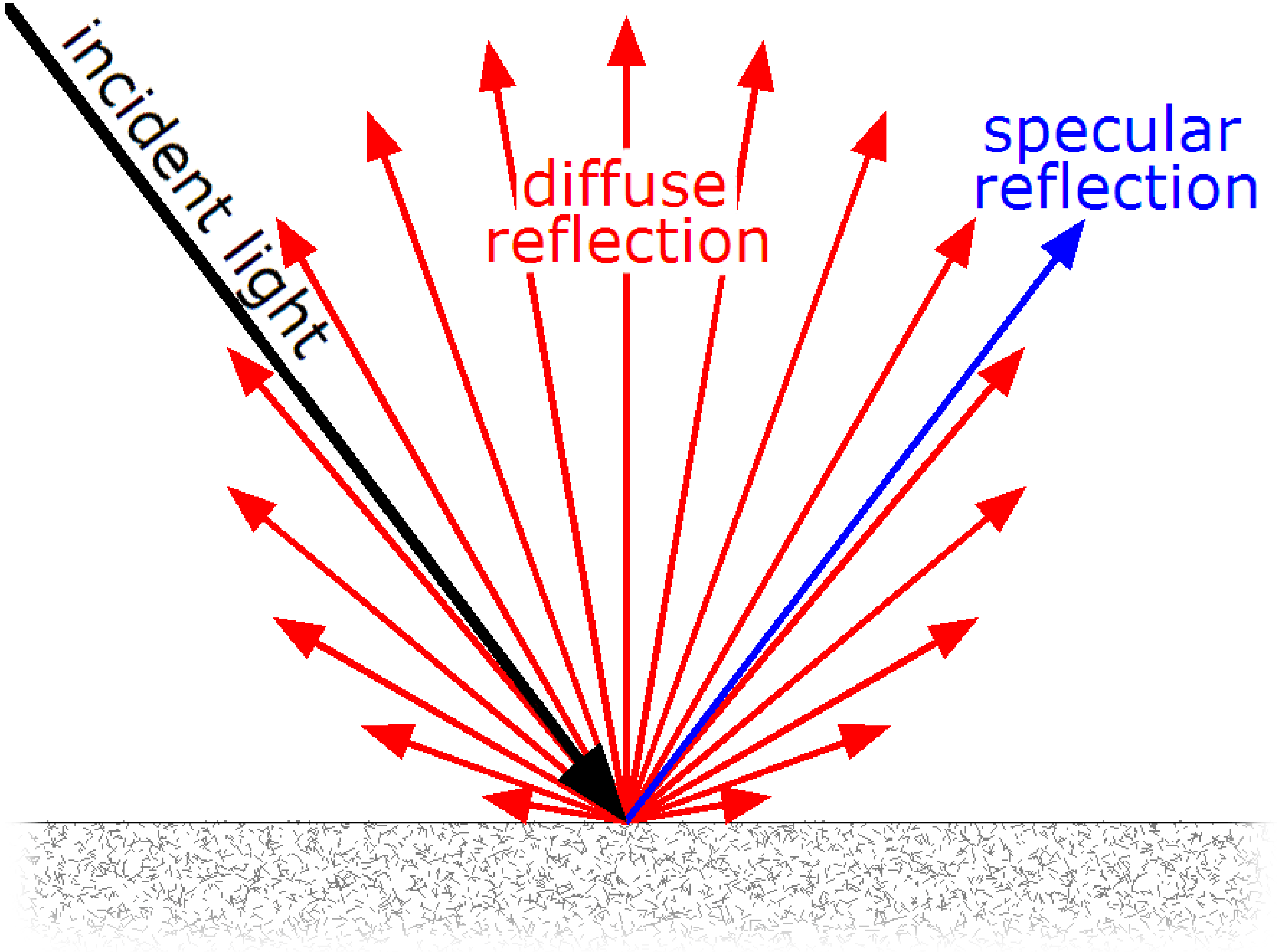


Area Lights

Simple solution: Approximate using multiple point/spot lights

Analytical solution: https://labs.unity.com/article/real-time-polygonal-light-shading-linearly-transformed-cosines

Bounces

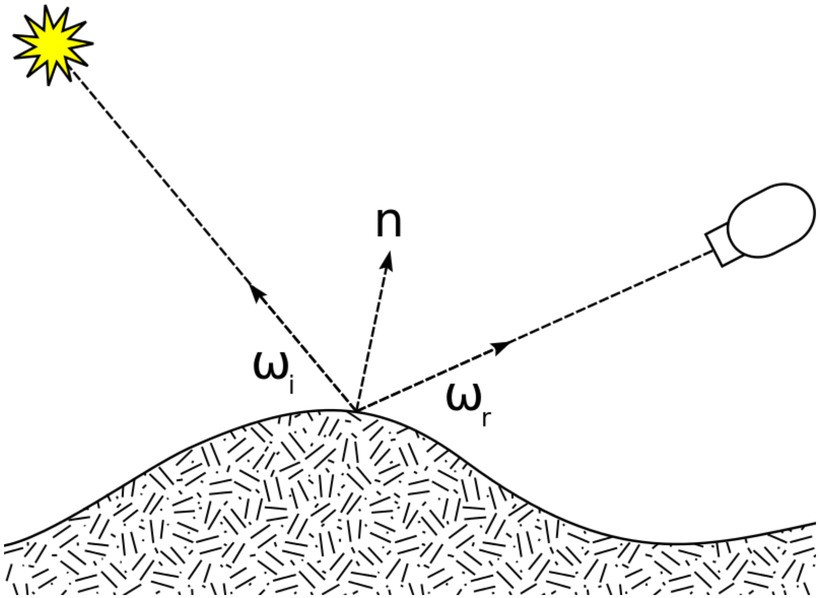


BRDF

Bidirectional reflectance distribution function

- incoming light direction
- outgoing direction (for example to the camera)
- returns the ratio of reflected radiance

$$f_r(\omega_i, \omega_r)$$



Path Tracing

Extended Raytracing

foreach (pixel)

- bounce around a lot

Use BRDF at each collision

Very slow

- but useful to create reference images
- and for prerendered lighting information

Realtime Lighting

TECHNISCHE UNIVERSITÄT DARMSTADT

Consider only light rays from direct light sources

- First bounce

Use shadow maps

- Second bounce

Ignore further light bouncing

- No reflections
- No ambient light

Image-Based Lighting

Put surroundings in cube map

- Use for example path tracing to generate the cube map

Ignore lights, instead sample cube map

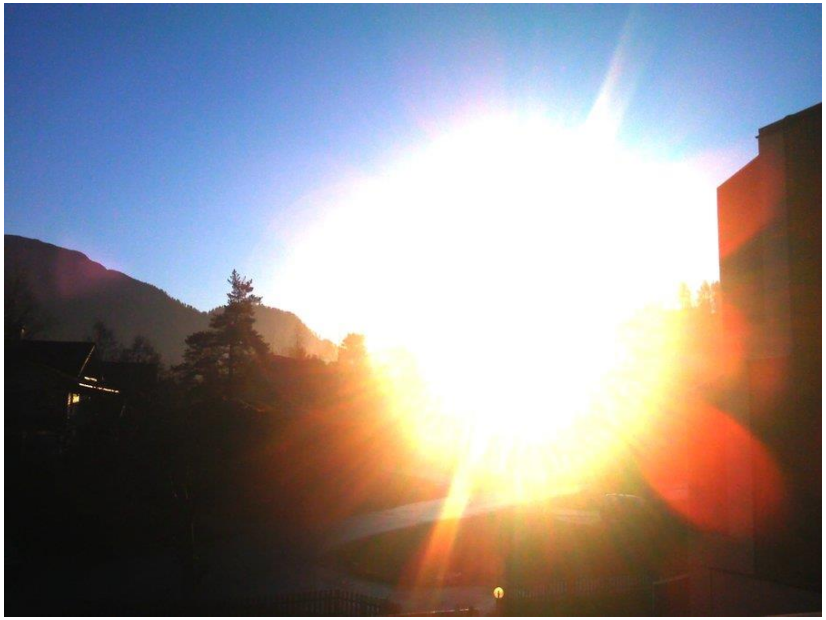
A cube map is only correct for one position

Ignores dynamic objects

HDR

„High dynamic range“

Use more than 32 bits of data for one pixel

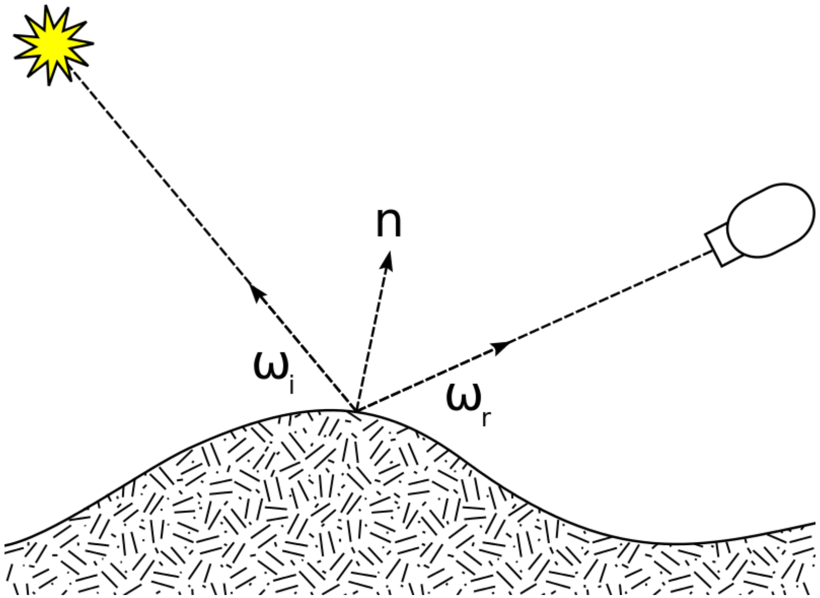


BRDF

Bidirectional reflectance distribution function

- incoming light direction
- outgoing direction (for example to the camera)
- returns the ratio of reflected radiance

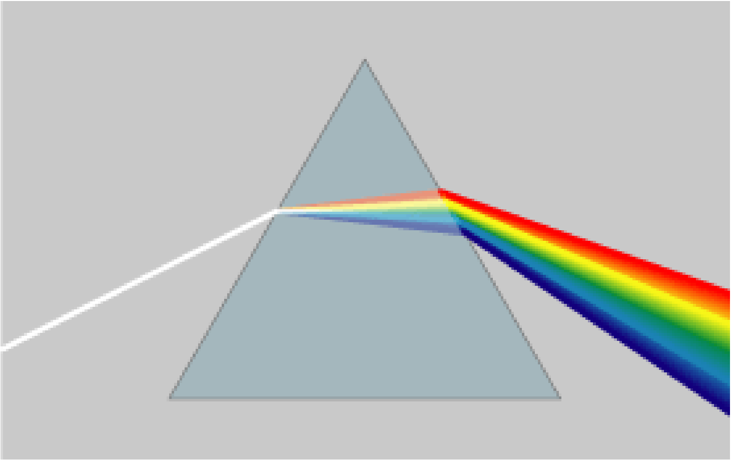
$$f_r(\omega_i, \omega_r)$$



BRDF Shortcomings

Subsurface-Scattering

Wavelength dependence



BRDF

Only positive light

$$f_r(\omega_i, \omega_r) \geq 0$$

BRDF

Inverted

$$\boldsymbol{f_r(\omega_i, \omega_r) = f_r(\omega_r, \omega_i)}$$

BRDF

Energy conserving

$$\forall \boldsymbol{\omega_i}, \int_{\boldsymbol{\Omega}} \boldsymbol{f_r}(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}) \boldsymbol{cos}(\boldsymbol{\theta_r}) \boldsymbol{d\omega_r} \leq \mathbf{1}$$

Phong Lighting

TECHNISCHE UNIVERSITÄT DARMSTADT

color = ambient + diffuse + specular

Gamma Correction / sRGB

TECHNISCHE UNIVERSITÄT DARMSTADT

See Exercise 1

Transform textures to linear (pow 2.2)

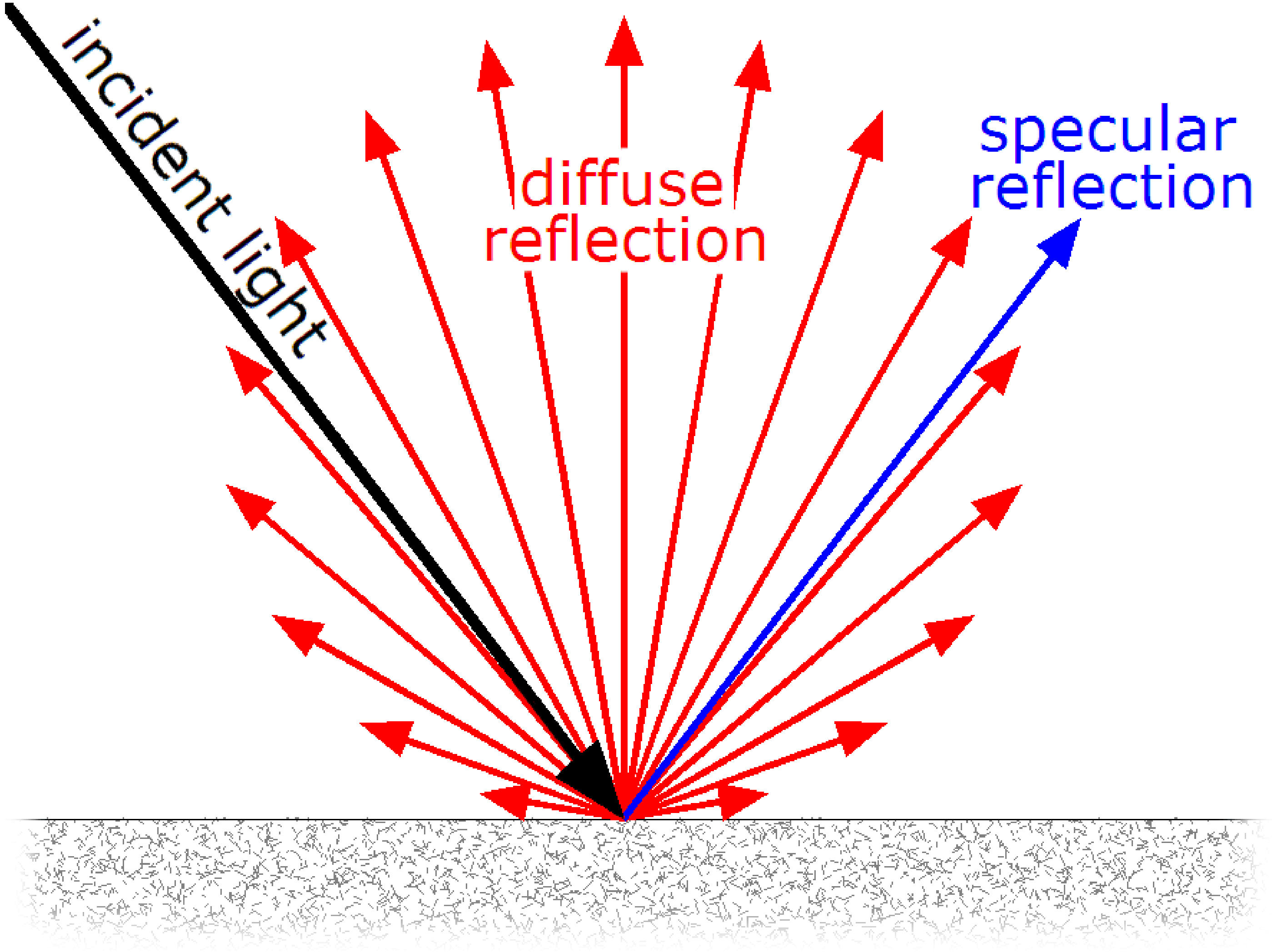
- Or use sRGB texture reading (also allows proper filtering)

Lighting calculations in linear space (gamma 1)

Then transform for sRGB (pow 1 / 2.2)

Diffuse & Specular

TECHNISCHE
UNIVERSITÄT
DARMSTADT



Diffuse

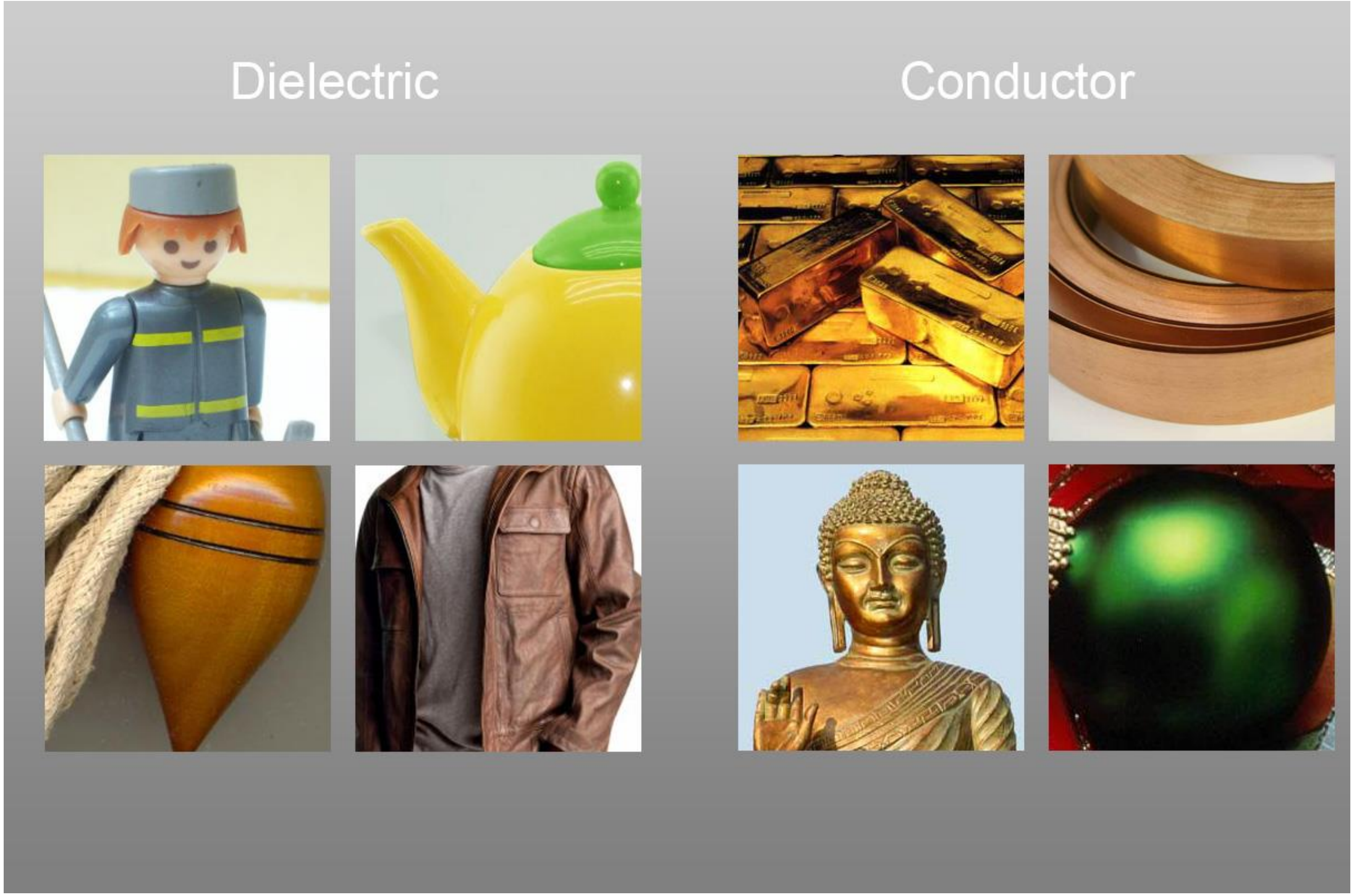
Lambertian reflectance / Phong diffuse

I = L*N

Good enough for modern engines

- Used for example in Unreal Engine 4

Metals and Dielectrics



Metals and Dielectrics

Metals

- No diffuse
- High Specular

Dielectrics

- Diffuse
- Lower but still surprisingly high Specular

Note: Specular value is specified at low angles

Polarization of Reflected Light

Specular Reflection

- Polarization does not change

Diffuse Reflection

- Polarization is randomized

Cardboard

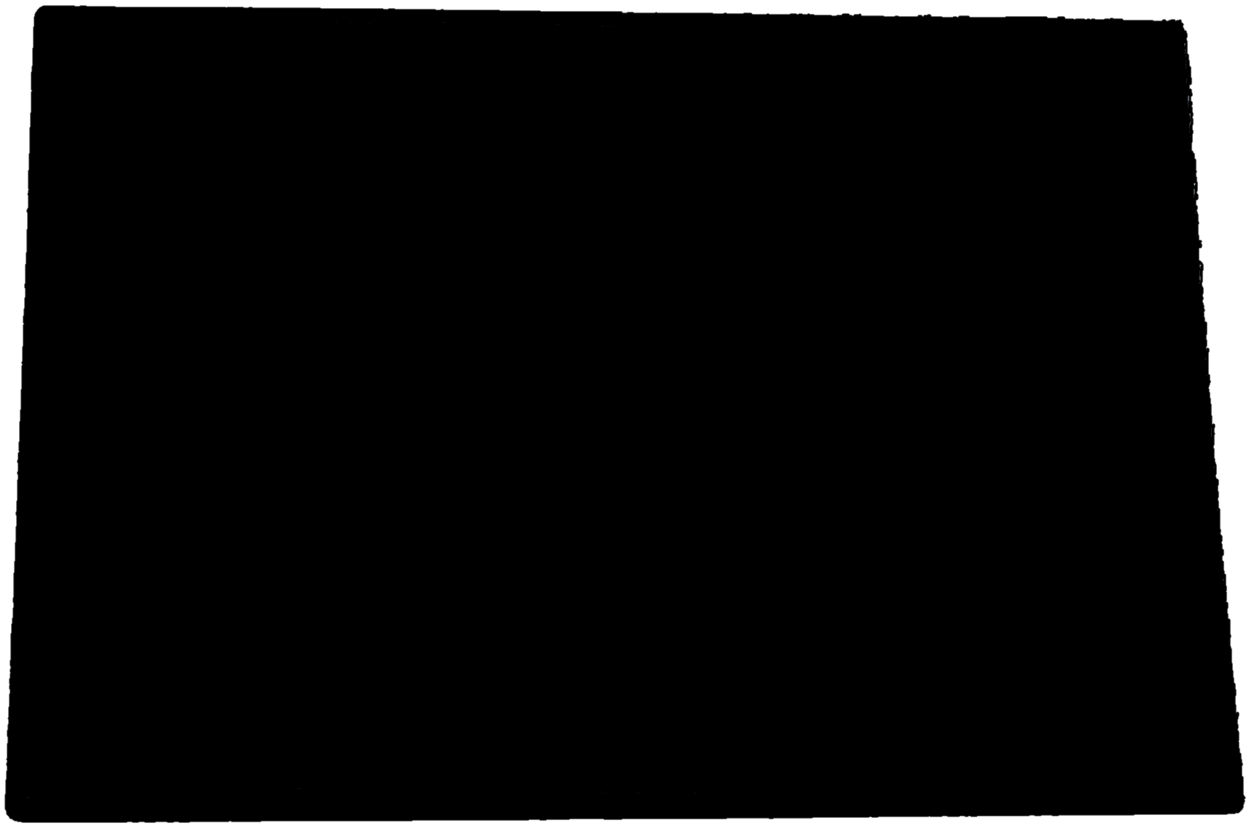
Cardboard Diffuse

Cardboard Specular

Metal

Metal Specular

Metal Diffuse

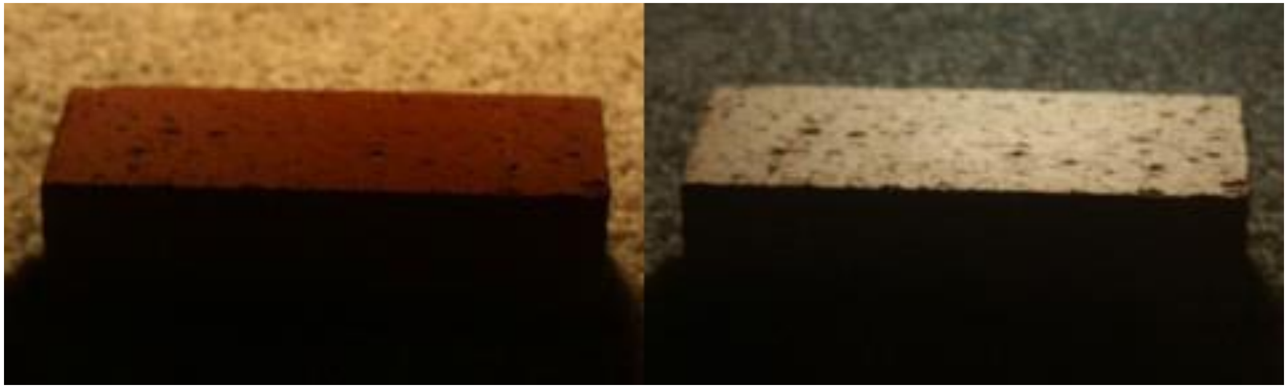


Specularity

angle(normal, light) = angle(normal, camera)

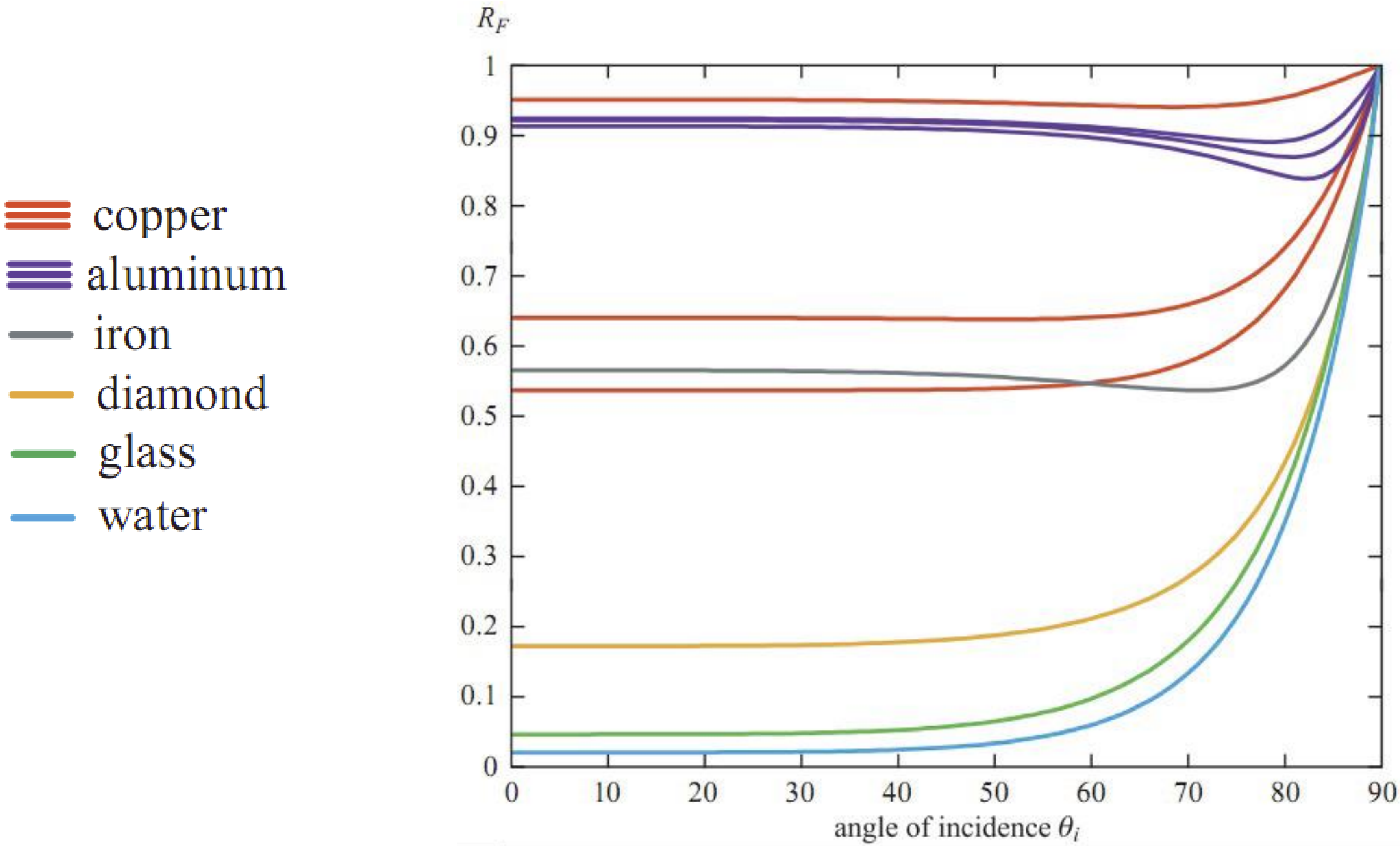
Brick

angle(normal, light) = angle(normal, camera)



Fresnel

TECHNISCHE UNIVERSITÄT DARMSTADT



Fresnel

Schlick Approximation

Schlick(spec, light, normal) = spec + (1 - spec) (1 - (light*normal))^5

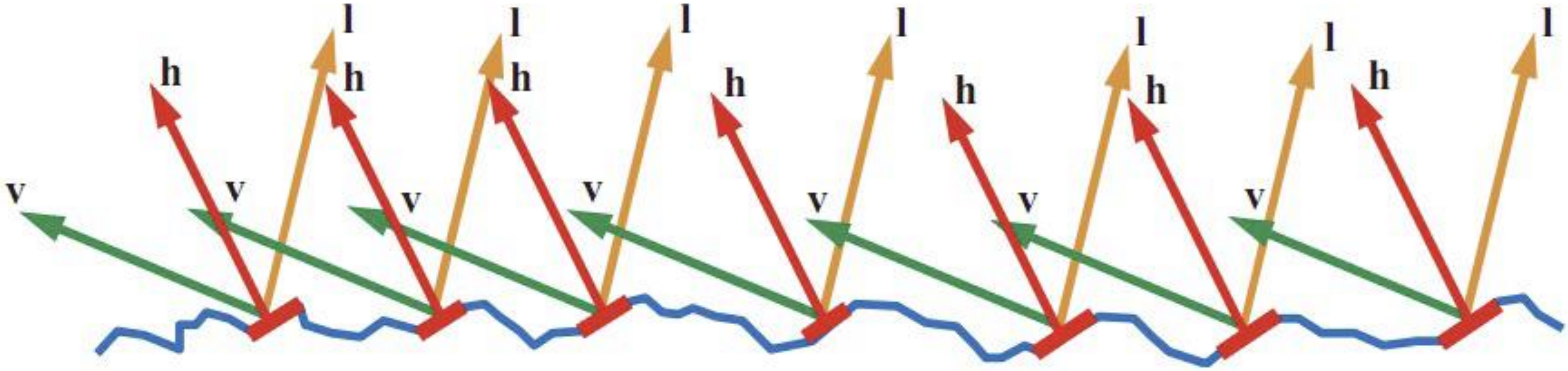
- light*normal = 1 → Schlick = spec
- light*normal = 0 → Schlick = 1

spec

- characteristic specular reflectance
- specular color

Material	$F(0°)$ (Linear)	$F(0°)$ (sRGB)	Color
Water	0.02,0.02,0.02	0.15,0.15,0.15	
Plastic / Glass (Low)	0.03,0.03,0.03	0.21,0.21,0.21	
Plastic High	0.05,0.05,0.05	0.24,0.24,0.24	
Glass (High) / Ruby	0.08,0.08,0.08	0.31,0.31,0.31	
Diamond	0.17,0.17,0.17	0.45,0.45,0.45	
Iron	0.56,0.57,0.58	0.77,0.78,0.78	
Copper	0.95,0.64,0.54	0.98,0.82,0.76	
Gold	1.00,0.71,0.29	1.00,0.86,0.57	
Aluminum	0.91,0.92,0.92	0.96,0.96,0.97	
Silver	0.95,0.93,0.88	0.98,0.97,0.95	

Microfacet Model



Microfacet BRDF

$$f(l,v) = \frac{F(l,h)G(l,v,h)D(h)}{4(n \cdot l)(n \cdot v)}$$

Microfacet BRDF

Fresnel Reflectance

$$f(l,v) = \frac{\boldsymbol{F(l,h)}G(l,v,h)D(h)}{4(n\cdot l)(n\cdot v)}$$

Microfacet BRDF

Active microfacets

$$f(l,v) = \frac{F(l,h)\boldsymbol{G(l,v,h)}\boldsymbol{D(h)}}{4(n \cdot l)(n \cdot v)}$$

Microfacet BRDF

Normal Distribution Function

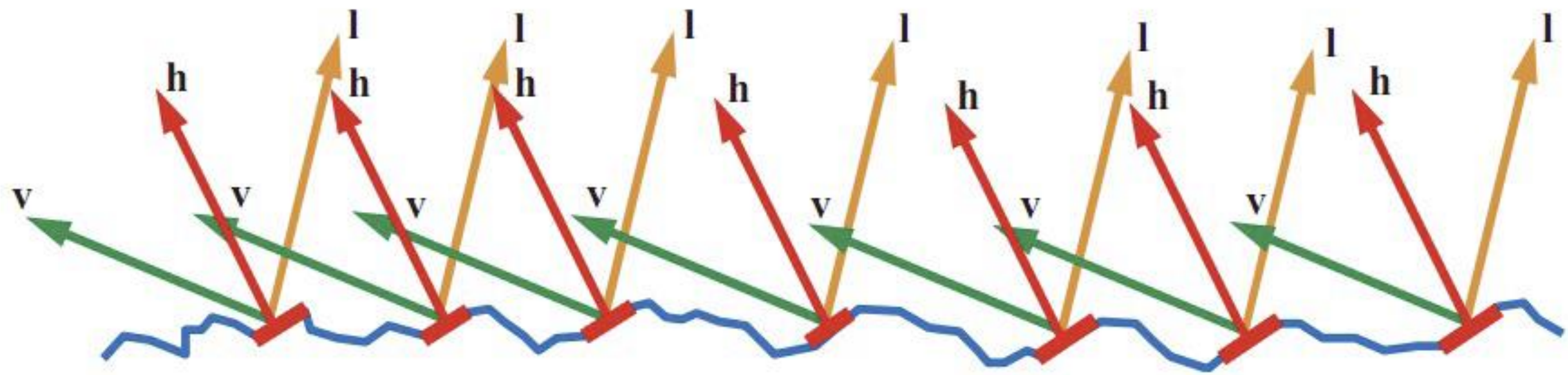
$$f(l,v) = \frac{F(l,h)G(l,v,h)\boldsymbol{D}(\boldsymbol{h})}{4(n \cdot l)(n \cdot v)}$$

Normal Distribution Function D

Evaluated for h

The concentration of microfacets that have an orientation so they could reflect light to the camera

- Might still be occluded, …



Normal Distribution

TECHNISCHE UNIVERSITÄT DARMSTADT

D(h)

Portion of microfacets pointing to h

$$D_{tr}(m) = \frac{\alpha_{tr}^2}{\pi\left((n \cdot m)^2(\alpha_{tr}^2 - 1) + 1\right)^2}$$

Trowbridge-Reitz (GGX)

α: Roughness

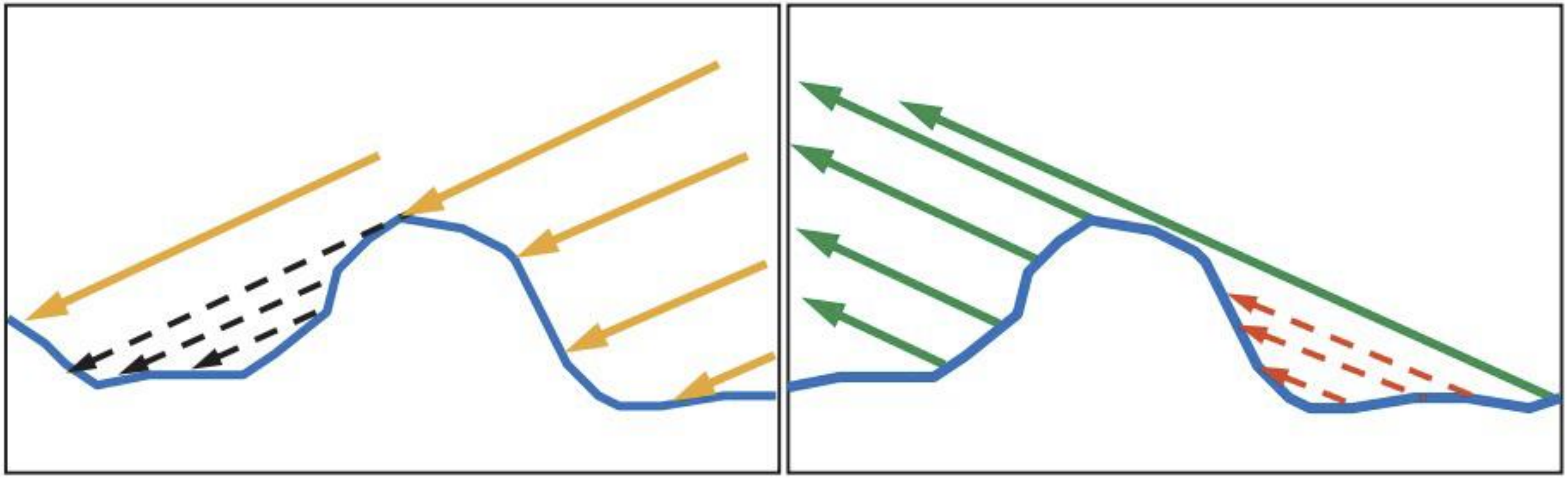
Microfacet BRDF

Shadow Masking Function

$$f(l,v) = \frac{F(l,h)\boldsymbol{G(l,v,h)}D(h)}{4(n \cdot l)(n \cdot v)}$$

Shadow Masking Function G

- Also referred to as the Geometry Function
- Probability that microfacets with normal h are visible from both the light and the view direction



Geometry Factor

- **G(l, v, h)**
- **Cook-Torrance:**

$$G_{ct}(l, v, h) = \min\left(1, \frac{2(n \cdot h)(n \cdot v)}{(v \cdot h)}, \frac{2(n \cdot h)(n \cdot l)}{(v \cdot h)}\right)$$

Microfacet BRDF

TECHNISCHE UNIVERSITÄT DARMSTADT

$$f(l,v) = \frac{F(l,h)G(l,v,h)D(h)}{4(n \cdot l)(n \cdot v)}$$

Physically Based Rendering

TECHNISCHE UNIVERSITÄT DARMSTADT

In practice:

- Gamma correction
- Microfacet BRDF
- Lots of Cube Maps

Textures

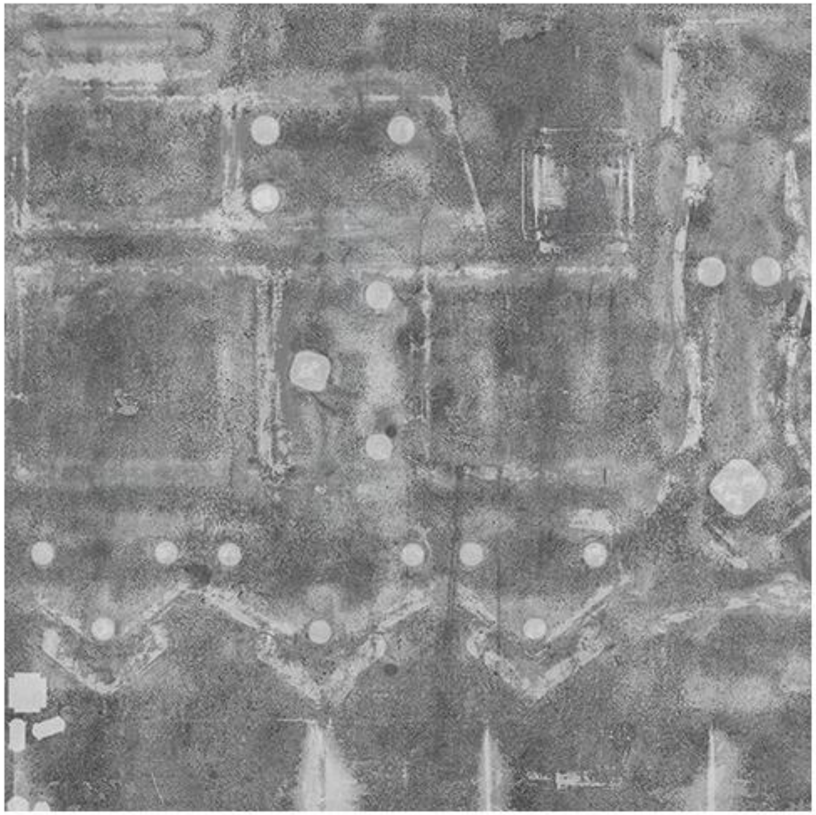
Typical Setup

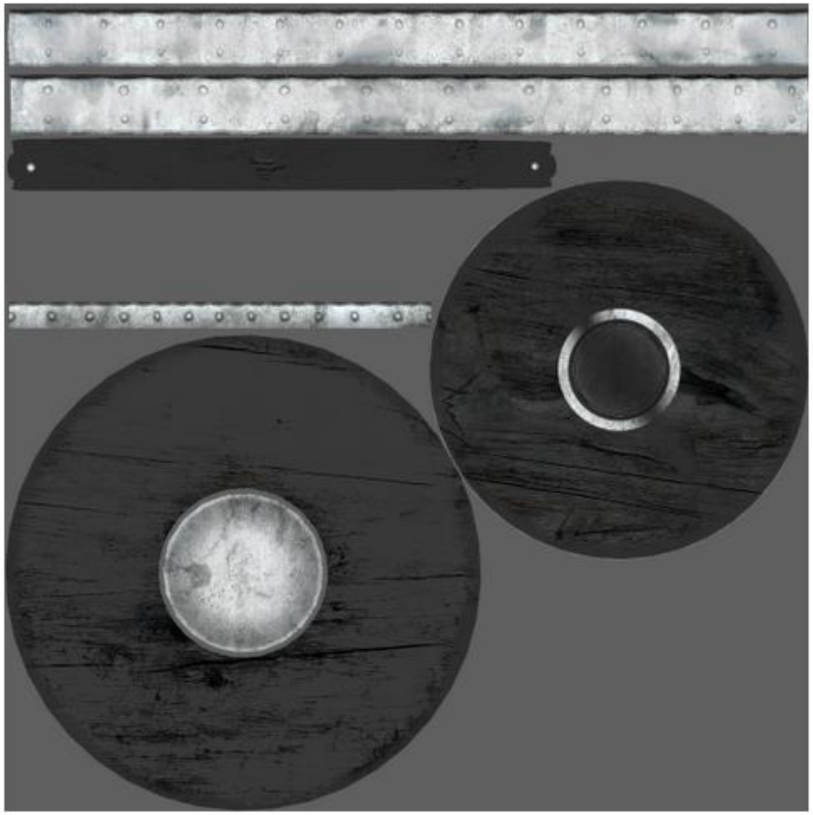
- Diffuse texture
- Specular texture
- Roughness texture
- Normal Map

Diffuse Texture

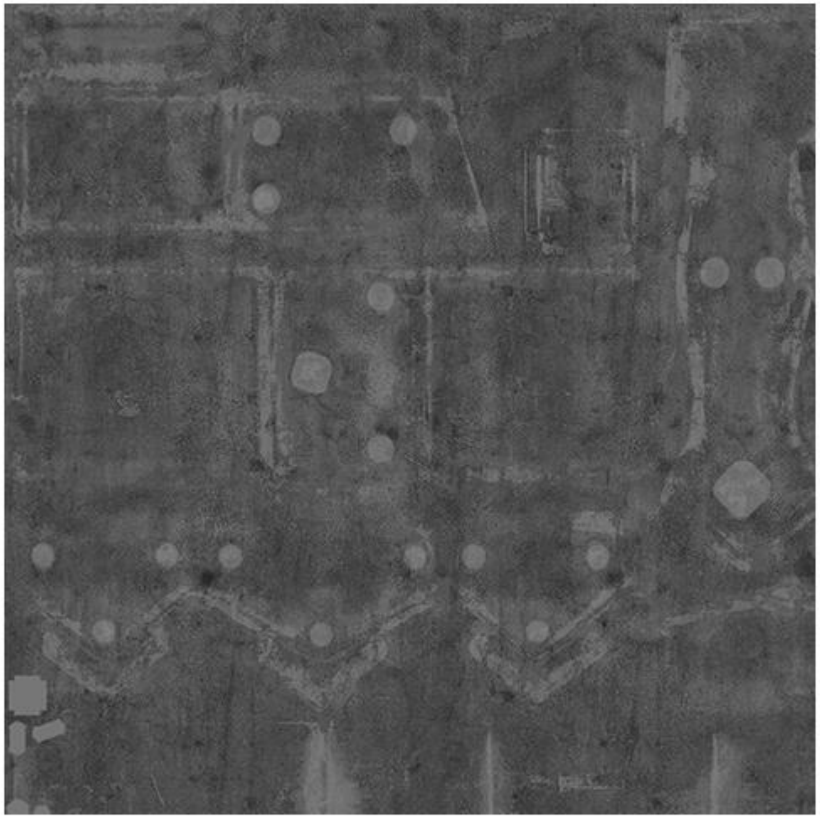
TECHNISCHE UNIVERSITÄT DARMSTADT

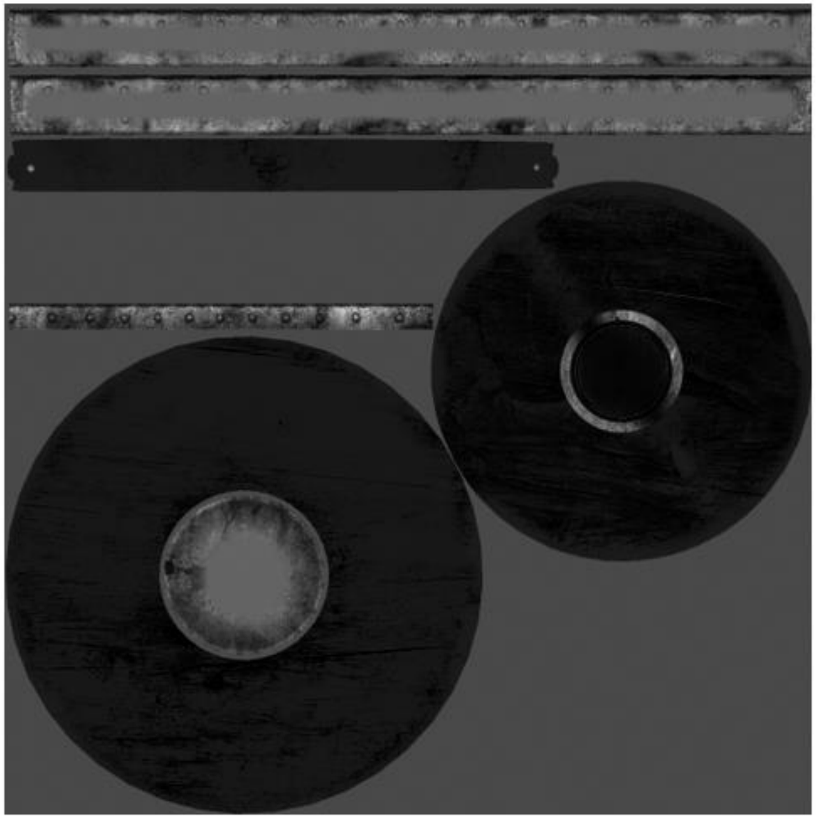
Specular Texture





Smoothness Texture



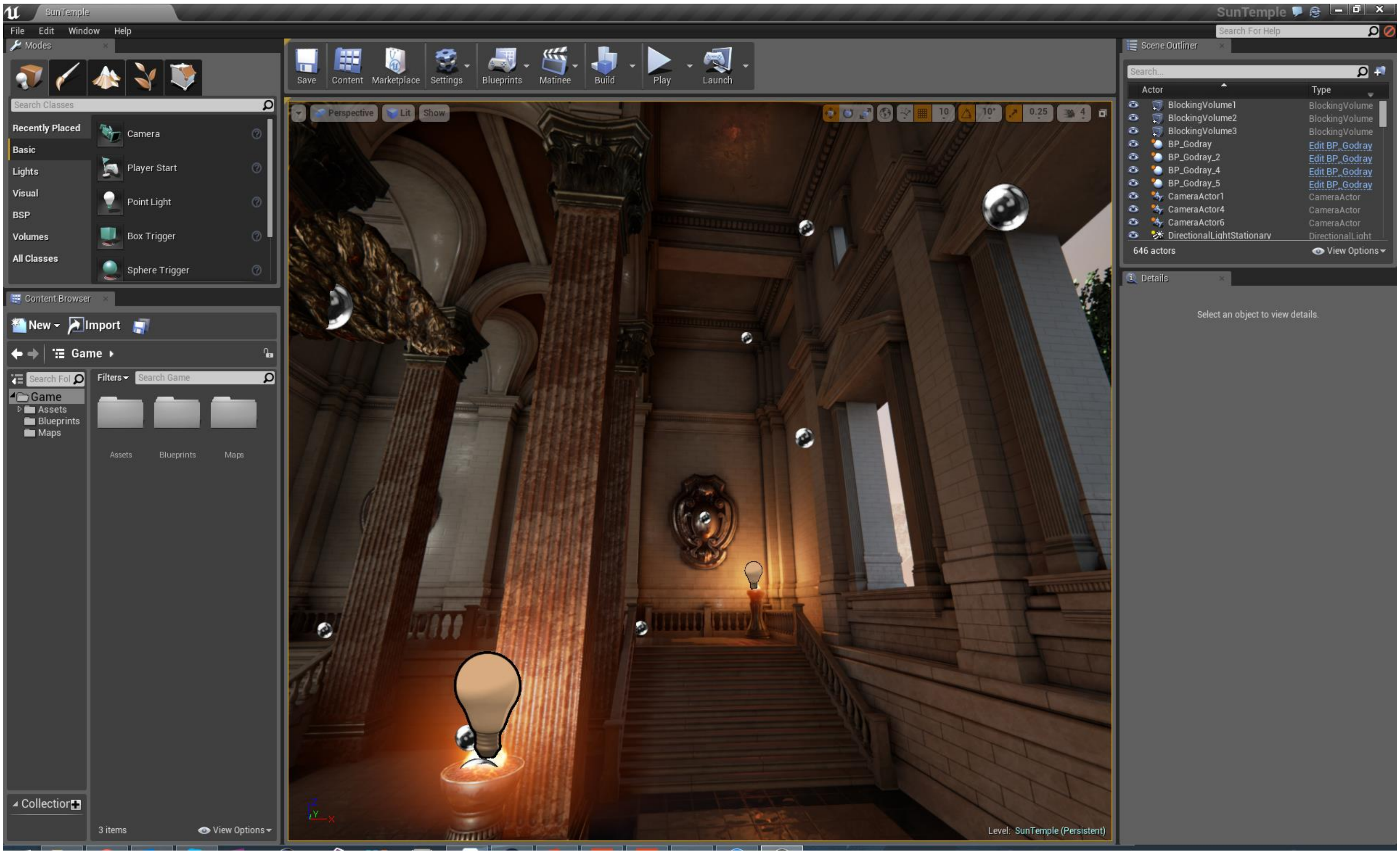


Incorporating Image Based Lighting

Precalculate Cube Maps

- Lots of Cube Maps
- Manually placed in level editor

Cube Maps


SunTemple
File
Edit
Window
Help
Modes
Search Classes
Recently Placed
Basic
Lights
Visual
BSP
Volumes
All Classes
Camera
Player Start
Point Light
Box Trigger
Sphere Trigger
Content Browser
New
Import
Game
Filters
Search Game
Assets
Blueprints
Maps
Collections
3 items
View Options
Save
Content
Marketplace
Settings
Blueprints
Matinee
Build
Play
Launch
Perspective
Lit
Show
Level: SunTemple (Persistent)
Search For Help
Scene Outliner
Actor
Type
BlockingVolume1
BlockingVolume2
BlockingVolume3
BlockingVolume
BP_Godray
BP_Godray_2
BP_Godray_4
BP_Godray_5
Edit BP_Godray
CameraActor1
CameraActor4
CameraActor6
CameraActor
DirectionalLightStationary
DirectionalLight
646 actors
Details
Select an object to view details.

Cube Maps

TECHNISCHE UNIVERSITÄT DARMSTADT

Can be interpolated

- Which is a rough approximation

Can not capture dynamic objects

Reflection Rendering

TECHNISCHE UNIVERSITÄT DARMSTADT

As done in Unreal Engine 4, Killzone Shadow Fall,…

Deferred Rendering Pass

Raytrace depth buffer

If no hit

- Interpolate local cube maps

If no hit

- Use global cube map

Ambient Occlusion

Small notches are normally shadowed

- Unless lit directly

Calculations need very exact light bounces

Screen Space Ambient Occlusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Filter after rendering**
- **Darken at sharp normal changes**

Global Illumination

TECHNISCHE UNIVERSITÄT DARMSTADT

„Ambient light“

Spherical Harmonic Lighting

Voxel Cone Tracing

…

Summary

Stay closer to nature

- Energy conservation
- Metals vs. Dielectrics
- Textures, parameters can be capured from nature

More reusable assets

- No specific lighting information baked/painted into textures

Physically Based Shading in Theory and Practice Course @SIGGRAPH

http://blog.selfshadow.com/publications/

GPU Internals

NVidia GeForce GT 6600, 2004

Memory

Memory bandwidth is extremely important

- Textures
- Framebuffer
- Depth Buffer

Memory access times not very important

- Most data is streamed
- Access times can be hidden by switching tasks

Memory

Gigantic discrepancy from low-end to high-end

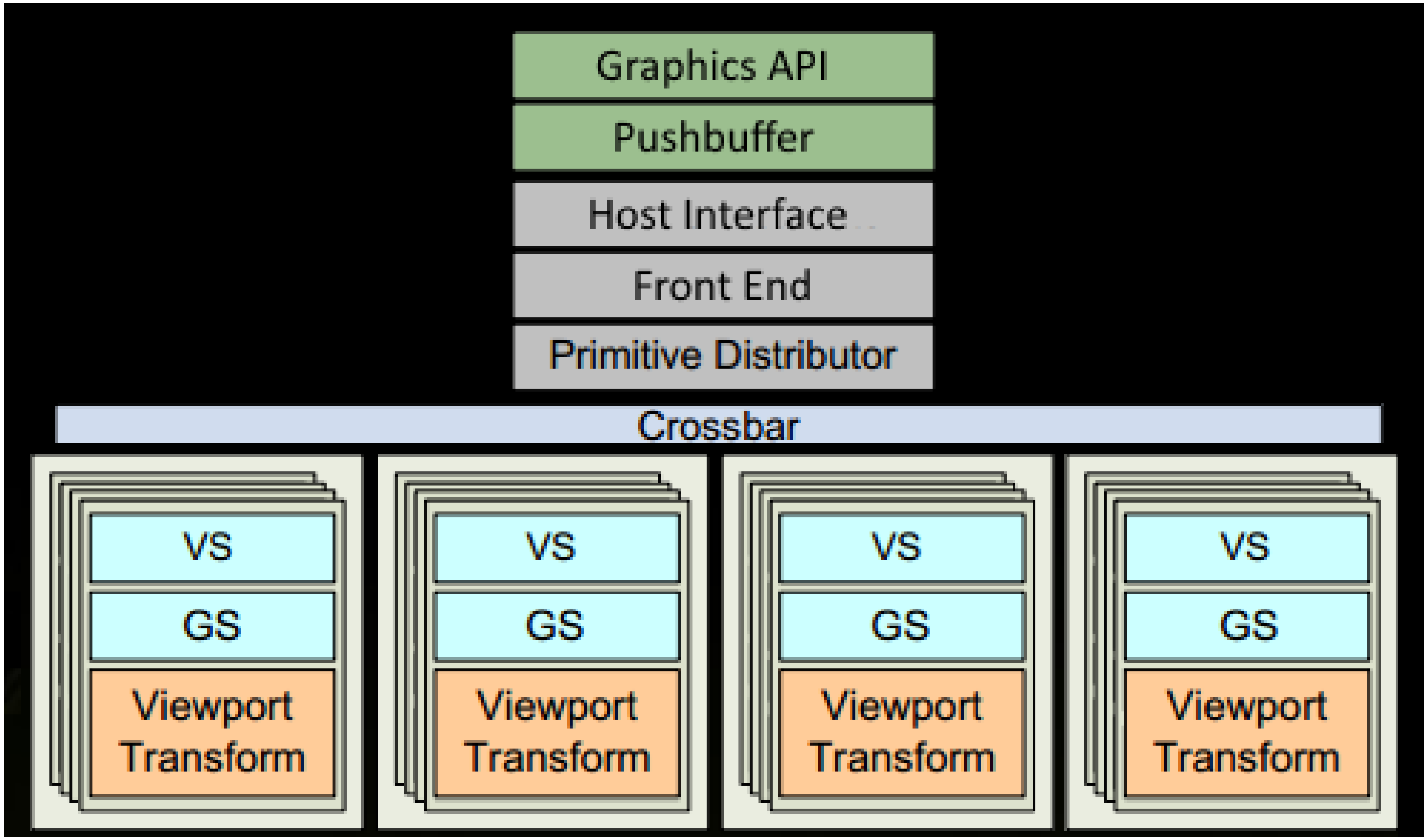
- High End CPU: 60 GB/s
- GeForce 1030: 48 GB/s
- PS4: 176 GB/s
- GeForce 1080 Ti: 484 GB/s

Vertex and Fragments Shaders

Run on the same hardware

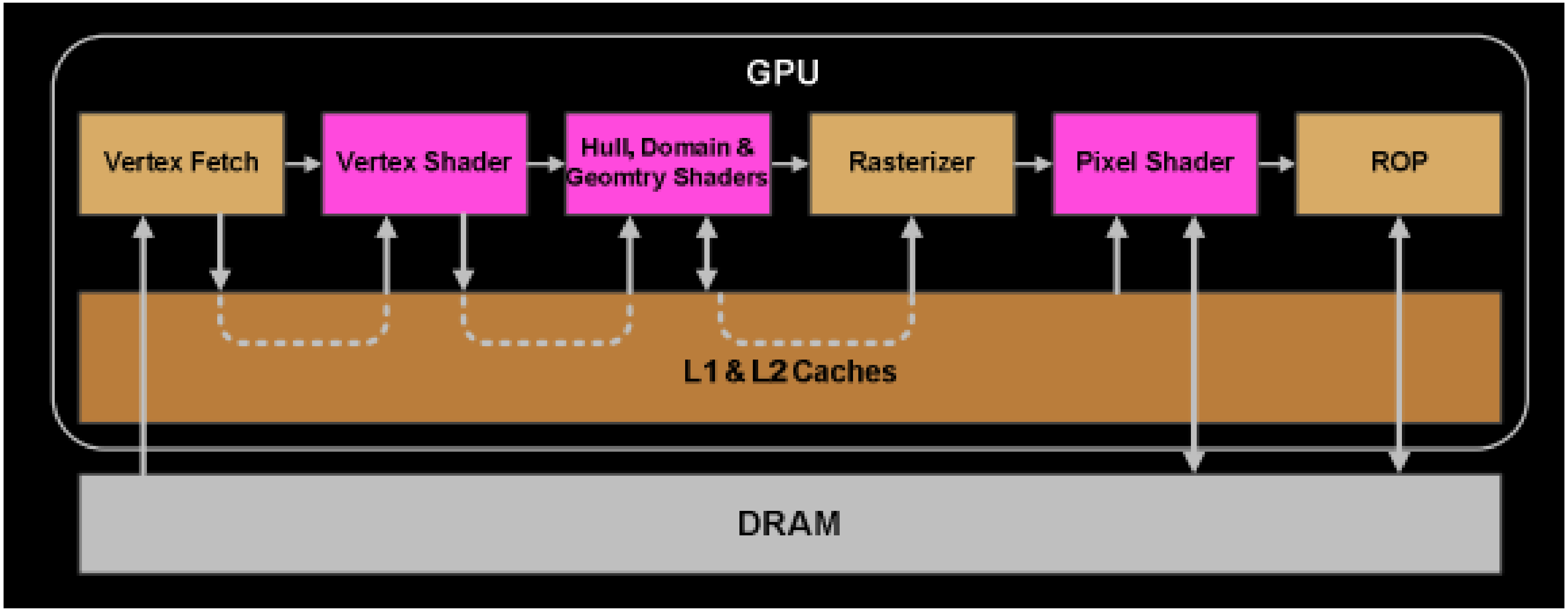
Dynamically scheduled

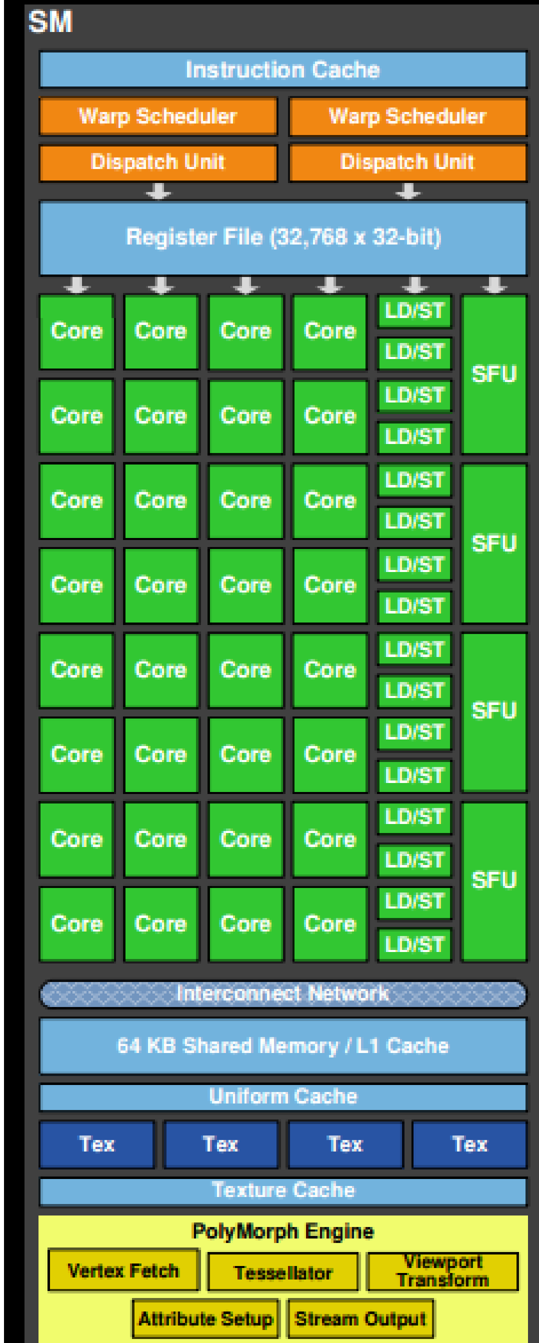
CPU - GPU

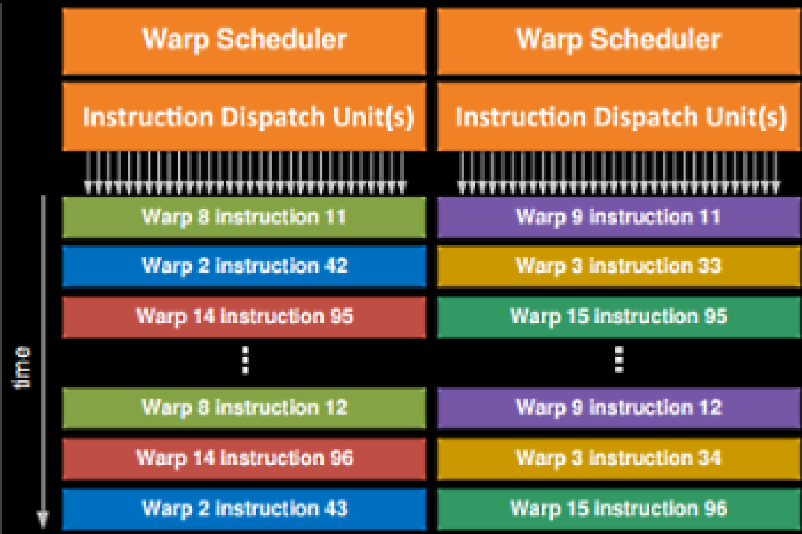


https://developer.nvidia.com/content/life-triangle-nvidias-logical-pipeline

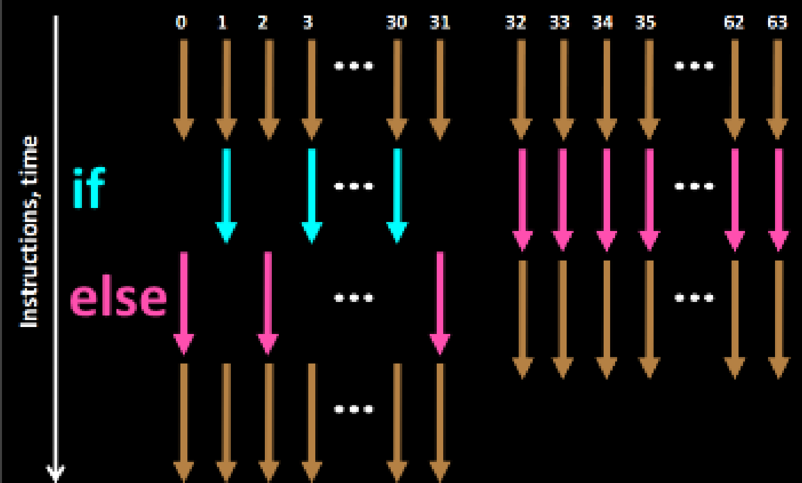
GPU


GPU
Vertex Fetch
Vertex Shader
Hull, Domain &
Geomtry Shaders
Rasterizer
Pixel Shader
ROP
L1 & L2 Caches
DRAM



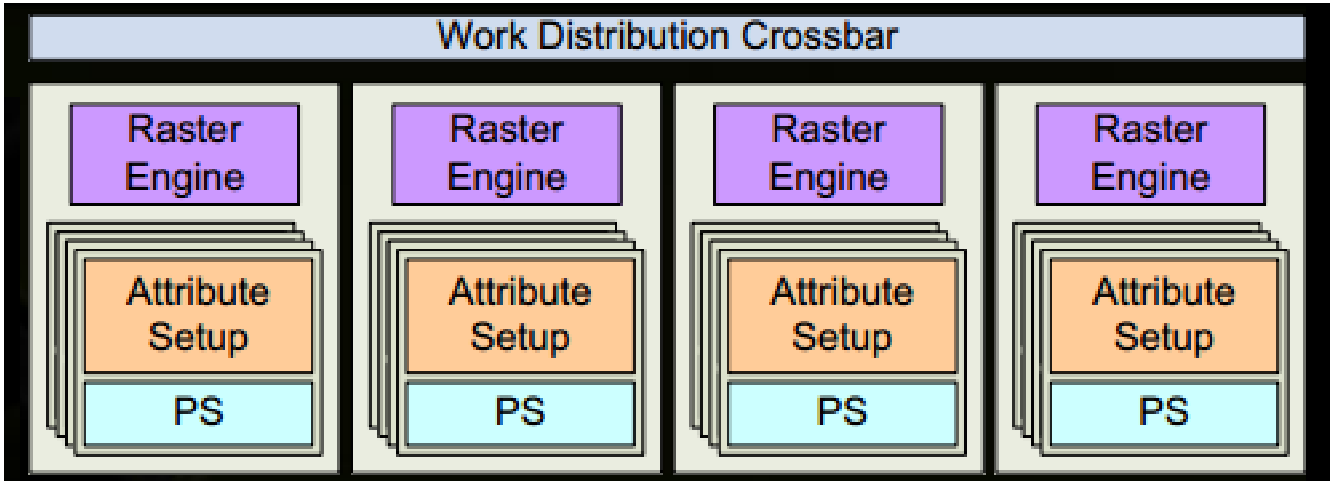


Warps (32 threads group) advance as a whole, with instructions managed by scheduler.

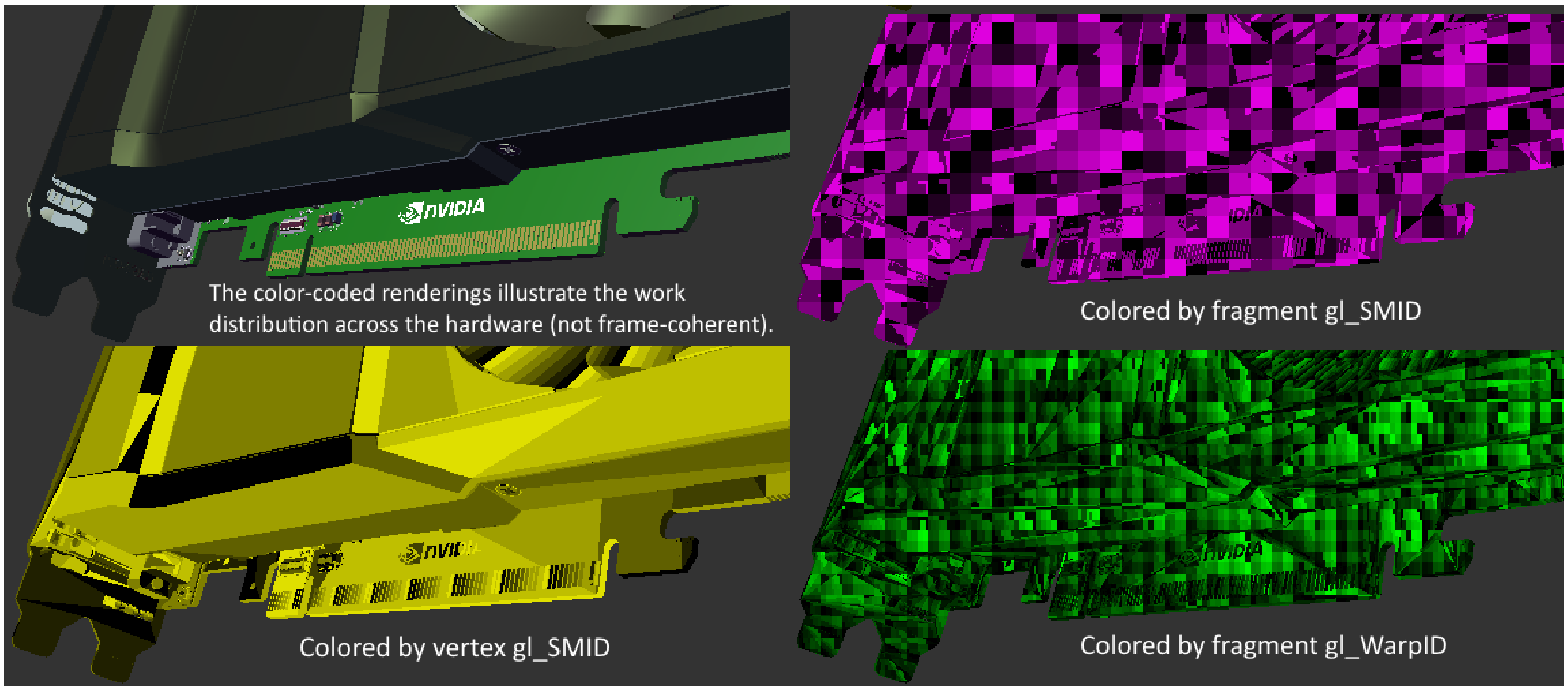


Due to this lock-step behavior the divergency in the left warp causes longer execution. The threads cannot advance individually.

Rasterization, Pixel Shader



Work distribution



SMT (Symmetric Multithreading)

Also used in CPUs (Hyperthreading)

Switch to different thread when stalled (for example waiting for memory)

SIMD

Compiler can put calculations on multiple vertices/pixels in one instruction

Problem: Flow control

- Wrong paths pseudo-executed
  - Can be efficient when all vertices/pixel in one pack take the same paths

Shader variants

- „Typically when you create a simple surface shader, it internally expands into 50 or so internal shader variants” (Shader Compilation in Unity 4.5)

Parallelization

TECHNISCHE UNIVERSITÄT DARMSTADT

Small work packages can prevent parallelization

- Performance dip for tiny triangles

CPU ←→ GPU

TECHNISCHE UNIVERSITÄT DARMSTADT

Biggest performance trap

Minimize state changes

Minimize draw calls

Send little data to the GPU

If possible never read data from the GPU

Summary

Broad overview of GPU internals

- CPU – GPU interaction
- Work distribution
- Massively parallel execution (vertex, pixel shaders)
- Multithreading and SIMD