目录

[1. 卡通渲染分类分类 1](#_Toc22161)

[1.1. 分类方法 1](#_Toc29490)

[1.2. 美式卡通 1](#_Toc26801)

[1.3. 日式卡通 2](#_Toc12420)

[2. 描边方法 2](#_Toc201)

[2.1. 基于视角的勾边 2](#_Toc30380)

[2.2. 基于几何生成方法的描边 2](#_Toc1816)

[2.3. 基于图像处理的描边 3](#_Toc19719)

[3. 着色方法 3](#_Toc27989)

[3.1. Cel shading 3](#_Toc28429)

[3.2. Tone Base Shading 4](#_Toc22313)

[4. 卡通渲染的游戏 5](#_Toc22390)

[5. 卡通着色相关参考 6](#_Toc27489)

# 卡通渲染分类分类

## 分类方法

卡通渲染最关键的特征包括不同于真实感渲染的艺术化光影效果和描边。以这两个关键的特征为卡通渲染分类的话，可以将近年来游戏中常用的卡通渲染分为美式卡通风格和日式卡通风格。

## 美式卡通

风格在色彩上比较连续，有渐变色，着色风格很大程度上依赖于艺术家定义的色调（tone），而在阴影和高光方面常常采取夸张和变形的做法，比较典型的是《军团要塞2》；

美式卡通中往往倾向于使用基于图像处理的描边方法来生成均匀一致的描边效果。在《英雄联盟》中小兵和英雄的勾边效果就是用Sobel算子对深度信息进行边缘检测来获得的。由于游戏中只需要针对小兵和英雄勾边而不需要对场景地图进行勾边，因此在LOL中，勾边的计算并非全屏后处理，而是逐物体进行的，这样的好处是可以随意控制哪些物体描边，每个物体可以单独指定描边颜色，缺点是当物体较多时（尤其是skinned mesh较多时）计算量会增大。一个折衷的方案是，在进行正常绘制的阶段用stencil buffer标记出需要描边的物体，然后用一个全屏的后处理，对stencil buffer标记的像素进行边缘检测，当然这样的话，就很难给每个物体单独指定描边颜色了。

实际上，在LOL中有两种类型的描边，一种是小兵和英雄的固定描边，另一种是防御塔发出攻击警报或者某个单位被点选时才产生的红色描边，这两种描边在处理上略有差别，前者直接使用边缘检测的结果作为最终描边，而后者则是对边缘检测结果再进行一次模糊，借此来扩大和柔化描边效果。

## 日式卡通

风格往往角色造型更写实，但在着色方面，则趋向于大片大片纯色色块，并有的明暗交界，例如《崩坏3》。

日式卡通中往往倾向于使用基于几何体生成的方法去描边，这类描边方法相较于另两类方法的好处在于线宽更容易为美术所控制，而在日式卡通中，往往需要粗细有变化的描边去体现角色不同部位的特征，例如在《GUILTY GEAR Xrd》中，角色的描边就是通过几何体生成的方法，结合了shell method和z-bias method，并引入了逐物体的顶点色来控制描边细节，同时也是为了保证描边粗细不会随着摄像机视距发生变化

# 描边方法

卡通渲染及其相关技术总结（介绍了日式和美式卡通渲染）

https://blog.uwa4d.com/archives/usparkle\_cartoonshading.html

中提到了3种

## 基于视角的勾边

这部分的计算依赖于我们的一个直觉观察：当我们的视线和某个表面相切时，这个表面上的像素点往往就是模型的边缘，基于这个观察，我们可以用dot(viewDir, normal)^{k} 来估计一个像素的“边缘程度”，当然，这个值也可以用来作为纹理坐标去采样一张预定义的“轮廓纹理”。

缺点是线宽粗细差别较大，不易控制

## 基于几何生成方法的描边

这类方法的特点是描边本身是一个单独的几何体，通过特殊的方法绘制出来，比较常见的做法是shell method，原理和实现都比较简单：首先在绘制结束正常的模型后，将需要描边的物体改用正面剔除再绘制一遍，在VS中将顶点沿着法线方向膨胀一定距离，然后在FS中将模型用纯色输出。另外一种叫做z-bias的方法，也是绘制背面，但不膨胀，而是把背面顶点的Z值稍微向前偏移一点点，使得背面的些许部分显示出来形成描边效果。

基于shell method的绘制方法，实现简单，线宽较为均匀

这种方式较常规

## 基于图像处理的描边

这类方法的实现可以说更接近于“边缘”这一概念的本质定义，什么是“边缘”呢？边缘就是在深度或者法线上不连续的位置。因此为了获取边缘，我们只需要在图片上找到深度或者法线不连续的位置即可，因此，我们需要将深度信息和法线信息以贴图的形式传入，运用边缘检测算法去寻找这些像素。

这类方法的优点是描边的线宽一致，缺点是需要额外的法线和深度信息，当然，由于近年来流行的延迟渲染框架，法线和深度本来就是G-Buffer的一部分，因此往往不需要额外绘制法线和深度的信息。

这种使用延迟渲染，目前不适合移动端

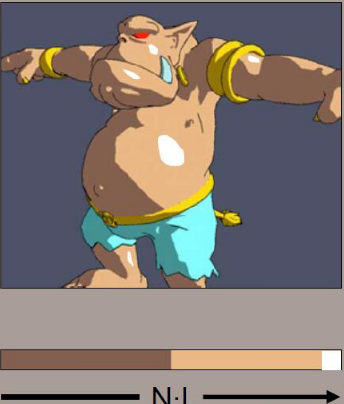
# 着色方法

## Cel shading

Cel Shading的基本思想是把色彩从多色阶降到低色阶，减少色阶的丰富程度，从而实现类似手工着色的效果,



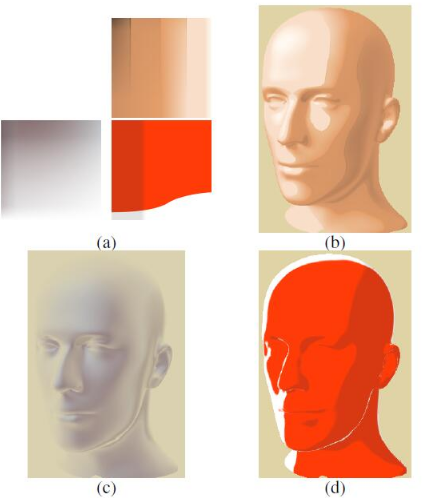
其中，Kd表示模型自身的贴图颜色，celCoord表示法线和光照方向的点积，用作一维色彩表的查找坐标，而paletteTex则是由美术绘制的一维色阶表，一般来说是由几个纯色色块组成的



从1维色阶表取色

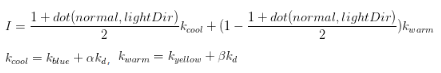
上述做法可以用于模拟卡通渲染的漫反射分量，却并没有考虑到视角相关的光照分量的模拟，因此很难实现类似菲涅尔效果的卡通渲染。实际上，也可以用类似的查找表的思路来视角相关光照分量的色阶离散化6，只需要将一维查找表扩展到二维即可：





## Tone Base Shading

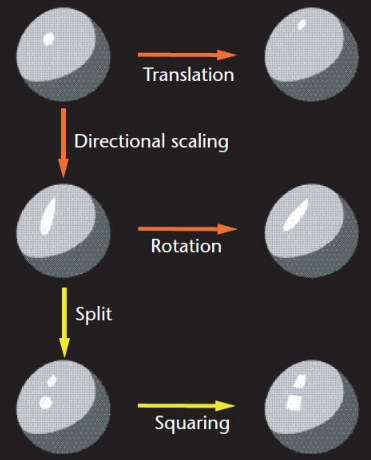
Tone Based Shading的风格化是基于美术指定的色调插值，并且插值得到的色阶是连续的。首先需要由美术指定冷色调和暖色调，而最终模型的着色将根据法线和光照方向的夹角，在这两个色调的基础上进行插值，具体算法如下



其中，Kd仍是模型自身色彩贴图，Kblue，Kyellow和alpha，beta则均是自定义的参数。



# 高光，阴影的风格化



# 卡通渲染的游戏

崩坏3



崩坏3游戏截图，着色以单色色块为主，有明显的明暗交界

军团要塞2



军团要塞2的卡通渲染，人物造型夸张，但着色连续，接近真实感光照

守望先锋

英雄联盟

dota2

罪恶装备

二之国

无主之地

塞尔达

# 卡通着色相关参考

卡通渲染及其相关技术总结

<https://blog.uwa4d.com/archives/usparkle_cartoonshading.html>

介绍了日式和美式卡通渲染，里面关于日式，美式的具体算法没细看也不是特别详细，需要的化可以当线索用

风格化的计算也没细看，这部分将的比较泛泛，且复杂不易懂