走近卡通渲染

- 关于Trick的二三事

@MIZI

Index

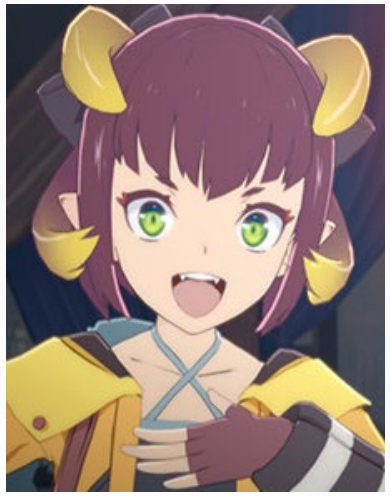
- 基于Trick的卡通世界
- 基于PBR的卡通表现

- 基于Trick的卡通世界
  - 概述
  - 基础表现「光照、描边」
- 基于PBR的卡通表现
  - PBR的优势
  - NPR与PBR的结合
  - Shading model
    - 面部 头发 眼睛 皮肤
- 其他
  - 面向镜头的形变
  - Tonemapping

基于Trick的卡通世界





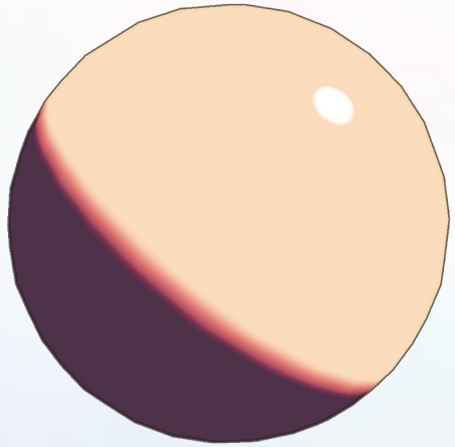






卡通渲染 Cel Shading | Toon Shading

- 基本表现要素：阴影、描边、高光
- 美术表现点：形态、色彩



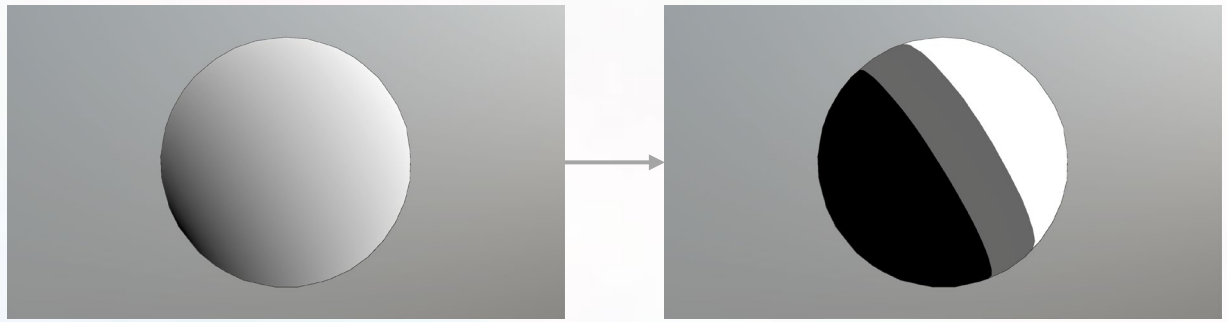
概述

修改光照模型的表现

- 对Diffuse的修改

  Step | Smoothstep 函数

  RampMap



基础表现

光照

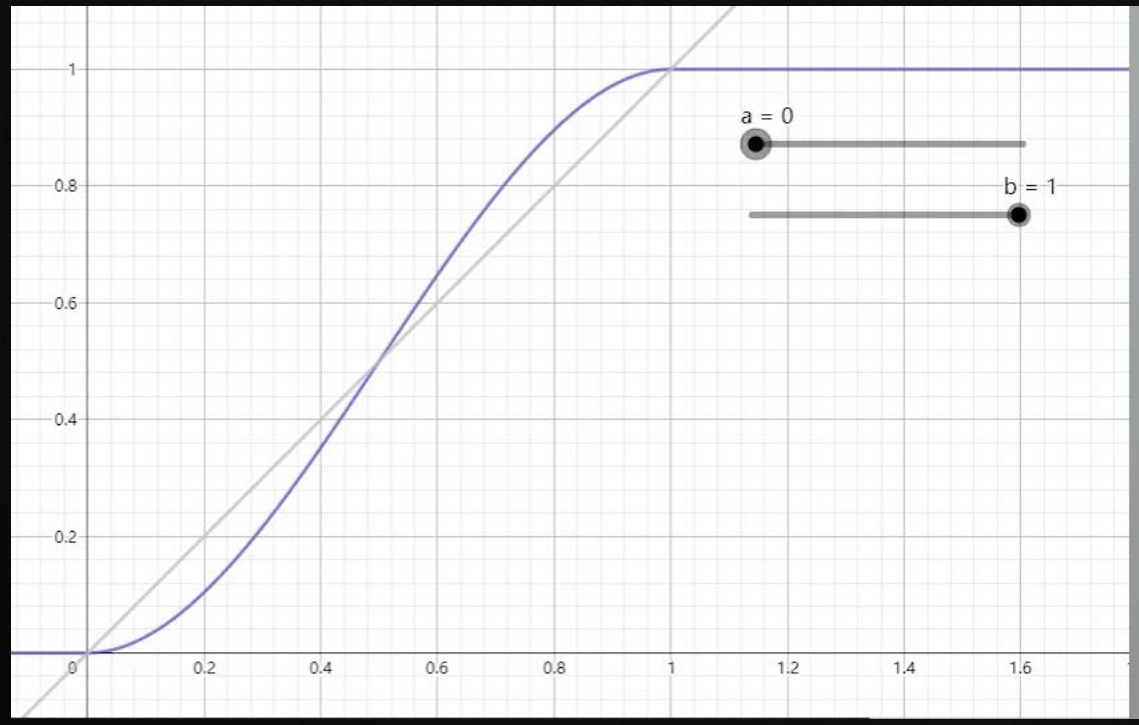
step | smoothstep 函数

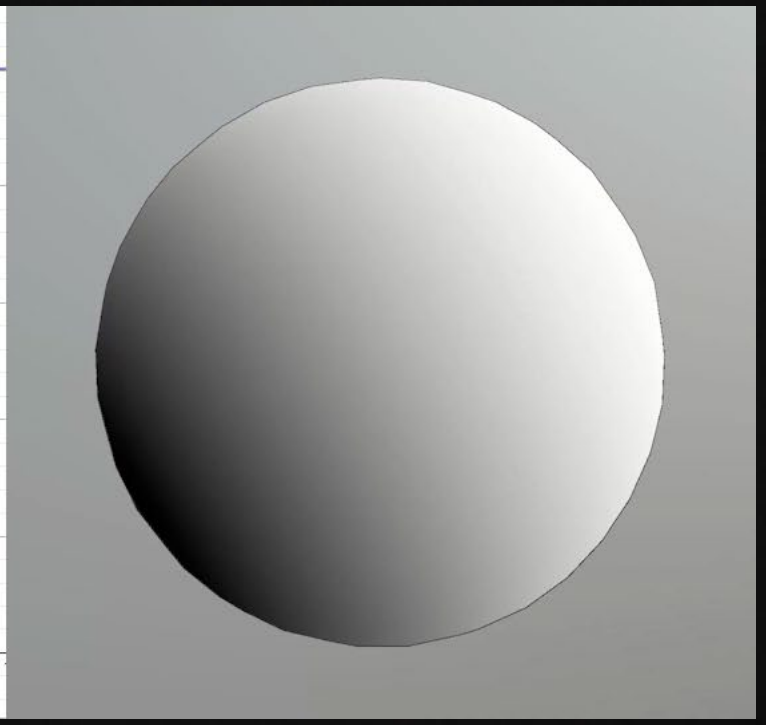
step：阶跃函数

step(a, x)

smoothstep(a, b, x)

```
float smoothstep(float a, float b, float x)
{
    float t = saturate((x - a)/(b - a));
    return t * t * (3.0 - (2.0 * t));
}
```



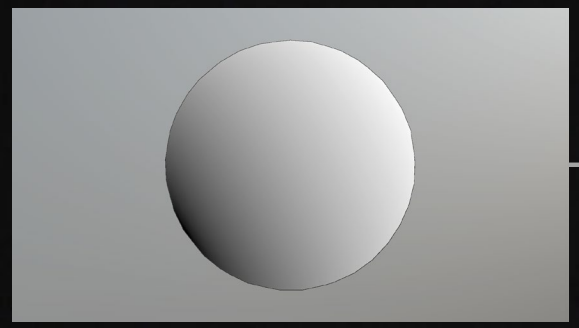


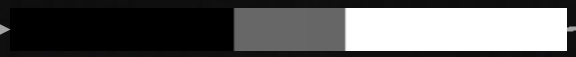
基础表现

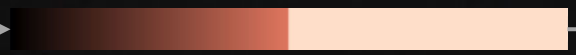
光照

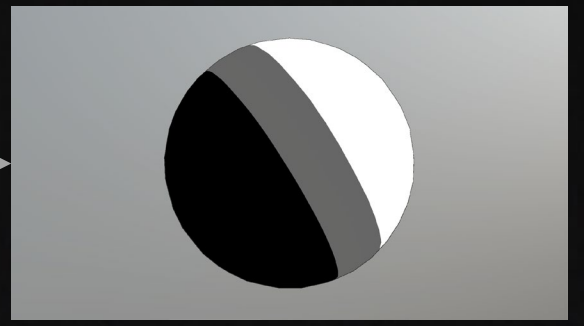
RampMap

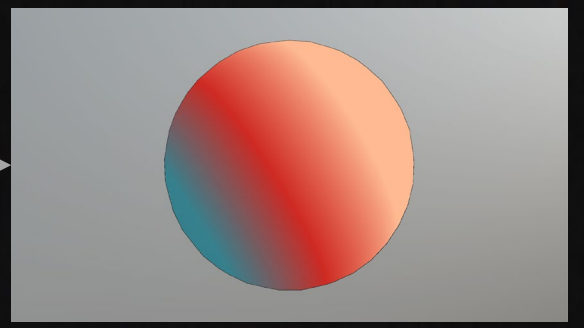
Half Lambert

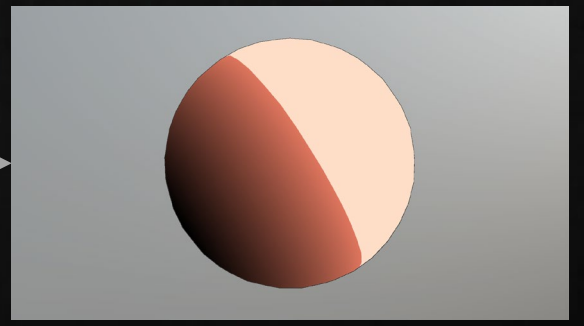












基础表现

光照

边缘光(Rim Light)

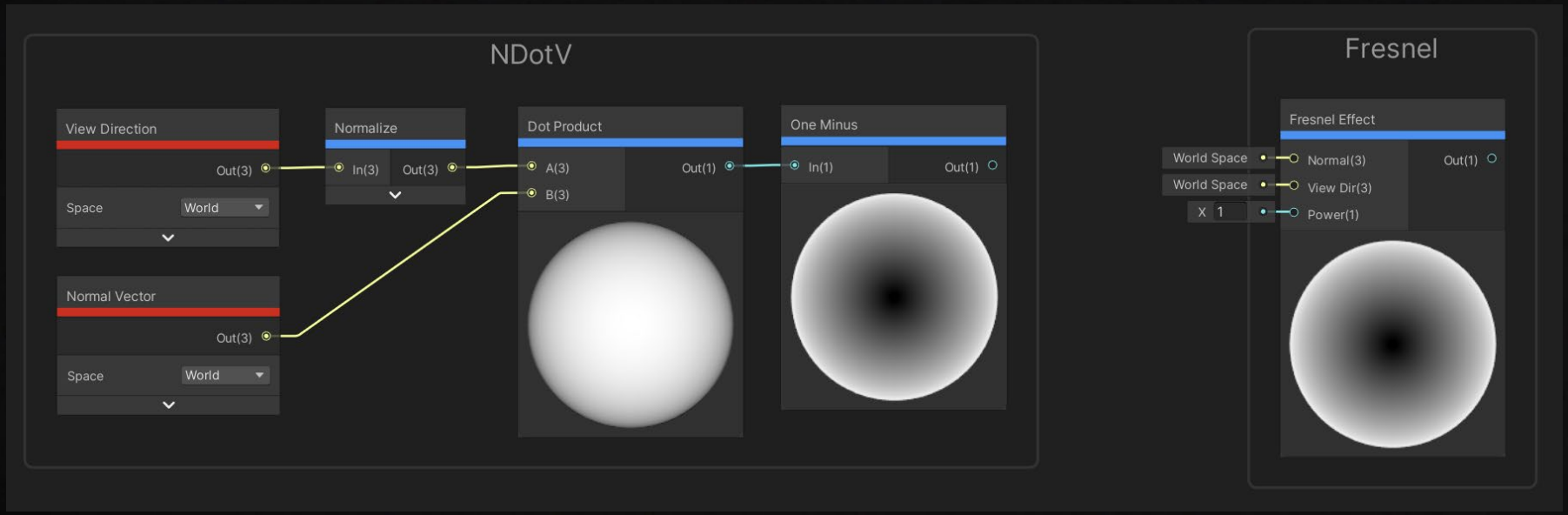
- Fresnel
- Depth Offset

基础表现

光照

Fresnel Rim Light

- 通常由NDotV获取



基础表现

光照

Depth Offset Rim Light

Exponential Tonemapping

```
//Light stored in buffer as exp2(-light)
float3 lightBuf = tex2D(_LightBuffer, uv);

//Invert to get tonemapped lighting info
float3 light = 1 - lightBuf;
```

Thin Depth Offset Rimlight

```
//Offset screenspace uv by a few pixel
//along view norm, sample depth at offset
float2 uvOfs = uv + vNorm.xy * rimWidth;
float zOfs = tex2D(_DepthBuffer, uvOfs);

//Get delta from regular frag depth
float rim = saturate(fragPos.z - zOfs);

//Apply rim to lighting with parameter
light += rim * light * rimFact;
```

Wide Rim aka Roughness

```
//Dot product between view dir and normal
//Then tone it down with parameter
float rough = saturate(dot(view, norm));
rough = rough * roughFact + 1 - rougFact;

//Converge lighting toward 1 along 'rough'
light /= rough * (1 - light) + light;
```

基础表现

光照

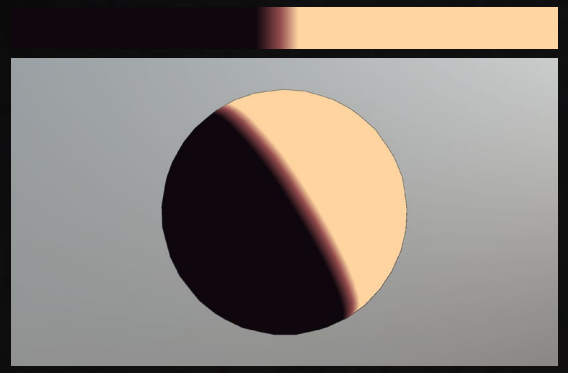
https://www.gdcvault.com/play/1024126/Huddle-up-Making-the-SPOILER

明暗交界线(Terminator Line)

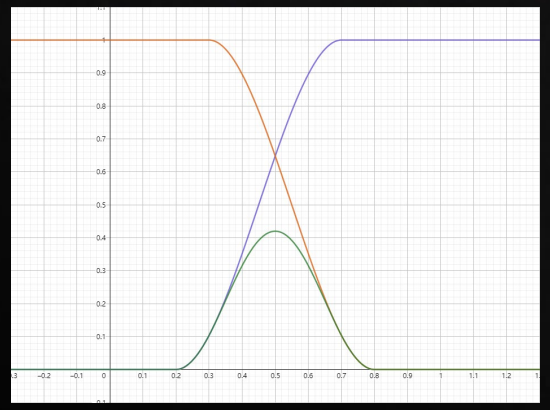
- 绘画用术语，即受光面与背光面的分界线
- 经常会对这一部分区域做特殊处理

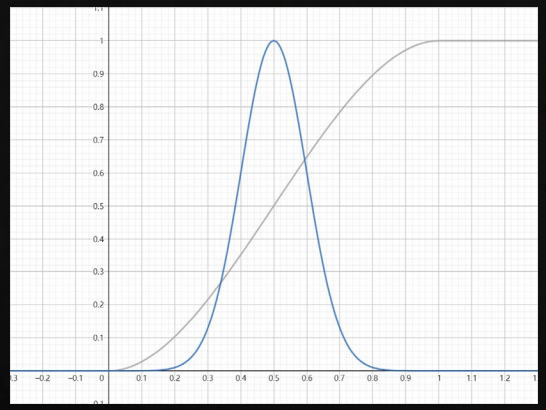
明暗交界线

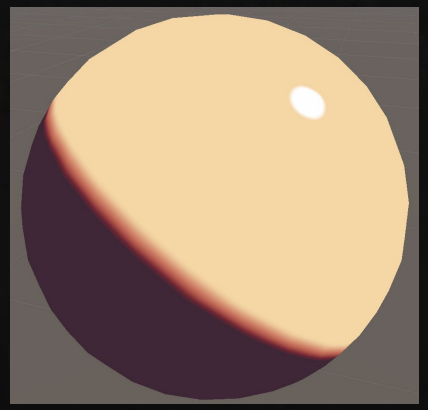
- Ramp中画出明暗交界线



- 使用函数计算，阶跃函数、高斯函数 $f(x) = a \cdot e^{\frac{-(x-b)^2}{2c^2}}$ 等。







基础表现

光照

描边 | 轮廓边

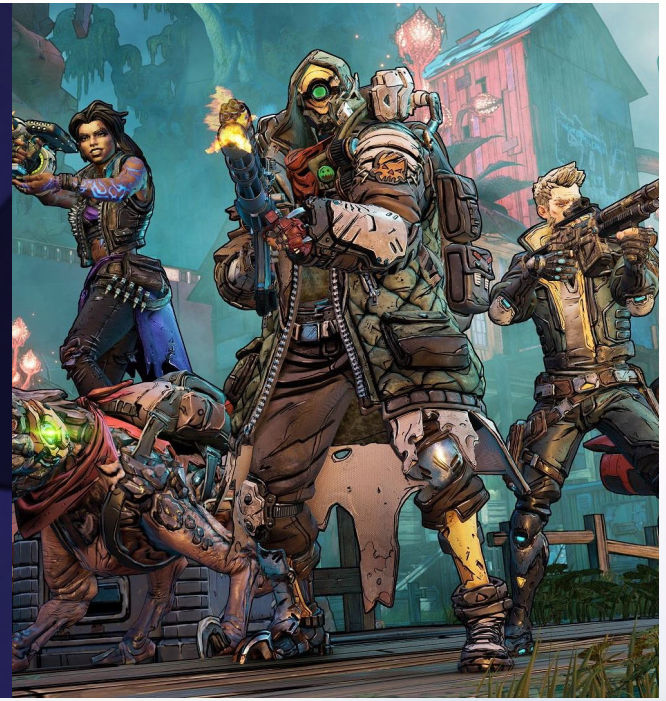
- 分割结构，补充轮廓信息
- 强调主体
- 风格化

基础表现

描边

描边 | 轮廓线

- 分割结构，补充轮廓信息
- 强调主体
- 风格化



基础表现

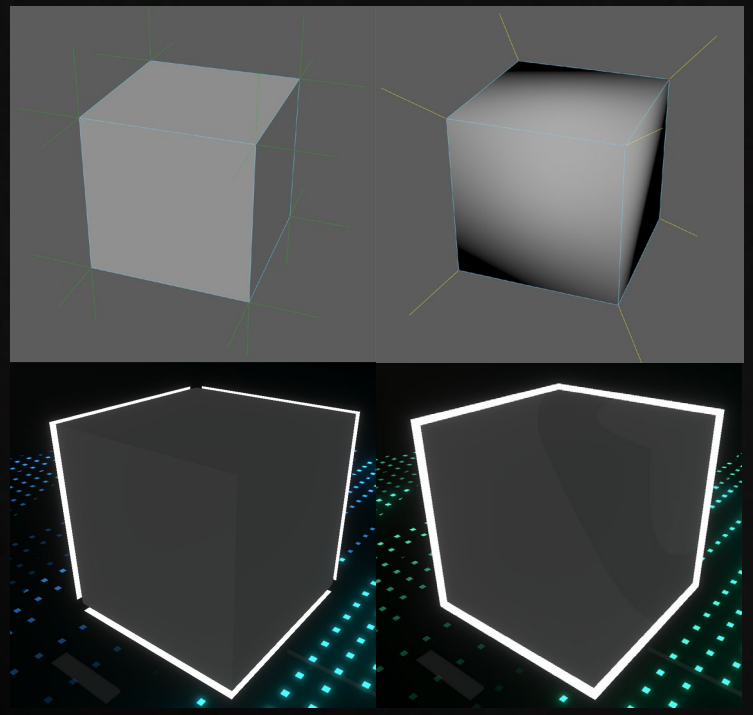
描边

常见的描边方式

- RimLight - 基于观察角度和表面法线/基于深度
- Backface - 过程式几何轮廓线渲染
- 边缘检测 - 基于图像处理的轮廓线渲染
- 基于轮廓边检测
- 本村氏线

Backface

- 2个Pass，一个渲染正面，一个渲染背面

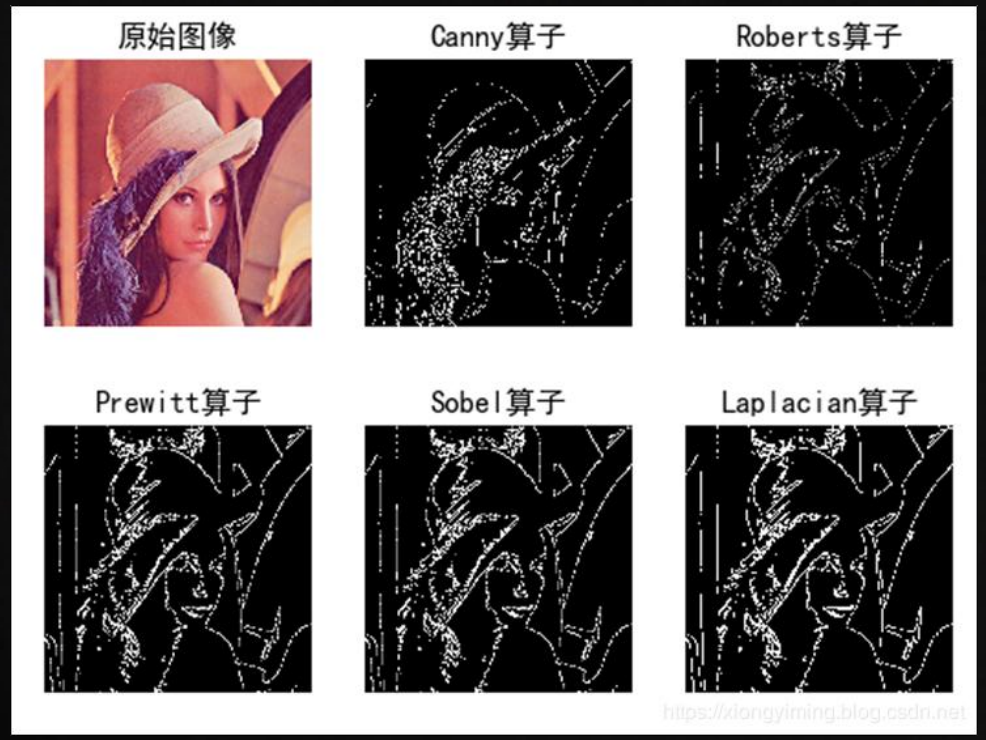


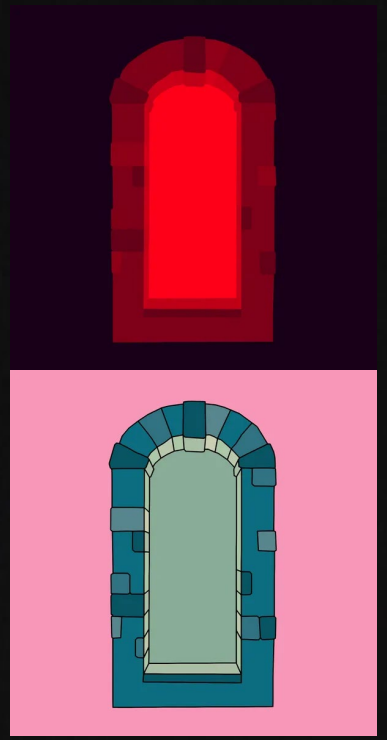
基础表现

描边

边缘检测

- 通过一阶导数或二阶导数的方式计算，利用不同卷积核对图像滤波。





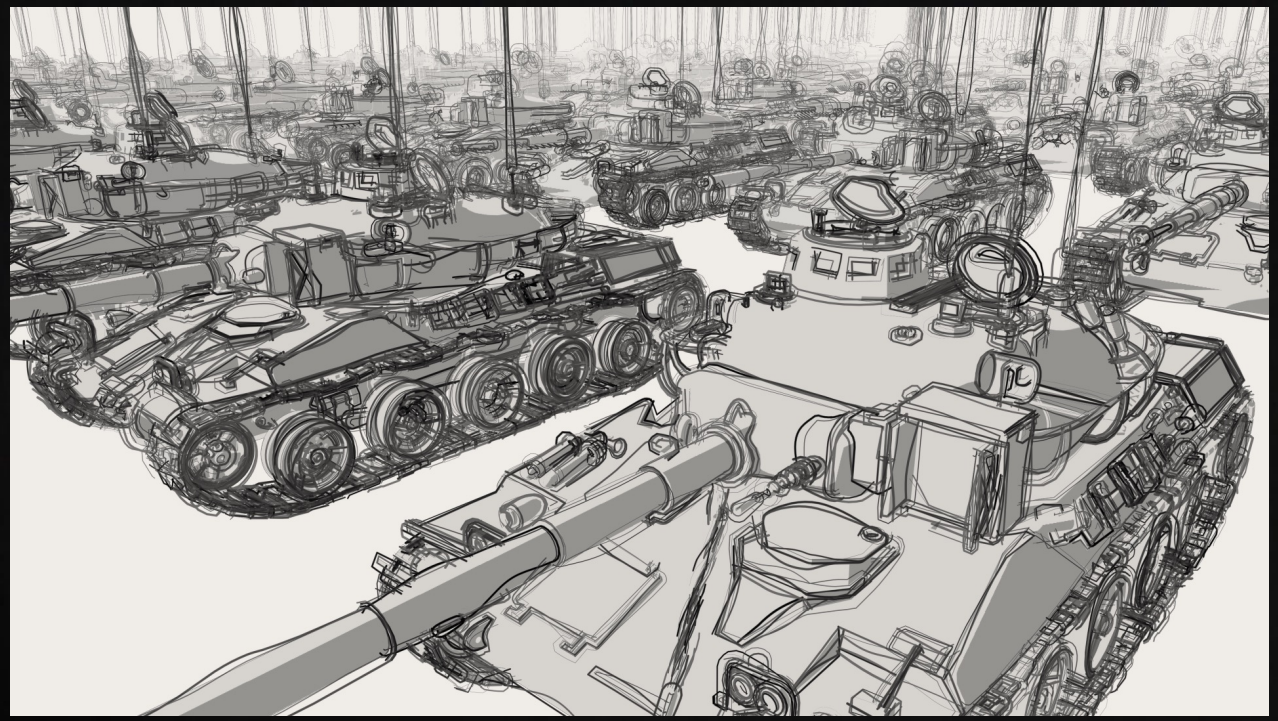
基础表现

描边

https://alexanderameye.github.io/notes/rendering-outlines/

基于轮廓边检测

- 多见于离线渲染



基础表现

描边

https://www.psoft.co.jp/en/product/pencil

本村式线条

- 罪恶装备中的角色表面描边方法
- 将画线位置的UV打直，只绘制垂直于U轴或V轴的直线，从而避免斜向线条的采样问题

基础表现

描边

https://www.4gamer.net/games/216/G021678/20140703095/index_2.html

基于PBR的卡通表现

基于PBR的卡通表现

- PBR基于物理的渲染
- NPR/卡通渲染基于现实概括+创作者的表达

优势

- 标准化工作流
- 高级质感
- 真实环境光照

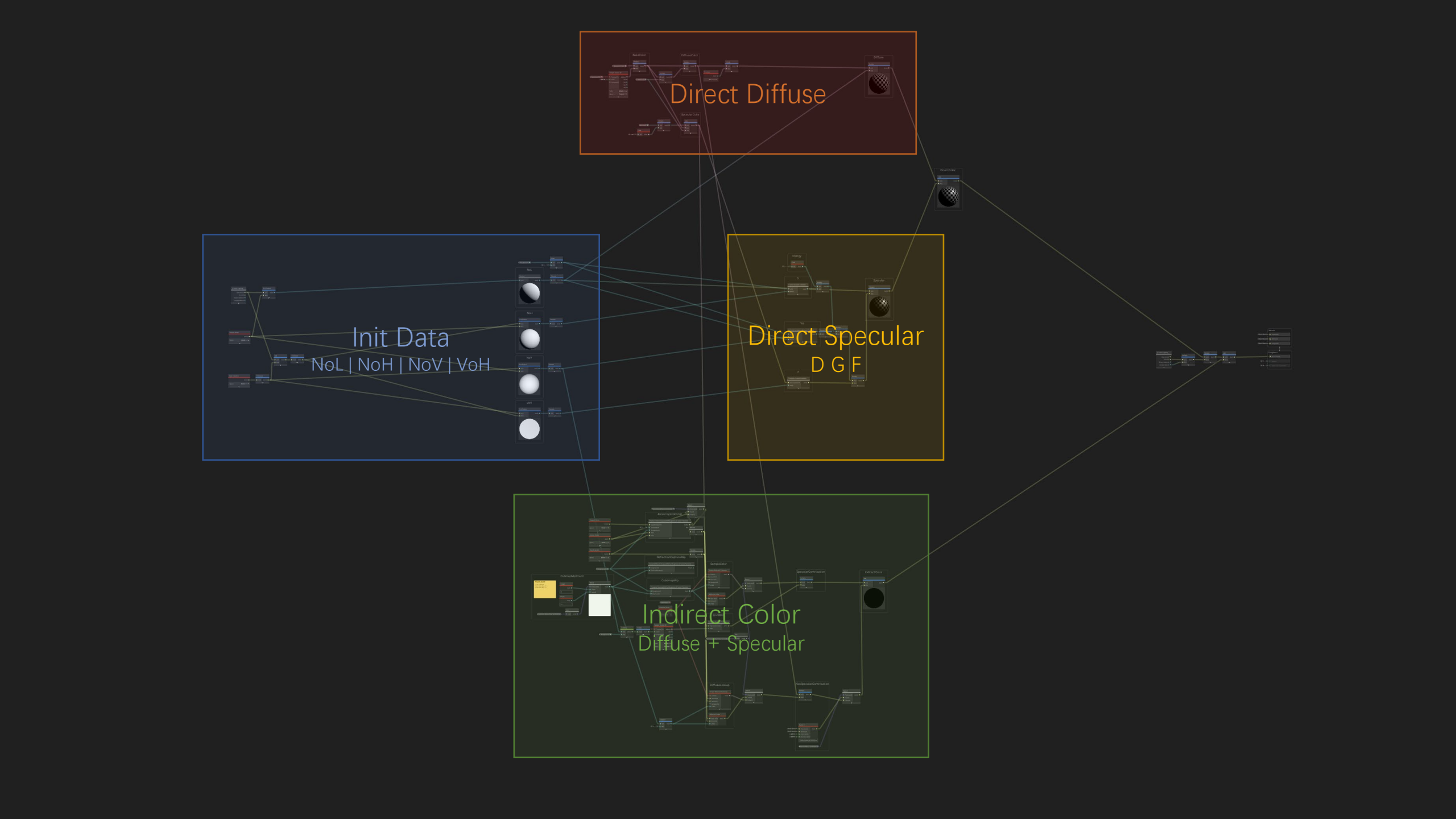
PBR
的优势

材质质感

Unlit

PBR(NPR)

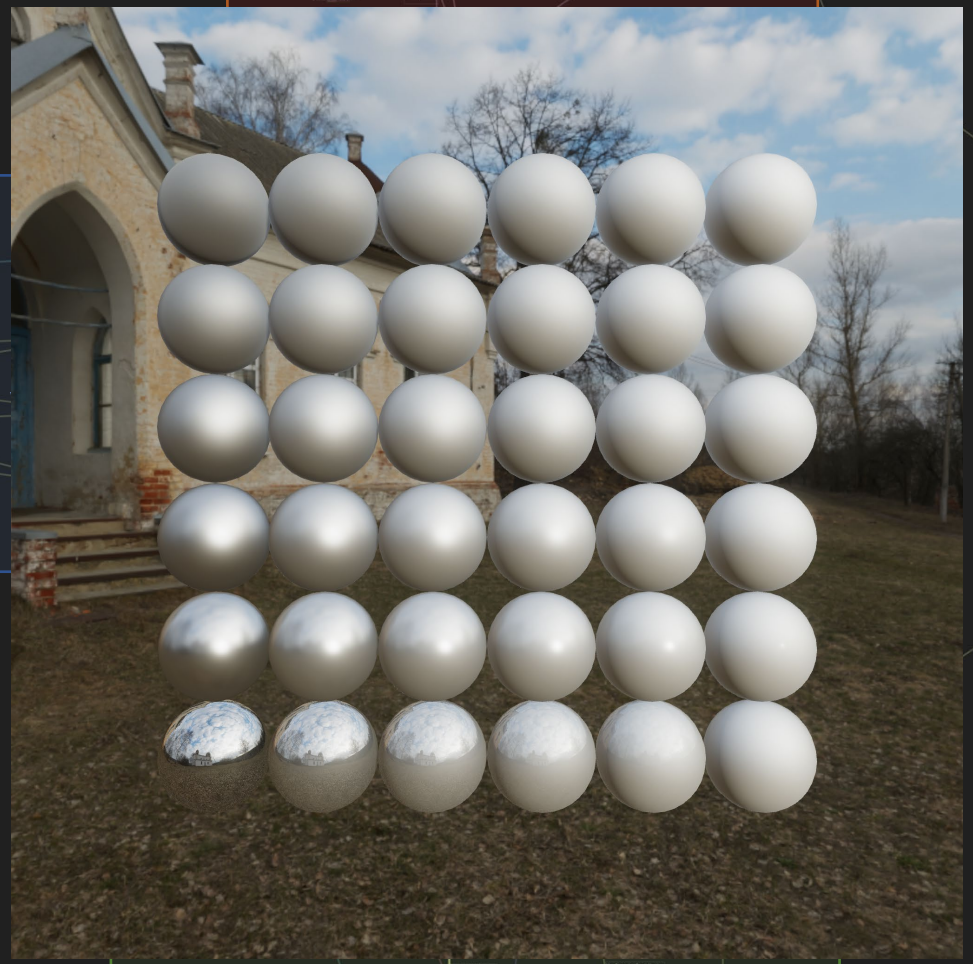
环境光照

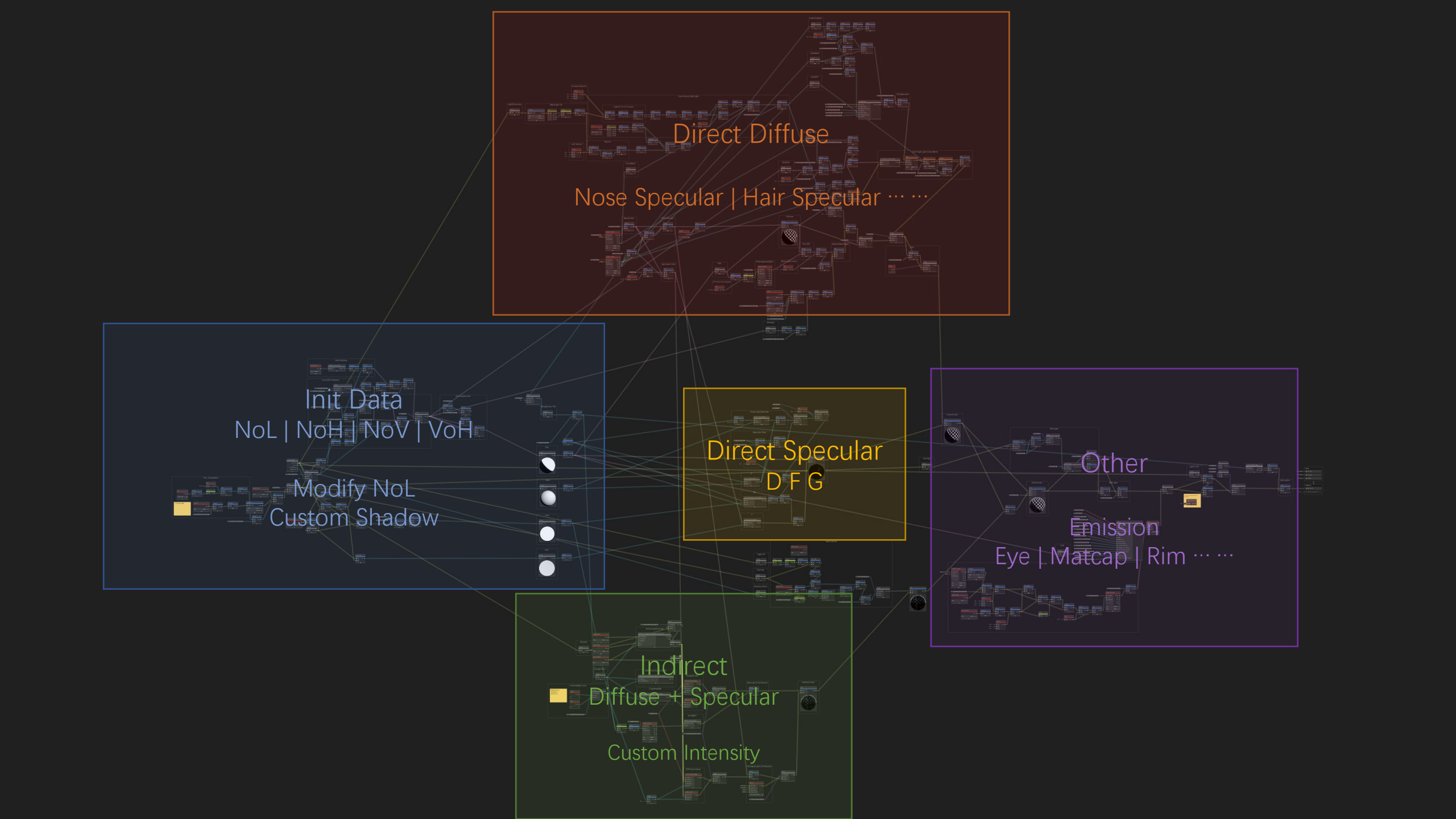

Direct Diffuse
Init Data
NoL | NoH | NoV | VoH
Direct Specular
D G F
Indirect Color
Diffuse + Specular

Direct Diffuse

Init Data

NoL | NoH | NoV |




Direct Diffuse
Nose Specular | Hair Specular ··· ···
Init Data
NoL | NoH | NoV | VoH
Modify NoL
Custom Shadow
Direct Specular
D F G
Other
Emission
Eye | Matcap | Rim ··· ···
Indirect
Diffuse + Specular
Custom Intensity

Direct Diffuse

Init Data

NoL | NoH | NoV | Vc

Modify NoL

Custom Shadow

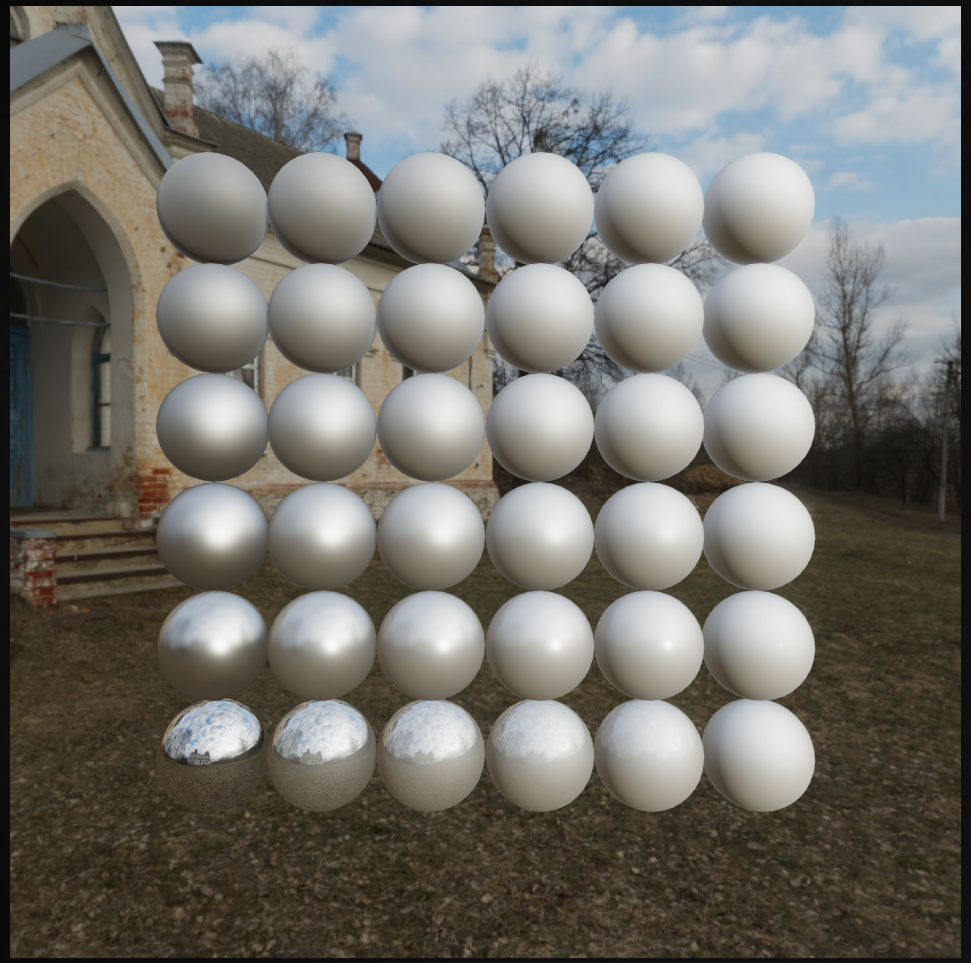
Other

Emission

Eye | Matcap | Rim ··· ···

Diffuse + Specular

Custom Intensity



3D还原2D画面存在的难点

- 高度概括的边界
- 不同视角的造型
- 违背物理的特征
- 高饱和的色倾向

3D还原2D画面存在的难点

- 高度概括的边界
- 不同视角的造型
- 违背物理的特征
- 高饱和的色倾向

3D还原2D画面存在的难点

- 高度概括的边界
- 不同视角的造型
- 违背物理的特征
- 高饱和的色倾向

光影修正

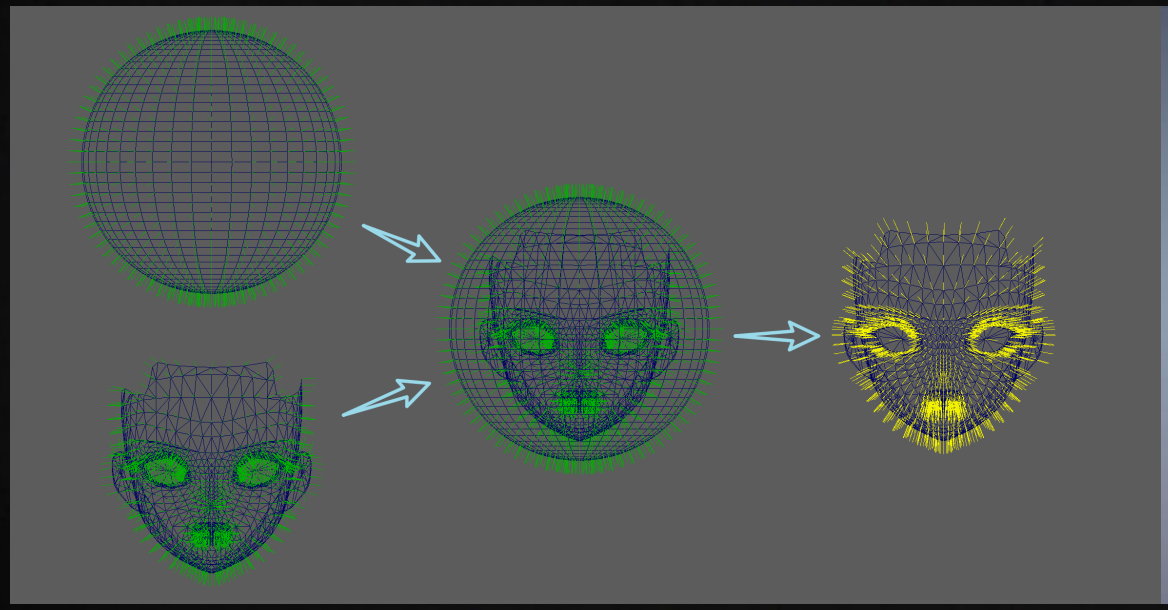
- 修正原因：二值化导致的阴影边缘杂乱
- 修正方式：法线、拓扑、有向距离场(SDF)

Shading Model

面部

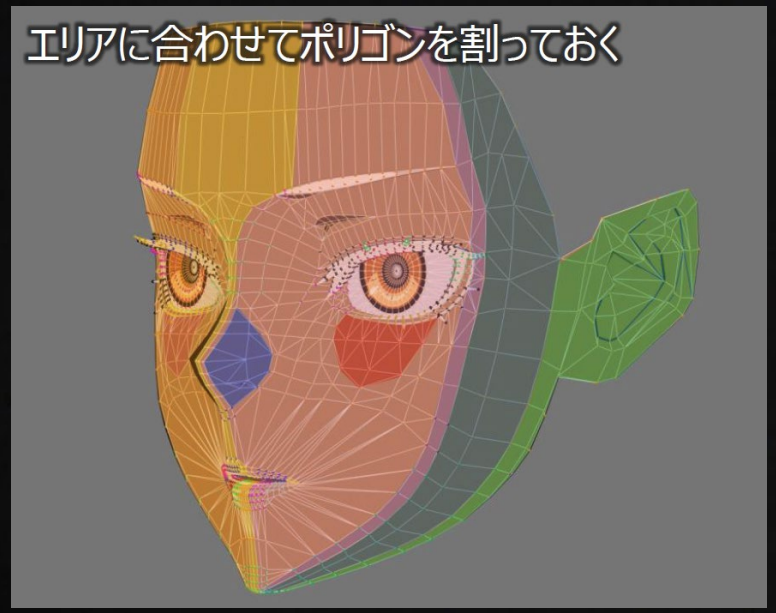
面部光影修正

- 包裹体法线



面部光影修正

- 模型拓扑+法线修改



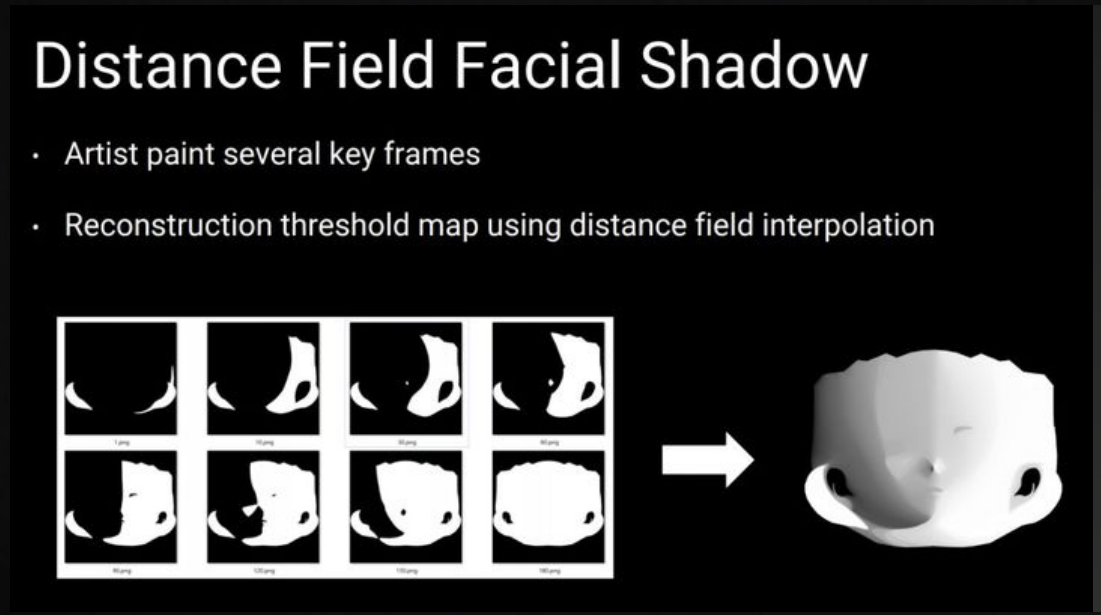
Shading Model

面部

https://www.slideshare.net/ASW_Yokohama/guilty-gear-xrdtips-124324946

面部光影修正

- 使用有向距离场（SDF）记录阴影



Shading Model

面部

https://twitter.com/madumpa/status/1093736946419916805

其他

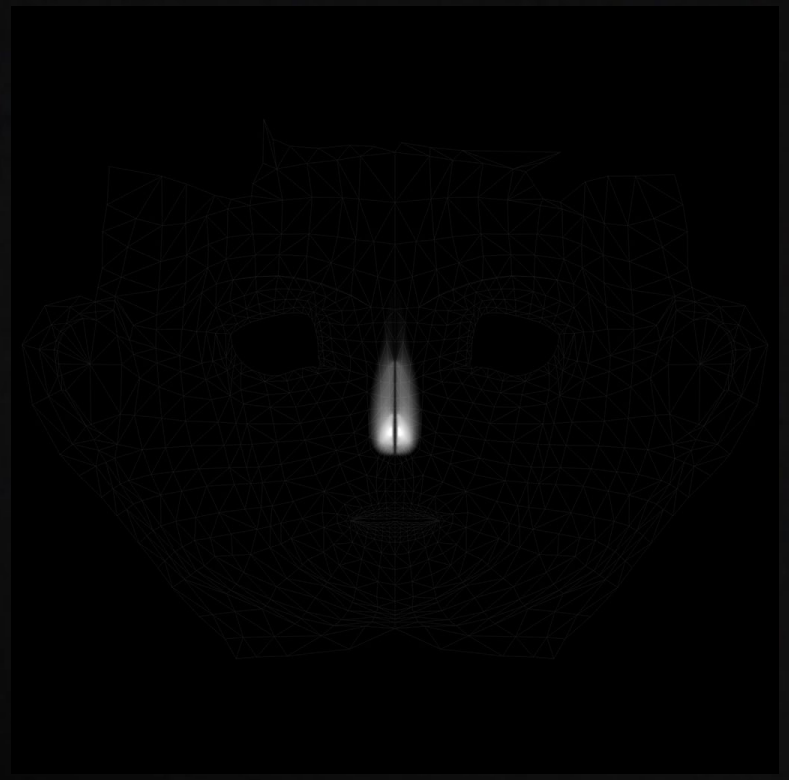
- 鼻尖高光

Shading Model

面部

鼻尖高光

- Mask(SDF)



Shading Model

面部

天使之环高光

- 头发各项异性高光的表现
  - 基于各向异性
  - 基于UV

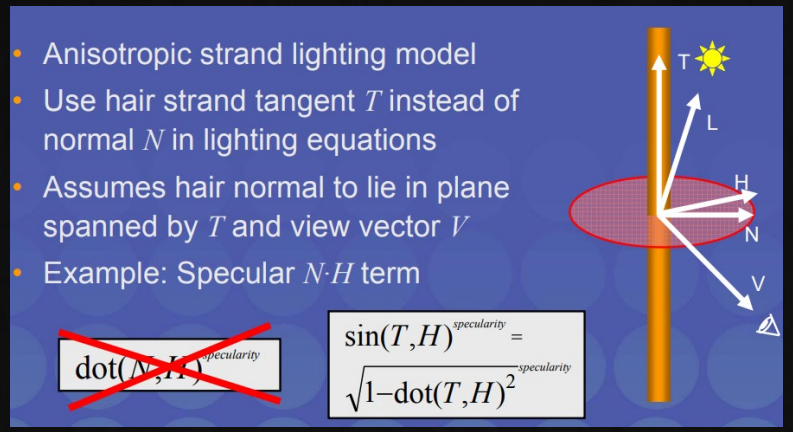
Shading Model

头发

https://google.github.io/filament/Filament.html

基于各项异性计算

- Kajiya-Kay Model



```
float3 ShiftTangent (float3 T, float3 N, float shift){
    float3 shiftedT= T + shift *N;
    return normalize (shiftedT);
}

float StrandSpecular (float3 T, float3 V, float3 L, float exponent)
{
    float3 H = normalize(L + V);
    float dotTH = dot(T, H);
    float sinTH = sqrt(1.0 - dotTH * dotTH);
    float dirAtten = smoothstep(-1.0, 0.0, dot(T, H));
    return dirAtten * pow(sinTH, exponent);
}
```

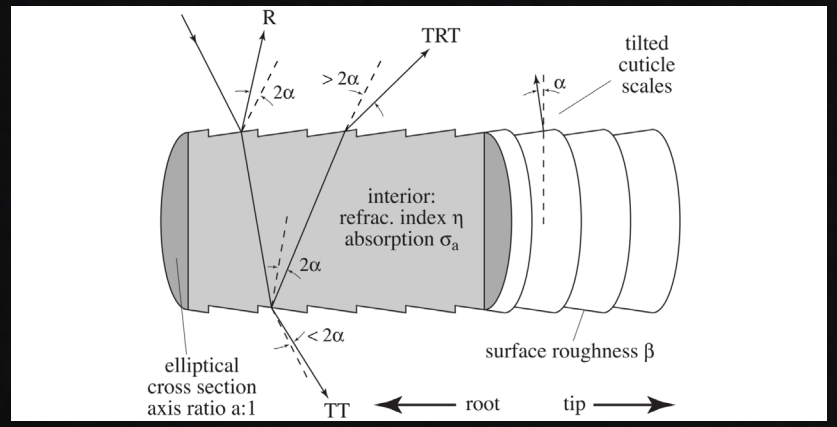
Shading Model

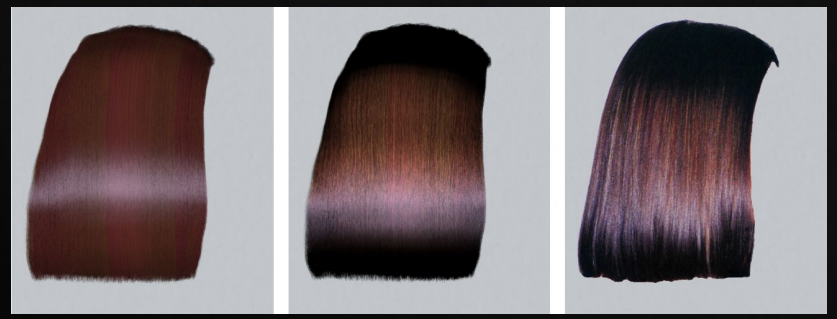
头发

https://developer.amd.com/wordpress/media/2012/10/Scheuermann_HairSketchSlides.pdf

基于各项异性计算

- Marschner Model
- 真实头发的光路是更加复杂，光路传播包含三部分：R,、TRT 、TT
- Hair Rendering and Shading中使用**双层高光**拟合了这种表现





Shading Model

头发

http://www.graphics.stanford.edu/papers/hair/hair-sg03final.pdf

基于各项异性计算

Kajiya-Kay Model

Add Mask | Noise

Shading Model

头发

基于各项异性计算

修正切线

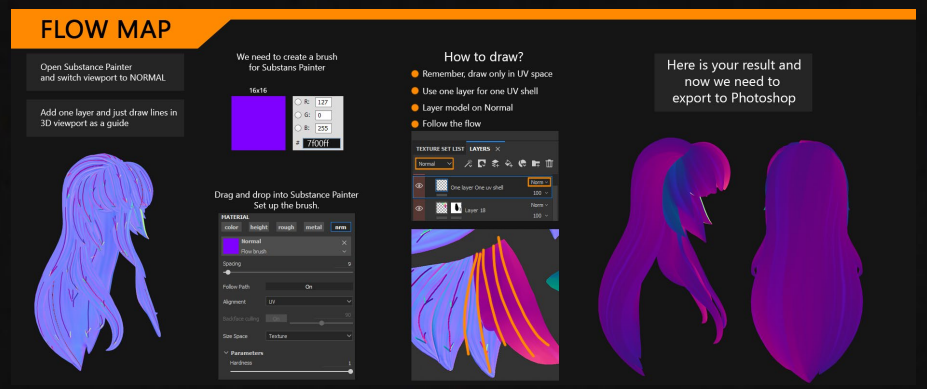
未修改切线

Shading Model

头发

基于各项异性计算 - 切线

- 各向异性的计算依赖于顶点切线，在头发上渲染出连续的各向异性高光需要连续的切线信息
- 切线修正方式
  - 打直UV
  - 绘制FlowMap [Houdini, Krita]



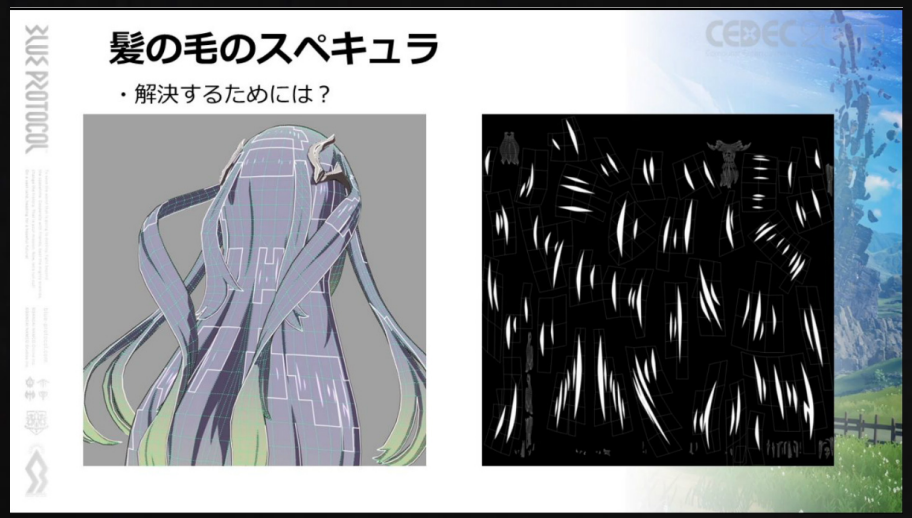
Shading Model

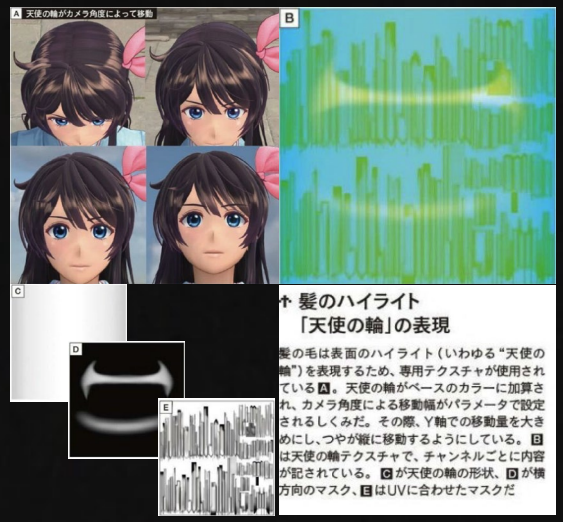
头发

https://www.youtube.com/watch?v=egHSE0dpWRw | https://twitter.com/JenVulpine/status/1215200431711965184

基于UV

- 各向异性的计算相对复杂，可控性较差，另一种思路是直接画在UV上





Shading Model

头发

https://www.onlinegamer.jp/news/202009040001 | CGWORLD 2021 01

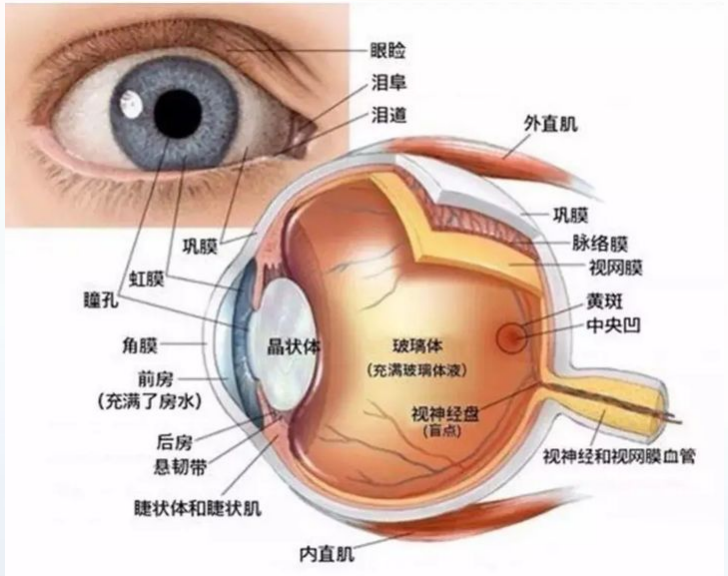
眼睛

Shading Model

眼睛

眼睛

- 为了表现透镜折射，从模型结构上主要有
  - 主体内陷的复合结构
  - 外轮廓的单层结构



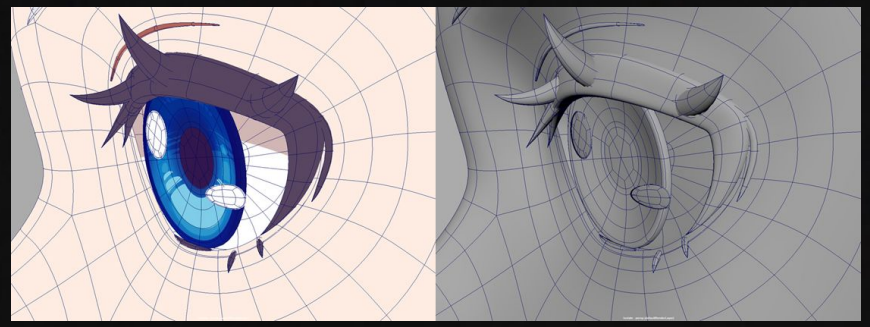
Shading Model

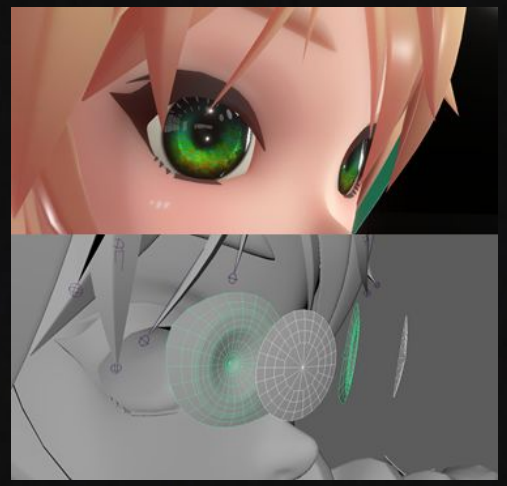
眼睛

眼睛结构

- 主体内陷的**复合结构**

  基础凹陷的眼底 + 突出的高光模型





- 外轮廓的**单层结构**

  单层的轮廓模型 + 视差计算

https://cgworld.jp/feature/201809-cgw241hs-houkai3rd01.html

Shading Model

眼睛

折射计算

- Parallax

```
float3 offset = viewT * _ParallaxScale;
texcoordOffset = texcoord - offset.xy;
```

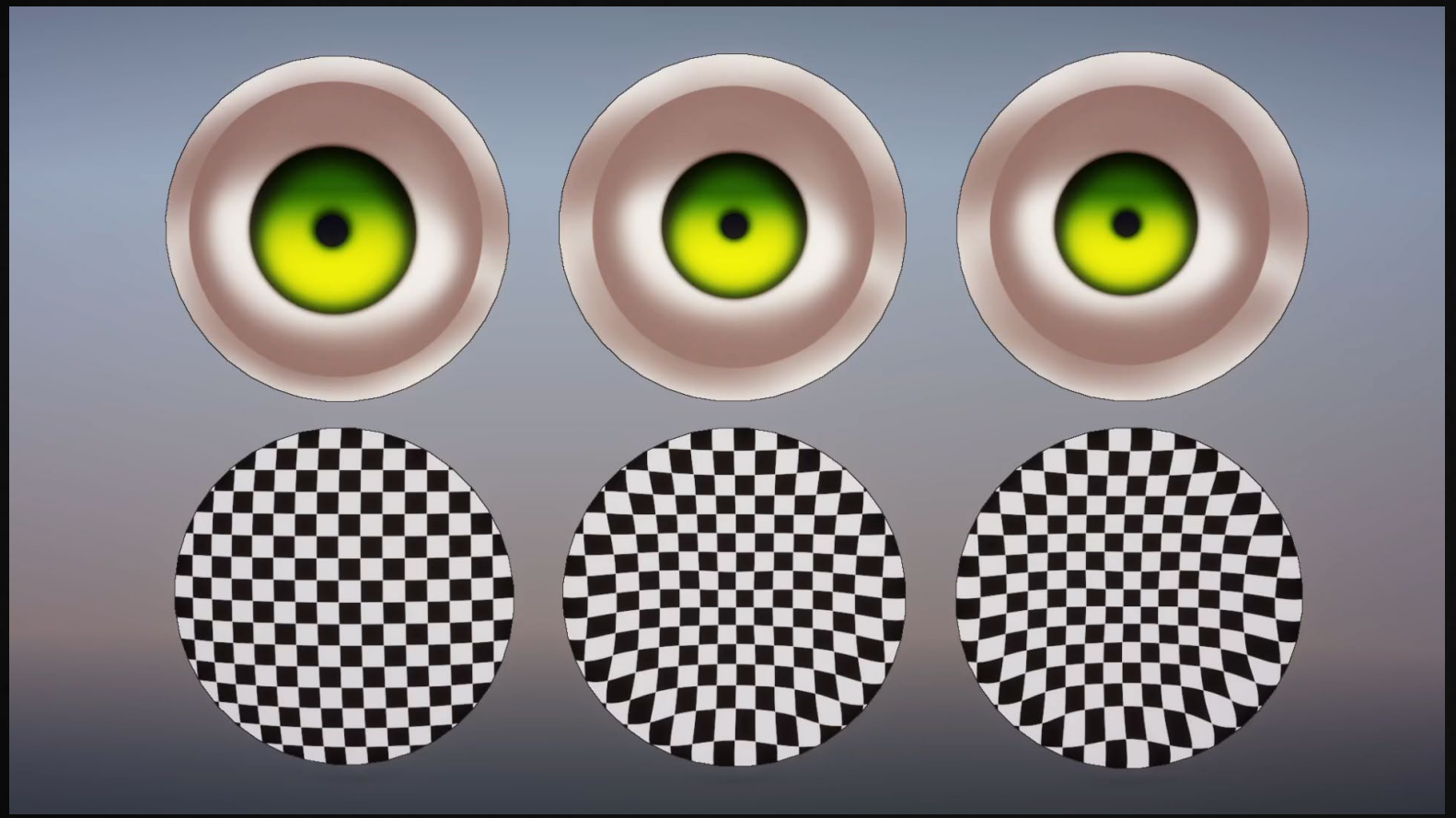
- Physically based refraction

```
float height = anteriorChamberDepth * saturate( 1.0 - 18.4 * radius * radius );

// refractedT
float w = n * dot(normalT, viewT);
float k = sqrt(1.0 + (w - n) * (w + n));
refractedT = (w - k) * normalT - n * viewT;

// calculate refracted
float cosAlpha = dot(frontNormalT, -refractedT);
float dist = height / cosAlpha;
offsetT = dist * refractedT;

texcoordOffset = -float2(mask, mask) * offsetT;
```

Shading Model

眼睛

折射计算



Shading Model

眼睛

http://www.iryoku.com/stare-into-the-future

眉眼遮挡显示

- 半透明
- 深度测试
- 模板测试
- RenderTexture

Shading Model

眼睛

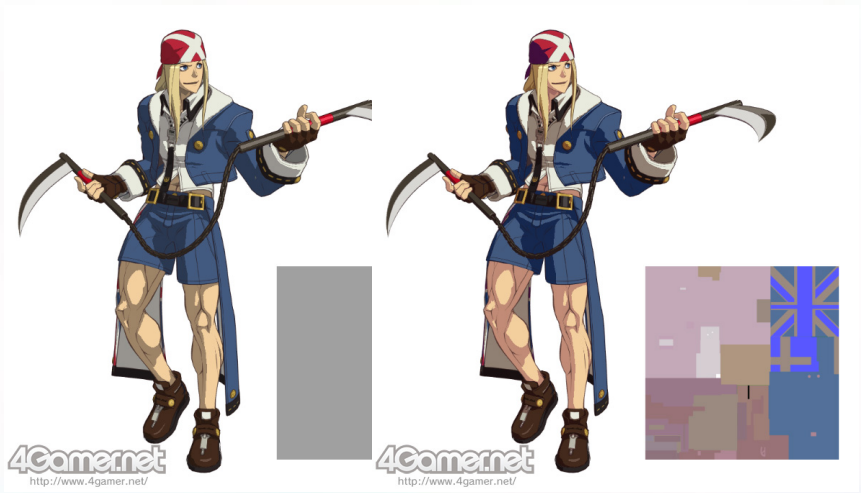
折射计算

Shading Model

眼睛

皮肤质感

- 次表面散射(SSS) | 影色偏移
- 高光



Shading Model

皮肤

https://www.4gamer.net/games/216/G021678/20140703095/index_2.html

影色

- 阴影着色倾向

Shading Model

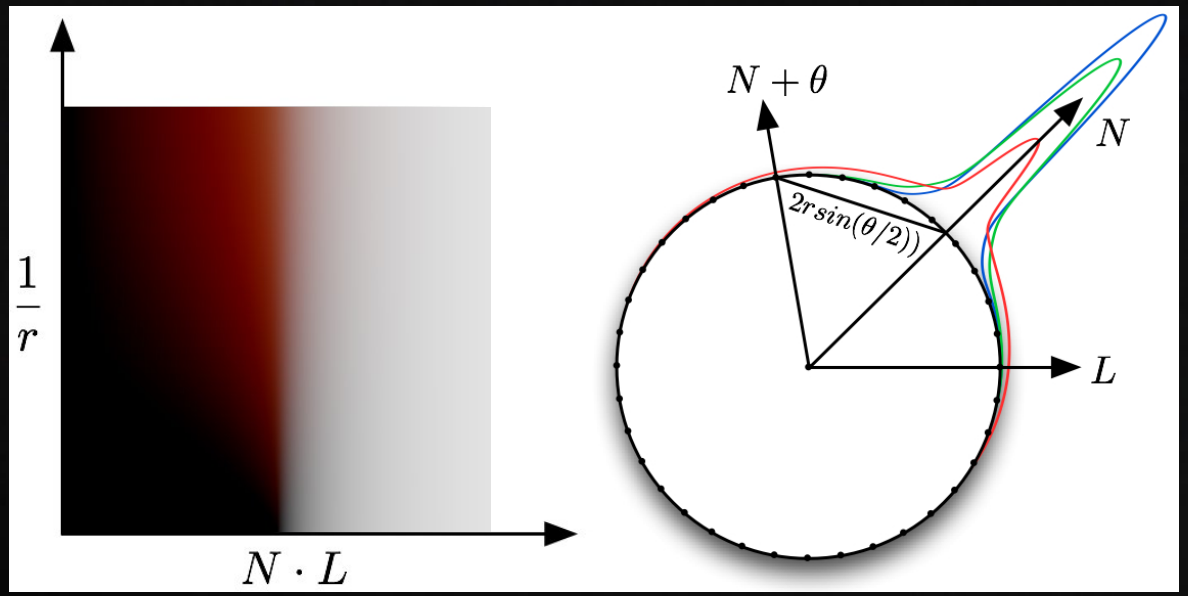
皮肤

https://www.4gamer.net/games/216/G021678/20140703095/index_2.html

皮肤质感 – SSS

- 次表面散射(Subsurface Scattering) ssss | sssss
- 预积分的皮肤渲染（Pre-integrated Skin Shading ）

  基于预计算的LUT



Shading Model

皮肤

http://advances.realtimerendering.com/s2011/

皮肤质感 – 高光

- 双镜叶高光（Dual Lobe Specular）

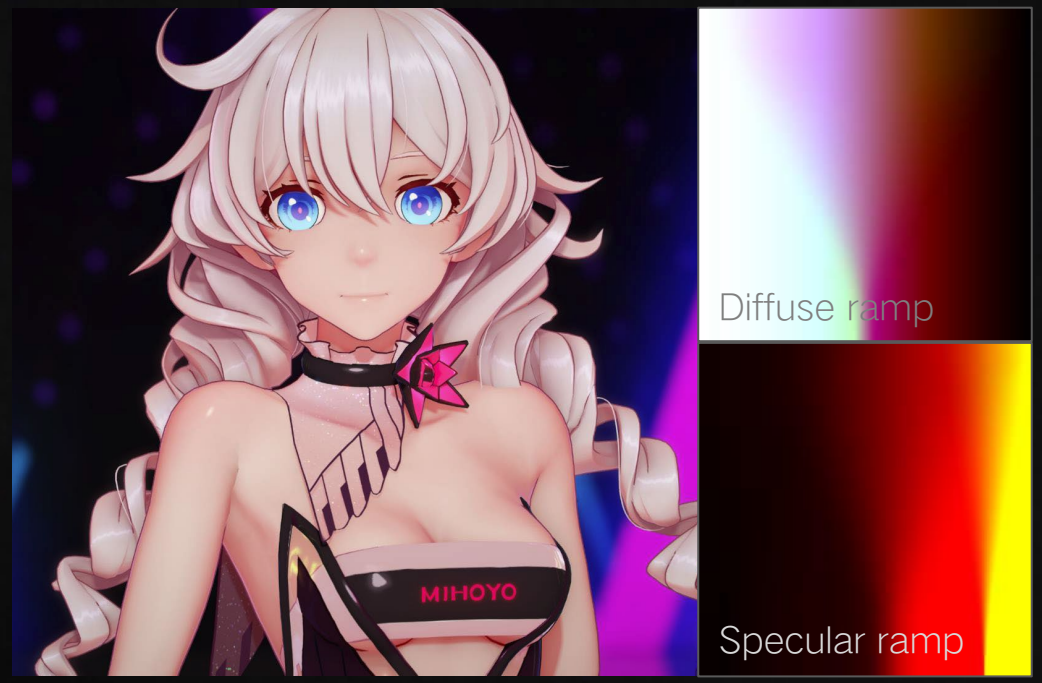
Shading Model

皮肤

https://docs.unrealengine.com/4.26/zh-CN/Resources/Showcases/DigitalHumans/

皮肤质感

- Unite Seoul 2018 - Achieving high quality Anime style rendering on Unity
- Diffuse ramp 与 Specular ramp

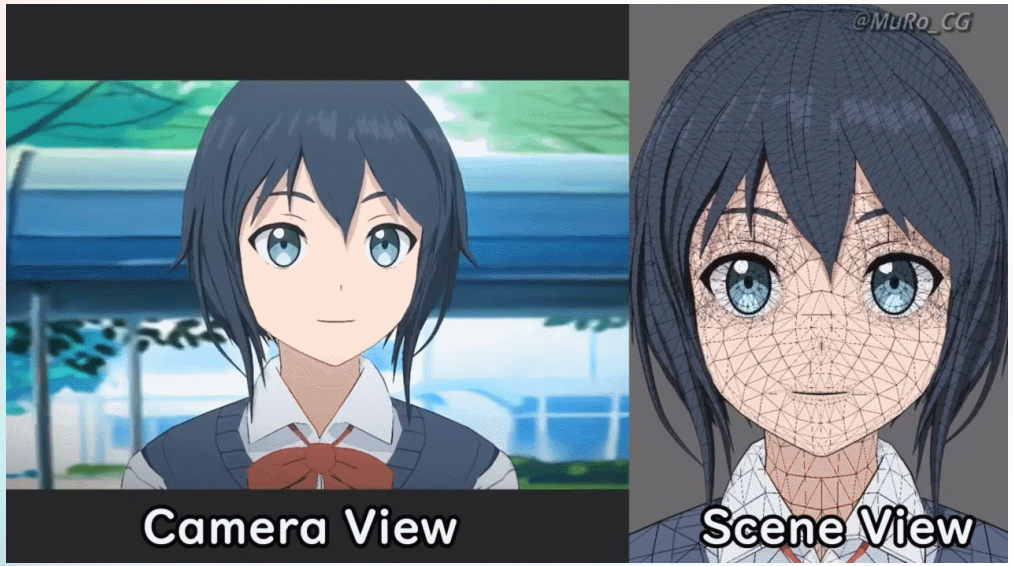


Shading Model

皮肤

https://www.youtube.com/watch?v=egHSE0dpWRw

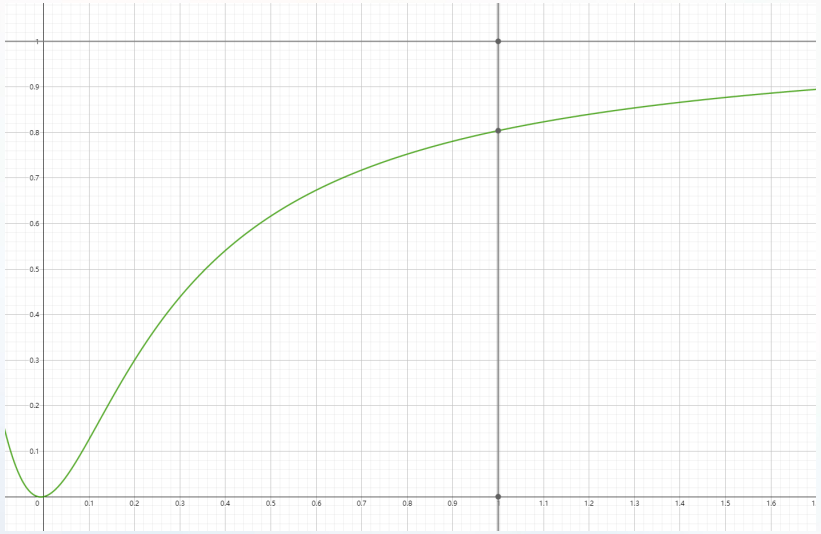
面向镜头的形变



其他

Tonemapping

- 将HDR数据映射到LDR的过程
- ACES 1.0 -> 0.803

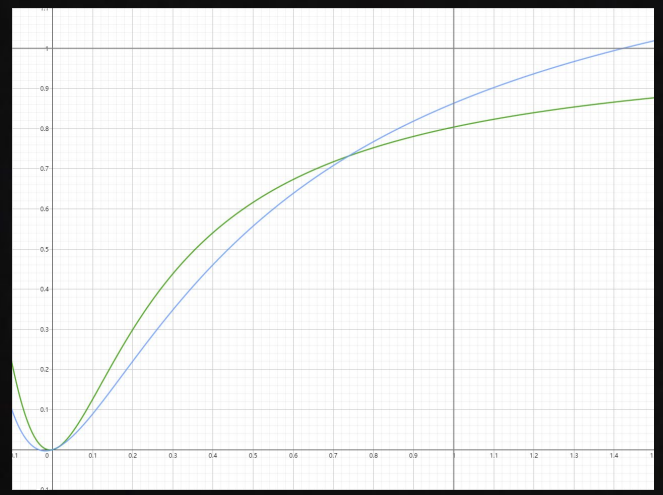


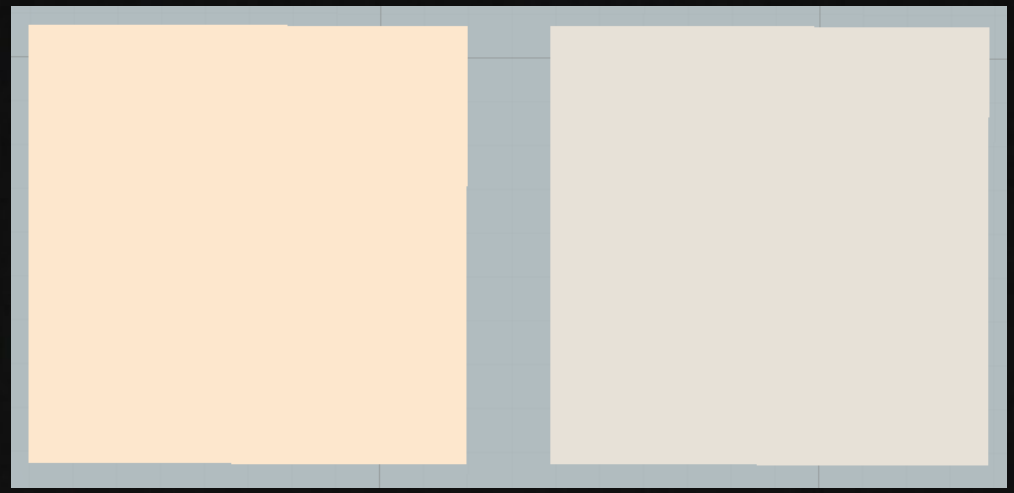
其他

Tonemapping - ACES

- 修改映射曲线
- 计算中补偿

$$A_{ces}(x) = \frac{x\,(2.51\,x + 0.03)}{x\,(2.43\,x + 0.59) + 0.14}$$





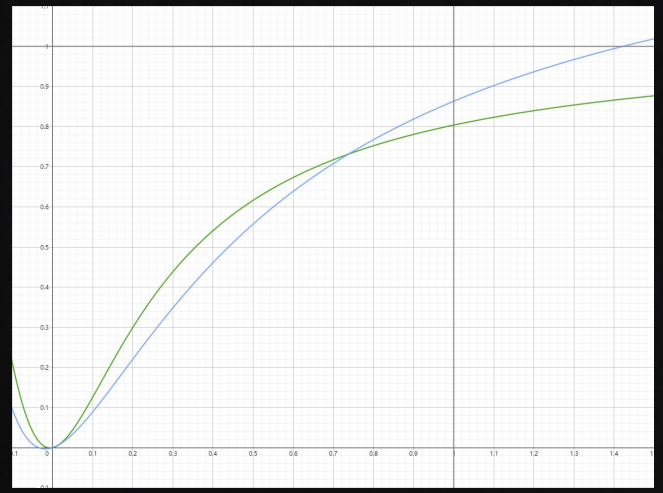
原有肤色　　ACES映射（绿色曲线）

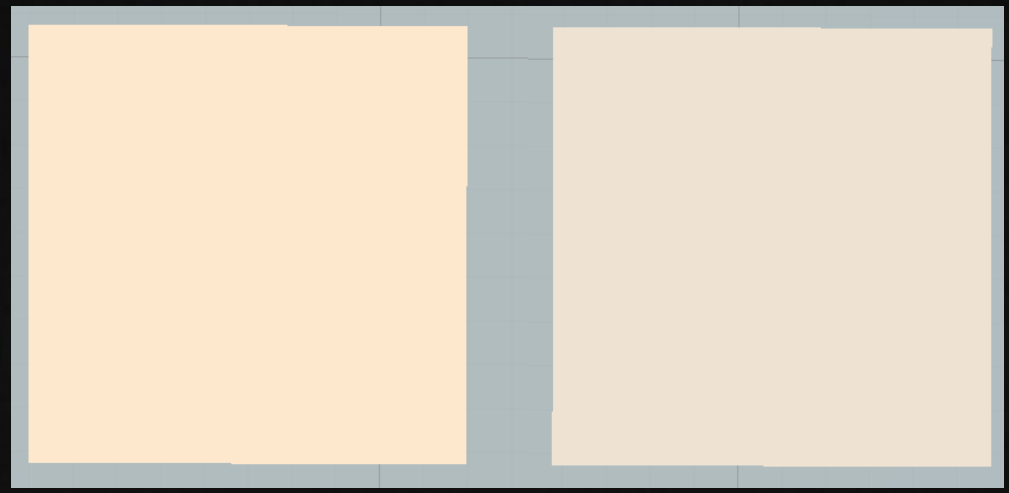
其他

Tonemapping - ACES

- 修改映射曲线
- 计算中补偿

$$A_{ces}(x) = \frac{x\,(2.51\,x + 0.03)}{x\,(2.43\,x + 0.59) + 0.14}$$





原有肤色　　修改后的ACES映射（蓝色）

其他

Tonemapping - ACES

其他

Thank you

zhihu.com/people/zi-xie-42-53

leiyuhangzxy@163.com