Journal of System Simulation

Volume 29 | Issue 11 Article 14

6-5-2020

Lightweight Real-Time and Realistic WebGL Rendering Algorithm Based on PBR

Weixin Zheng
1. Changchun University of Science and Technology, Changchun 130000, China;;

Jinyuan Jia
2. School of Software Engineering, Tongji University, Shanghai 201804, China;

Lightweight Real-Time and Realistic WebGL Rendering Algorithm Based on PBR

Abstract

Abstract: Considering the weak rendering ability of web browser, the authenticity and aesthetics of Web3D rendering are always hot topics.In particular, the realization of PBR in the web-side and even dynamic editable are difficult.The realization and dynamic edit of PBR in the web end are especially difficult. Based on the problem mentioned, this paper presents *a web-based lightweight real-time rendering algorithm with authenticity and high quality* based on the PBR algorithm, which simplifies the complex calculation steps of the algorithm and reduces the computational complexity for rendering. *The LPM (Lightweight Progressive Mesh) algorithm* is reused to simplify the model file and accelerate the model loading speed. Finally, a demo is presented to verify the algorithm's effectiveness. The experiment results show that the optimized algorithm can render editable high quality lightweight 3D model on the web end in real-time.

Keywords

physics based rendering, lightweight progressive meshes, WebGL, real-time rendering

Recommended Citation

Zheng Weixin, Jia Jinyuan. Lightweight Real-Time and Realistic WebGL Rendering Algorithm Based on PBR[J]. Journal of System Simulation, 2017, 29(11): 2693-2700.

第 29 卷第 11 期 系统仿真学报© Vol. 29 No. 11
2017 年 11 月 Journal of System Simulation Nov., 2017

基于 PBR 的轻量级 WebGL 实时真实感渲染算法

郑维欣[1]，贾金原[2]
(1.长春理工大学，长春 130000; 2.同济大学软件学院，上海 201804)

摘要：考虑到网页浏览器的渲染能力较弱，对 Web3D 渲染效果的真实性与美观性一直都是研究热点，尤其是基于物理的材质 PBR (Physical based Rendering)在网页端的实现乃至动态可编辑都是难点。基于上述问题本文提出了*基于 PBR 算法的 Web 端轻量级真实感高品质实时渲染算法*，将该算法涉及到的计算量较大的计算步骤简化，降低计算机渲染时的计算复杂度；同时提出了*基于重用度的 LPM (Lightweight Progressive Mesh)算法*来简化模型文件、加快模型的加载速度。最后通过上述算法所编写的 demo 验证了算法的有效性。实验结果表明，该优化后的算法能够在 Web 端实现实时渲染可编辑的高品质轻量级 3D 模型。

关键词：PBR；LPM；WebGL；真实感绘制
中图分类号：TP.319.9 文献标识码：A 文章编号：1004-731X (2017) 11-2693-08
DOI: 10.16182/j.issn1004731x.joss.201711014

Lightweight Real-Time and Realistic WebGL Rendering Algorithm Based on PBR

Zheng Weixin[1], *Jia Jinyuan*[2]
(1.Changchun University of Science and Technology,City Changchun; 2.School of Software Engineering, Tongji University, City Shanghai)

Abstract: Considering the weak rendering ability of web browser, the authenticity and aesthetics of Web3D rendering are always hot topics.In particular, the realization of PBR in the web-side and even dynamic editable are difficult.The realization and dynamic edit of PBR in the web end are especially difficult. Based on the problem mentioned, this paper presents *a web-based lightweight real-time rendering algorithm with authenticity and high quality* based on the PBR algorithm, which simplifies the complex calculation steps of the algorithm and reduces the computational complexity for rendering. *The LPM (Lightweight Progressive Mesh) algorithm* is reused to simplify the model file and accelerate the model loading speed. Finally, a demo is presented to verify the algorithm's effectiveness. The experiment results show that the optimized algorithm can render editable high quality lightweight 3D model on the web end in real-time.

Keywords: physics based rendering; lightweight progressive meshes; WebGL; real-time rendering

引言

模型真实性问题广泛存在于 3D 模型渲染、3D 游戏中。现有的一些 3D 模型在渲染时所引入的材质虽然对模型的真实性进行了不同程度的提高，但模型的真实性却并未达到与自然界物体相差无几的程度，主要原因有如下两方面：

1. 真实感绘制的计算量大，渲染光照计算公式复杂，尤其是达到物理级真实的材质效果，渲染时间往往较长，即便离线渲染也难以达到实时，更不要说在线了；

收稿日期：2017-07-29 修回日期：2017-09-30;
基金项目：国家自然科学基金面上项目(61272276)，吉林省重点科技攻关课题(20140204088GX)，长春高新区"长白慧谷"英才计划(3-2013006);
作者简介：郑维欣(1993-)，男，吉林吉林，硕士生，研究方向为 Web3D 渲染。

2. 网页浏览器的计算能力薄弱，WebGL 的渲染引擎仅仅支持简单的 Phong 光照模型，尤其是为模型引入较为高品质的材质时，渲染效果并不理想、渲染速度也会有所减缓，而且当模型量较大以及渲染光照比较复杂时会出现严重卡顿的现象。

本文的目标：通过 WebGL 以及基于 WebGL 的 Threejs 库函数在 Web 端的场景中实现具有 PBR 材质的模型渲染；减少所需渲染的模型文件的实际大小；提高模型在 Web 端渲染效果的真实感；加快模型的渲染与加载速度；最终在 Web 端实现一个可以在线编辑的基于PBR材质的3D模型材质编辑器。

1 问题概述与研究现状

1.1 Web3D 模型渲染问题

通过分析，将 Web3D 模型渲染的问题分为两个方面：模型的真实性、模型加载与渲染的速度。

模型(本文主要研究在 Web 端实现 PBR 算法，此算法对于金属材质、玻璃材质以及橡胶材质的模型的渲染效果较为真实，所以此处的模型指车模型，以下均为车模型)的真实性是指在 Web 端渲染 3D 模型时，模型渲染后的效果能够与真实的自然界中所存在的车所表现出的外观效果一致或极为接近。目前影响渲染效果的主要因素为 Web 端较弱的渲染能力以及在 Web 端渲染模型时所涉及到的大量的计算过程，对于现在已有的 PBR 材质算法，其所涉及的计算公式较多、计算量较大、计算复杂度较高，本文旨在将已有的 PBR 材质算法进行优化，将模型的真实性提高的同时也要提高模型的加载与渲染速度。

1.2 研究现状

文献[1]阐述了 PBR 算法是一个基于实物的阴影系统但却不同于在其之前的阴影系统、遵守能量守恒定律，并证明了扩散、反射与能量守恒就是 PBR 算法的核心思想。

文献[2]提出了一款比较完善的 PBR 材质模型，该算法基于 Beckmann 理论。由此文献[2]所提出的模型公式为 DGF 的乘积除以 π 与 NV、NL 的向量积。但是这个 Torrance 模型只是一个表面扩散模型，并且该模型忽略了材质的内部结构。

现在的 PBR 算法是以支持 Cook-Torrance Microsoft 的 BRDF 理论为基础的，并在直接照明与间接照明的 Shading Model 里使用，比如文献[3]在基于 BRDF 理论的基础上，提出一种新的 BRDF 映射技术，该方法主要由 MERL BRDF 和 Fresnel 理论为基础，可以更好的表达出物体对于光的反射效果。

文献[4]对运算进行 GPU 加速处理，使得运算速度得到整体的提升。

文献[5]使用了渲染管线为 G-Buffer+TBDR，由于基于 Lambert 的 BRDF 使用起来比较简单，而且易于维护，所以可以达到更好的渲染效果。

对于上述文献，所提出的 PBR 算法的基本原理、以及对于该原理的使用及优化，并不能减少 Web 端在渲染高品质真实性较高的 3D 模型时所需的资源消耗。

基于此，本文通过使用基于 WebGL 的 Threejs 源生 API 以及编写 WebGL 的底层代码，在 Web 端实现车模型的 PBR 渲染效果，并将优化后的 Web3D-PBR 渲染算法作用于模型上；同时为了提高加载和渲染的速度，引入基于重用度的 LPM 轻量化算法，减少浏览器在加载与渲染 3D 模型时的负担，以达到流畅的加载与渲染效果。

2 模型预处理

本文采用的模型文件格式为 JSON 文件，对于原文件，一次性加载则会造成用户等待 Web 端加载的时长过长的现象，为了解决等待时长的问题，本文在文件初始化加载时预先采用 LPM 轻量化算法预处理 JSON 模型文件，分为两个方面：LM 模型轻量化预处理、PM 模型轻量化预处理。

2.1 基于 LM 的模型轻量化预处理

此处采用模型重用的思想，将重复使用的构件

DOI: 10.16182/j.issn1004731x.joss.201711014

进行简化删除。例如一辆车子有四个车轮，对这四个轮子删减为一个轮子以及 3 个索引 l_1、l_2、l_3。

对于简化后的模型的构件的顶点做顶点塌陷工作，例如将 4 个顶点中的一个进行塌陷，柔和成 3 个顶点，以此方式进行迭代，将顶点多次塌陷，简化为一个顶点数最少的模型和一个塌陷矩阵 A。

2.2 基于 PM 的模型轻量化预处理

对于 JSON 模型文件，采用 PM 的方法简化文件分为两个部分：基网格文件、增量网格文件。(LM、PM 是并列的运算过程，没有固定的优先过程)

基网格文件的要求：文件大小足够小但应能在渲染后展现出模型的大体外观效果，即该基础面文件在 Web 端可以展现出一个模糊的有模型的大体外观的渲染效果。

增量网格文件的要求：每个文件应足够细小，且包含所有的模型的细节部分。

对于每一个增量网格文件，会存储一个与之对应的过程信息 I，该过程信息可以用来在渲染时恢复原模型的具体外貌。

此处，对于已经预处理好的 JSON 文件进行两段渲染，首先加载只有基网格的 JSON 文件，会在 Web 页面渲染出一个具有模型的外形的效果，然后加载并渲染含有渐进网格文件的多个 JSON 文件，达到补充模型的细节的目的。最后会在减少加载时间的情况下，达到与渲染整个未处理的 JSON 文件的渲染效果相同的目的。

基于上述流程，对已有的模型文件进行文件简化处理，得到如下表格：

表 1 模型简化前后对比表
Tab. 1 Model simplification comparison table

复杂度	截图	原面片数	基网格	简化比%
低		200	30	15
低		1 492	110	14
低		1 938	192	10
高		3 098	850	25
高		4 096	200	20%

通过以上表格，可以看出经过简化后的 JSON 文件面片数较简化之前减少为原面片数的 10%~25%，可以有效地提升 Web 端 3D 模型的加载与渲染速度。

3 Web3D-PBR 材质算法

3.1 菲涅尔效应

在计算机图形学中，菲涅尔效应(后面简称 Fresnel)指的是不同的角度下不同的反射率。光线以一个入射余角照射到一个表面上时比光线直接照射到表面上的情况会更容易会发生发射。同时在临近这个入射余角的边界时会出现更亮的反射。对于 PBR 着色器会添加一小部分重要的修正到 Fresnel 计算公式中。

关于 Fresnel 属性的第二个现象是不同材质的反射率/角度曲线不会变化很大。其中金属有最大的不一致效果[6-7]。如图 1 所示。

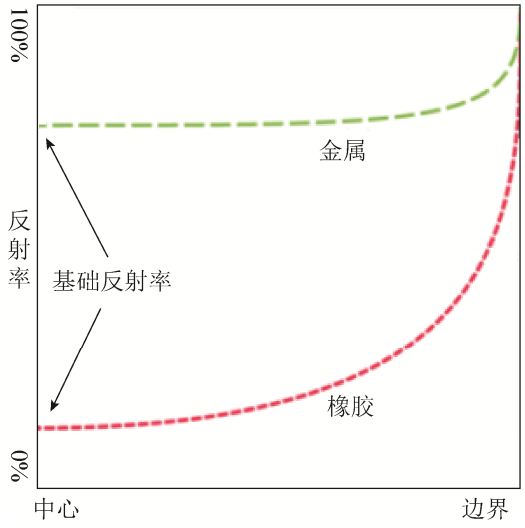


图 1 反射率与角度变化曲线
Fig. 1 Reflectance and angle change curve

菲涅耳反射率的值取决于两点：入射角度(光矢量和表面法线之间的角度)和材料的折射率。

因为表面不能反射小于 0%或超过 100%的入射光。在反射平面中活跃的微晶片 $m=h$，菲涅耳反射率的入射角实际上在 1 和 h 之间。

由此整理出图表，如图 2[8]：

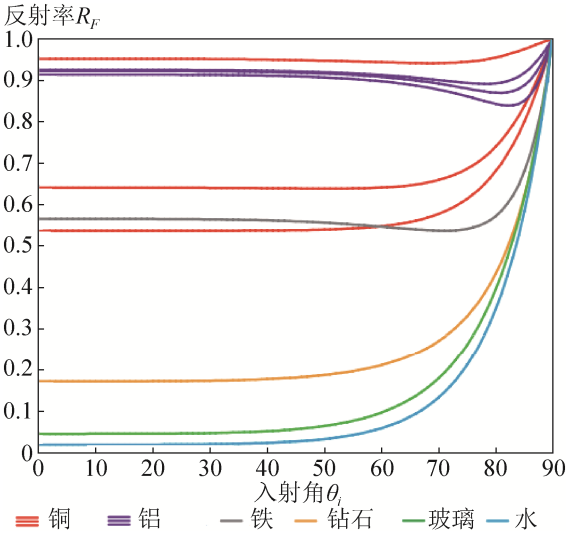


图 2 入射角与菲涅耳反射率曲线

Fig. 2 Incident angle and Fresnel reflectivity curve

可见在 0°~45°之间的入射角反射率几乎不变。在 45°左右的反射率会产生明显地变化(但并不总是有所增加)，增加到约 75°。最后，在 75°和 90°之间反射率总是快速变为 1。由于菲涅耳反射率在大多数范围内保持接近一个稳定的值，例如铁的菲涅耳反射率大多数情况下接近于 0.6，即 F_0=0.6，而水则是接近于 F_0=0，所以我们可以令这个值 F_0 作为材料的特征镜面反射率。

3.2 Shading Model

PBR 算法可以分为三个部分：Shading Model，Lighting Model，Material Model。由于 Shading Model 部分所涉及到的公式较多、计算较为复杂，所以本文主要是针对 PBR 材质算法中的第一部分 Shading Model 进行优化。

由于 PBR 中最大的特点是引入了 Microfacet (微平面)的概念，即着色平面不是一个完美的反射平面，故而有了粗糙度的概念[9]。如图 3 所示。

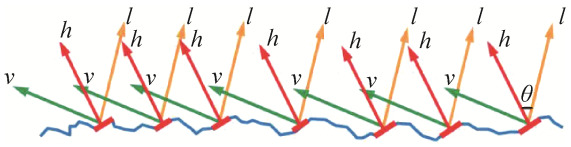


图 3 光在反射面的反射情况

Fig. 3 Reflection of light in the reflection surface

L：光源，v：视线，h：法向，θ：入射光源与 h 的夹角；

Microfacet 的 specular BRDF 可以表达成：

$$\rho(\vec{l},\vec{v})=\frac{F(\vec{l},\vec{h})G(\vec{l},\vec{v},\vec{h})D(\vec{h})}{4(\vec{n}\cdot\vec{l})(\vec{n}\cdot\vec{v})} \quad (1)$$

$F(\vec{l},\vec{h})$ 是有效 Microfacet 产生的 Fresnel 反射。

$G(\vec{l},\vec{v},\vec{h})$ 是有效 Microfacet 中没有被 shadow 或 mask 的比例。

$D(\vec{h})$ 是 Microfacet 的法线分布函数，即 Microfacet 的法线等于 h 的密度。

$4(\vec{n}\cdot\vec{l})(\vec{n}\cdot\vec{v})$ 是个校正因子，用来校正从 Microfacet 的局部空间转到整体表面的数量差异[10-12]。

对于公式(1)中的 F(反射率)有如下公式：

当出射光的能量小于入射光的能量时：

取入射光的位置信息 $P_1(x_1,y_1,z_1)$，法向与入射光的交点 $P_2(x_2,y_2,z_2)$：

$$\theta=arctan(\frac{\sqrt{(x_1-x_2)^2+(z_1-z_2)^2}}{\sqrt{(y_1-y_2)^2}})$$

$$F=f_0(1-f_0)2^{-9.60232(cos^8\theta)-8.58092cos\theta} \quad (2)$$

f_0 是正常镜面的反射率，对 F 的 $cos^8\theta$ 化简拆分，可得到结果为：

$$C_2=\frac{35+\cos(8\theta)+28\cos(4\theta)}{128}$$

$$C_2=\frac{48\cos(4\theta)+16\cos(4\theta)\cos(2\theta)}{128}$$

$$COS^8\theta=C_1+C_2 \quad (3)$$

由公式(3)的计算方法可见将 8 次方化简为 1 次方与二次方的加法运算，减少计算机的运算时间。

4 基于 WebGL 的 PBR 算法流程

4.1 PBR 算法运算流程

(1) 对于 Shading Model 中的 Diffuse BRDF 和 Specular BRDF 进行简化，方法见 3.2 的公式(2)、(3)；

(2) 通过公式

https://dc-china-simulation.researchcommons.org/journal/vol29/iss11/14
DOI: 10.16182/j.issn1004731x.joss.201711014

$$D(h)=\frac{\alpha^2}{\pi((n\cdot h)^2(\alpha^2-1)+1)^2}\quad(\alpha=Roughness^2)\quad(4)$$

(此处的 α 为粗糙度的平方)判断出射光能量是否小于入射光能量；

(3) 若步骤(2)判断结果为是，则将公式带入 3.2 的公式(1)；

(4) 对于 Lighting Model 中的公式

$$\int_H L_i(l)f(l,v)\cos\theta_l dl\approx\frac{1}{N}\sum_{k=1}^{N}\frac{L_i(l_k)f(l_k,v)\cos\theta_{l_k}}{p(l_k,v)}\quad(5)$$

$$F_{schlick}=(1.0-F_0)(1-\cos\theta)^5+F_0\quad(6)$$

公式(5)中 v 是视点方向，Li 是每一个入射光。公式(6)中 F_0 是垂直入射时的反射率所得出的计算结果 pre-fileter 与 cubemap 相乘，即可得到 Lighting Model 的最终结果

$$M_1=F_0\int_H\frac{f(l,v)}{F(v,h)}(1-(l-v-h)^5)\cos\theta_l dl$$

$$M_2=\int_H\frac{f(l,v)}{F(v,h)}(1-vh)^5\cos\theta_l dl$$

$$\int_H f(l,v)\cos\theta_l dl=M_1+M_2\quad(7)$$

4.2 基于 LPM 的渐进式加载算法

(1) 对于 PM 处理过的模型文件，优先加载基网格文件；

(2) 将 PM 的增量网格文件乘以 $\boldsymbol{\Gamma^{-1}}$，逐步加载所有的增量网格文件。

(3) 对于 LM 处理好的模型文件进行检测，判断文件是否可以进行恢复工作，如果可以，则将文件的顶点信息乘以塌陷矩阵的逆矩阵 $\boldsymbol{A^{-1}}$，并进行多次反向迭代；如果不可以，则检测下一个文件；

(4) 通过文件所对应的索引找出文件的重复次数 N，将文件通过 l_1、l_2.....l_N 这 N 个索引进行构件的复制与位置的移动。

4.3 基于 PBR 算法的车模型渲染算法

对于上述算法进行整合，得到整体运算流程图，如图 4。

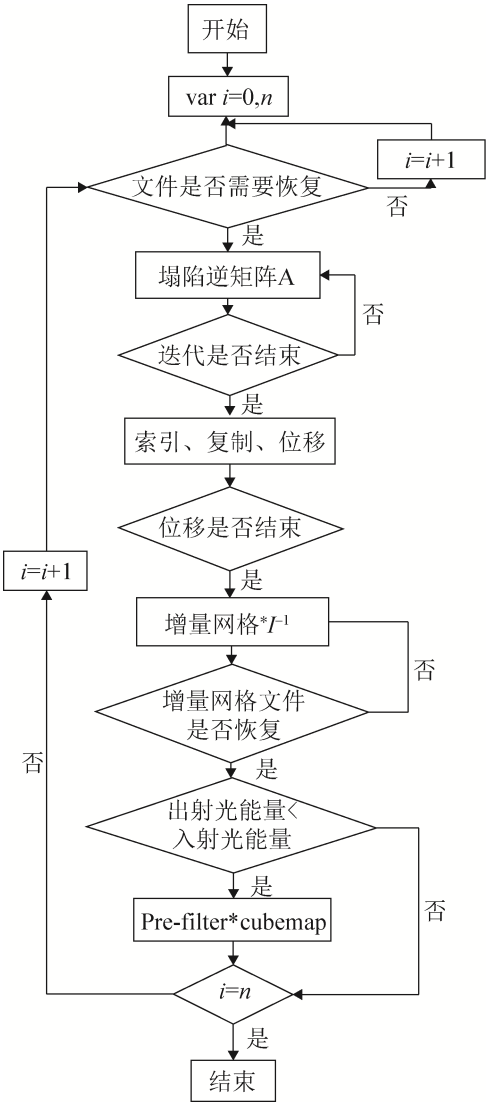


图 4 渲染流程图
Fig. 4 Render the flow chart

5 算法结果与分析

为验证本文算法的有效性，作者应用 JetBrains WebStorm 11.0.1 开发平台，使用 WebGL 和 Threejs 编写了一个在 Web 端展示的汽车模型的例子。实验计算所使用的计算机配置为：Lenovo、操作系统 Windows 7 旗舰版 64bit、CPU 主频双核 3.60GHz、内存 8G。

5.1 实验一

对于已有算法进行性能测试，实验结果如表 2：

表 2 PBR 模型的性能对比表
Tab. 2 Performance Comparison of PBR Model

性能测试	加载时间/s	内存消耗/Mb	FPS	真实性
无 PBR 车模型	1.68s	260.41 Mb	43	低
有 PBR 车模型	1.63s	210.53 Mb	43	高

图 5 汽车模型对比
Fig. 5 Comparison of car models

图 5 左侧为引入 PBR 材质算法的汽车模型在 Web 端展现的效果截图、右侧为未引入 PBR 材质算法的汽车模型在 Web 端展现的效果截图。由上述两图可以看出，在引入了 PBR 材质算法后的汽车模型，能够较好的展示出金属在经过光源照射后所达到的反射光源的效果，同时在车身上能够映衬出环境的效果；并且在加载时间上并没有相差甚多；对于汽车模型的真实性，左侧比右侧的汽车模型更加真实；同时也能够较好地减少内存的消耗。

5.2 实验二

对于现有的渲染算法进行渲染效果的对比测试实验，实验结果如图 6、图 7：

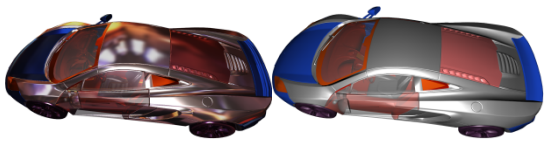


图 6 汽车模型对比
Fig. 6 Comparison of car models

图 7 汽车模型对比
Fig. 7 Comparison of car models

图 6、图 7 分别为未引入 PBR 算法的汽车模型在 Web 端的不同角度的展示效果、引入 PBR 算法的汽车模型在 Web 端的不同角度的展示效果。

由图 6、图 7 可以看出在不同的入射光角度下，光源会对车身的不同位置产生不同的反射效果，且该反射效果会根据车模型的不同构件的属性有所不同。例如图 6 和图 7 的入射光位置不同，所以车身的反射效果也有所不同。

5.3 实验三

通过交互面板，改变在 Web 端渲染的车模型构件的属性值，进而改变车模型的整体渲染外观，实验结果如图 8。

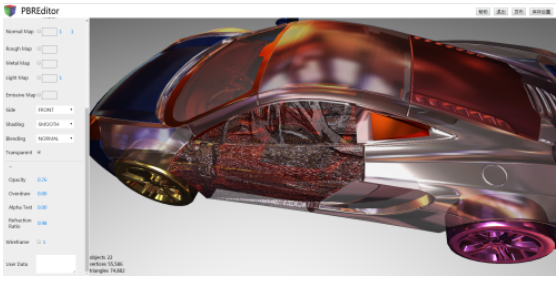


图 8 调节参数后的汽车模型
Fig. 8 Adjust the parameters of the car after the model

通过图 8 左侧的控制面板可以随意更改汽车模型中的任何构件的构建属性，并通过个人喜好获得最终的汽车模型展示效果。例如可以为左侧车门添加贴图、改变车轮颜色，改变车身反光度以及金属度等。

5.4 实验四

对于不同材质的构件进行 PBR 效果测试，测试结果如图 9。

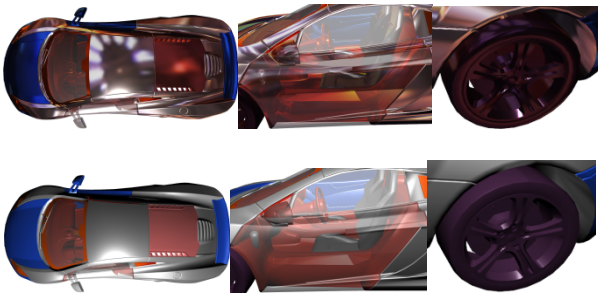


图 9 不同材质对比图
Fig. 9 Comparison of different materials

https://dc-china-simulation.researchcommons.org/journal/vol29/iss11/14
DOI: 10.16182/j.issn1004731x.joss.201711014

车模型主要是由金属、玻璃和橡胶三种属性的构件构成，所以此处选择最有代表性的三个构件--汽车模型的车身、汽车模型的左侧玻璃车门以及汽车模型的轮胎。图 9 上半部为有 PBR 材质效果的车构件，下半部为无 PBR 材质效果的车构件，对于这三种材质的构件，得出测试表格，如表 3：

表 3 不同材质构件的属性对比表

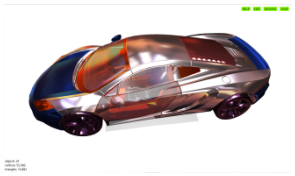
Tab. 3 Comparison of attributes of different material components

类型	金属度	粗糙度	透明度	反光能力
金属车身	高	低	低	强
挡风玻璃	低	低	高	较弱
轮胎	低	高	低	弱

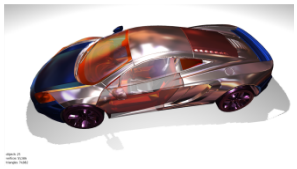
由于构件属性的不同，所以对于构件本身的反光能力也会有所不同，最终会导致构件与构件之间的渲染效果产生差异，即不同类型的车构件产生的不同渲染效果。

5.5 实验五

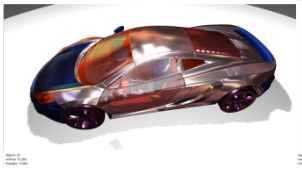
在场景中添加不同的光源，测试车模型在不同光源以及多光源下的渲染效果以及阴影效果，如图 10 所示。



(a) 添加方向光渲染效果图



(b) 添加点光源渲染效果图



(c) 添加聚光灯渲染效果图

(d) 添加三种光渲染效果图

图 10 添加光源渲染效果

Fig. 10 Rendering effect with light

对于本文的最终展示效果，不仅提供了图片作为参考，同时可以访问如下两个网页链接：smart3d.tongji.edu.cn/LPM/volvo;smart3d.tongji.edu.cn/PBREditor，直接查看渲染效果(无需任何插件，即可浏览，建议使用 Chrome、Firefox 等浏览器浏览)，在访问第二个链接时也可以通过编辑菜单对车模型进行基础信息编辑、材质编辑、光源编辑、标签设置等操作。

6 结论

本文主要是针对金属、玻璃以及橡胶这三种材质属性的物体，所以采用车模型。由文中图片可以看出，采用 PBR 材质的车模型能够很好地反映出汽车模型应有的真实渲染效果，而且对于金属、玻璃以及橡胶这三种材质属性的物体的渲染效果也会有非常明显的差异，来较为明显且真实的表达该构件的本质属性，最终得到较为真实的 Web 端车模型渲染效果。

但是对于 PBR 算法的适用性而言，并没有对所有的材质属性进行测试，所以可能会产生对于某种特殊属性的物体不适用的现象，例如混合属性的物体(水泥--由液体水和固体石灰混合而成的胶装的泥)。所以接下来的工作内容主要是针对这些文中并未进行测试的其他材质属性物体、尤其是混合材质属性的物体的渲染。

参考文献:

[1] Jeff Rusell. Basic Theory of Physically-Based Rendering[M]. (2015-11-01) [2017-09-20]. https://www.marmoset.co/posts/basic-theory-of-physically-based-rendering/.

[2] Mathieu Robart, Mathias Paulin, Rene Caubet. Material Model for Physically-based rendering[M].Part of the EUROPTO Conference on Polarization and Color Techniques in Industrial Inspection, 1999, 3826: 10-20.

[3] Jannik Boll Nielsn, Henrik Wann Jensen, Ravi Ramamoorthi. On Optimal, Minimal BRDF Sampling for Reflectance Acquisition[J]. ACM Transactions on Graphics (S0730-0301), 2015, 34(6): 1-10.

[4] Soham Uday Mehta, Kihwan Kim, Dawid Pajak. Filtering Environment Illumination for Interactive Physically-Based Rendering in Mixed Reality[M]. (2015-11) [2017-09-20]. http://dx.doi.org/10.2312/sre. 20151172.

（下转第 2708 页）