北京语言大学

学 士 学 位 论 文

论文题目： 基于 Unity PBR 框架下的风格化角色渲染

姓　　名：闫硕

学　　号：202111580835

院　　系：信息科学学院

专　　业：数字媒体技术

指导教师：安维华

二〇二五年六月

北京语言大学学士学位论文

(2021 级)

论文题目：基于 Unity PBR 框架下的风格化角色渲染

院　　系：信息科学学院

专　　业：数字媒体技术

学 生 姓 名：闫硕

指导教师姓名：安维华

论文完成日期：2025 年 6 月

Stylized Character Rendering in Unity under the PBR Framework

by

Shuo Yan

Supervised

by

Weihua An

Submitted to School of Information Science

in partial fulfillment of the requirements for the degree of

Bachelor of Engineering

at

Beijing Language and Culture University

June 2025

北京语言大学
BEIJING LANGUAGE AND CULTURE UNIVERSITY

论文原创性声明

本人郑重声明：所呈交的论文，是本人在导师指导下，独立进行的研究工作及取得的研究成果。尽我所知，除了文中已经注明引用和致谢的地方外，论文中不包含其他人或集体已经发表或撰写的研究成果，也不包含为获得北京语言大学或其他教育机构的学位或证书所使用过的材料。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

签　　名：闫硕

日　　期：2025年6月1日

学位论文知识产权权属声明

本人郑重声明：本人所呈交论文，是在导师指导下所完成的，论文知识产权归属北京语言大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版本，允许论文被查询和借阅，将论文编入有关数据库进行检索等。本人离校后发表或使用学位论文或与该论文直接相关的学术论文获成果时，署名仍为北京语言大学。

签　　名：闫硕

导师签名：安维华

日　　期：2025年6月1日

目　录

基于 Unity PBR 框架下的风格化角色渲染

作者：闫硕　　指导教师：安维华

摘要：随着计算机图形学的不断发展，实时渲染技术已成为数字内容创作中不可或缺的部分。传统的实时渲染方法多侧重于追求真实感。而近年来，风格化渲染(Non-Photorealistic Rendering, NPR)作为一种艺术表现手段，逐渐受到关注。NPR 技术通过模拟传统艺术风格，如水彩、素描和卡通等风格，为数字图像赋予独特的视觉效果 。然而，NPR 技术在实际应用中面临诸多挑战。首先，缺乏统一的标准和规范，导致不同项目间的风格实现差异较大，增加了开发和维护的难度。其次，NPR 渲染框架通常需要针对特定风格进行定制，缺乏通用性和扩展性，限制了其在多样化项目中的应用 。

本文旨在探索在 Unity 通用渲染管线（Universal Render Pipeline）框架下，结合 PBR（Physically Based Rendering）和 NPR 技术，实现一种半写实、半风格化的卡通渲染效果。通过对 URP 的 PBR 框架进行改造，构建一个兼具真实感与艺术风格并且有统一规范的渲染框架。

本文的主要工作包括对 Unity URP 默认 PBR 框架的光照重写与处理，常见的卡通渲染效果的实现，特殊材质的开发，配套工具脚本和 GUI 的开发，以及使用插件实现的骨骼物理动画。本文的最终效果实现了一套通用且灵活的半写实、半风格化的卡通渲染框架。

关键词：非真实感渲染，计算机图形学，基于物理的渲染，风格化渲染，Unity

Stylized Character Rendering in Unity under the PBR Framework

Author: Shuo Yan **Tutor: Weihua An**

Abstract: With the continuous advancement of computer graphics, real-time rendering has become an integral component of digital content creation. While traditional real-time rendering methods predominantly focus on photorealistic fidelity, Non-Photorealistic Rendering (NPR) has emerged as a prominent artistic expression technique in recent years. By emulating traditional artistic styles such as watercolor painting, sketch illustration, and cel-shading, NPR endows digital imagery with distinctive visual aesthetics. Nevertheless, practical implementations of NPR face significant challenges. Primarily, the absence of unified standards and specifications leads to substantial stylistic discrepancies across different projects, exacerbating development complexity and maintenance costs. Furthermore, existing NPR frameworks often require style-specific customization, lacking the versatility and scalability necessary for diverse application scenarios.

This study proposes a hybrid rendering framework that harmonizes Physically Based Rendering (PBR) fundamentals with NPR techniques within Unity's Universal Render Pipeline (URP) architecture. Through systematic modification of URP's PBR workflow, we aim to establish a standardized rendering solution that achieves an optimal balance between physical authenticity and artistic stylization. The proposed framework seeks to deliver semi-realistic cartoon rendering effects while maintaining cross-project consistency and extensibility in stylized visual development.

This work primarily involves rewriting and handling the lighting system within Unity URP's default PBR framework, implementing common cartoon rendering effects, developing special materials, creating supporting tool scripts and GUIs, and utilizing plugins to achieve skeletal physics animations. The final result is a versatile and flexible cartoon rendering framework that strikes a balance between semi-realistic and semi-stylized effects.

Key Words: Non-Photorealistic Rendering, Computer Graphics, Physics based Rendering, Stylized Rendering, Unity

第一章 引言

1.1 研究背景与意义

1.1.1 PBR 的发展历程

计算机图形学的发展历程中，渲染技术一直致力于追求视觉效果的真实感。早期的渲染方法，如 Phong 模型和 Blinn-Phong 模型，虽然在一定程度上模拟了光照与材质的相互作用，但由于其简化的假设和参数设置，难以准确模拟现实世界中的光照表现。随着计算机算力的提升和渲染算法的进步，研究者开始探索更为精确的物理模型，以实现更高质量的视觉效果。

于是，基于物理的渲染（Physically Based Rendering，以下简称 PBR）应运而生。它旨在通过模拟光与物质的物理交互，提供更真实的视觉呈现。PBR 的核心理念是，基于能量守恒和微表面理论，采用双向散射分布函数（BSDF）等数学模型，准确描述光在材质表面的反射、折射和散射行为[1,2,3]。

其实早在 20 世纪，PBR 就已经在图形学业界有了一些讨论。在 2010 年的 SIGGRAPH 会议上就已经有公开讨论的讲座，例如《SIGGRAPH 2010 Course: Physically-Based Shading Models in Film and Game Production》。在当时，PBR 大都需要大量复杂而且不直观的参数，它的优势并没有那么明显。

到了 2012 年，迪士尼的 Brent Burley 等人在 SIGGRAPH 会议上提出了迪士尼原则的 BRDF（Disney Principled BRDF）（Bidirectional Reflective Distribution Function，双向反射分布函数，以下简称 BRDF），并指出他们的着色模型是艺术导向型的，不一定是要完全的物理正确[4]，并且对 BRDF 的各项参数都进行了严谨的调查，并提出了清晰明确简单的解决方案。该模型具有高度的通用性，它将材质复杂的物理属性，用非常直观的少量变量表达了出来（如金属度 metallic、粗糙度 roughness），以此能让美术师们能用非常直观的少量参数，以及非常标准化的工作流，就能快速实现涉及大量不同材质的真实感的渲染工作。而这对于传统的着色模型来说，是不可能完成的任务。

在经历了迪士尼原则的 BRDF 革命之后，PBR 可以以相同的标准和规范大规模的批量生产风格高度一致的美术资产。这在电影工业界和游戏工业界引起了不小的轰动。从此，基于物理的渲染正式进入大众的视野。

1.1.2 NPR 的特点

非真实感渲染（Non- Photorealistic Rendering，以下简称 NPR）是相对于真实感渲染的技术，它旨在通过模拟艺术风格来生成图像，而非追求现实世界的逼真再现。

NPR 专注于为数字艺术启用各种表现风格[5]，作为一种完全以艺术风格为主导的渲染方法，其技术框架与评判标准往往更加主观和多变，这使得 NPR 的生产流程和规范化管理显得十分困难。

由于风格各不相同、常常缺乏统一的标准，NPR 的创作过程往往依赖于经验和个人技巧，容易导致制作团队之间在同一项目中的风格不一致，甚至影响到最终作品的质量[6,7]。此外，NPR 渲染虽然在艺术风格上具有强大的表现力，但其在一些复杂材质（如金属、玻璃等）的表达能力上较为薄弱，往往无法满足多种多样材质的渲染需求[7]。

1.1.3 次世代 NPR

近年来，随着 PBR 在游戏、电影和虚拟现实等领域越来越广泛的应用，工业界开始尝试将 PBR 与 NPR 相结合，力图在保留风格化效果的同时，弥补传统 NPR 在材质表达上的不足。

首先 PBR 提供了一套严谨的基于物理的渲染模型，能够更准确地模拟金属、玻璃等材质的反射和折射等特性，同时也能够在遵循能量守恒定律的基础上提供多种复杂的更加真实的光照与材质表现[7]。

其次，Disney PBR 下简洁直观的参数调节方式，为美术师提供了足够的自由度来控制和调整风格化效果，同时 PBR 框架的引入弥补了 NPR 缺乏统一规范和标准的缺点。

因此，将 PBR 的渲染技术与 NPR 的艺术风格结合，可以在增强渲染效果表现力的同时，也为 NPR 渲染的规范化和生产效率提升提供了支持[7]。而这种在 PBR 框架的基础和规范上进行的 NPR 渲染，我们通常将其称之为次世代 NPR。

1.2 研究内容

本文工作的主要目标是基于 Unity 通用渲染管线(Universal Rendering Pipeline，以下简称 URP)实现一套通用且灵活的风格化角色渲染框架。基于上述目标，本文工作包括如下几个部分。

首先，准备适合于本文研究的美术资产。在这里，本文收集了游戏《少女前

线 2：追放》的 3 套游戏角色模型和贴图。采用该模型的原因是，该游戏模型提供了一套完整的 PBR 金属工作流的美术资产，并且模型为卡通角色模型，符合本文的研究主题。

其次，本文使用 Blender 软件对美术资产进行了格式转换，网格、材质的分离合并等处理操作。

最后，在 Unity 通用渲染管线（URP）中通过修改内置的 PBR 框架和渲染特性开发，实现通用的、有标准生产规范的风格化角色渲染流程。这是本文的工作重点。主要工作包括：重写并修改 Unity URP 的默认 PBR 光照，实现卡通渲染常见的效果，特殊材质的开发，配套脚本工具和 ShaderGUI 的开发，基于 Dynamic Bone 插件的骨骼物理动画。

1.3 论文结构安排

本文共分为八章。各章节的内容安排如下。

第一章：引言。主要介绍了研究背景和意义，进行了相关技术概述，简要介绍 PBR 和 NPR 的基本原理及发展历程。并介绍了研究内容。

第二章：美术资产的收集与处理。包括使用 Blender 软件分离、合并网格模型，分离、合并材质素材，转换模型格式等。

第三章：Unity PBR 框架的重构。对 Unity URP 的 PBR 框架进行分析，并在此基础上进行重构，包括对漫反射和高光的处理等。

第四章：卡通渲染特性的实现。阐述对卡通渲染常见渲染特性的编写，如描边、边缘光、SDF 面部阴影和后处理等效果。

第五章：特殊材质的实现。包括基于多 Pass 的绒毛和丝袜材质开发等。

第六章：骨骼的物理效果实现。基于 Unity Dynamic Bone 插件，对角色的头发和衣物实现动态物理效果。

第七章：最终效果展示。使用 3 个不同的角色模型，展示并分析本文渲染框架的最终效果。

第八章：结论。对本文研究过程和结果进行总结，并提出感悟与展望。

第二章 美术资产的收集与处理

2.1 模型与贴图的收集

本文所采用的角色模型和贴图资源来自于“模之屋”网站(https://www.aplaybox.com/)。它是一个专注于模型分享与交流的平台，涵盖了动漫、游戏、电影等领域的各类模型资源。用户可以在这里找到海量的模型作品，并与其他模型爱好者进行交流与学习。

本文在该网站上共收集了三套来自游戏《少女前线 2：追放》的角色模型和贴图，用于对本文渲染效果进行演示分析。这三个角色模型如图 2.1 所示。

另外,收集的贴图分为两类,一类为 PBR 贴图(如图 2.2 所示),它包括 Albedo（基础色），Normal（法线），RMO（混合贴图）三个分量。其中，RMO 贴图的 R 通道记录了粗糙度，G 通道记录了金属度，B 通道记录了环境遮蔽信息等 PBR 计算需要用到的属性。

另一类为特殊贴图，如图 2.3 所示。其中，Face SDF 记录了面部的阴影信息，它用于在 Shader 中计算面部的 SDF 阴影；Hair_SpecMask 记录了头发的高光遮罩，用来控制高光的范围；Fur_Threhold 为绒毛渲染的阈值贴图，用来实现多层的绒毛效果。

图 2.1 本文采用的角色模型

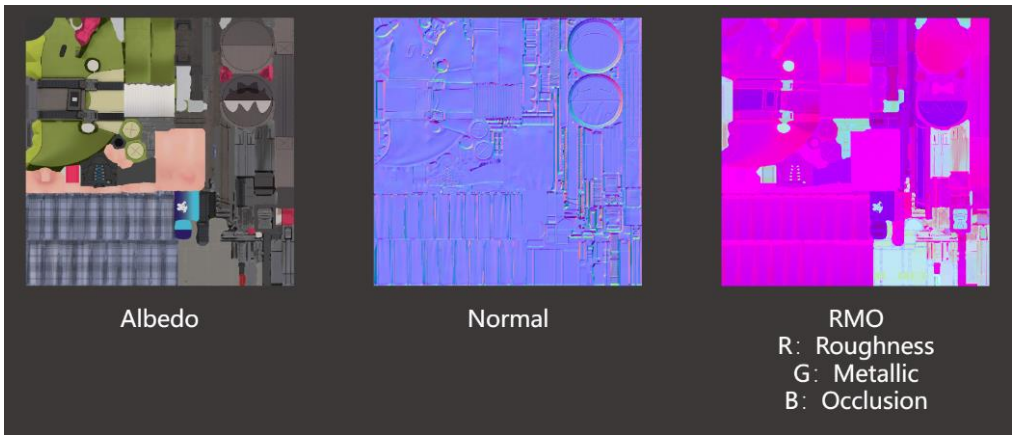


图 2.2 PBR 贴图

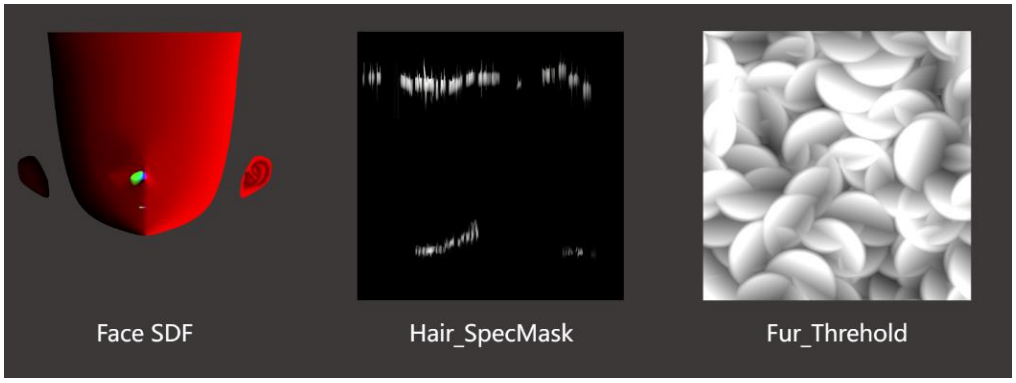


图 2.3 特殊贴图

2.2 基于 Blender 软件的网格处理

首先，原始角色模型是用来制作 MMD（MMD，全称 MikuMikuDance，是一个免费的 3D 动画制作软件)动画的专用模型，为 pmx 格式。而在游戏开发中，我们通常使用的文件格式是 fbx 格式，所以本文在 Blender 软件中使用 cats-blender 插件对角色模型进行了格式转换。

其次，由于角色模型在渲染计算上的特殊性(例如，有些时候角色的眼睛和其他部位的光照计算方式可能不同)，需要对每个子网格单独赋予相应的材质信息。所以，这里就要对网格进行分离。为了节省渲染的开销，本文将拥有同一套贴图 UV 的网格和其材质合并到了一起，这样可以在 Unity 内进行合批渲染，减少绘制命令调用的次数。

使用 Blender 分离、合并整理后的网格和材质列表如图 2.4 所示，三套模型

都遵循此思路处理。在图 2.4 中，左边为分离、重新合并后的材质列表，右边为分离、重新合并后的网格列表。

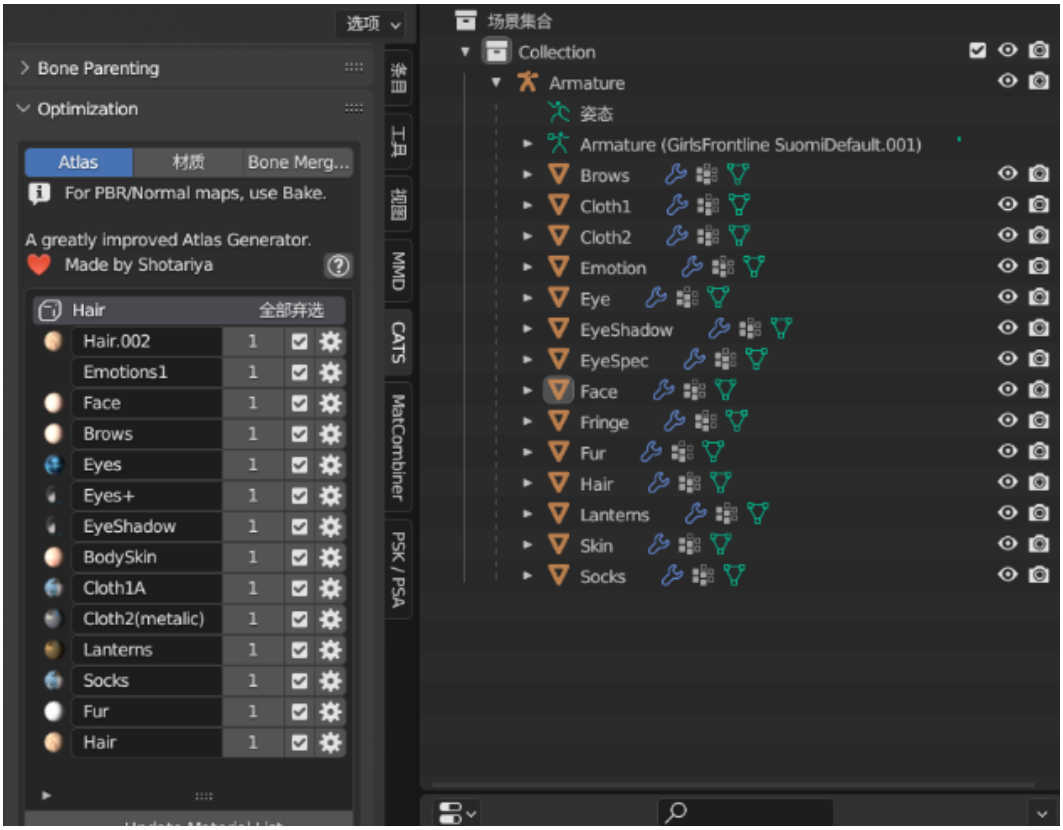


图 2.4 Blender 内整理后的模型网格和材质结构

第三章 Unity PBR 框架的重构

3.1 Unity URP 的 PBR 框架分析

几乎所有现代游戏引擎的 PBR 框架都基于公式(3.1)所示的反射方程。这个方程完整描述了一个基于物理的渲染模型。它表示：对于一个点 p 的反射光能，完全由所有入射到点 p 的光能及其方向、观察方向和物体表面的属性所决定。

其中，$\left(k_d \frac{c}{\pi} + k_s \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}\right)$就是根据统计学近似的 BRDF。加号左边为漫反射的 BRDF，c 表示基础色；右边为镜面反射的 BRDF，D 为法线分布函数，它从统计学上近似地表示了与某些半程向量 h（反射方向和法线的半角向量）

取向一致的微平面的比率。G 项（也可以叫 V 项，下文的 V 项和 G 项意义相同）则为几何遮蔽，它从统计学上近似的求得了微平面间相互遮蔽的比率，这种相互遮蔽会损耗光线的能量。F 项为菲涅尔项，描述的是被反射的光线对比光线被折射的部分所占的比率，这个比率会随着我们观察的角度不同而不同。

$$L_o(p,\omega_o) = \int_{\Omega} \left(k_d \frac{c}{\pi} + k_s \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} \right) L_i(p,\omega_i) n \cdot \omega_i d\omega_i \tag{3.1}$$

在实际应用中，公式(3.1)可以简化为公式(3.2)所示。其中,L_o为反射光，L_i为入射光，f_r为 BRDF，n为法线朝向，ω_i为入射光方向。

公式(3.2)中的f_r可以具体表述为公式(3.3)。这里，k_d表示漫反射系数；k_s表示镜面反射系数;$f_{lambert}$为漫反射 BRDF,一般采用 Lambert 模型;$f_{cook-torrance}$为镜面反射 BRDF，一般采用 Cook-Torrance 模型。

$$L_o = L_i * f_r * (n \cdot \omega_i) \tag{3.2}$$

$$f_r = k_d f_{lambert} + k_s f_{cook-torrance} \tag{3.3}$$

3.1.1 直接光的计算

Unity URP PBR 的直接光分为直接漫反射和直接镜面反射，为了节省性能开销，直接漫反射着色使用传统的 Lambert 模型计算得到，而非 Disney Lambert。而对于直接镜面反射，Unity 官方在 2015 年的 SIGGRAPH 会议上演示了它们对传统 Cook-Torrance BRDF 所作的优化后的版本。具体方法为，在 Cook-Torrance BRDF 的基础上,法线分布函数不变,对几何遮蔽函数 G 项进行了更改,从 SmithGGX 模型改为了更快速的 SKSm 拟合模型，并且对几何遮蔽 G 项和菲涅尔 F 项进行了合并,并使用机器学习方法对该结果进行简化拟合从而得到了一种非常快速的近似 BRDF[8]。

SmithGGX 模型的几何遮蔽计算如公式(3.4)所示，其中参数 k 的计算如公式(3.5)所示。SKSm 模型的几何遮蔽计算如公式(3.6)所示。Schlick 模型的菲涅尔项如公式(3.7)所示(其中 F_0 为基础反照率，h 为视线和法线的半程向量)。V 项和 F 项的合并拟合结果如公式(3.8)所示。

$$V_{SmithGGX} = \frac{1}{\big((N \cdot L) \cdot (1-k) + k\big)\big((N \cdot V) \cdot (1-k) + k\big)} \tag{3.4}$$

$$k = \frac{(roughess + 1)^2}{8} \tag{3.5}$$

$$V_{SKSm} = \frac{1}{(L \cdot H)^2(1 - roughness^2) + roughness^2} \tag{3.6}$$

$$F_{Schlick} = F_0 + (1 - F_0)\big(1 - (h \cdot v)\big)^5 \tag{3.7}$$

$$(V \cdot F)_{approx} = \frac{F_0}{(L \cdot H)^2\big(roughness + 0.5\big)} \tag{3.8}$$

3.1.2 间接光的计算

Unity 的间接光同样分为漫反射和镜面反射。在实时渲染中，间接光通常使用 IBL(Image Based Lighting，意为基于图像的光照)来进行计算。IBL 通常使用一张 HDR 全景图来记录场景中环境光的集合，我们可以将全景贴图的每个像素视为光源，在渲染方程中直接使用它。这种方式可以有效地捕捉环境的全局光照和氛围，使物体更好地融入其环境。

在间接光漫反射方面，由于漫反射的 BRDF 是一个常数，那么我们可以对漫反射的结果进行预先计算，并使用球谐函数进行投影，烘焙存储到多项式中，并在进行计算时直接对球谐函数进行采样得到[9]。间接漫反射的计算如公式(3.9)和其变换结果公式(3.10)所示。在预积分完成后，相当于对整张图像做了一次滤波，用该结果乘以漫反射系数即为间接光的漫反射结果。所以，实际应用的计算形式如公式(3.11)所示。

$$L_o(p, \omega_o) = \int_{\Omega} \left(k_d \frac{c}{\pi}\right) L_i(p, \omega_i) n \cdot \omega_i d\omega_i \tag{3.9}$$

$$L_o(p, \omega_o) = k_d \frac{c}{\pi} \int_{\Omega} L_i(p, \omega_i) n \cdot \omega_i d\omega_i \tag{3.10}$$

$$L_{indirect\ diffuse} = k_d * IBL_{pre\ filtering} \tag{3.11}$$

间接光镜面反射方面，由于高光通常受视角ω_o，光照方向ω_i和粗糙度的影响(如公式 3.12 所示)，因此常规的计算中，通常会使用两种参数构建一张二维的 LUT（Look Up Table,查找表，以下简称 LUT）来存储采样的 BRDF[10]，如在 Unreal 引擎中采用的就是这种方法。

但 Unity URP 中，为了节省采样贴图的带宽开销并没有使用这种方法，而是使用了粗糙度和 F_0 等参数进行拟合得到近似的 BRDF[8]，如公式(3.13)所示。光照的处理则和市面上大多算法相同，积分的第一部分仍然为预滤波环境贴图，但这次考虑了粗糙度。因为随着粗糙度的增加，参与环境贴图卷积的采样向量会更分散，导致反射更模糊，所以对于卷积的每个粗糙度级别，我们将按顺序把模糊后的结果存储在预滤波贴图的 mipmap 中。光照则最终来自用粗糙度计算得到的 mip 等级采样的 IBL 的不同层级预滤波计算好的 Mipmap[10]。Mip 等级的计算如公式(3.14)所示；Mip 采样的 IBL 结果如公式(3.15)所示。最终的光照计算结果如公式(3.16)所示。

$$L_o(p,\omega_o) = \int_{\Omega} L_i(p,\omega_i) n\cdot\omega_i d\omega_i * \int_{\Omega} fr(p,\omega_i,\omega_o) n\cdot\omega_i d\omega_i \tag{3.12}$$

$$BRDF_{env_{specular}} = \big(1 - max(roughness, N\cdot V)\big)^3 + F_0 \tag{3.13}$$

$$miplevel = \ roughness * (1.7 - 0.7 * roughness) \tag{3.14}$$

$$IBL_{mip} = SampleTextureLod\big(IBL_{Mips}\ ,\ miplevel\big) \tag{3.15}$$

$$L_{indirectspecular} \ = \ IBL_{mip} * \ BRDF_{envspecular} \tag{3.16}$$

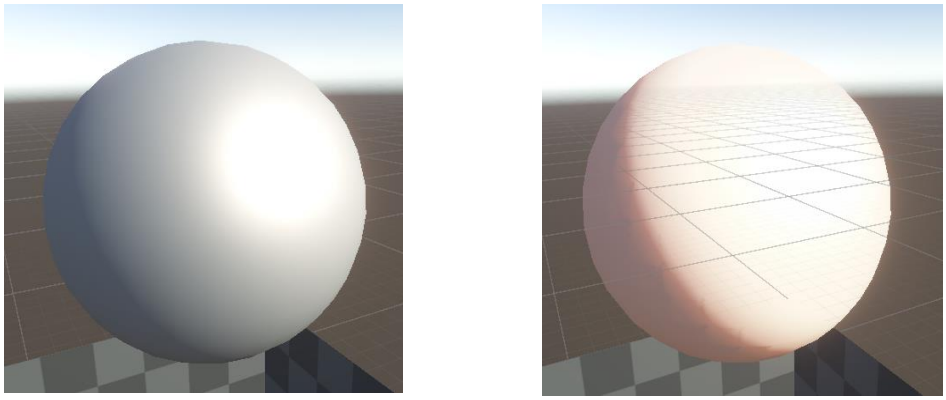
3.2 Unity URP 的 PBR 框架重写与处理

卡通渲染和真实感渲染的一大区别就在于，卡通渲染的色阶更少。对此，我们通常可以使用 step 函数或者 smoothstep 函数对原本的光照结果进行处理，并对其进行手动颜色分阶处理。但手动分阶可控性较差，因此我们还可以使用 ramp 贴图对光照结果进行映射的方法来处理色阶，其中最主要的是对直接光的$n\cdot\omega_i$项和高光 BRDF 的法线分布函数 D 项进行映射[11,12,13,14]。

对于间接光，它们虽然可以显著增加渲染的立体感，但这并非卡通渲染的本意，所以为了达到一个平衡的效果，我们可以人为减弱间接光的立体感，比如对于间接漫反射可以拍平采样的法线，而对于间接高光我们可以使用 Ramp 映射成较少的色阶[12,13]。

在本文中，角色模型的每个部位都配备了 4 条带状的 Ramp 贴图。它们分别用于对$n\cdot\omega_i$、高光 BRDF 的法线分布函数 D 项、间接高光、额外光源四个部分的映射处理。图 3.1(a)和(b)分别展示了相同参数下，在使用 Ramp 贴图（图 3.2）

处理之前和处理之后的光照结果。



(a)未 Ramp 的 PBR 光照　　(b)Ramp 后的 PBR 光照

图 3.1 Ramp 前后的光照对比

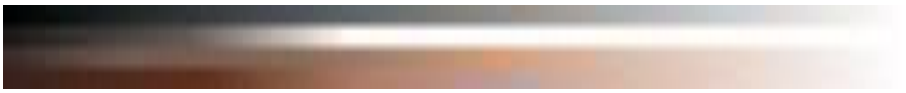


图 3.2

第四章 卡通渲染特性的实现

对于一个卡通动画来说，角色身上常见的表现效果如图(4.1)所示。本文的渲染框架着重实现了图中的几种特性，包括描边、面部阴影、刘海投影、刘海透眉、边缘光、头发高光，还包括提升氛围感的后处理。最终达到了有卡通动画感的效果。

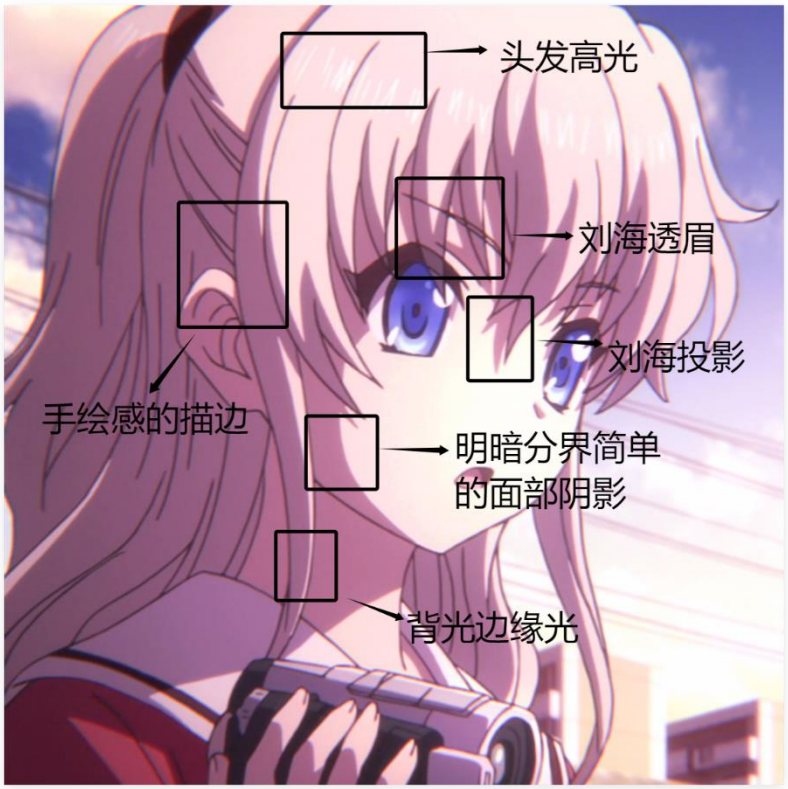


图 4.1 卡通动画角色身上的常见表现

4.1 描边

描边作为卡通渲染中常见的特性，多年来发展出了各种算法。但目前游戏工业界最常用的仍然是 Back Facing 算法，因为它的计算消耗很低。该算法的主要思路是，将模型顶点沿法线方向偏移，并开启正面剔除，只渲染背面，从而在原模型外部留下黑色描边[14,15]。本文采用的也是这种算法。

需要说明的是，这种算法存在一些问题：首先，在模型的硬表面的拐点，由于存在 2 个位置相同但法线朝向不同的顶点，导致在顶点沿法线外扩后会出现描边断裂的现象[15]，如图 4.1(a)所示。其次，由于它的本质是渲染了第二次模型，

这会导致它会受到顶点着色器的几何变换影响。

针对第一个问题，本文在 Unity 中开发了一个用于平滑法线的脚本，它会遍历模型所有的顶点，而针对位置相同的顶点，该脚本会对包含这些顶点的所有三角形遍历，获得与该顶点相邻两条边的夹角作为权重，然后以此作为权重重新计算该位置的所有顶点法线朝向。

针对第二个问题，本文选用在视图空间进行顶点的法线方向外扩，这样可以很容易的获取到摄像机到模型的深度，从而根据深度和 FOV 共同对描边的粗细进行调整，其中 FOV 使用脚本从相机组件向 Shader 进行参数传递。

描边算法的伪代码见算法 4.1 所示，法线平滑算法的伪代码见算法 4.2 所示。使用模型自带法线描边和使用平滑后的模型法线描边的对比效果，见图4.1，其中(a)为未平滑，(b)为平滑后的。

算法 4.1

```
Vertex Shader：
    ……
    Cull Front   //剔除正面
    depthFactor = abs(posVS.z/1000) //深度影响描边粗细
    posVS += (normalVS.xy, _Flatten) * _Width * depthFactor * FOV //最终的外扩计算
    ……
```

算法 4.2

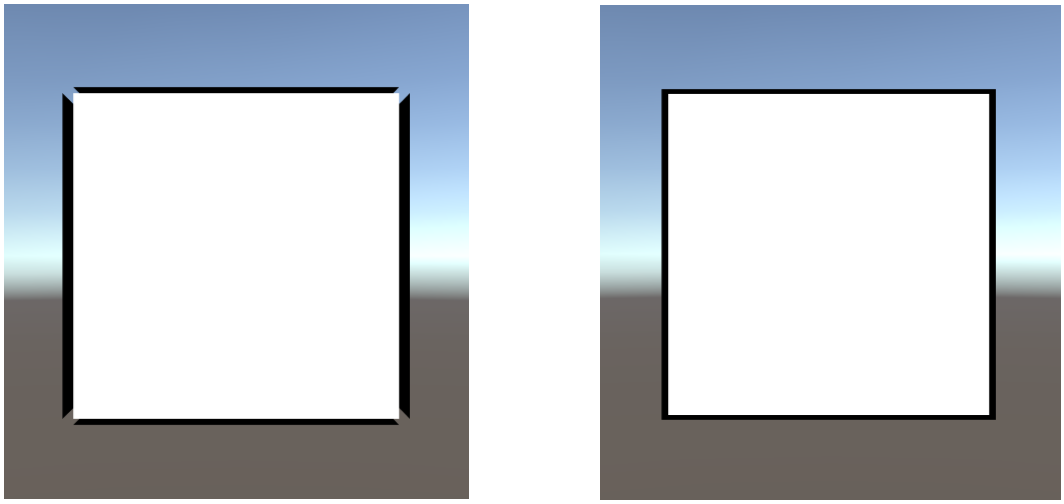
```
For each triangle in mesh:
    v0, v1, v2 = triangle.vertices

    for each vertex in [v0, v1, v2]:
        //计算当前顶点在三角形中的两条边向量
        edgeA = next_vertex.position - current_vertex.position
        edgeB = prev_vertex.position - current_vertex.position
        // 计算面法线和角度权重
        angle = arccos(normalize(edgeA) · normalize(edgeB))
        face_normal = cross(edgeA, edgeB).normalized()
        store(vertex, face_normal, angle)

```

```
For each vertex in mesh:
    //加权混合所有关联面法线
    blended_normal = Σ(face_normal * (angle / total_angle))
    blended_normal.normalize()

    //转换到切线空间
    TBN = build_TBN_matrix(vertex.tangent, vertex.normal)
    smooth_normal = (TBN.transposed() * blended_normal).normalized()
    save_to_uv7(smooth_normal)
```



(a)未平滑法线的描边　　(b)平滑法线后的描边

图 4.2

4.2 边缘光

边缘光在卡通动画和动漫中也是一种经常出现的效果，它被用来模拟现实世界中的菲涅尔现象。尤其是在角色背光情况下，效果会非常明显。它可以增强角色的轮廓感和氛围感。

本文对边缘光的计算算法，采用了屏幕空间 UV 的深度偏移计算阈值来实现。具体思路如下：

首先，在 Unity URP 管线中，我们可以插入如下渲染流程：在正式计算着色之前，渲染场景中所有不透明物体的深度，并把它记录到一张全局的深度图上。

然后，在计算着色时，我们在片元着色器中计算角色所在片元的屏幕空间坐标，以此作为 UV 去采样深度图。

接着，将屏幕空间的 UV 向屏幕空间的法线方向偏移，再次采样深度图。这

里设定一个阈值。如果两次采样的深度大于一个阈值，我们便可以认定，这是一个边缘，并进行着色计算。

同时边缘光还会受到灯光颜色，灯光方向的影响。前面提到，边缘光通常用于表现角色背光的氛围。因此，本文还对边缘光乘以了一个强度系数，强度系数由 1 减去角色的朝向向量和灯光的方向向量的点积乘以灯光强度和灯光颜色得到。算法 4.3 展示了边缘光算法的伪代码。角色在背光状态下，关闭和开启边缘光的效果分别如图 4.3 和图 4.4 所示。可以看到开启边缘光后角色的边缘出现了亮光效果。

算法 4.3

- Fragment Shader:
- ……
- rimIntensity = 1 - (dot(viewDirWS,lightDir)*0.5+0.5) //背光强度变化
- OriginDepth = SampleDepthTex(ScreenUV) //原始深度
- OffsetPosVS = float3(posVS.xy + normalVS.xy * _OffsetMul, posVS.z)
- OffsetPosCS = TransformViewToHClip(OffsetPosVS)
- OffsetScreenUV = TransformHClipToViewPortPos(OffsetPosCS).xy
- OffsetDepth = SampleDepthTex (OffsetScreenUV)//偏移后的深度
- diffDepth = OffsetDepth – OriginDepth //原深度和偏移后深度的差值
- RimMask = step(_Threshold, diffDepth) //使用差值作为阈值生成 Mask
- RimLight = RimMask*_RimLightColor * (diffDepth*0.3)* rimIntensity//最终计算
- ……

图 4.3 无边缘光

图 4.4 有边缘光

4.3 SDF 面部阴影

有向距离场（Signed Distance Field，以下简称 SDF）是一种用于表示空间中物体形状的数学工具，它为每个点分配一个数值，记录该点到最近物体表面的距离。在卡通渲染中，SDF 被广泛应用于生成面部阴影，尤其是在二维风格化角色的渲染中。

SDF 面部阴影是目前卡通渲染风格游戏工业中常用到的算法。由于卡通角色的面部模型的拓扑结构的特殊性以及 ShadowMap 的精度问题，想在卡通角色模型上渲染阴影和漂亮的伦勃朗光非常困难。因此工业界常常把希望展现出来的不同光照角度的阴影画在一张张图上，然后计算它们的 SDF 并进行混合得到一张融合了所有光照情况阴影 SDF 的图中。在着色计算时，根据光照方向和面部朝向的点积来对采样出来的 SDF 贴图进行 step 运算得到当前时刻的阴影[16]。

由于贴图只能记录一种顺序的光照变化，（例如如果记录了从左向右，那么则无法记录从右向左），因此，采样时还可以根据光照方向和面部向右的方向，计算此时灯光是在角色的左半边还是右半边。以此来决定采样 SDF 贴图时是否反转 UV 的 x 轴来渲染正确的光照结果，伪码见算法 4.4。

算法 4.4

```
Fragment Shader:
    ……
//判断光在角色左/右
    faceLightDotX = dot(lightDirWS.xz, faceRightDirWS.xz) //判断光在角色左/右
//用于判断阴影边界的阈值
    ctrl = 1-(dot(lightDirWS.xz, faceFrontDirWS.xz)*0.5+0.5)
    faceUV.x = faceLightDotX > 0 ? 1 - faceUV.x : faceUV.x
```

- faceShadow = step(ctrl, SampleTex(FaceSDF, faceUV))
- ……

4.4 刘海投影

在 4.3 节中提到，ShadowMap 在渲染精细的角色阴影时会出现精度问题。在渲染刘海的投影时，同样会出现这个问题。由于 ShadowMap 通常会记录整个场景的阴影信息，而角色在场景中又仅占很小一部分，所以使用默认的 ShadowMap 渲染角色身上的高质量投影非常困难。对此，通常有两种做法，一种是对角色做包围盒，然后投影到光源空间，单独渲染一张 ShadowMap 去渲染高质量的角色阴影。另一种做法则是使用模板测试偏移。本文中采用了第二种方法来渲染高质量的刘海投影。具体思路如下。

在第一个 Pass 中，对角色的刘海进行渲染并写入固定模板值，并向灯光方向偏移一定距离；然后在第二个 Pass 中渲染脸部，模板测试设置为等于固定模板值才通过，这样就可以得到两个模型相交的部分。此时再对第二个 Pass 中的脸部进行阴影的着色计算就可以得到如图所示的高质量刘海投影了。思路见算法 4.5。

图 4.5 展示了在侧光下若不使用 SDF 面部阴影和模板测试刘海投影，仅凭 Unity 默认的 ShadowMap 渲染出来的低质量阴影。图 4.6 展示了在侧光下开启 SDF 面部阴影和使用模板测试高质量投影的效果。可以看到，使用 Unity 默认 ShadowMap 渲染出来的刘海投影，由于精度过低把角色的整张脸全部挡住了。而使用了 SDF 面部阴影和模板测试生成的刘海投影的角色，面部的明暗部能根据光照的方向进行明显的区分。

算法 4.5

- Pass1(Hair)
- Stencil Ref 1, Comp NotEqual, Pass Zero
- ColorMask0
- Vertex Shader:
- ……
- posCS.xy += lightDir.xy * _Offset //渲染头发网格并向灯光方向偏移
- ……
-
- Pass2(Face)

- Stencil Ref 3, Comp Greater, Pass Keep //渲染脸模型并与头发求交
- Blend Op Min, Blend SrcAlpha OneMinusSrcAlpha
- …
- //NPR Shading
- …

图 4.5 使用 Unity 默认的 ShadowMap

图 4.6 使用 SDF 面部阴影和模板测试刘海投影

4.5 刘海透眉

刘海透眉同样是在卡通动画和动漫中经常出现的效果。它指的是，在手绘的动画中，角色的眉毛虽然以物理空间来说是应该被遮挡的，但是始终飘在角色的刘海上面，或是以半透明的形式叠加在上面的效果。在 3D 渲染中，如果想实现这种错误的遮挡关系同样需要使用模板测试来实现。

在常见的渲染管线中，当片元着色器完成了着色计算时，片元还要经过深度测试和模板测试才会显示在屏幕上，因此可以依靠模板测试来实现这种物理不正确的遮挡关系。

首先，渲染角色的眼睛眉毛部分，写入模板值 x(本文中，x=4)；然后可以渲染脸部，写入模板值 x-1；最后渲染刘海，模板参考值设为 x-1，模板测试设置为大于等于 x-1。这样，由于角色刘海在眉毛的部分虽然通过了深度测试，但未通过模板测试，就会显示在刘海前方，如图 4.7 所示。此时，再次渲染一遍头发，并设置透明度混合，即可得到眉毛半透明且透过刘海的效果，如图 4.8 所示。

图 4.7 刘海飘眉效果

图 4.8 刘海透眉（半透明）

4.6 后处理

4.6.1 Bloom 效果

Bloom 在卡通渲染中属于一种非常常见的后处理。但与真实感渲染不同，Bloom 不单单用来表达光晕效果，而是用于提升一种画面整体的氛围感。本文中直接使用了 URP 中后处理组件 Global Volume 中提供的 Bloom 效果。关闭和开启 Bloom 效果如图 4.9 和图 4.10 所示。

图 4.9 无 Bloom

图 4.10 开启 Bloom

4.6.2 ToneMapping

传统的真实感渲染中，常常使用 ACES 曲线进行 Tonemapping，它将 HDR 映射为 LDR。但在卡通渲染中，ACES 曲线会导致画面的饱和度大大降低，这有悖于艺术导向的渲染风格。所以目前工业界中，卡通渲染常用的两种曲线为 Gran Turismo 曲线和 Simple-ACES 曲线。这两种曲线可以有效缓和 ACES 的饱和度降低问

题。

Gran Turismo 曲线是 Polyphony Digital 在研发《Gran Turismo》系列游戏时为其定制的色调映射曲线。该曲线具备可变峰值、线性中段、可调肩部/足部。在保证高光不过曝、阴影不塌陷的同时，保留了中间调的真实细节[17]。同时也正是因为其不会导致饱和度降低，现在被许多卡通渲染游戏所使用。

Simple-ACES 则是传统 ACES 曲线的调整版本。传统的 ACES 曲线会导致图片的色调变化过大。卡通渲染的颜色一般都指定好的，不希望在渲染过程中发生改变。所以在卡通渲染中，Tonemapping 的曲线在一般 0 到 0.5 之间都是线性的[18]。

两种曲线的走势如图 4.11 和 4.12 所示。图 4.13 显示了关闭 ToneMapping 的计算效果；图 4.14 显示了开启 Gran Turismo ToneMapping 的效果；图 4.15 显示了开启 Simple-ACES ToneMapping 的效果。

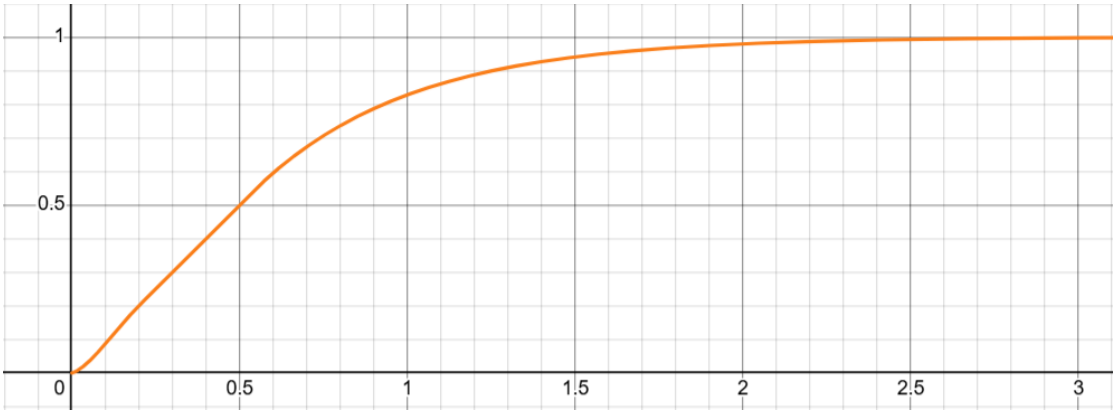


图 4.11 Gran-Turismo Curve

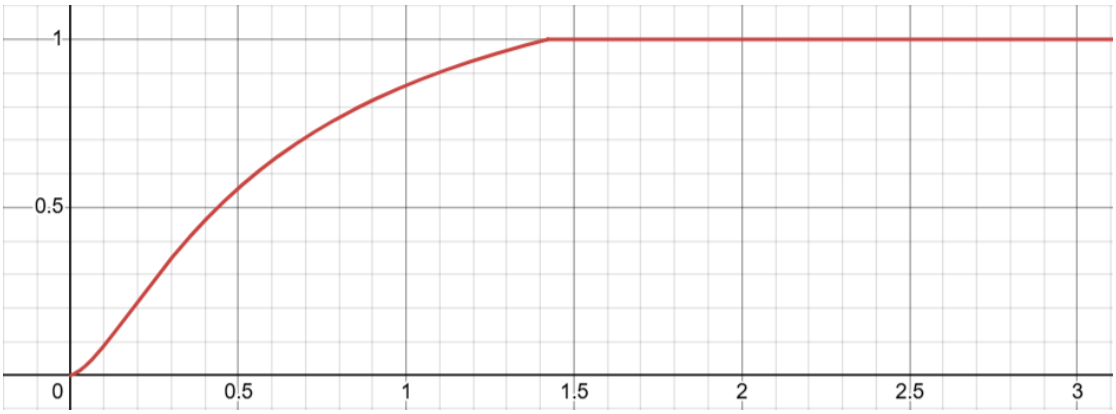


图 4.12 Simple-ACES Curve

图 4.13 无 ToneMapping

图 4.14 Gran Turismo

图 4.15 Simple-ACES

第五章 特殊材质的开发

5.1 绒毛

由于收集到的角色模型身上有明显的毛发区域，所以本文额外开发了绒毛材质。绒毛采用了多 Pass 方法，对同一模型进行多次渲染，并且每次对顶点沿法线方向外扩，同时采样阈值图，并逐级缩减像素剔除的阈值和混合透明度[19,20]。本研究并没有对绒毛进行额外的光照处理，它和角色其他部位的光照计算相同，仍然接收修改后的 PBR 框架处理，包括多光源的处理。绒毛材质算法伪代码见算法 5.1，效果如图 5.1 所示。

算法 5.1

```
inter = 1/_StepCount
For each Pass i from 0 to _StepCount:
    Furoffset = i * inter
    Vertex Shader:
        posOS += normalOS * Furoffset *_FurLength //顶点外扩
        ……
    Fragment Shader:
        ……
        Threhold = SampleTex(ThreholdTex,FurUV) //采样阈值图
        Alpha = 1-step(lerp(0,_CutOffEnd, Furoffset), Threhold) //渐变透明度
        Clip(Threhold - FurOffset) //阈值剔除
        ……
```

图 5.1 绒毛材质

5.2 丝袜

同样的，收集到的角色模型身上有丝袜区域，所以额外开发了丝袜材质。根据丝袜的光照特性，和视角越接近的地方，丝袜应该更会透露出皮肤的颜色，而越是和视角不接近的地方，则会呈现出丝袜的颜色。因此，本研究采用了类似菲尼尔项，也就是视角与法线的点积来插值给定的丝袜颜色和皮肤颜色，用这个颜色值来作为 PBR 计算的 Albedo，即可得到如图 5.2 所示效果。

图 5.2 丝袜材质效果展示

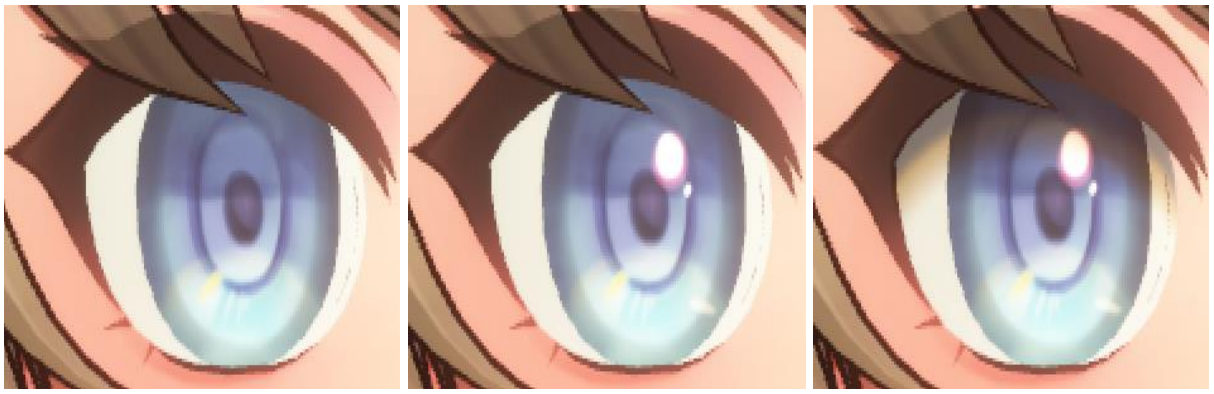
5.3 头发高光

卡通渲染中角色头发的高光通常有两种方法。为了渲染头发的各向异性高光，一种是使用经验模型如 Kajiya-Kay，使用切线和光照方向的正弦计算[21]。另一种则采用常规的光照模型最后乘以一个各向异性的遮罩。本研究采用了第二种方法，因为收集到的美术资产中提供了各向异性遮罩，并且头发的模型已经烘焙了球形法线。在第二种方法的基础上，本研究还对各向异性遮罩的采样 UV 加入了视角的影响，来模拟真实世界中头发当视角方向变化时高光偏移的效果。图 5.3 展示了不同视角下头发高光位置的偏移。

图 5.3 不同视角下的高光位置表现

5.4 眼睛

收集到的美术资产中，角色的眼睛模型和贴图被分为了三部分：眼睛的阴影、眼睛的高光、眼球。因此，在制作眼睛的材质时屏蔽了引擎的光照，而采用无光照的模式渲染眼球；眼睛的阴影和高光则采用透明度混合叠加到眼睛上来制作。图 5.4 的(a)(b)(c)分别展示了眼球、眼球加高光、眼球加高光加阴影的渲染效果。



(a)眼球　　(b)眼球+高光　　(c)眼球+高光+阴影

图 5.4

5.5 统一的 Custom Shader GUI

本研究还实现了配套的自定义 ShaderGUI。基于 Unity 的 ShaderGUI API，实现了对 Shader 面板的高效控制和管理，方便美术师调参使用。面板基本属性如图 5.5 所示，下面进行具体解释。

- Toggle 折叠页可以选择材质适用的区域，比如是眼睛，头发，脸，还是丝袜或其他部位；并且可以选择是否使用 RMO 贴图。使用 RMO 贴图时，PBR 的光照计算则采用 RMO 贴图记录的信息；否则会出现滑条调整对应参数值。自定义透明度则决定，半透明物体的混合透明度是手动指定，还是由 Albedo 贴图的 Alpha 通道决定。
- RimLight 开关和自发光开关决定是否启用两种效果，并且勾选了相应开关会自动出现调节对应效果的参数页。
- Settings 则可以对渲染状态调整，如剔除模式，深度写入，混合模式，模板测试等。

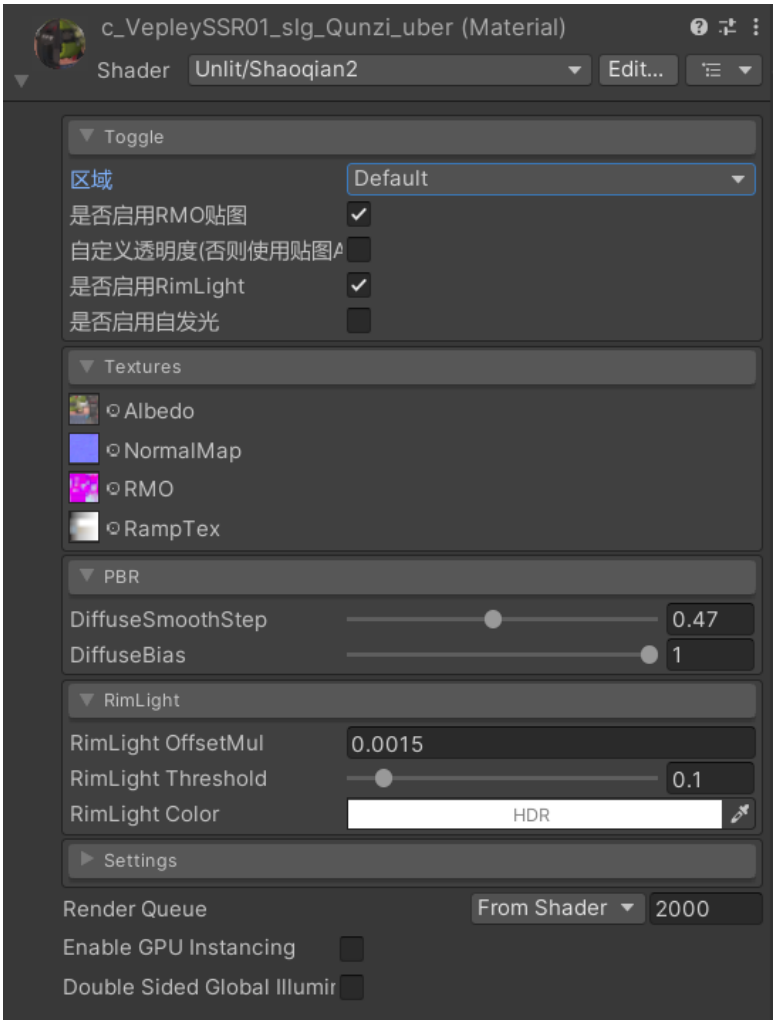


图 5.5 配套 Custom ShaderGUI 界面展示

第六章 骨骼的物理效果实现

DynamicBone 是一个基于弹簧振子算法，实现树状柔体物理模拟的插件。它可以用来快速在 Unity 中实现骨骼的物理效果。本研究中，使用该插件对角色的头发、裙子的骨骼添加了物理效果。在播放动画时，相应的骨骼会控制网格进行摆动。

图 6.1 为该插件的设置界面。其中，Root 绑定了角色需要有物理动画的骨骼的根骨；Upadate Rate 为刷新率，设为了 60 帧；调节了 Damping（阻尼），Elasticity（弹性），Stiffness（刚性），Inert（惰性）等参数，为相应部位的

骨骼添加了碰撞体，并调节了碰撞体的半径 Radius。

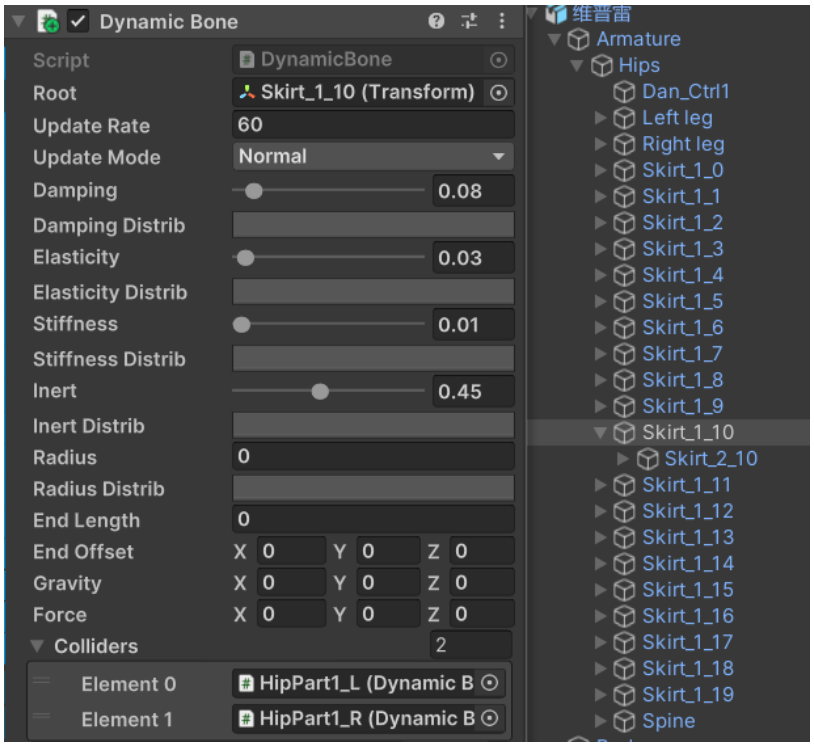


图 6.1 对角色骨骼使用 Dynamic Bone 插件

第七章 最终效果展示

基于本文的计算框架，角色模型在不同光照角度、多光源情况下，都可以正确表现出预期的卡通渲染特性，满足卡通渲染的表现。其中描边、边缘光、面部阴影、刘海投影、绒毛、丝袜等均可以表现出预期的光照效果。

7.1 角色正面受光表现

图 7.1，图 7.2 和图 7.3 分别展示了三个角色在本渲染框架下正面受光的近镜头表现。图 7.4 展示了三个角色在本框架下正面受光的整体表现。由这些图示可知，本渲染框架对不同模型的渲染效果具有一致性。衣服具有相同的 PBR 表现，面部、头发等具有统一风格的卡通渲染表现。

图 7.1 维普雷正面受光近镜头表现

图 7.2 索米正面受光近镜头表现

图 7.3 莱娜正面受光近镜头

图 7.4 三个演示角色正面受光的整体表现

7.2 角色侧面受光表现

图 7.5 至图 7.7 分别展示了三个角色在本渲染框架下侧面受光的近镜头表现。图 7.8 展示了三个角色在本框架下侧面受光的整体表现。可以看到，所有角色的面部阴影、刘海投影、明暗分界线和明暗处的边缘光变化，都可以正确表现出来。

图 7.5 维普雷侧面受光近镜头表现

图 7.6 索米侧面受光近镜头表现

图 7.7 莱娜侧面受光近镜头表现

图 7.8 三个演示角色侧面受光的整体表现

7.3 角色背面受光表现

图 7.9 至图 7.11 分别展示了三个角色在本渲染框架下背面受光的近镜头表现。图 7.12 展示了三个角色在本框架下背面受光的整体表现。可以观察到，三个角色边缘光的整体情况和衰减都可以正确表现出来。

图 7.9 维普雷背面受光近镜头表现

图 7.10 索米背面受光近镜头表现

图 7.11 莱娜背面受光近镜头表现

图 7.12 三个演示角色背面受光的整体表现

7.4 多光源下的表现

图 7.13 和图 7.14 分别展示了在本渲染框架下单个角色和多个角色的多光源表现。这里，在角色之间分别放置了一个红色点光源和一个蓝色点光源。可以看到，包括丝袜、绒毛在内的材质都可以正确受到多光源的强度和颜色影响。并且包括边缘光、SDF 面部阴影等都可以正确接收到多个光源的强度和颜色影响。

图 7.13 角色多光源下的近镜头表现

图 7.14 三个演示角色多光源下的整体表现

第八章 结论

8.1 总结

本文基于 Unity URP 管线，开发了一套完备的 PBR 框架下的 NPR 渲染框架，

实现了一种半写实半卡通的渲染风格，并使用游戏《少女前线 2：追放》官方公开的美术资产加以展示。其包含一套完整的卡通角色渲染材质和配套 ShaderGUI，以及常见的卡通角色渲染 Feature 及其配套工具链。理论上，只要有配套的符合标准的美术资产，那么这套渲染框架可以进行通用。并且，由于基于 PBR 框架开发，本套渲染流程修改和维护的成本极低，并且可以通过调参和修改框架实现灵活的定制化开发。

8.2 展望

PBR 和 NPR 诞生之初就没有对立关系，近些年来诞生的 PBR 和 NPR 融合风格的游戏作品更是印证了这一点。需要看到的是，从微平面理论提出到 Disney PBR 一统江湖，再到结合 PBR 理论落地的 NPR，近些年来的 PBR 和 NPR 的生产都愈发成熟。很大一部分原因在于，在工业界，人们更关心的是生产流程的可规范性和可落地性。统一简易的标准是可以大规模量产的前提。并且，对于一个基于游戏引擎的庞大且周期很长工程来说，整个代码框架的可维护性和灵活性往往比其它东西重要。因此，如何将理论转化为实际生产中实用的规范在未来将越来越重要。本框架的价值正在于通过将一个个风格化算法封装为标准化的通用渲染模块，既保证了渲染的通用性，又通过参数化接口为美术定制创作保留了弹性空间。

最后，随着游戏工业化，游戏越来越精品化，上下游产业链越趋于完备，一个理论的普及将更加取决于其与工业化管线的契合度。当理论突破真正转化为流水线上的标准化工具时，图形学的学术智慧才能最终转化为触手可及的艺术表达力，这正是游戏工业化进程中工程师与科学家共同追寻的技术诗篇。而在未来，凭借愈发强大的 GPU 算力和深度学习神经网络介入，我相信，实时渲染终将模糊现实与虚拟的界线——就像《头号玩家》中指尖轻触即可重构世界的愿景，我们正在用数学的严谨编织幻梦的经纬，让每一行着色器代码都成为艺术家手中的画笔，在硅基与碳基的共鸣中书写属于这个时代的数字浪漫主义。

参考文献

[1] Cook R. L., Torrance K. E. A Reflectance Model for Computer Graphics[J]. Computer Graphics (SIGGRAPH 81 Proceedings), 1981, 15(3): 307–316.
[2] Walter B., Marschner S. R., Li H., Torrance K. E. Microfacet Models for Refraction Through Rough Surfaces[J]. Computer Graphics Forum, 2007, 26(4): 195–204.
[3] Yan L. Q. GAMES202 Lecture 10[EB/OL]. (2021-05-08)[2025-03-02]. University of California, Santa Barbara. https://sites.cs.ucsb.edu/~lingqi/teaching/resources/GAMES202_Lecture_10.pdf
[4] Burley B. Physically Based Shading at Disney[C]//Proceedings of the 21st Symposium on Computer Animation. Los Angeles: ACM SIGGRAPH, 2012: 7–12.
[5] Wikipedia. Non-Photorealistic Rendering[EB/OL]. (2002–)[2025-05-01]. https://en.wikipedia.org/wiki/Non-photorealistic_rendering
[6] Yan L. Q. GAMES202 Lecture 11[EB/OL]. (2021-05-15)[2025-03-02]. University of California, Santa Barbara. https://sites.cs.ucsb.edu/~lingqi/teaching/resources/GAMES202_Lecture_11.pdf
[7] Zhou C. Finding Harmony in Anime Style and Physically Based Rendering[C]//Proceedings of the Game Developers Conference 2022. San Francisco: UBM Tech, 2022.
[8] Unity Technologies. PBR Optimization[C]//Zioma R. Moving Mobile Graphics: ACM SIGGRAPH 2015 Courses. Los Angeles: ACM, 2015: Course 18, PBR Optimization.
[9] Ramamoorthi R., Hanrahan P. An Efficient Representation for Irradiance Environment Maps[C]//Proceedings of SIGGRAPH 2001. New York: ACM, 2001: 497–500.
[10] Karis B. Real Shading in Unreal Engine 4[C]. Raleigh, NC: Epic Games, 2013.
[11] Gooch A., Gooch B., Shirley P. A Non-Photorealistic Lighting Model for Automatic Technical Illustration[C]//Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering. San Diego: ACM SIGGRAPH, 2002: 19–26.
[12] Unity 官方. Unity 大咖作客丨网易 TA 大咖带你走近卡通渲染（下）：基于 PBR 的卡通表现[EB/OL]. (2021-11-08)[2025-03-02]. https://zhuanlan.zhihu.com/p/430557149
[13] Unity 官方. Unity 大咖作客丨网易 TA 大咖带你走近卡通渲染（上）：基于 Trick 的卡通世界[EB/OL]. (2021-11-10)[2025-03-02]. https://zhuanlan.zhihu.com/p/430471476
[14] Hutchins A., Kim S. Advanced Real-Time Cel Shading Techniques in OpenGL[EB/OL]. (2012)[2025-05-03]. Rensselaer Polytechnic Institute, Dept. of Computer Science.
[15] 2173.【01】从零开始的卡通渲染-描边篇[EB/OL].(2021-01-11)[2025-03-02]. https://zhuanlan.zhihu.com/p/109101851
[16] EricHu33. Face Shadow Map – Creation & Baking Workflow[EB/OL]. (2022-01-15)[2025-05-02]. GitHub.
[17] Uchimura H., Suzuki K. Practical HDR and Wide Color Techniques in Gran Turismo SPORT[C]//SIGGRAPH Asia 2018 Courses. Tokyo: ACM SIGGRAPH Asia, 2018: Course Note, Practical HDR and WCG.
[18] 蛋白胨. DanbaidongRP Documents[EB/OL].(2025-01-26)[2025-03-02]. https://miusjun13qu.feishu.cn/docx/EXPtdrNmnox8hkx4mnCcy8QNn2b
[19] Lengyel J., Praun E., Finkelstein A., Hoppe H. Real-Time Fur Over Arbitrary Surfaces[C]//Proceedings of the 2001 Symposium on Interactive 3D Graphics. New York: ACM, 2001: 59–62.
[20] Isidoro J., Mitchell J. L. User Customizable Real-Time Fur[C]//ACM SIGGRAPH 2002 Conference Abstracts and Applications. New York: ACM, 2002: 273.
[21] Scheuermann T. Hair Rendering and Shading[C]//Proceedings of the 18th Annual Game Developers Conference. San Jose Convention Center, San Jose, California, USA: Game Developers Conference, 2004.

致 谢

首先，要感谢我的导师安维华老师，安维华老师是一位对教学和论文指导都非常认真负责的老师，并且给我提供了很大的自由度让我可以研究我感兴趣的内容。

其次，感谢 GAMES Webinar（计算机图形学与混合现实在线平台）和加州大学圣塔芭芭拉分校的闫令琪老师，在学习图形学和渲染的道路上，GAMES 平台上闫令琪老师的 GAMES101 和 GAMES202 系列对我受益匪浅，还要感谢 B 站 up 主@霜狼_may 的技术美术百人计划公开课对我在游戏渲染技术的学习上提供了非常大的帮助。

最后，感谢这四年来北京语言大学信息科学学院负责 2021 级数字媒体技术专业课授课的所有任课老师对我提供过的帮助，祝大家都能在自己热爱的学科领域中取得自己满意的成就。