

订阅DeepL Pro以编辑此演示文稿。  
访问[www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document)，了解更多信息。

**冯氏镶嵌法**

Tamy BoubekeurMarc Alexa柏林大学

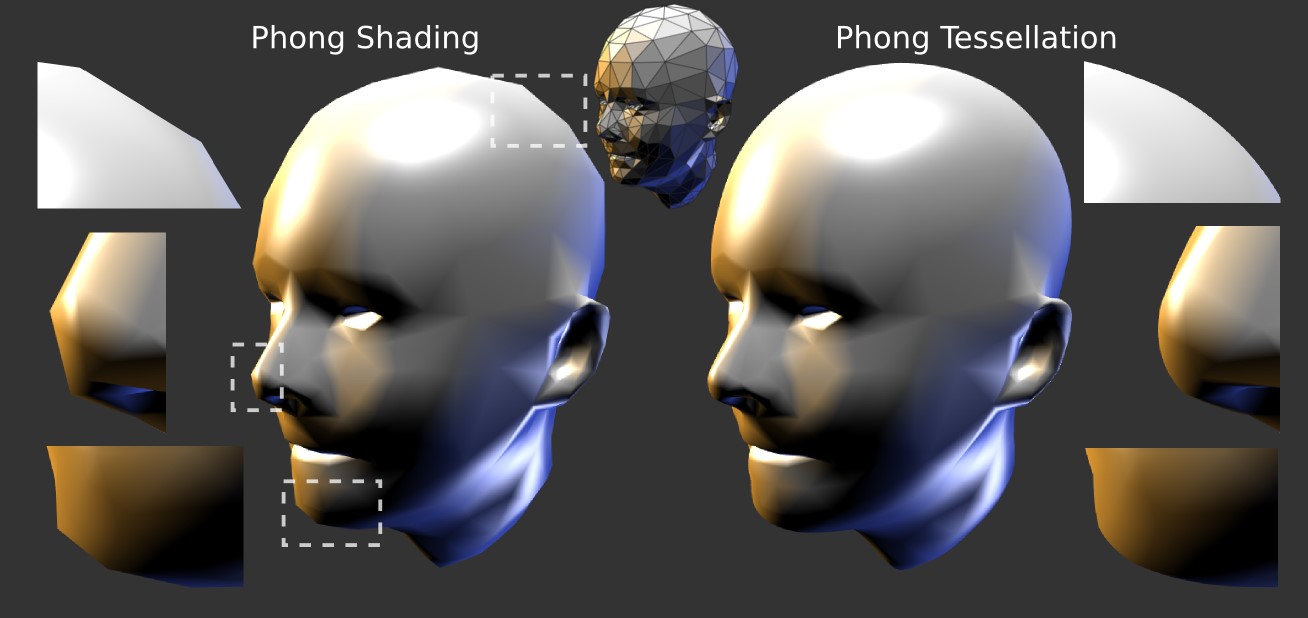


图1：*Phong Tessellation完成了Phong Shading。*

# 摘要

实时应用中使用的现代3D引擎提供的着色，使用调制法线、纹理和色调映射来掩盖形状内部高阶连续性的缺乏--如果表面几何形状不光滑，伪影只留在内部轮廓和轮廓上。本文的基本思想是应用一个纯粹的局部细化策略，使几何体膨胀到足以避免这些伪影。我们的技术是Phong法线插值的几何版本，不是应用于法线，而是应用于顶点位置。我们把这种策略称为Phong Tessellation。

CR类别：I.3.3 [计算机图形]：线条和曲线生成；I.3.3 [图片/图像生成]：显示算法；I.3.1 [硬件结构]：图形处理器；

关键词：实时拼合、网格细化、视觉连续性

# 简介

交互式渲染逐渐采用了由离线渲染技术激发的效果：HDR色调映射、柔和的阴影、环境遮蔽、色彩渗入、运动模糊等，现在都可以自由选择。

在交互式三维引擎中经常出现。由于高质量的光滑表面，如NURBS和细分表面，如今已成为离线渲染的标准几何表现形式，因此在实时应用中建立它们似乎也是很自然的。然而，视频游戏和其他交互式应用的要求通常完全是视觉上的，而 "视觉平滑度 "的很大一部分已经用Phong Shading和法线贴图生成。因此，剩下要解决的问题是改善那些不能基于阴影产生视觉平滑度的几何图形：轮廓和剪影。

以图1和图2为例，对人脸使用高质量的光滑表面表示法不会对阴影图像的内部产生重大影响，因为纹理、Phong法线插值和照明已经提供了一个真实的视觉复杂性。然而，剪影则受到底层多边形表示的影响。在试图改善沿轮廓线的视觉伪影时，有两个相互竞争的目标：

* 沿着轮廓线产生的几何图形是平滑的，这样就可以避免视觉伪影，而
* 用尽可能少的操作创建这个几何体，因为轮廓周围的区域只占图像的一小部分，因此，整体的视觉印象，其生成需要大部分的CPU和GPU周期。

我们本着这种精神设计了*Phong Tessellation*，它几乎不比单独的Phong Shading贵：它基于使用重心插值和正交投影，只需要三角形在GPU上自然携带的信息，即其顶点位置和顶点法线。我们表明，它的效率几乎与标准的线性（平面）三角形镶嵌一样高，同时在大多数情况下提供了明显改善的轮廓线。如图1所示，它自然地补充了Phong Shading。



图2：*如这些来自游戏《毁灭战士3》（id Software）的截图所示，法线贴图和Phong阴影为物体内部提供了合理的阴影，但沿轮廓线的分片线性形状清晰可见。*

# 以前的工作

实时镶嵌最近在计算机图形学中受到了广泛的关注。原因是，应用程序通常为动画、交互、物理模拟而保持3D模型的粗略近似，而GPU将有能力光栅化大量的基元。因为CPU和GPU之间的总线仍然是一个限制，所以将粗略的模型发送到GPU，在投影和扫描转换之前生成额外的顶点和多边形是有意义的。

一些技术被开发出来用于在GPU上生成光滑的表面（NURBS[Guthe等人，2005]或细分表面[Shiue等人，2005]），然而，它们被认为对于整合到实时引擎来说太慢。一般来说，在GPU上生成依赖于维护网格拓扑结构的光滑表面需要多次渲染，而且内存和计算量很大，这与实时3D引擎的要求不相符合。

一个成功的解决方案来自于Gouraud[Gouraud 1971]和Phong[Phong 1975]的开创性想法：*视觉平滑性*，即意识到在大多数情况下，只要表面看起来是平滑的，那么确切的*几何平滑性*就不是关键，因为这是着色技术的结果。在着色计算中插值pervertex法向量可以实现这种视觉平滑性，仍然可以避免对拓扑邻域的了解，并独立处理多边形。Valchos等人[Vlachos等人，2001]在这一原则的基础上：弯曲的PN三角形是一个纯粹的局部方案，根据´三角形的三个位置和三个法线构造一个立方贝塞尔贴片。继Van Overveld和Wyvill[van Overveld和Wyvill 1997]之后，构建了一个四维补丁来评估连续法线场与*C*0  边缘的连续性相遇，并产生一个连续的阴影。Boubekeur和Schlick[Boubekeur和Schlick 2007]使用细分方案[Loop 1987; Zorin等人1996]，引导局部二次元近似，导致视觉上比PN三角形更平滑的表面，同时保持高帧率。最近，Loop和Schaefer[Loop and Schaefer 2008]使用低度四边形补丁，同样具有独立的法线场，来逼近CatmullClark表面。一个 "视觉平滑 "的几何升采样的想法可以应用在输入网格的任何地方，也可以只应用在特定的位置，如轮廓。例如，Dyken*等人*[Dyken等人，2008]只在剪影上使用PN三角形。

所有这些技术以及屏幕空间方法[Max 1989]都使用相同的基本原理：每个输入的多边形都被一个多项式补丁取代。我们建议的Phong细分化操作避免了明确的修补，因此更简单、更高效，同时在大多数情况下仍然具有视觉说服力。

# 冯氏镶嵌法

考虑一个三角形t，索引三个顶点{v*i* ,v*j* ,v*k* }，其中v =

{p*,*n}，p∈IR3  的位置，n∈IR3  的法向量。一般来说，t的镶嵌产生一组新的顶点，位于t上定义的表面上。线性镶嵌只是在t定义的平面上产生顶点。线性镶嵌中产生的每个顶点p(*u,v*)对应于一个arycentric坐标(*u,v,w*)*，u,v∈*[0*,*1]*，w*=1*-*u-v，定义为

p(*u,v)* = (*u,v,w*)(p*i* ,p*j* ,p )*k*T  （1）

冯氏法线插值是使用同样的过程，只是在最后将结果归一化：

 (2)

显然，Phong法线是成功的，因为它们尽可能地简单。它们有许多众所周知的缺陷（例如，它们不能提供拐点），但它们仍然是首选，因为与高阶变体相比，它们产生可见伪影的少数情况被它们的计算简单性远远超过了。我们认为，实时网格细化算子应该像Phong法线插值一样高效和简单，以便在交互式应用中具有同样的吸引力。我们在下文中激励和描述的程序似乎是最简单的，仍然可以提供弯曲的几何图形。

注意，在每个顶点周围，由顶点法线定义的切平面是适当的局部几何。因此，从概念上讲，我们将三角形t投影到顶点v*i*  的切平面上，并在切平面内进行arycentric插值来定义v*i* 附近的几何体。相对于v*j*  和v*k*  的几何体也是类似的定义。然后，在整个三角形t上定义的几何体是由投影三角形内的三个arycentric插值产生的。由于投影与arycentric插值互换，点的计算可以简化为以下程序：

1. 计算线性嵌片，那么
2. 将得到的点正交地投射到由三角形顶点定义的三个切平面上，最后
3. 计算这三个投影的arycentric插值。

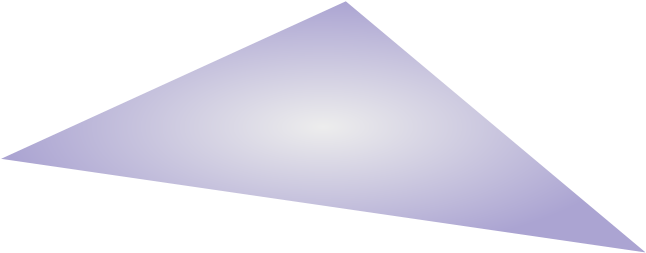


图3：*Phong Tessellation原理。我们不是像Phong Shading那样插值法线，而是插值投影到顶点的切平面上，为每个三角形定义一个曲线几何。*

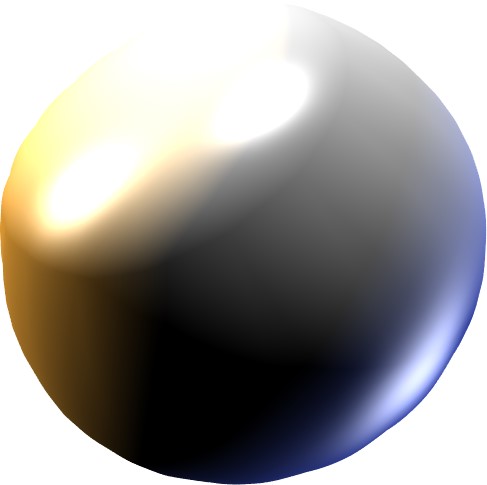
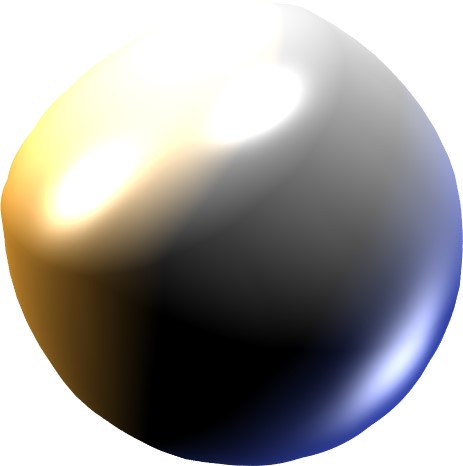
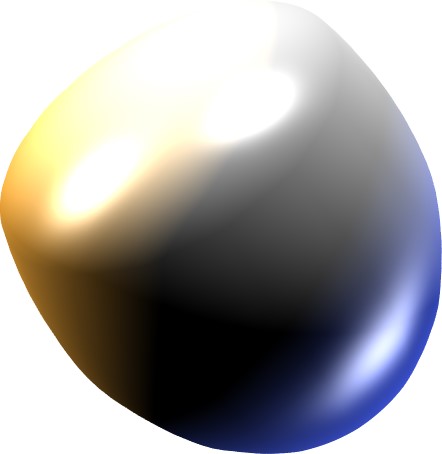
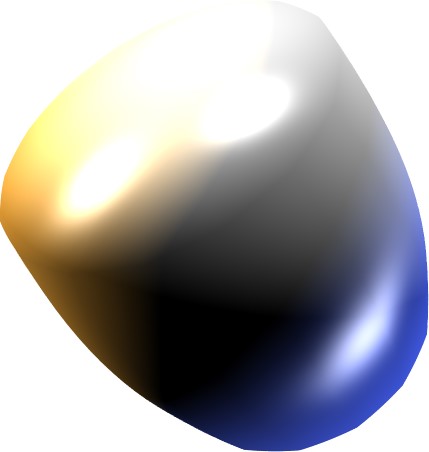
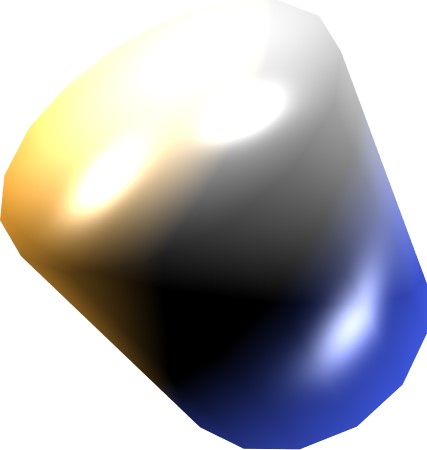
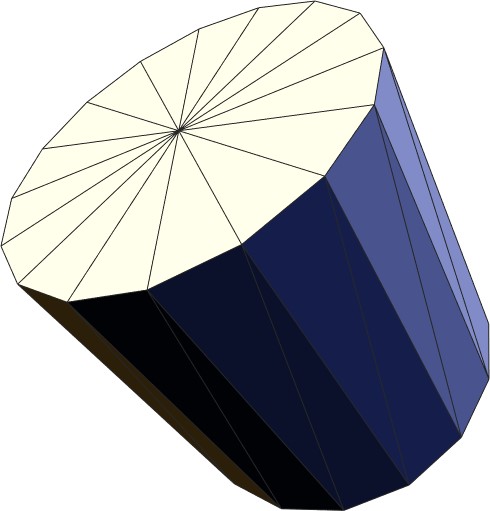
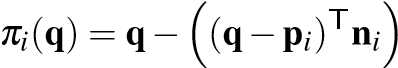
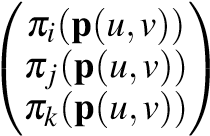


图4：*粗略的网格，然后是α等于*0*、*1/4*、*1/2*、*3/4*和*1的*各种Phong Tessellations。*

让

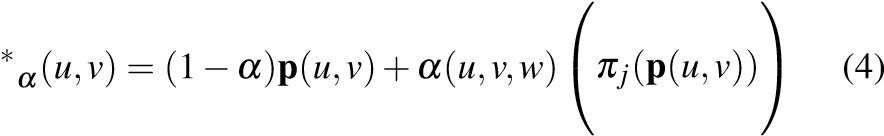
n*i*

是q的正交投影到由p*i*  和n*i* 定义的平面上，那么Phong Tessellation被定义为：

p∗ *(u,v*) = (*u,v,w*) (3)

与线性细分法（公式1）相比，该运算符增加了三个正交投影和一个额外的重心插值。

形状因子*α*可以用来在线性（平坦）和Phong Tessellation之间插值，控制在一个三角形上产生的曲率大小。事实上，我们在(*u,v*)处使用下面的定义来插入顶点，用于Phong tessellation：

*πi* (p(*u,v)*)

p *πk* (p(*u,v*))

图4显示了这个形状参数对最终几何形状的影响。在我们的实验中，我们将*α*=3/4全局固定下来，因为这个值在大多数情况下提供了令人信服的结果。然而，*α*也可以在每个顶点的基础上进行设置，并在三角形上进行插值。

# 财产

Phong Tessellation的重要计算特性是：(1)只使用与GPU上的三角形相关的信息；(2)评估没有副作用：请注意，我们所知道的在GPU上评估补丁的所有其他技术[Vlachos等人，2001年；Boubekeur和Schlick，2007年；Loop和Schaefer，2008年]都需要两个渲染过程（一个是通过将必要的补丁信息写入GPU的内存来设置补丁，另一个是将补丁信息写入GPU）。2001年；Boubekeur和Schlick 2007年；Loop和Schaefer 2008年]需要两次渲染过程（一次是通过将必要的补丁信息写入GPU的内存来设置补丁，另一次是评估这些信息），或者为每个细化的顶点进行完整的补丁构建。我们的构造可以在单程中进行评估，因为顶点位置和法线是三角形信息的一部分。

我们的定义所产生的几何体是一个二次修补，通过写出定义可以立即看到：

p∗ vw 

二次元补丁直接显示出，只要顶点法线彼此不同，曲面就会是弯曲的（见图5）。对于相同的法线，表面与平坦的三角形相同，类似于Phong法线，对于恒定的输入是恒定的。请注意，一个四阶补丁不能提供拐点。我们觉得在实际应用中不需要高阶补丁，这也类似于高阶法线插值在实践中被广泛认为没有必要。

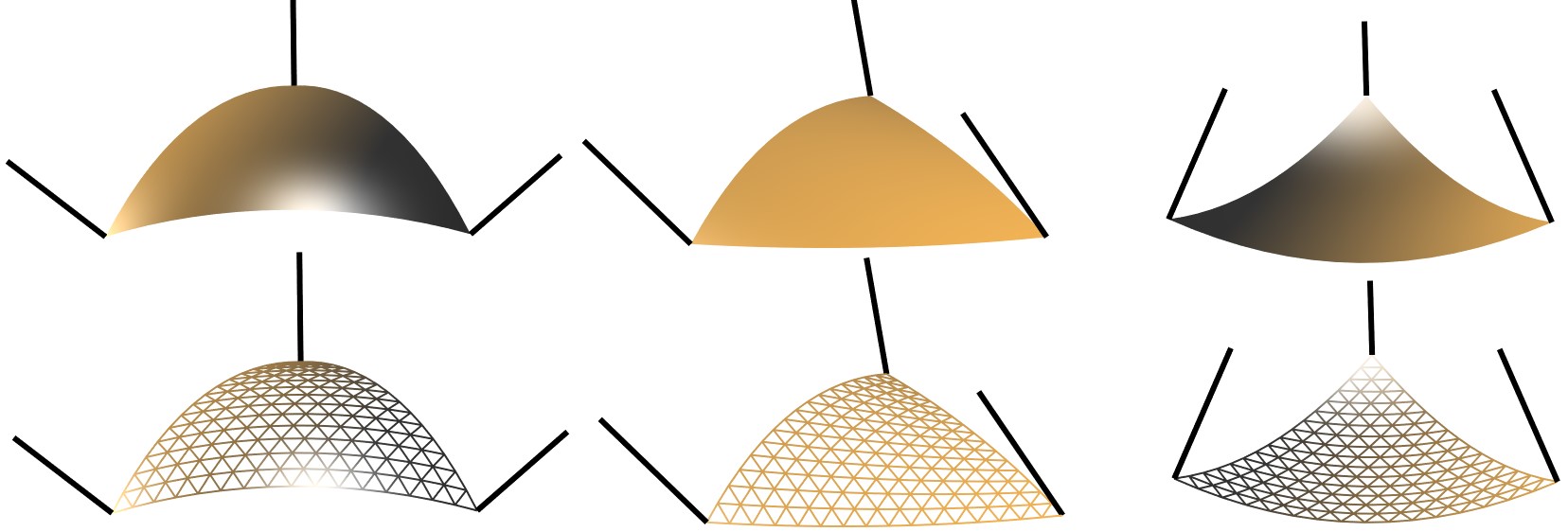


图5：*凸面、鞍面和凹面法线配置。*

虽然斑块是弯曲的，但二次斑块的集合形成了一个*C*0  连续的表面：沿着每条边，arycentric插值导致只使用入射在边上的顶点的位置和法线，而且这些信息对于两个相邻的三角形是相同的。跨越边缘或顶点的切线是不连续的。然而，将Phong Tessellation与Phong法线插值结合起来进行着色的结果是*视觉上的平滑性*，Phong法线在网格边缘提供平滑变化的阴影，Phong顶点形成弯曲的剪影和轮廓线（见图6）。

Phong Tessellation对输入的网格顶点进行插值，不表现出近似细化方案（如细分）的收缩效应。这允许重用实时应用中存在的所有现有网格，而不需要重新设计它们，使其极限面达到所需的形状。与其他插值网格顶点的技术如PN三角形或修正的Butterfly细分相比，视觉效果同样令人信服（见图9），同时效率明显提高。当然，Phong tessellation，如弯曲的PN三角形，可能会产生有明显切线不连续的轮廓线，而细分面总是会提供至少*C*1  的轮廓曲线。

由Phong细分法定义的曲面几何的质量取决于提供顶点的法线。在我们的测试中，我们使用经典的入射面法线的角度加权组合。我们观察到使用其他权重也有类似的好结果，但略有不同。还要注意的是，像尖锐边缘这样的奇异现象可以用标量标签来控制[Boubekeur和Schlick 2005b]。

# 业绩

镶嵌法中单个顶点的生成仅代表几行着色器代码。请注意，计算是基于arycentric插值和三个正交投影的，而直接评估二次元形式则需要更多的操作或两个渲染过程。我们观察到我们的变体平均快了40%。

由于镶嵌器单元仍在开发中（已宣布用于DX 11），我们使用几何着色器单元（在低镶嵌率下）或按照Boubekeur和Schlick[Boubekeur和Schlick 2005a]的精神使用通用的GPU细化内核对其进行仿真。在

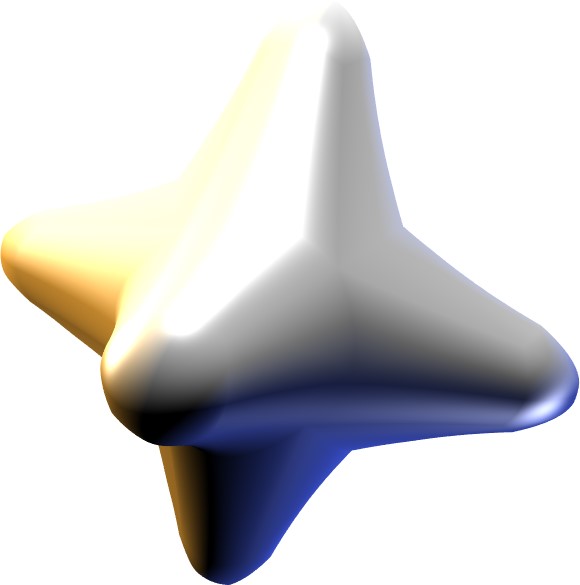
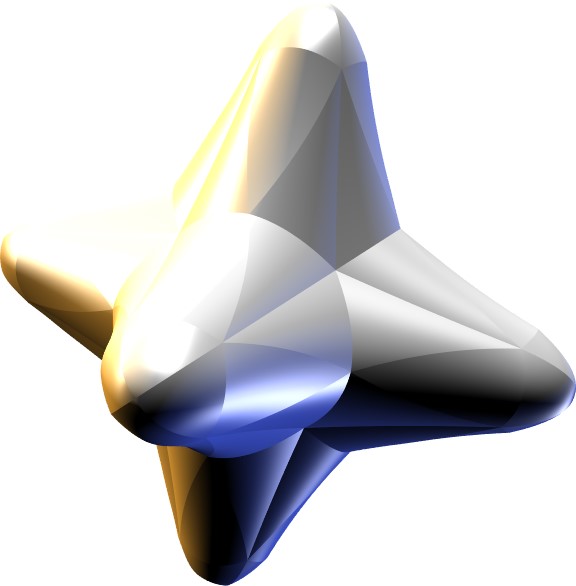
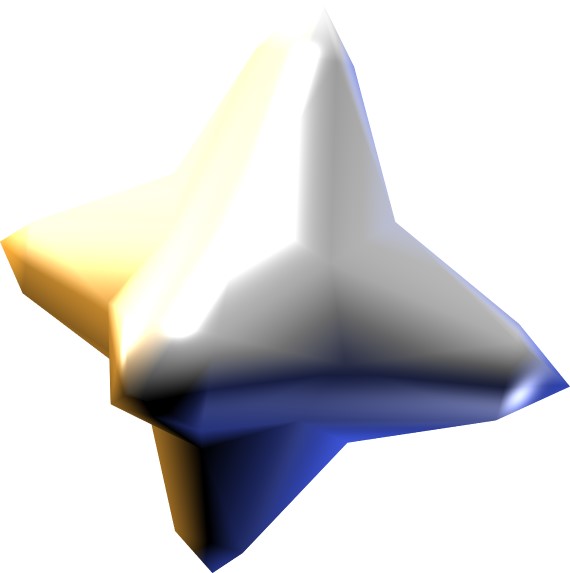
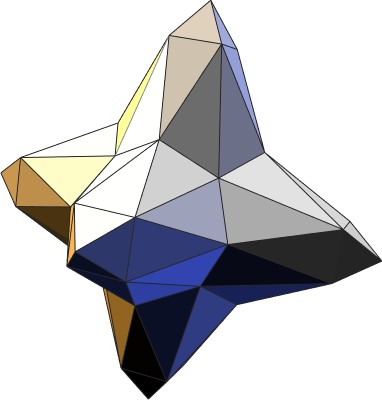
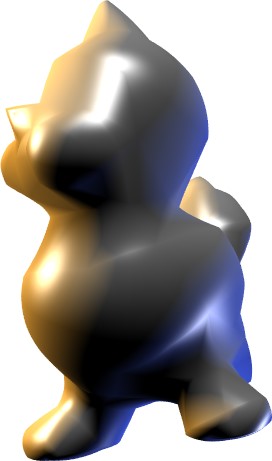
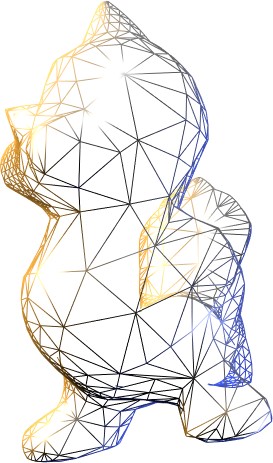
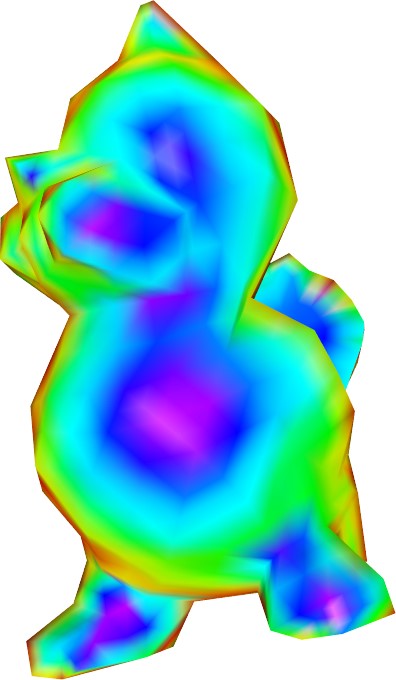
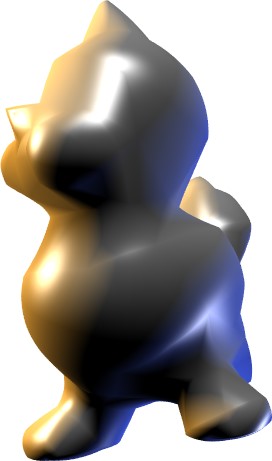
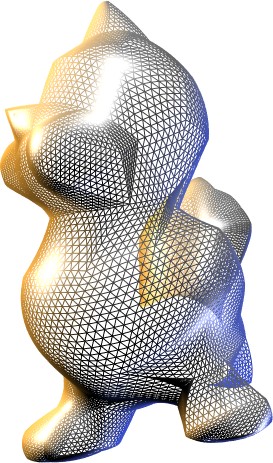
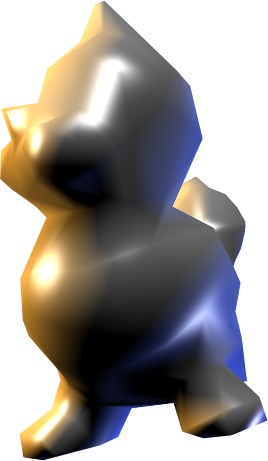
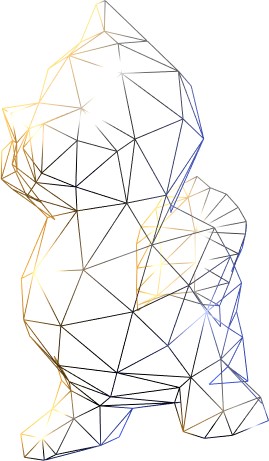


图6：*Phong的渲染。从左到右：网格、标准的Phong Shading、带有分析法线的Phong Tessellation和带有Phong法线的Phong Tessellation。*



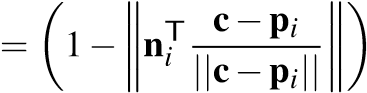
(a) 输入 (b) 统一的 (c) 适应性的视线依赖

图7：*自适应视图依赖的Phong Tessellation，只对剪影和轮廓进行实时几何升采样。*

在每种情况下，我们通过为每个顶点提供一个额外的*深度*值来执行自适应网格细化[Boubekeur and Schlick 2008]，该*深度*值对应于其附近所需的细分比例。

如图7所示，Phong细分的主要影响出现在轮廓和剪影上，因此我们在这些位置进行了自适应细化，考虑了顶点法线和视图矢量之间的角度，并依靠Phong阴影来产生形状内部必要的视觉平滑度。

在我们的实验中，我们使用一个简单的措施来计算围绕轮廓平滑增长的细化深度：

*di*  *m*

c是相机的位置，*m*是最大细化深度。这可以捕捉到剪影和内部轮廓，用几何体合成来处理它们，使我们的解决方案比剪裁技术[Sander等人，2000]更容易、更有效。为了避免强烈的弹出假象，可以通过将细化深度*di*  映射到*α*上，在粗略的多边形和细化的多边形之间进行逐步过渡。

图8显示了Phong Tessellation与线性镶嵌（在三角形上产生平面几何图形）相比的帧率。我们在GeForce 8800 GTX上测量了实际渲染条件下的帧速率，768 Mb。我们观察到，在几何处理工作负载中使用Phong Tessellation时，开销约为10%（禁用着色）。然而，在现实的渲染环境中--启用阴影、1600x1200的全图像合成、3个光源和几何体的动态变形（图8中的黄色和绿色条）--这一成本在比例上可以忽略不计。本文介绍的不同图片说明了视觉上的改进。

380

540

882

1246

1994

2720

0

200

400

600

800

1000

1200

1400

1600

1800

2000

线性

冯

线性+阴影

冯+阴影

每秒帧数

输入三角形的数量

图8：*线性（平坦）和Phong细分法之间的帧速率比较，有阴影和无阴影（最大深度64x64）。*

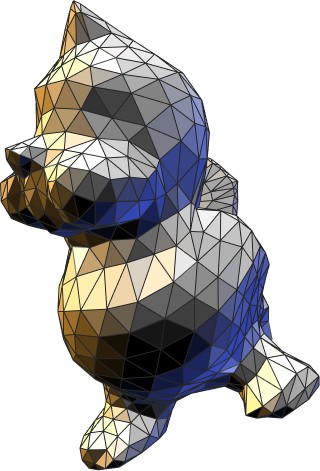


图9：*比较：修正的蝴蝶细分（左中），PN三角形（右中）和Phong细分（右）。*

# 总结

法线贴图和目前所有的着色技术已经提供了光的反射，看起来是来自于形状内部的弯曲几何体--但在轮廓线和轮廓线上仍然表现为片状的线性几何。我们认为，在许多实时场景中，在CPU上实际生成光滑的表面或在GPU上生成近似的表面是浪费的。相反，*Phong Tessellation*的简单程序，只需要基于顶点位置和法线的正交投影和线性插值，就可以用来生成弯曲的轮廓线。结合Phong Shading生成连续变化的阴影，在大多数情况下，它可以提供足够的视觉平滑度，以获得令人信服的渲染效果--而且是以最小的额外成本。

Phong tessellation可以在渲染管道中透明地实现，也就是说，用户不需要调整他们的渲染引擎来使用它。它可以自动处理任何基于网格的几何体，包括动态更新顶点位置或法线的情况。它可以很容易地被用于即将到来的镶嵌器单元中。

# 参考文献

Boubekeur, T., and Schlick, C. 2005.GPU上的通用网格细化。In *Proceedings of ACM SIGGRAPH/Eurographics Graphics Hardware*, 99-104.

Boubekeur, T., and Schlick, C. 2005.Scalar tagged PN triangles.In *Proceedings of Eurographics 2005 (short papers)*, 17-20.

Boubekeur, T., and Schlick, C. 2007.QAS：细分曲面的实时二次逼近。在*2007年太平洋图形会议上*，453-456。

Boubekeur, T., and Schlick, C. 2008.用于GPU上自适应网格细化的灵活内核.*Computer Graphics Forum 27*, 1, 102-114.

Dyken, C., Reimers, M., and Seland, J. 2008.使用自适应混合bezier补丁的实时GPU剪影细化。*Comp.Graph.Forum 27*, 1, 1-12.

GOURUD, H. 1971.弯曲表面的连续遮蔽。*IEEE Transactions on Computers 20*, 6, 623-628.

Guthe, M., Balzs, ., and Klein, R. 2005.基于GPU的Nurbs和T-spline曲面的修剪和细分。In *Proceedigns of ACM SIGGRAPH*, 1016-1023.

loop, C., and Schaefer, S. 2008.用二次元补丁逼近catmull-clark细分曲面。*ACM Transaction on Graphics 27*, 1, 1-8.

LOOP, C. 1987.*基于三角形的平滑细分曲面*。硕士学位论文，犹他大学。

MAX, N. 1989.多边形表面的平滑外观。*The Visual Computer 5*, 3 (Mai), 160-173.

PHONG, B.T. 1975.计算机生成的图片的照明。*Comm.ACM 18*, 6 (June), 311-317.

Sander, P., Gu, X., Gortler, S., Hoppe, H., and Snyder, J. 2000。Silhouette clipping.在*ACM SIGGRAPH的会议上*，327-334。

Shiue, L.-J., Jones, I., and Peters, J. 2005.A realtime GPU subdivision kernel.In *Proceedings of ACM SIGGRAPH*, 1010-1015.

van overveld, C. W. A. M., and WYVILL, B. 1997.Phong normal interpolation revisited.*ACM Transaction on Graphics 16*, 4, 397-419.

Vlachos, A., Peters, J., Boyd, C., and Mitchell, J. 2001.弯曲的PN三角形。*ACM交互式3D研讨会论文集*，159-166。

Zorin, D., Schroeder, P., and Sweldens, W. 1996.具有任意拓扑结构的网格的插值细分。In *Proceedings of ACM SIGGRAPH*, 189-192.