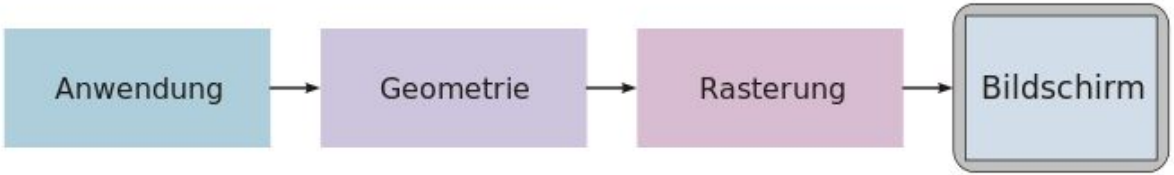
Shader

Beleuchtungsmodelle,
C for Graphics,
Unity Shader Graph

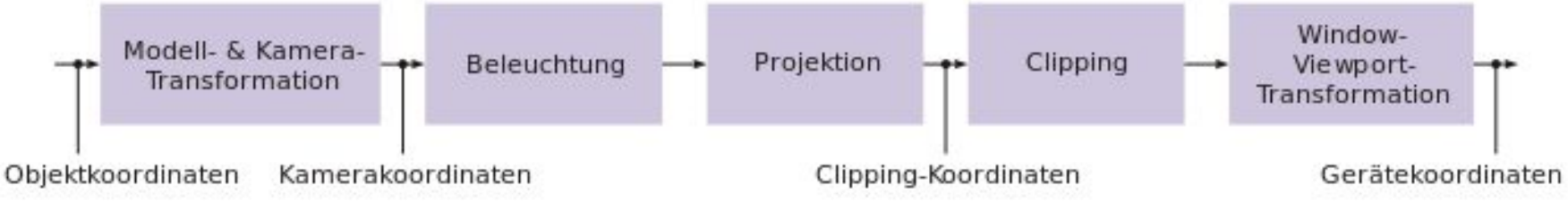
Graphikpipeline

- Anwendung
  - Content, Logik
  - Interaktion
- Geometrie
  - Modellbeschreibung
  - Transformation
  - Shader (3D)
- Rasterung
  - Postprocessing (2D)
- Bildschirm
  - Ausgabe auf Endgerät
  - Responsive



(Quelle: Wikipedia)

Geometrie



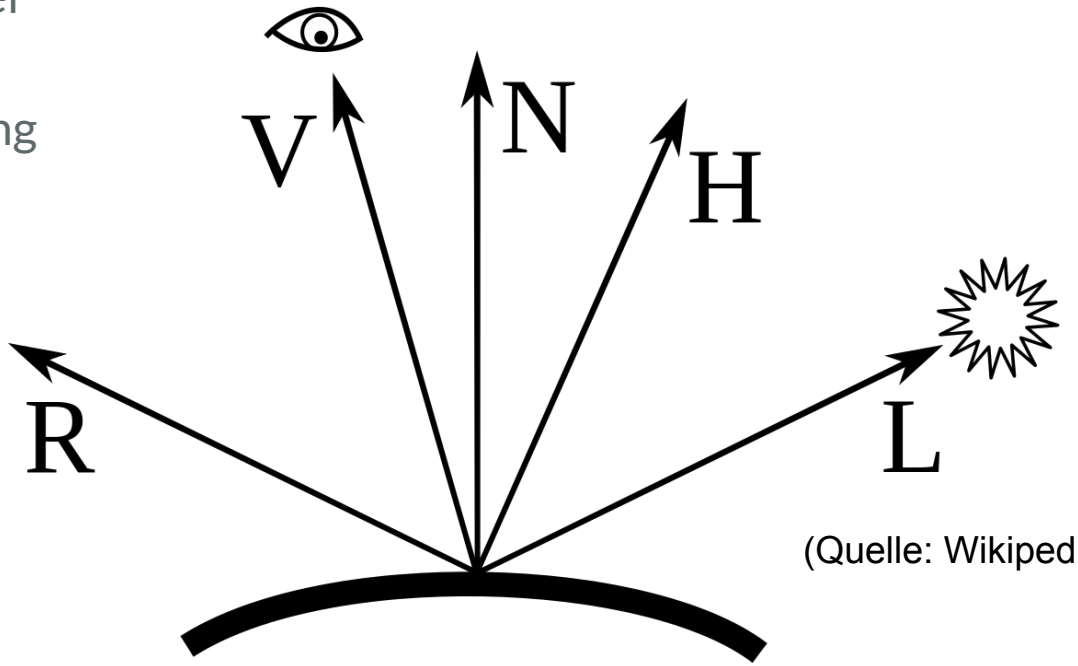
(Quelle: Wikipedia)

- Beleuchtung
  - Lichtquellen, Global Illumination, Real-time vs. Baked Lighting
  - Farbe, Farbsysteme
  - Beleuchtungsmodelle, Shader

Beleuchtungsmodelle

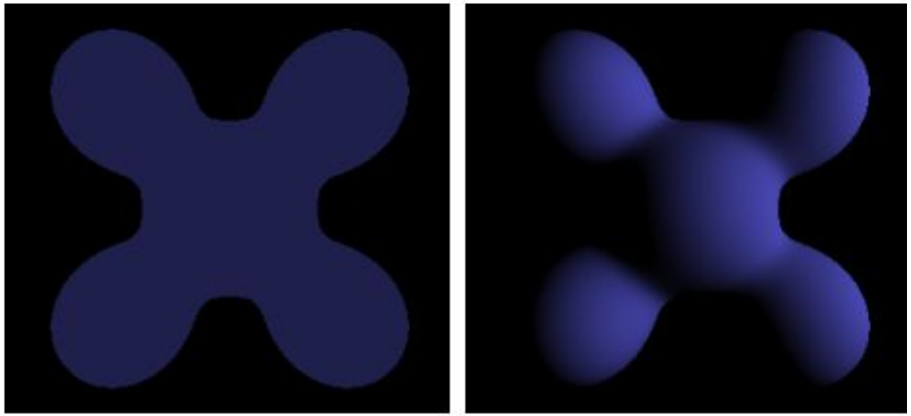
Beleuchtung

- Lambert: Einfallswinkel
- Phong: Reflexionen
- Blinn: Vereinfachung



(Quelle: Wikipedia)

Lambert Beleuchtung



(Quelle: Wikipedia)

Ambiente Beleuchtung:

$$I_{\text{ambient}} = I_{\text{a}}\, k_{\text{ambient}}$$

mit

- I_{a} ... Intensität des Umgebungslichts
- k_{ambient} ... Materialkonstante

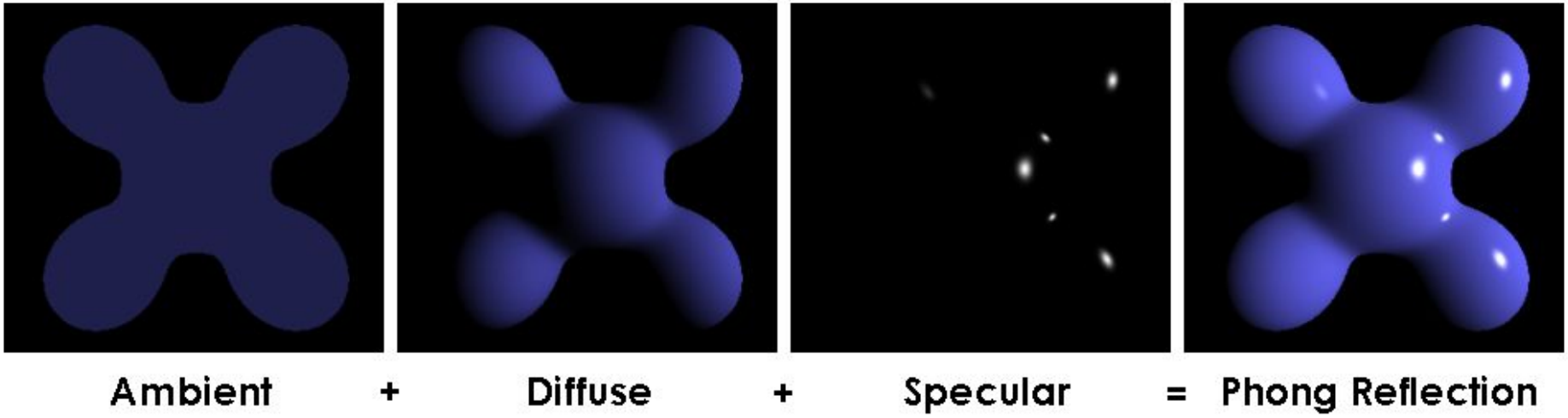
Diffuse Beleuchtung:

$$\begin{aligned} I_{\text{diffus}} &= I_{\text{in}}\, k_{\text{diffus}}\, \cos\varphi \\ &= I_{\text{in}}\, k_{\text{diffus}}\, (\vec{L} \cdot \vec{N}) \end{aligned}$$

mit

- I_{in} ... Lichtstärke des einfallenden Lichtstrahls
- k_{diffus} ... empirisch bestimmter Reflexionsfaktor
- φ ... Winkel zwischen Normalenvektor

Phong Beleuchtung



(Quelle: Wikipedia)

Specular:

$$
\begin{aligned}
I_{\text{specular}} &= I_{\text{in}}\, k_{\text{specular}}\, \cos^n \theta \\
&= I_{\text{in}}\, k_{\text{specular}}\, (\vec{R} \cdot \vec{V})^n
\end{aligned}
$$

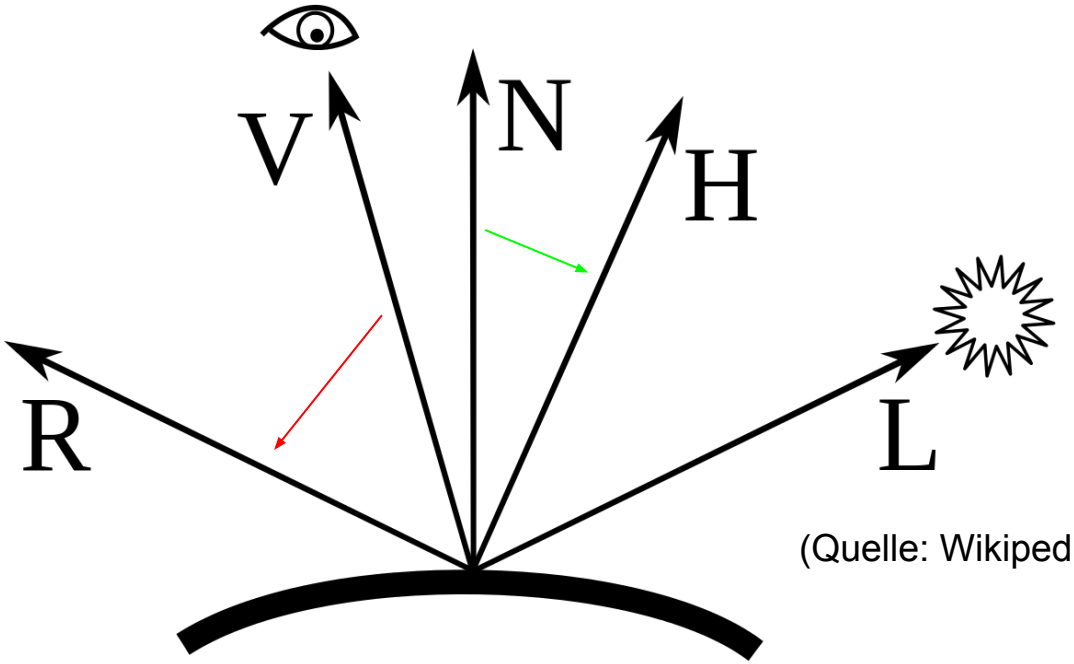
mit

- I_{in} ... Lichtstärke des einfallenden Lichtstrahls der Punktlichtquelle
- k_{specular} ... empirisch bestimmter Reflexionsfaktor für spiegelnde Komponente der Reflexion
- θ ... Winkel zwischen idealer Reflexionsrichtung des ausfallenden Lichtstrahls $\vec{R}$ und Blickrichtung des Betrachters $\vec{V}$
- n ... konstanter Exponent zur Beschreibung der Oberflächenbeschaffenheit

Blinn-Phong Beleuchtung

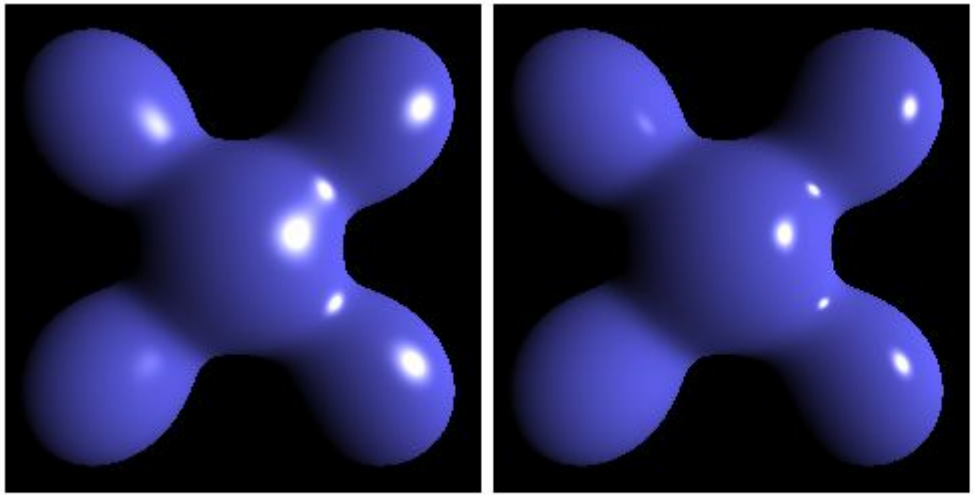
- Phong: cos(theta) = R * V
- Blinn: cos(theta') = N * H

- spart Berechnung von R!



(Quelle: Wikipedia)

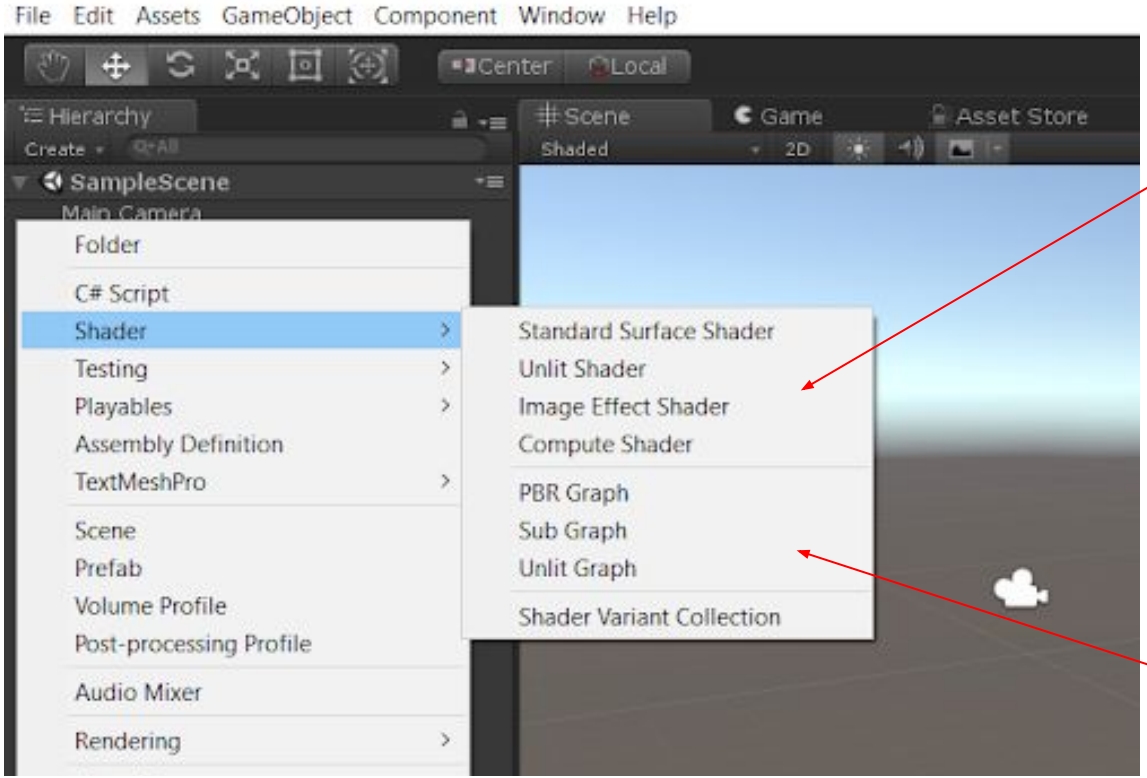
Blinn-Phong Beleuchtung



(Quelle: Wikipedia)

C for Graphics (CG)

Shader



Standard, programmierbar

Shadergraph

Shader

- **Properties**
  - aus Unity Inspector
  - Deklaration im Programm
- **CG (C for Graphics) Programm**
  - aka HLSL (High Level Shader Language)
  - Input aus dem Modell
  - “surf”: Berechnungsblock
- **FallBack**
  - Default Shader
- **Lambert**
  - Beleuchtungsmodell
  - Standard: Blinn-Phong

```
Shader "HSA/SimpleSurfaceShader" {

    Properties {
        _myColour ("Example Colour", Color) = (1,1,1,1)
        _myEmission ("Example Emission", Color) = (1,1,1,1)
    }

    SubShader {

        CGPROGRAM
            #pragma surface surf Lambert

            struct Input {
                float2 uvMainTex;
            };

            fixed4 _myColour;
            fixed4 _myEmission;

            void surf (Input IN, inout SurfaceOutput o){
                o.Albedo = _myColour.rgb;
                o.Emission = _myEmission.rgb;
            }

        ENDCG
    }

    FallBack "Diffuse"
}
```

= “surface” shader, Typ: Lambert

Properties

```
Shader "HSA/AllProps"
{
    Properties {
        _myColor ("Example Color", Color) = (1,1,1,1)
        _myRange ("Example Range", Range(0,5)) = 1
        _myTex ("Example Texture", 2D) = "white" {}
        _myCube ("Example Cube", CUBE) = "" {}
        _myFloat ("Example Float", Float) = 0.5
        _myVector ("Example Vector", Vector) = (0.5,1,1,1)
    }
```

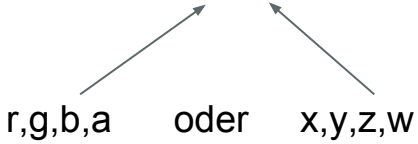
Properties Datentypen

- float 32-bit floating point number
- half 16-bit floating point number
- int 32-bit integer
- fixed 12-bit fixed point number
- bool boolean variable
- sampler* texture object

```
fixed4 _myColor;
half _myRange;
sampler2D _myTex;
samplerCUBE _myCube;
float _myFloat;
float4 _myVector;
```

Einschub: Packed Arrays

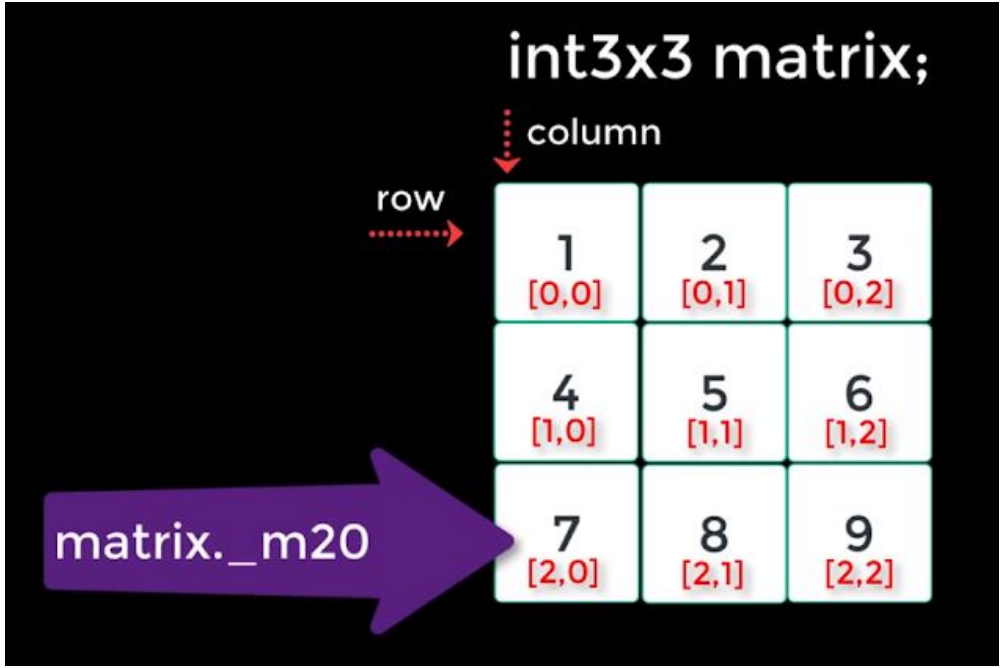
```
fixed4 colour1 = (0, 1, 1, 0);

fixed4 colour2 = (0, 1, 1, 0);
```



Häufig:

```
fixed3 colour3;
colour3 = colour1.rgb;
```

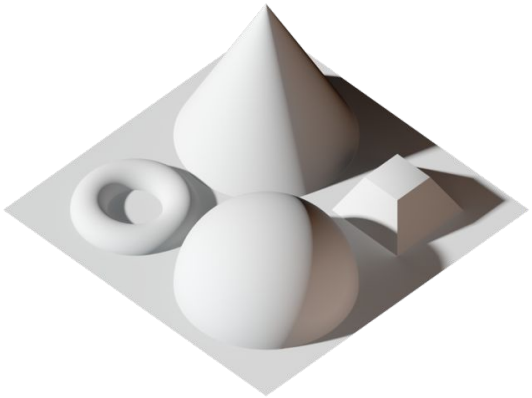
Einschub: Packed Matrices

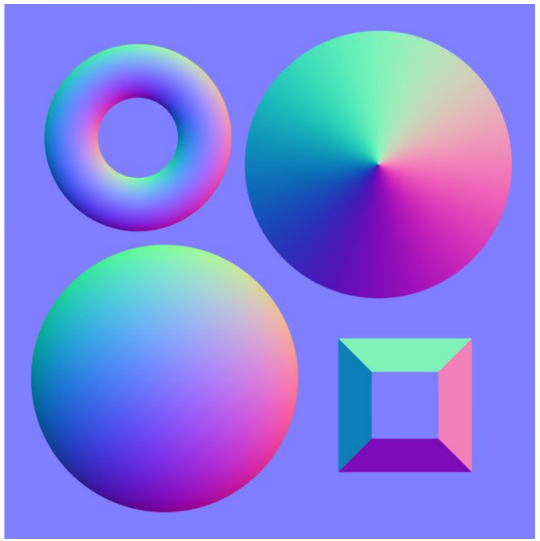


Beispiel: Normal Map

- **Texture2D (= fixed4 Array)**

- **R** 0..255 X: -1 .. +1
- **G** 0..255 Y: -1 .. +1
- **B** 128..255 Z: 0 .. +1





(Quelle: Wikipedia)

Input Structure

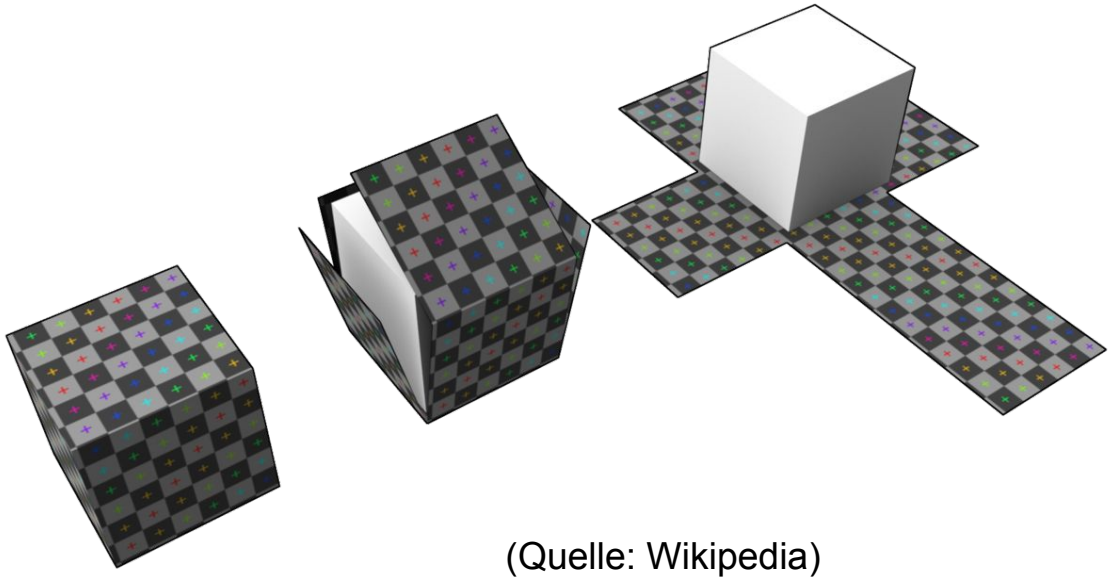
- float2 uv_MainTex;
- float2 uv2_MainTex;
- float3 viewDir; // Betrachtungswinkel
- float3 worldPos; // Modellkoordinaten (3D)
- float3 worldRefl; // Reflektionen

```
struct Input {
    float2 uv_mainTex;
    float2 uv2_mainTex;
    float3 viewDir;
    float3 worldPos;
    float3 worldRefl;
};
```

Beispiel: UV Map

- Zwischen 0 und 1
- Vertex-spezifisch

- float2 uv_MainTex;



(Quelle: Wikipedia)

Surface Output

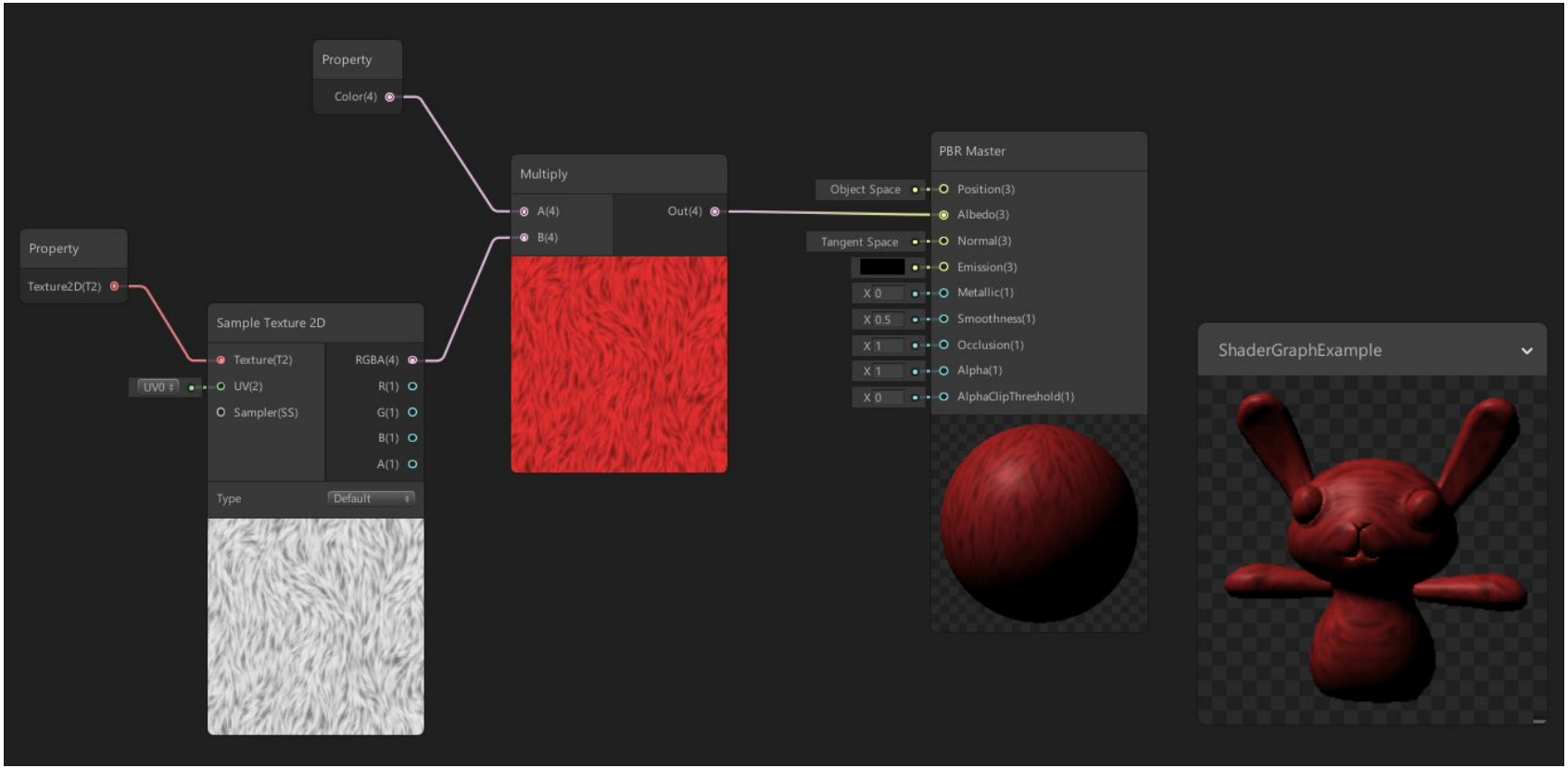
- fixed3 Albedo; // diffuse color
- fixed3 Normal; // Normalenvektor
- fixed3 Emission; // Leuchtkraft
- half Specular; // Blinn-Phong (Specular Highlight)
- fixed Gloss; // Strength of Specular Reflection
- fixed Alpha; // Transparency

Shader Graph

Voraussetzungen

- Lightweight Renderpipeline
- Create/Rendering/Lightweight Pipeline Asset
- Edit/Project Settings/Graphics → Drag & Drop Asset

Beispiel


Property
Color(4)
Multiply
A(4)
B(4)
Out(4)
PBR Master
Object Space
Position(3)
Albedo(3)
Tangent Space
Normal(3)
Emission(3)
X 0
Metallic(1)
X 0.5
Smoothness(1)
X 1
Occlusion(1)
X 1
Alpha(1)
X 0
AlphaClipThreshold(1)
Property
Texture2D(T2)
Sample Texture 2D
Texture(T2)
RGBA(4)
UV0
UV(2)
R(1)
Sampler(SS)
G(1)
B(1)
A(1)
Type
Default
ShaderGraphExample