DOI: 10.1111/j.1467-8659.2011.01993.x

Eurographics Symposium on Rendering 2011
Ravi Ramamoorthi and Erik Reinhard
(Guest Editors)

Volume 30 (2011), Number 4

Direct Ray Tracing of Phong Tessellation

Shinji Ogaki[†] and Yusuke Tokuyoshi[‡1]

[1]Square Enix Co., Ltd., Japan

Abstract

There are two major ways of calculating ray and parametric surface intersections in rendering. The first is through the use of tessellated triangles, and the second is to use parametric surfaces together with numerical methods such as Newton's method. Both methods are computationally expensive and complicated to implement. In this paper, we focus on Phong Tessellation and introduce a simple direct ray tracing method for Phong Tessellation. Our method enables rendering smooth surfaces in a computationally inexpensive yet robust way.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Rendering smooth surfaces is important in various fields such as industrial design, film, and games. In the gaming industry, we frequently use cinematic scenes that are often pre-rendered. It is not difficult to render parametric surface in real time graphics because modern GPUs support hardware tessellation. However, it is not easy for offline rendering. In order to obtain exactly the same appearance both in real time graphics and in cinematic scenes, offline renderers and game engines have to use the same tessellation method. It is very helpful to generate high quality images from low polygonal models since artists are freed from the need to create both assets for offline renderers and game engines.

To render a smooth surface, a number of parametric surfaces have been developed such as Cubic A-Patches [BCX95], Curved PN Triangle [VPBM01], NURBS, and T-Spline [SZBN03]. NURBS is known as one of the most flexible parametric surfaces and the standard for industrial design. In the gaming industry, simple tessellation algorithms including Curved PN Triangle are prefered since they fit well for the current rendering pipeline. An offline renderer handles parametric surfaces mainly in two ways. One approach is to perform intersection tests with tessellated triangles. The other method is to directly calculate intersection points. Analytic solutions are available only for limited cases. Many intersection tests for low-degree primitives such as triangles and quadrics can be found in the literature such as Graphics gems series ([Gla90], [Arv91], [Kir92], [Hec94], [Pae95]) and [AMHH08]. A comprehensive study can be found in [SE02]. They also exist for low order parametric surfaces such as bilinear patch [RPH04]. In general, numerical methods are necessary to render higher order parametric surfaces. An intensive study has been done to efficiently render parametric surfaces. Nevertheless, it still remains an open problem.

Among various parametric surfaces, Phong Tessellation [BA08] has recently increased in popularity because it produces nice results despite its simplicity. Phong Tessellation has a couple of nice properties which make it attractive: the shape is determined only by its vertices and vertex normals, and analytic solutions exist for the ray-Phong Tessellation intersection problem. In this paper, we therefore focus on Phong Tessellation and present a simple and robust direct ray tracing method for it. Besides the intersection test, we introduce three techniques to efficiently render Phong Tessellation: 1) an axis aligned bounding box (AABB) overlap test, 2) a surface area heuristic (SAH) kd-tree construction method, and 3) a normal smoothing method.

2. Related Work

Tessellated Triangles Many efficient ray tracing methods for tessellated polygons have been proposed, such as [SSS00]. This method tessellates triangles at runtime and performs intersection tests along a ray hence additional memory is not required over original triangle models. State-

† ogaki@square-enix.com
‡ tokuyosh@square-enix.com

© 2011 The Author(s)
Computer Graphics Forum © 2011 The Eurographics Association and Blackwell Publishing Ltd. Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

of-the-art methods can handle motion blur of micropolygons (see for example [HQL*10]). To achieve high performance, tessellated triangles must be cached and rays have to be traced in a coherent way so that cache performance is maximized. Additionally, in order to select the appropriate tessellation level, vital for achieving a high quality result, ray differentials [Ige99] must be used. View-dependent adaptivity metrics such as [Bou10b] are also useful to increase performance.

Numerical Method Intersection points can be found via implicitization such as [SA84]. Another common method is to express a ray as an intersection of two orthogonal planes and obtain intersection points as the intersections of two planar curves. In order to solve these problems, numerical methods including Newton's method, Laguerre's method, and Bézier Clipping [SN90] can be used. For higher order parametric surfaces, sophisticated root finding algorithms such as [EM95] can be used, however rendering speed might be inferior to numerical methods. We can simplify the problem by removing one variable from bivaritate planar curve equations with the Sylvester resultant [Kaj82]. This enables us to obtain analytic solution up to a quadratic parametric surface. However this is susceptible to numeric errors. Increasing ray coherency, for example, by reordering rays, can improve performance further [JB86]. Wang et al. combined Newton's method and Bézier Clipping to increase robustness [WSC02]. Their method uses a previus intersection point as the initial value of Newton's method to find the next intersection point. If the next intersection point is invalid, the algorithm switches to use Bézier Clipping. Toth [Tot85] applied Interval Arithmetic to find intersection points. This approach was later improved with SIMD [KHH*07]. Recently, Knoll et al. [KHK*09] proposed a rendering method of arbitrary implicit surfaces with Reduced Affine Arithmetic. Loop et al. [LB06] used GPU to accelerate rendering of piecewise algebraic surfaces, and also introduced a preconditioning technique to obtain more robust solutions.

Avoiding Redundant Intersection Tests In order to reduce the number of redundant intersection tests, mail-boxing can be used. Inverse mail-boxing [SK07] is more suitable when the rendering process is parallelized. Hunt introduced cost function modification regarding inverse mail-boxing [Hun08].

Shading Normals Other related research is normal smoothing. Breen [Bre86] proposed to use Phong normal to achieve smooth shading of Steiner Patch. Phong interpolation has been improved in the literature such as in [vOW97]. Consistent Normal Interpolation [RSM10] removes artifacts at grazing angles. These algorithms cannot improve the quality of shadow boundaries and silhouettes of objects. Hence we have to rely on numerical methods or tessellation to improve image quality.

3. Our Method

This section describes the details of our method. First, we briefly review Phong Tessellation. Next we explain the ray-Phong intersection test and Phong Tessellation-AABB overlap test. Then a SAH kd-tree construction method is introduced. Finally we describe how to calculate shading normals.

3.1. Phong Tessellation

The definition of Phong Tessellation is given as follows. Let (u,v,w) be a barycentric coordinate and P_i a vertex. Phong Tessellation for a triangle is defined as

$$P_\alpha(u,v) = (1-\alpha)P(u,v) + \alpha P^*(u,v), \tag{1}$$

where

$$\begin{aligned} P(u,v) &= (u,v,w)(P_1,P_2,P_3)^T \\ P^*(u,v) &= (u,v,w)\begin{pmatrix} \pi_1(P(u,v)) \\ \pi_2(P(u,v)) \\ \pi_3(P(u,v)) \end{pmatrix}, \end{aligned}$$

and $w = 1-(u+v)$. Letting N_i be a vertex normal vector, the function π_i is given as

$$\pi_i(Q) = Q - ((Q-P_i)^T N_i)N_i. \tag{2}$$

The factor α is used for controlling the shape. Phong Tessellation is extended for a quadrangle as

$$\begin{aligned} P(u,v) &= uvP_1 + (1-u)(1-v)P_4 \\ &+ (1-u)vP_2 + u(1-v)P_3 \\ P^*(u,v) &= uv\pi_1(P(u,v)) + (1-u)(1-v)\pi_4(P(u,v)) \\ &+ (1-u)v\pi_2(P(u,v)) + u(1-v)\pi_3(P(u,v)). \end{aligned}$$

See [BA08] and [Bou10a] for more details.

3.2. Ray Phong Tessellation Intersection Test

Since Phong Tessellation defines a quadratic parametric surface, the intersection test can be rearranged into the intersection problem of two quadratic curves.

Every ray can be represented as an intersection of two planes, say $Plane_1(x,y,z) = D_1 \cdot (x,y,z)^T + O_1$ and $Plane_2(x,y,z) = D_2 \cdot (x,y,z)^T + O_2$. The intersection X lies on both the planes. Therefore we have

$$0 = D_1 \cdot X + O_1 \tag{3}$$
$$0 = D_2 \cdot X + O_2. \tag{4}$$

Substituting the right-hand side of Equation (1) for X of Equation (3) and (4) gives two quadratic curves:

$$0 = F(u,v) = au^2 + bv^2 + c + duv + eu + fv \tag{5}$$
$$0 = G(u,v) = lu^2 + mv^2 + n + ouv + pu + qv, \tag{6}$$

© 2011 The Author(s)
© 2011 The Eurographics Association and Blackwell Publishing Ltd.

where

$$\begin{aligned} a &= -D_1 \cdot C_{31} \\ b &= -D_1 \cdot C_{23} \\ c &= D_1 \cdot P_3 + O_1 \\ d &= D_1 \cdot (C_{12} - C_{23} - C_{31}) \\ e &= D_1 \cdot (C_{31} + E_{31}) \\ f &= D_1 \cdot (C_{23} - E_{23}) \\ l &= -D_2 \cdot C_{31} \\ m &= -D_2 \cdot C_{23} \\ n &= D_2 \cdot P_3 + O_2 \\ o &= D_2 \cdot (C_{12} - C_{23} - C_{31}) \\ p &= D_2 \cdot (C_{31} + E_{31}) \\ q &= D_2 \cdot (C_{23} - E_{23}) \end{aligned}$$

and E_i and C_i are given as

$$E_{ij} = P_j - P_i \tag{7}$$

$$C_{ij} = \alpha((N_j \cdot E_{ij})N_j - (N_i \cdot E_{ij})N_i). \tag{8}$$

Two quadratic curves have four intersections in general, hence this problem can be solved analytically. (It is noteworthy that Phong Tessellation can be regarded as the integral Steiner, thus intersection points can also be obtained with the implicitization method described in [SA84]. However, this method is slightly complicated since a 10×10 linear system has to be solved.) We can obtain the intersections of the two planar curves $F(u,v)$ and $G(u,v)$ with [Kaj82]. To save space, we only describe the result. If F and G have one or more common roots, the determinant of the Sylvester matrix is zero. Hence we have

$$0 = a_4 u^4 + a_3 u^3 + a_2 u^2 + a_1 u + a_0, \tag{9}$$

where

$$\begin{aligned} a_4 &= abo^2 + a^2m^2 + d^2lm + b^2l^2 - admo - bdlo - 2ablm \\ a_3 &= beo^2 + d^2mp \\ &- admq - bdlq - bdop - afmo - demo - bflo \\ &+ 2(aem^2 + b^2lp + aboq + dflm - abmp - belm) \\ a_2 &= abq^2 + f^2lm + bco^2 + d^2mn + b^2p^2 + e^2m^2 \\ &- bfop - bdno - efmo - cdmo \\ &- bdpq - afmq - demq - bflq \\ &+ 2(b^2ln + acm^2 + beoq + dfmp - bemp - abmn - bclm) \\ a_1 &= beq^2 + f^2mp \\ &- bfpq - bdnq - efmq - cdmq - bfno - cfmo \\ &+ 2(cem^2 + b^2np + bcoq + dfmn - bcmp - bemn) \\ a_0 &= bcq^2 + b^2n^2 + f^2mn + c^2m^2 - bfnq - cfmq - 2bcmn. \end{aligned}$$

Solving this quartic equation, values for u are obtained. Substituting the values for u of Equation (5) or (6), we obtain values for v.

However, we observed that Kajiya's method generates artifacts because of numerical errors. Therefore, we propose a simpler and more robust method which makes use of a pencil of curves (see [Hos92] for more details). The pencil P is defined as

$$0 = P = xF(u,v) + G(u,v) = (u,v,1)M(u,v,1)^T, \tag{10}$$

where

$$M = \begin{pmatrix} xa+l & (xd+o)/2 & (xe+p)/2 \\ (xd+o)/2 & xb+m & (xf+q)/2 \\ (xe+p)/2 & (xf+q)/2 & xc+n \end{pmatrix}. \tag{11}$$

If $0 = |M|$, P can be represented as a product of two lines. This is a special case of a hyperbola. If this is not clear, remember that $u^2/s^2 = v^2/t^2$ can be expressed as $(tu+sv)(tu-sv) = 0$. By letting

$$L_1 = \alpha_1 u + \beta_1 v + \gamma_1 \tag{12}$$

$$L_2 = \alpha_2 u + \beta_2 v + \gamma_2, \tag{13}$$

we have

$$(u,v,1)M(u,v,1)^T = L_1 L_2. \tag{14}$$

The factorization is done by comparing the coefficients of u^2, v^2, uv, u, v, and 1. Dividing the both sides of Equation (14) by M_{11} or M_{22}, whichever has a greater absolute value, makes the calculation easier and more robust. In the case of $M_{11} > M_{22}$, the two lines L_1 and L_2 are obtained as

$$L_1 = u + \left(M'_{12} + \sqrt{M'^2_{12} - M'_{22}}\right) v + \left(M'_{13} \pm \sqrt{M'^2_{13} - M'_{33}}\right)$$

$$L_2 = u + \left(M'_{12} - \sqrt{M'^2_{12} - M'_{22}}\right) v + \left(M'_{13} \mp \sqrt{M'^2_{13} - M'_{33}}\right),$$

where $M'_{ij} = M_{ij}/M_{11}$. The coefficients γ_1 and γ_2 are chosen so that $M'_{23} = \beta_1\gamma_2 + \beta_2\gamma_1$. Similary, in the case of $M_{22} > M_{11}$, L_1 and L_2 are obtained as

$$L_1 = \left(M'_{12} + \sqrt{M'^2_{12} - M'_{11}}\right) u + v + \left(M'_{23} \pm \sqrt{M'^2_{23} - M'_{33}}\right)$$

$$L_2 = \left(M'_{12} - \sqrt{M'^2_{12} - M'_{11}}\right) u + v + \left(M'_{23} \mp \sqrt{M'^2_{23} - M'_{33}}\right),$$

where $M'_{ij} = M'_{ij}/M_{22}$. The coefficients γ_1 and γ_2 are chosen so that $M'_{13} = \alpha_1\gamma_2 + \alpha_2\gamma_1$.

Let $\{\theta \cap \phi\}$ denote the set of the intersections of θ and ϕ. The set of the intersections is decomposed as

$$\{F(u,v) \cap G(u,v)\} = \{F(u,v) \cap L_1\} + \{F(u,v) \cap L_2\} \tag{15}$$

since

$$\begin{aligned} \{F(u,v) \cap G(u,v)\} &= \{F(u,v) \cap (F(u,v) + G(u,v))\} \\ &= \{F(u,v) \cap (xF(u,v) + G(u,v))\} \\ &= \{F(u,v) \cap P\} \\ &= \{F(u,v) \cap L_1 L_2\}. \end{aligned}$$

© 2011 The Author(s)
© 2011 The Eurographics Association and Blackwell Publishing Ltd.

Thus, the problem can be simplified by finding the value of x such that $0 = |M|$. The values of x are given by solving the following cubic equation

$$0 = a_3x^3 + a_2x^2 + a_1x + a_0, \tag{16}$$

where

$$\begin{aligned} a_3 &= abc + (def - af^2 - be^2 - cd^2)/4 \\ a_2 &= abn + amc + lbc - (afq + bep + cdo)/2 \\ &+ (oef + deq + dpf - lf^2 - me^2 - nd^2)/4 \\ a_1 &= amn + lbn + lmc - (lfq + mep + ndo)/2 \\ &+ (dpq + oeq + opf - aq^2 - bp^2 - co^2)/4 \\ a_0 &= lmn + (opq - lq^2 - mp^2 - no^2)/4. \end{aligned}$$

We only need to obtain one value of x, which significantly reduces the cost of root finding. Substituting Equation (12) and (13) into Equation (5) or (6) gives two quadratic equations. By solving them, we obtain values for u and v. The parameters u, v, and $w = 1 - (u+v)$ must lie between 0 and 1. Otherwise there is no intersection. Note also that the intersection test is immediately terminated if $0 < M_{11}M_{22} - M_{12}M_{21}$ since $0 = (u,v,1)M(u,v,1)^T$ becomes an ellipsoid whose area is zero.

3.3. Phong Tessellation - AABB Overlap Test

In order to build a high quality acceleration structure, we need the Phong Tessellation-AABB overlap test. We perform the triangle-AABB overlap test [AM01] against an enlarged AABB instead of calculating exact intersection points.

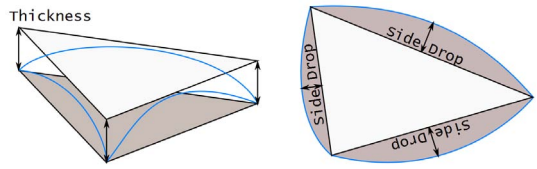

Figure 1: *Thickness and side drops of Phong Tessellation*

An AABB is enlarged by an offset which is the maximum of *thickness* and all *side drops* (Figure 1). *Thickness* is the maximum distance between $P_\alpha(u,v)$ and the triangle $P_1P_2P_3$ (Figure 1 Left), which is defined as

$$thickness = max(|D(u,v)|), \tag{17}$$

where

$$D(u,v) = N_G \cdot (P_\alpha(u,v) - P_1) \tag{18}$$

and $N_G = (E_{12} \times E_{13})/|E_{12} \times E_{13}|$. Generally, this problem is solved by minimizing the L_2-norm of $D(u,v)$. It is worth noting that in most cases the following condition holds:

$$0 \leq N_G \cdot N_i \tag{19}$$

for $i = 1,2,3$. In this case we can significantly simplify the problem assuming that $0 \leq D(u,v)$ or $0 \geq D(u,v)$ for every u and v such that $0 \leq u$, $0 \leq v$, and $0 \leq w$. Letting

$$0 = \frac{\partial D}{\partial u} = \frac{\partial D}{\partial v} \tag{20}$$

yields two line equations. Finding their intersection, we obtain the estimates of u and v, $\hat{u}$ and $\hat{v}$, which are given as

$$\begin{aligned} \hat{u} &= K(2(N_G \cdot C_{23})(N_G \cdot (C_{31} + E_{31})) \\ &+ (N_G \cdot (C_{23} - E_{23}))(N_G \cdot (C_{12} - C_{23} - C_{31}))) \quad (21) \\ \hat{v} &= K(2(N_G \cdot C_{31})(N_G \cdot (C_{23} - E_{23})) \\ &+ (N_G \cdot (C_{31} + E_{31}))(N_G \cdot (C_{12} - C_{23} - C_{31}))), \quad (22) \end{aligned}$$

where

$$K = \frac{1}{4(N_G \cdot C_{23})(N_G \cdot C_{31}) - (N_G \cdot (C_{12} - C_{23} - C_{31}))^2}.$$

However $\hat{u}$ and $\hat{v}$ may not lie between 0 and 1. If this is the case, we also have to evaluate the points $(\hat{u},0)$ and $(0,\hat{v})$. *Side drop* S_{ij} is the maximum distance between the edge E_{ij} and edge of $P_\alpha(u,v)$ (Figure 1 Right), which is given as

$$S_{ij} = max(|(P_j - P_\alpha(u,v)) \times (P_i - P_\alpha(u,v))|/|E_{ij}|). \tag{23}$$

Obtaining *side drop* is easy thanks to the property of Phong Tessellation. Since each edge is a quadratic curve, we only have to evaluate the midpoint i.e. $(u,v,w) = (0,0.5,0.5)$, $(0.5,0,0.5)$, and $(0.5,0.5,0)$.

Once an offset is computed, it is reused during an acceleration construction process.

3.4. SAH KD-Tree Construction

In our implementation, kd-tree is used as an acceleration structure. SAH has to be used as a cost metric to construct a high quality kd-tree. Since obtaining exact terminals and sorting them are time consuming operations, we use the min-max binning algorithm [SSK07] with a slight modification. Our algorithm performs min-max binning with the previously described AABB overlap test in a similar manner to binary search. To update min-bins set, we split the current node and perform an overlap test with child node boxes (light blue and red boxes as shown in Figure 2). If the lower node box overlaps, it is further subdivided. If not, the upper node box is subdivided. This process is recursively performed until it reaches a user specified maximum depth level (Figure 2 Top). The same approach is used to update max-bins set (Figure 2 Bottom). The binary search part can be made quaternary or octary by using SIMD instruction sets such as Intel AVX. This speeds up the kd-tree construction process.

Since our system handles various types of primitives, we need to modify the cost metric SAH. SAH is defined as

$$\begin{aligned} SAH &= SA(Node_L) \times T_L + SA(Node_R) \times T_R \\ &+ NodeTraversalCost, \end{aligned} \tag{24}$$

© 2011 The Author(s)
© 2011 The Eurographics Association and Blackwell Publishing Ltd.

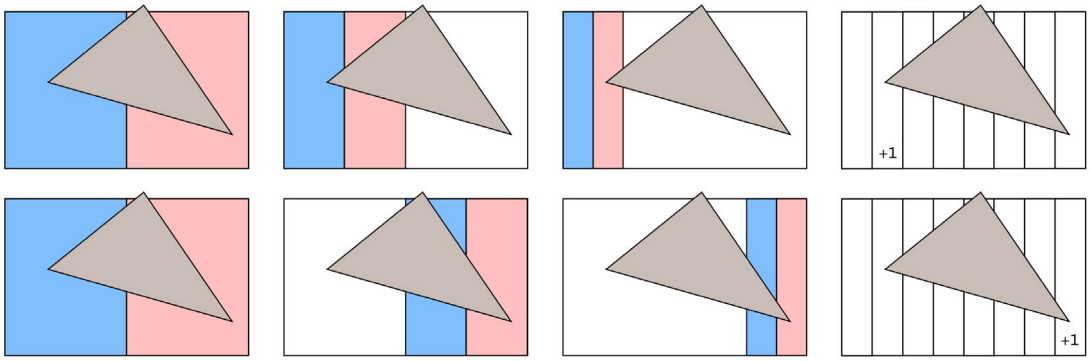

Figure 2: *Binary search based binning method. Top: Finding lower bin. Bottom: Finding upper bin.*

where SA denotes the surface area of a child node, and T_L and T_R are the number of triangles in the left node and right node, respectively. By introducing the ratio r based on the cost of ray-triangle interscction test, SAH is modified as

$$\begin{aligned} SAH &= SA(Node_L) \times (T_L + r \times PT_L) \\ &+ SA(Node_R) \times (T_R + r \times PT_R) \\ &+ NodeTraversalCost, \end{aligned} \tag{25}$$

where PT_L and PT_R are the number of Phong Tessellations in the left and right node, respectively. Note that our cost function does not take into account inverse mail-boxing. Rendering performance varies depending on r. Through our experiments we observed that $r = 4$ is suitable as the cost ratio.

3.5. Shading Normal

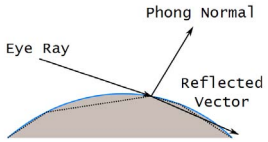

Figure 3: *Shading Normal. At grazing angles, reflected vectors might go underneath the surface.*

For smooth rendering, we use the Phong-interpolated normal N_P as in [Bre86]. It is given as

$$N_P = (uN_1 + vN_2 + wN_3)/|uN_1 + vN_2 + wN_3|. \tag{26}$$

In general, using N_P produces good results. However, this leads to unpleasant artifacts at grazing angles since N_P is not necessarily equivalent to the surface normal of Phong Tessellation N_S. To address this issue, we use N_S instead if the inner product of N_S and the reflection vector r is negative (Figure 3). Letting v be the viewing direction, r is given as

$$r = v - 2N_P(v \cdot N_P). \tag{27}$$

The surface normal N_S can be derived as

$$N_S = \left(\frac{\partial P_\alpha(u,v)}{\partial u} \times \frac{\partial P_\alpha(u,v)}{\partial v}\right) \Bigg/ \left|\frac{\partial P_\alpha(u,v)}{\partial u} \times \frac{\partial P_\alpha(u,v)}{\partial v}\right|. \tag{28}$$

The partial derivatives $\frac{\partial P_\alpha(u,v)}{\partial u}$ and $\frac{\partial P_\alpha(u,v)}{\partial v}$ are obtained as

$$\frac{\partial P_\alpha(u,v)}{\partial u} = (w-u)C_{31} + v(C_{12} - C_{23}) + E_{31} \tag{29}$$

$$\frac{\partial P_\alpha(u,v)}{\partial v} = (w-v)C_{23} + u(C_{12} - C_{31}) - E_{23}. \tag{30}$$

We cannot directly apply Consistent Normal Interpolation since Phong Tessellation is not flat. However, this simple method works well in practice since the gap between the geometric normal and Phong normal is relatively small.

4. Result

We implemented the above described algorithm in our renderer. Figure 4 (Top) shows a low polygon sphere model and Figure 4 (Middle) shows the same model rendered with our method. Our normal smoothing method enables us to properly render a reflective material at grazing angles. Figure 4 (Bottom) shows the same scene rendered with our method and Kajiya's method [Kaj82]. Cracks can be seen in the image rendered with Kajiya's method while no visible artifacts are generated with our method. In our implementation quartic equations are solved with Ferrari's method and cubic equations are solved with Cardano's method. Other root finding methods might be able to remove cracks. However, this result shows that our method gives better results with standard techniques.

To further improve the AABB overlap test, we can split Phong Tessellation into N small patches and operate in parallel because modern CPUs support SIMD instruction. We tested this with $N = 4$, however, very little performance gain was obtained. This is because our intersection test is implemented together with inverse mail-boxing hence a small amount of redundant intersection tests can be ignored.

© 2011 The Author(s)
© 2011 The Eurographics Association and Blackwell Publishing Ltd.

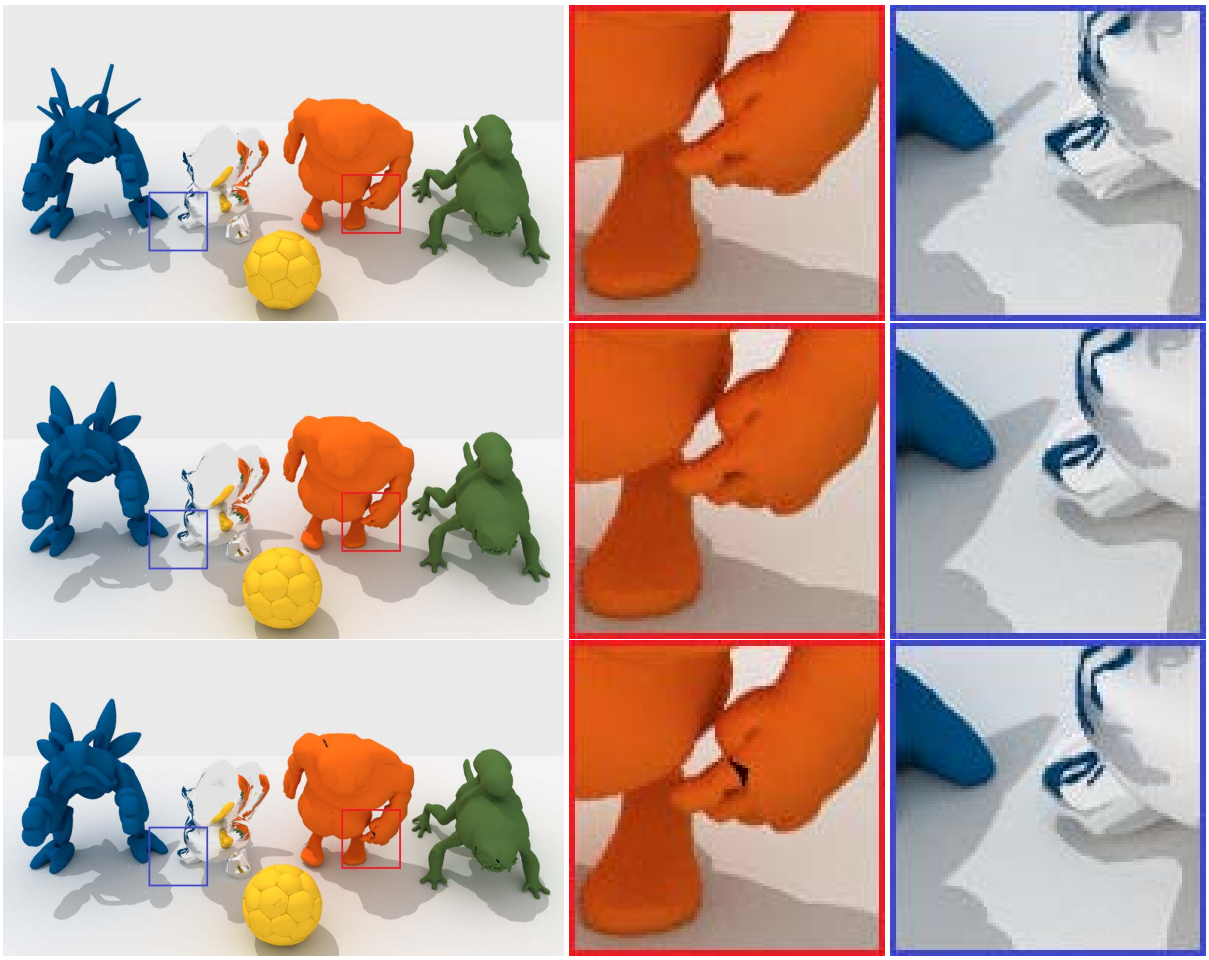

Figure 4: *Top: Low polygon models (in signle precision, 960x540pixels, 24seconds). Middle: Low polygon models rendered with our method (in single precision, 960x540pixels, 64seconds). Bottom: Low polygon models rendered with Kajiya's method (in double precision, 960x540pixels, 103seconds). All images were rendered on a dual Xeon W5590 system.*

5. Discussion and Future Work

Intersection Test Kajiya's method needs to solve a quartic equation while ours solves a cubic equation. This fact makes our algorithm more robust against numerical errors, hence no visually perceptible noise is produced. During the intersection test, we could obtain three x's such that $0 = |M|$. Choosing x that minimizes $M_{12}M_{21} - M_{11}M_{22}$ is reasonable. However, it is difficult to see how the choice of x affects the numerical accuracy. Interval Analysis is one such technique to estimate the range of numerical errors. However, the plain use of Interval Arithmetic may result in a range explosion. We want to find a suitable quantitative evaluation method. We also would like to compare the accuracy of our method with other numerical methods.

If the shape controlling factor α is near zero, or all vertex normals are very similar to each other, small cracks might be generated. This is because in these cases the intersection test is done by treating a thin patch as a flat triangle. Even though it is unnoticeable until an image is rendered at very high resolution, we would like to develop a better technique to address this issue.

SAH KD-Tree Construction Our kd-tree construction is based on the min-max binning algorithm. Its memory requirement is significantly lower compared to other algorithms because only AABB overlap tests are used. The resulting kd-tree is not optimal, however a reasonably high quality tree can be built quickly with the help of SIMD instructions. We believe this algorithm is very useful especially in memory limited environments. An interesting research direction is to develop a fast SAH kd-tree construction algorithm for an arbitrary parametric surface. A better exploitation of SIMD instruction sets may also be helpful.

© 2011 The Author(s)
© 2011 The Eurographics Association and Blackwell Publishing Ltd.

6. Conclusion

In this paper, we introduced a simple and robust ray tracing method for Phong Tessellation which makes use of a pencil of curves. Image quality is high enough for non-product design fields, especially for organic shapes. We believe our algorithm is suitable to render smooth surfaces since high quality images can be rendered from low polygon models without changing the renderer architecture. We also described the Phong Tessellation-AABB overlap test and SAH kd-tree construction algorithm. They are very simple thanks to the property of Phong Tessellation. The idea of our overlap test and kd-tree construction method is applicable to other parametric surfaces and other data structures such as Bounding Volume Hierarchy or Hierarchical Grid.

7. Acknowledgement

We would like to thank Tamy Boubekeur and Yuta Araki for allowing us to use the low polygon models, and Arun Mehta and Junko Asakura for their valuable comments. We also thank anonymous reviewers for their helpful suggestions.

References

[AM01] AKENINE-MOLLER T.: Fast 3d triangle-box overlap testing. *journal of graphics, gpu, and game tools 6*, 1 (2001), 29–33. 4

[AMHH08] AKENINE-MÖLLER T., HAINES E., HOFFMAN N.: *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008. 1

[Arv91] ARVO J. (Ed.): *Graphics Gems II*. Academic Press, Inc., 1991. 1

[BA08] BOUBEKEUR T., ALEXA M.: Phong tessellation. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–5. 1, 2

[BCX95] BAJAJ C. L., CHEN J., XU G.: Modeling with cubic a-patches. *ACM Trans. Graph. 14* (April 1995), 103–133. 1

[Bou10a] BOUBEKEUR T.: *GPU Pro - Advanced Rendering Techniques*. ShaderX Book Series. A.K. Peters, 2010, ch. As-Simple-As Possible Tessellation for Interactive Applications. 2

[Bou10b] BOUBEKEUR T.: A view-dependent adaptivity metric for real-time mesh tessellation. In *IEEE International Conference on Image Processing (ICIP 2010)* (2010), pp. 3969 – 3972. 2

[Bre86] BREEN D. E.: Creation and smooth-shading of steiner patch tessellations. In *Proceedings of 1986 ACM Fall joint computer conference* (Los Alamitos, CA, USA, 1986), ACM '86, IEEE Computer Society Press, pp. 931–940. 2, 5

[EM95] EDELMAN A., MURAKAMI H.: Polynomial roots from companion matrix eigenvalues. *Math. Comput. 64* (April 1995), 763–776. 2

[Gla90] GLASSNER A. (Ed.): *Graphics Gems*. Academic Press, Inc., 1990. 1

[Hec94] HECKBERT P. (Ed.): *Graphics Gems IV*. Academic Press, Inc., 1994. 1

[Hos92] HOSAKA M.: *Modeling of curves and surfaces in CAD/CAM*. Springer-Verlag New York, Inc., New York, NY, USA, 1992. 3

[HQL*10] HOU Q., QIN H., LI W., GUO B., ZHOU K.: Micropolygon ray tracing with defocus and motion blur. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), ACM, pp. 1–10. 2

[Hun08] HUNT W.: Corrections to the surface area metric with respect to mail-boxing. In *IEEE/EG Symposium on Interactive Ray Tracing 2008* (Aug 2008), IEEE/EG, pp. 77–80. 2

[Ige99] IGEHY H.: Tracing ray differentials. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 179–186. 2

[JB86] JOY K. I., BHETANABHOTLA M. N.: Ray tracing parametric surface patches utilizing numerical techniques and ray coherence. *SIGGRAPH Comput. Graph. 20* (August 1986), 279–285. 2

[Kaj82] KAJIYA J. T.: Ray tracing parametric patches. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1982), SIGGRAPH '82, ACM, pp. 245–254. 2, 3, 5

[KHH*07] KNOLL A., HIJAZI Y., HANSEN C., WALD I., HAGEN H.: Interactive ray tracing of arbitrary implicits with simd interval arithmetic. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 11–18. 2

[KHK*09] KNOLL A., HIJAZI Y., KENSLER A., SCHOTT M., HANSEN C. D., HAGEN H.: Fast ray tracing of arbitrary implicit surfaces with interval and affine arithmetic. *Comput. Graph. Forum 28*, 1 (2009), 26–40. 2

[Kir92] KIRK D. (Ed.): *Graphics Gems III*. Academic Press, Inc., 1992. 1

[LB06] LOOP C., BLINN J.: Real-time gpu rendering of piecewise algebraic surfaces. *ACM Trans. Graph. 25* (July 2006), 664–670. 2

[Pae95] PAETH A. (Ed.): *Graphics Gems V*. Academic Press, Inc., 1995. 1

[RPH04] RAMSEY S., POTTER K., HANSEN C.: Ray bilinear patch intersections. *Journal of Graphics Tools 9*, 3 (2004), 41–47. 1

[RSM10] RESHETOV A., SOUPIKOV A., MARK W. R.: Consistent normal interpolation. *ACM Trans. Graph. 29* (December 2010), 142:1–142:8. 2

[SA84] SEDERBERG T. W., ANDERSON D. C.: Ray tracing of steiner patches. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), SIGGRAPH '84, ACM, pp. 159–164. 2, 3

[SE02] SCHNEIDER P. J., EBERLY D.: *Geometric Tools for Computer Graphics*. Elsevier Science Inc., New York, NY, USA, 2002. 1

[SK07] SHEVTSOV A. S. M., KAPUSTIN. A.: Ray-triangle intersection algorithm for modern cpu architectures. In *Conference on Computer Graphics and Vision* (2007). 2

[SN90] SEDERBERG T. W., NISHITA T.: Curve intersection using bézier clipping. *Computer-Aided Design 22*, 9 (1990), 538–549. 2

[SSK07] SHEVTSOV M., SOUPIKOV A., KAPUSTIN A.: Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. *Comput. Graph. Forum 26*, 3 (2007), 395–404. 4

[SSS00] SMITS B., SHIRLEY P., STARK M. M.: *Direct Ray Tracing of Smoothed and Displacement Mapped Triangles*. Tech. rep., 2000. 1

© 2011 The Author(s)
© 2011 The Eurographics Association and Blackwell Publishing Ltd.

[SZBN03] SEDERBERG T. W., ZHENG J., BAKENOV A., NASRI A.: T-splines and t-nurccs. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 477–484. 1

[Tot85] TOTH D. L.: On ray tracing parametric surfaces. *SIGGRAPH Comput. Graph. 19* (July 1985), 171–179. 2

[vOW97] VAN OVERVELD C. W. A. M., WYVILL B.: Phong normal interpolation revisited. *ACM Trans. Graph. 16* (October 1997), 397–419. 2

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), ACM Press, pp. 159–166. 1

[WSC02] WANG S.-W., SHIH Z.-C., CHANG R.-C.: An efficient and stable ray tracing algorithm for parametric surfaces. *J. Inf. Sci. Eng. 18*, 4 (2002), 541–561. 2

© 2011 The Author(s)
© 2011 The Eurographics Association and Blackwell Publishing Ltd.