Jurnal Informatika, Vol. 12, No. 1, Mei 2014, 27-32 DOI: 10.9744/informatika.12.1.27-32
ISSN 1411-0105

# PEMANFAATAN KURVA B-SPLINES UNTUK MEMPERHALUS VISUALISASI OBYEK MESH

**Liliana, Gregorius Satia Budhi[1], Erick Leonardo[2], Evans Sanjaya[3]**
[1,2,3] Program Studi Teknik Informatika, Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121-131, Telp. 62-31-2983455
E-mail: lilian@petra.ac.id, greg@petra.ac.id

**Abstrak:** Ray tracing merupakan salah satu metode rendering yang *photorealistic*. Ray tracing membutuhkan waktu proses rendering yang lama karena adanya perhitungan efek pencahayaan dan efek optik yang disebabkan oleh material dari obyek. Lamanya waktu rendering bertambah seiring dengan semakin kompleksnya bentuk obyek, seperti *mesh object*. Untuk meminimalkan waktu proses rendering dengan metode Ray Tracing, umumnya dilakukan dengan mengurangi kompleksitas pada mesh, yaitu mengurangi jumlah face yang membentuk *mesh object*. Cara mengurangi jumlah face adalah dengan cara menggabungkan *face* yang berdekatan sehingga face yang terbentuk, luas. Kelemahan dari cara ini adalah tampilan permukaan obyek yang dihasilkan akan terlihat kasar/bersegi-segi (*checkered*). Metode yang diajukan dalam penelitian ini bertujuan untuk memperhalus visualisasi dari obyek yang kompleksitasnya tidak tinggi. Permukaan mesh yang bersegi-segi tampak karena perhitungan warna yang sama untuk setiap face yang membentuk mesh. Jika pada sebuah face mempunyai gradasi warna, maka permukaan mesh akan tampak halus. Untuk menghasilkan gradasi warna pada sebuah face yang luas, digunakan perhitungan normal yang bergradasi pula pada sebuah *face*, yaitu secara proporsi yang mengadopsi perhitungan kurva *b-splines*. Pengujian dilakukan dengan membandingkan hasil penggunaan normal b-splines terhadap metode *simple subdivision* (metode yang membagi face yang luas menjadi beberapa *face* yang lebih kecil, sehingga permukaan mesh semakin halus) dengan metode rendering *ray tracing*. Dari hasil percobaan, dapat disimpulkan bahwa hasil ray tracing dengan menggunakan normal b-splines jika dibandingkan dengan *subdivision*, hasilnya tidak terlalu berbeda. Dari aspek lama proses rendering, semakin banyak jumlah face dari sebuah mesh, maka semakin meningkat pula efektifitas waktu proses renderingnya. Namun demikian, peningkatannya tidak berbanding lurus dengan peningkatan jumlah mesh.

**Kata kunci:** *Ray tracing*, rendering, kurva b-spines, *smooth surface*, 3D mesh.

***Abstract:*** Ray tracing is one of the photorealistic rendering method. Ray tracing process takes a long time because of the calculation of lighting effects and optical effects caused by the material of the object. The length of rendering time increases with the increasing complexity of the shape of the object, such as a mesh object. To minimize processing time, generally done by reducing the complexity of the mesh, which reduces the number of face that forms a mesh object. How to reduce the number of face is to combine adjacent face to face. The downside of this method is that the object surface resulting display will look grainy/triangular - facet (checkered). The proposed method in this study aims to refine the visualization of objects that complexity is not high. Triangular surface mesh - terms appear due to the calculation of the same color for each face that form a mesh. If the face has a color gradation, the mesh will look smooth surface. To produce a color gradation on a broad face, normal calculation was used, ie in proportion adopting b - splines curve calculations. Testing is done by comparing the results of the use of normal B - splines to the simple method of subdivision (a method of face vast divide into several smaller face , so that the finer the mesh surface) with ray tracing rendering method. From the experimental results, it can be concluded that the results of ray tracing using b - splines normal when compared with the subdivision, the results are not too different. From the aspect of long rendering process, the more the number of face of a mesh, it also increase the effectiveness of rendering time process. However, the increase is not proportional to the increase in the number of mesh

***Keywords****: Ray tracing, rendering, b-spline curve, smooth surface*, 3D mesh

## PENDAHULUAN

Dalam grafika komputer, obyek 3D bisa dimodelkan dengan menggunakan persamaan matematika atau dan ditampilkan dalam bentuk mesh atau wireframe. Untuk mensimulasikan penampilan permukaan obyek 3D yang menyerupai realita, maka digunakan proses rendering. Dalam proses ini, permukaan obyek akan diwarnai menurut jenis dan sifat material serta efek pencahayaan pada permukaan obyek. Selain itu disimulasikan juga adanya efek optik seperti pencerminan. *Ray tracing* merupakan

salah satu metode *rendering* yang banyak digunakan dalam aplikasi komersial yang ada sekarang ini, seperti renderMan dan 3DS max.

Kelebihan dari metode *ray tracing* kemampuannya untuk menghasilkan simulasi penampilan obyek 3D yang menyerupai aslinya (*photorealistic*). Untuk menghasilkan simulasi permukaan obyek yang demikian, dilakukan banyak proses perhitungan yang detail untuk setiap obyek primitif. Obyek primitif yang dimaksud adalah bidang datar, bola, kerucut dan tabung. Sebuah obyek 3D yang lebih kompleks bentuknya disusun dari gabungan obyek-obyek primitif. Salah satu bentuk kompleks dari obyek 3D adalah mesh. Mesh tersusun dari banyak segitiga (bidang datar) yang menutupi seluruh permukaan obyek.

Semakin banyak obyek primitif yang digunakan maka semakin lama waktu yang dibutuhkan untuk melakukan proses rendering bagi obyek tersebut. Untuk mendapatkan obyek dengan permukaan yang halus dan tidak terkesan bersegi-segi maka biasanya digunakan ukuran segitiga yang cukup kecil, dengan teknik subdivision[1]. Hal ini mengakibatkan jumlah obyek primitif meningkat. Implikasinya, waktu proses rendering akan meningkat juga. Pada penelitian yang pernah dilakukan sebelumnya, dilakukan perhitungan normal pada setiap titik pembentuk segitiga yang merupakan rata-rata dari beberapa normal segitiga yang bersinggungan pada titik tersebut[2]. Kemudian normal *face* akan didapat dari rata-rata ketiga normal titik pembentuk segitiga tersebut. Hasil dari penelitian ini masih belum memuaskan karena setiap *face* masih diwarnai dengan warna yang sama.

Dalam menentukan warna sebuah segitiga, normal bidang menjadi penentu[2]. Jika sebuah segitiga hanya mempunyai sebuah normal bidang, maka segitiga tersebut akan diwarnai dengan satu warna yang sama. Sementara itu, pada obyek primitif bola, setiap titik pada permukaan bola mempunyai normal tersendiri. Hal ini menyebabkan terjadinya gradasi warna yang halus pada permukaan bola. Dari hal ini, dapat diambil hipotesa sebagai berikut, jika sebuah bidang mempunyai normal di setiap titik pembentuk bidang, maka pada bidang tersebut akan muncul banyak warna yang diharapkan akan menampilkan gradasi warna yang halus.

Pada metode *phong shading*, gradasi warna pada permukaan obyek dihasilkan dengan cara mewarnai permukaan secara scan line (menampilkan garis demi garis secara horizontal sampai seluruh permukaan terwarnai)[3]. Setiap garis yang terbentuk mempunyai warna tersendiri karena normal garis yang digunakan merupakan interpolasi dari normal kedua titik ujung garis. Dan normal kedua ujung garis merupakan hasil interpolasi dari titik-titik pembentuk segitiga yang dihitung secara proporsi. Dengan mengadopsi metode phong shading, untuk memunculkan normal pada setiap titik pada permukaan bidang, maka akan dilakukan perhitungan proporsi normal bidang pada sebuah titik dalam segitiga terhadap normal dari tiga buah titik pembentuk segitiga. Warna tidak hanya ditampilkan dengan metode blending tetapi akan dihitung untuk setiap titik pada segitiga.

Selain itu, terinspirasi dari permukaan bola yang merupakan lengkungan, maka jika dilakukan pembentukan permukaan virtual dengan menggunakan kurva b-splines, akan mendapatkan normal bidang yang berbeda pada setiap titik pembentuk permukaan segitiga. Dengan adanya kurva, maka permukaan yang datar akan tampil seolah-olah melengkung. Kurva b-splines dipilih karena kurva ini yang menghasilkan lengkungan yang lebih halus dan lebih mendekati titik acuan kurva (knot vector)[4].

Pada penelitian ini, dilakukan implementasi dari kedua metode tersebut di atas.

## PERHITUNGAN WARNA DALAM *RAY TRACING*

Untuk mendapatkan warna atau intensitas pada obyek yang realistik, maka ada dua aspek yang diperhitungkan dalam metode Ray Tracing, yaitu efek pencahayaan dan efek optik. Efek pencahayaan memperhitungkan intensitas yang dipengaruhi oleh keberadaan sumber cahaya, sedangkan efek optik adalah tambahan intensitas yang didapat karena material dari obyek lainnya, misalnya pantulan dari bola kaca. Efek pencahayaan memperhitungkan tiga hal, yaitu ambient, diffuse, dan specular.

Ambient adalah estimasi pencahayaan untuk penerangan global dalam suatu lingkungan tidak memperhitungkan arah datang cahaya. Efek dari ambient akan mempengaruhi terang atau redupnya suatu lingkungan. Intensitas ambient pada suatu objek dapat dicari dengan persamaan 1 [5]:

$$I = I_a * K_a \quad (1)$$

dimana: $I_a$ merupakan intensitas ambient dan $K_a$ merupakan koefisien ambient.

Diffuse adalah pencahayaan yang tergantung dari besarnya sudut yang dibentuk antara sinar dari lampu ke titik tabrak pada objek dengan normal objek. Posisi lampu sangat mempengaruhi efek diffuse ini. Intensitas diffuse dapat dicari dengan persamaan 2 menggunakan hukum Lambertian sebagai berikut:

$$I = I_d * K_d(L \bullet N) / d \quad (2)$$

dimana: $I_d$ merupakan intensitas diffuse, $K_d$ merupakan koefisien diffuse, L merupakan vektor dari titik tabrak ke sumber cahaya, N merupakan vektor normal objek, dan d merupakan jarak antara sumber cahaya dengan titik tabrak.

Specular merupakan efek pencahayaan dimana bayangan sumber cahaya terlihat pada permukaan objek yang mengkilap. Semakin mengkilap permukaan suatu objek maka makin jelas bayangan sumber cahaya yang terlihat. Untuk mencari intensitas specular maka dapat digunakan persamaan 3 sebagai berikut:

$$I = I_p * K_s(R \bullet V)^n / d \quad (3)$$

dimana:

$$R = -S + 2*(S \bullet N) * N \quad (4)$$

$I_p$ merupakan intensitas specular, $K_s$ merupakan koefisien specular, R merupakan vektor arah pantulan, V merupakan vektor negasi arah sinar, d merupakan jarak antara sumber cahaya dengan titik tabrak, S merupakan vektor dari titik tabrak ke sumber cahaya, dan N merupakan vektor normal dari objek.

## NORMAL B SPLINES

*B-splines* merupakan salah satu metode untuk membentuk kurva dari kombinasi titik kontrol [4],[6]. Metode *B-splines* merupakan pengembangan dari metode *Bezier*. Kelebihan dari metode *B-splines* dibandingkan metode *Bezier* adalah derajat kontinuitas yang dapat diubah sampai jumlah titik kontrol, sehingga metode *B-splines* dapat meghasilkan kurva yang sangat halus.

*B-splines* menggunakan *basis function* untuk memperkirakan sejumlah **n** titik dengan kurva *polynomial* berderajat **d** yang memberikan kontinuitas $\mathbf{C^{(d-1)}}$.

Persamaan 5 merupakan formula umum untuk membentuk kombinasi liner dalam *B-splines*[6].

$$f(t) = \sum_{i=1}^{n} p_i b_{i.k}(t) \quad (5)$$

dimana $p_i$ adalah koefisien titik kontol berjumlah *n*, dan $b_i$ adalah *basis function*.

### *Uniform Cubic B-splines*

*Uniform Cubic B-splines* adalah *B-splines* dengan derajat d = 3. *Basis function* dari *Uniform Cubic B-splines* menggunakan persamaan 6[9].

$$b_{i,4}(t) \begin{cases} \frac{1}{6}u^3 & \text{jika } i \le t \le i+1 \text{ dengan } u = t - i \\ \frac{1}{6}(-3u^3+3u^2+3u+1) & \text{jika } i+1 \le t \le i+2 \text{ dengan } u = t-(i+1) \\ \frac{1}{6}(3u^3-6u^2+4) & \text{jika } i+2 \le t \le i+3 \text{ dengan } u = t-(i+2) \\ \frac{1}{6}(-u^3+3u^2-3u+1) & \text{jika } i+3 \le t \le i+4 \text{ dengan } u = t-(i+3) \\ 0 & \text{selainnya} \end{cases} \quad (6)$$

*Basis function* dapat dinotasikan dengan matriks sebagaimana dalam persamaan 7[9].

$$M_b = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \quad (7)$$

### *B-splines Surface*

*B-splines Surface* merupakan pengembangan dari *B-splines Curve*. Permukaan dibentuk dari langkah-langkah berikut[9]:

1. Titik kontrol sejumlah *m* baris dan $n$ kolom.
2. *Knot vector* sejumlah *h* pada sumbu *u*, $U = \{u_1, u_2, u_3, \ldots u_h\}$.
3. *Knot vector* sejumlah *k* pada sumbu *v*, $V = \{v_1, v_2, v_3, \ldots v_h\}$.
4. *v*, $V = \{v_1, v_2, v_3, \ldots v_h\}$.
5. Derajat *p* pada sumbu *u*
6. Derajat *q* pada sumbu *v*

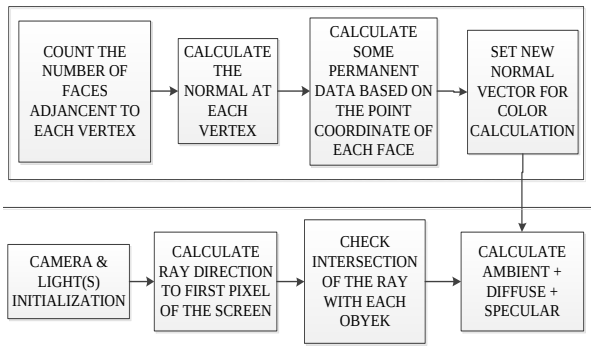
Permukaan dibentuk dengan interpolasi sumbu *u* dan sumbu *v*. Formula untuk membentuk permukaan *B-spline* adalah sebagai berikut[4],[6]:

$$f(u,v) = \sum_{i=1}^{m}\sum_{j=1}^{n} b_{i.p}(u) b_{j-q}(v) p_{i.j} \quad (8)$$

dimana $b_{i,p}(u)$ dan $b_{j-q}(v)$ adalah *basis functions*. Nilai *h* diperoleh dari $m + p + 1$, demikian pula untuk nilai *k* diperoleh dari $n + q + 1$.

## ARSITEKTUR SISTEM DAN FLOWCHART

### Arsitektur Sistem



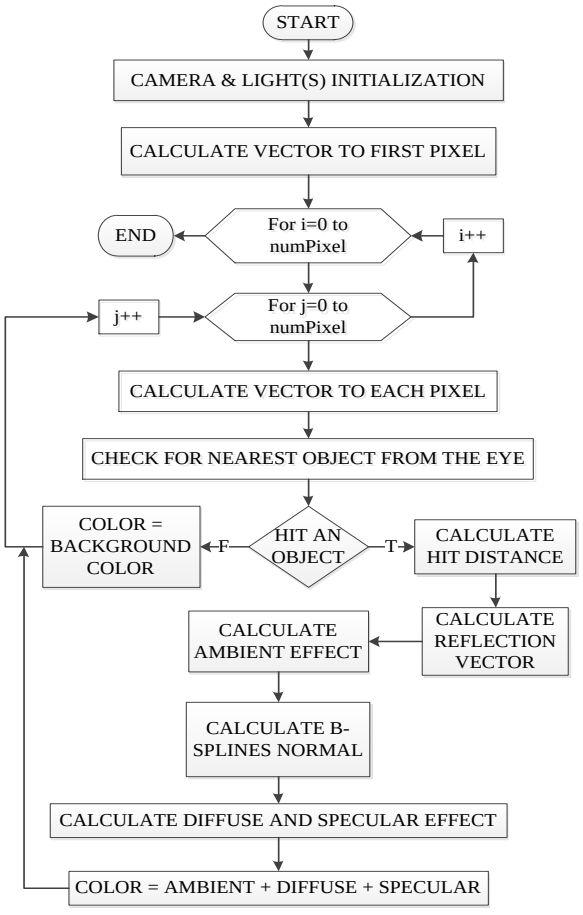

**Gambar 1.** Arsitektur sistem

Proses perhitungan dibagi menjadi dua bagian. Bagian pertama, menghitung semua variable yang dapat dihitung terlebih dahulu sebelum melakukan proses rendering. Hal ini akan menghemat banyak waktu proses karena tidak menghitung variable yang sama berulang-ulang. Yang harus dihitung terlebih

dahulu adalah normal pada setiap titik dalam mesh. Cara menghitungnya adalah dengan merata-rata normal face yang bertumpu pada titik yang dimaksudkan. Setelah mendapatkan normal dari setiap titik pada obyek mesh, maka perhitungan berikutnya adalah menghitung panjang dua sisi dari setiap face terhadap titik pada face tetangganya. Panjang sisi ini nantinya akan digunakan dalam menghitung bobot (u) pada persamaan 6.

Proses kedua adalah proses rendering. Perbedaan dengan proses rendering biasanya adalah adanya perhitungan normal yang berbeda untuk setiap titik tabrak pada sebuah face. Jika rendering biasa menggunakan normal face, maka untuk rendering dengan barycentric normal ini menghitung normal yang berbeda untuk setiap titik pada permukaan face. Proses perhitungannya adalah pada saat titik tabrak sudah diketahui koordinatnya, maka akan dihitung bobot berdasarkan jarak ke titik-titik pada face yang bertetangga. Kemudian dari bobot tersebut digunakan persamaan 7 yang dikalikan dengan normal titik yang bersesuaian untuk mendapatkan normal pada titik tabrak tersebut. Proses rendering dapat dilihat pada flowchart di Gambar 2.

**Flowchart**




**Gambar 2.** Flowchart Ray tracing dengan perhitungan Normal Spline

Pengujian Sistem dan Pembahasan

Pengujian sistem dilakukan mengunakan komputer dengan spesifikasi sebagai berikut:

Microprocessor : 2.40 GHz Intel Core i5-520M processor
Microprocessor cache : 3 MB L2 Cache
Memory : 4 GB DDR3
Video graphics : ATI Mobility Radeon HD 4550 Graphics
Video memory : 512 MB GDDR3 dedicated memory
Operating system : Windows 7 32 bit

Pengujian dilakukan dengan menggunakan obyek primitif (obyek sederhana) dan obyek kompleks. Perbandingan dilakukan terhadap proses raytracing tanpa metode penghalusan, dengan menggunakan kurva b-splines. Aspek yang diperbandingkan adalah lama proses rendering dan penggunaan memory.

Aspek yang mempengaruhi lama proses *rendering* pada ray tracing adalah resolusi screen yang akan dihasilkan, jumlah obyek yang dirender, dalah hal ini adalah jumlah face, posisi kamera yang akan mempengaruhi presentase visualisasi obyek terhadap resolusi screen

Pengujian dilakukan dengan dua aspek pengamatan, yaitu lama proses dan penggunaan memory. Hasil perbandingan lama proses rendering antara ray tracing biasa, ray tracing dengan perhitungan *barycentric* normal dapat dilihat pada Tabel 1 dan perbandingan dalam aspek penggunaan memory dapat dilihat pada Tabel 2.

**Tabel 1.** Peningkatan lama proses ray tracing normal b-splines terhadap ray tracing biasa.

| No. | Nama Obyek | Jumlah face | *Ray tracing* dengan normal b-splines |
|---|---|---|---|
| 1 | *Ball.3ds* | 960 | 232,33% |
| 2 | *Torus.obj* | 1152 | 288,62% |
| 3 | *Cone.3ds* | 62 | 190,87% |
| 4 | *Icosphere.obj* | 80 | 144,98% |
| 5 | *Gourd.obj* | 648 | 215,94% |
| 6 | *Magnolia.obj* | 1372 | 219,55% |
| 7 | *Unicorn.obj* | 1572 | 213,18% |
| 8 | *Monkey.obj* | 968 | 310,49% |
| 9 | *Dolphin.obj* | 1692 | 218,96% |
| 10 | *Skull.obj* | 7120 | 291,37% |

Dari tabel 1 dapat dilihat bahwa waktu proses raytracing dengan menggunakan normal b-spline lebih lama 2-3 kali dibandingkan dengan ray tracing biasa. Sedangkan dari tabel 2 dapat dilihat bahwa penggunaan memory tidak meningkat secara kentara

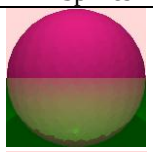
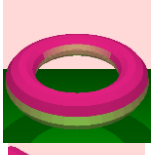
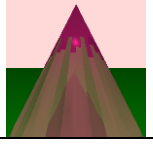
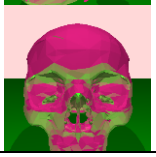
karena dilakukan secara terpisah antara proses perhitungan normal dengan rendering. Bagian perhitungan normal dilakukan terlebih dahulu.

**Tabel 2.** Peningkatan penggunaan *memory* dari proses ray tracing dengan normal b-splines terhadap ray tracing biasa

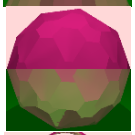
| *No.* | *Nama Object* | *Ray tracing* dengan normal b-splines |
|---|---|---|
| 1 | *Ball.3ds* | 100,1% |
| 2 | *Torus.obj* | 100 % |
| 3 | *Cone.3ds* | 100,35% |
| 4 | *Icosphere.obj* | 100% |
| 5 | *Gourd.obj* | 100,35% |
| 6 | *Magnolia.obj* | 100,41% |
| 7 | *Unicorn.obj* | 100% |
| 8 | *Monkey.obj* | 100% |
| 9 | *Dolphin.obj* | 100% |
| 10 | *Skull.obj* | 100% |

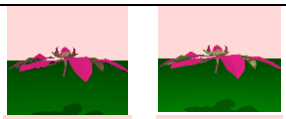
Pengujian terhadap hasil render dilakukan dengan menggunakan perhitungan barycentric normal dibandingkan dengan jika menggunakan meode *subdivision*. Untuk pengujian ini digunakan obyek primitive dan obyek kompleks. Obyek primitif mempunyai permukaan yang cenderung lebih halus karena bentuknya yang sederhana dan cenderung conveks. Walaupun demikian, jika obyek primitif dibentuk dengan menggunakan sedikit face maka tetap akan terlihat bersiku-siku. Dengan menggunakan perhitungan barycentric normal maka hal tersebut bisa diatasi dengan baik seperti terlihat pada Tabel 3. Empat obyek pertama dari Tabel 3 adalah obyek primitif, sedang 6 obyek selanjutnya adalah obyek yang kompleks. Pada pecobaan ini, sebagian besar obyek berhasil dirender dengan halus. Hanya pada obyek "unicorn.obj" hasil kurang maksimal. Hal ini disebabkan oleh jumlah face yang relatif sedikit dibandingkan dengan ukuran obyek.

**Tabel 3.** Hasil rendering ray tracing dengan normal b-splines dibandingkan dengan subdivision

| No. | Nama Objek | Jumlah Face | Hasil dengan B-Splines | Hasil dengan Subdivision |
|---|---|---|---|---|
| 1 | Ball.3ds | 960 |  |  |
| 2 | Torus.obj | 1152 |  |  |
| 3 | Cone.obj | 62 |  |  |
| 4 | Icosphere.obj | 80 |  |  |
| 5 | Gourd.obj | 648 |  |  |
| 6 | Magnolia.obj | 1372 |  |  |
| 7 | Unicorn.obj | 1572 |  |  |
| 8 | Monkey.obj | 968 |  |  |
| 9 | Dolphin.obj | 1692 |  |  |
| 10 | Skull.obj | 7120 |  |  |

**Tabel 4.** Hasil rendering ray tracing dengan normal b-splines dibandingkan dengan ray tracing biasa

| No. | Nama Objek | Jumlah Face | Hasil dengan B-Splines | Hasil dengan ray tracing biasa |
|---|---|---|---|---|
| 1 | Ball.3ds | 960 |  |  |
| 2 | Torus.obj | 1152 |  |  |
| 3 | Cone.obj | 62 |  |  |
| 4 | Icosphere.obj | 80 |  |  |
| 5 | Gourd.obj | 648 |  |  |

| | | | |
|---|---|---|---|
| 6 | Magnolia.obj | 1372 |  |
| 7 | Unicorn.obj | 1572 |  |
| 8 | Monkey.obj | 968 |  |
| 9 | Dolphin.obj | 1692 |  |
| 10 | Skull.obj | 7120 |  |

## KESIMPULAN

Dari hasil percobaan, dapat disimpulkan bahwa hasil *ray tracing* dengan menggunakan normal b-splines lebih halus dibandingkan dengan ray tracing biasa. Sementara jika dibandingkan dengan sub-division, hasilnya juga tidak terlalu berbeda. Dari aspek lama proses rendering, semakin banyak jumlah face dari sebuah mesh, maka peningkatan waktu proses rendering juga semakin meningkat. Namun demikian, peningkatannya tidak berbanding lurus dengan peningkatan jumlah mesh.

Desain program yang dilakukan dengan cara membagi proses perhitungan, proses sebelum dan proses pada saat rendering sudah berhasil menekan peningkatan lama proses rendering.

## DAFTAR PUSTAKA


1. Benthin, C., Boulos, S., Lacewell, D. and Wald, I. *Packet-based Ray Tracing of Catmull Clark Subdivision Surfaces*.
2. Liliana. *Memperhalus Permukaan Obyek Mesh yang Dirender Dengan Menggunakan Ray Tracing*. Prosiding Seminar Nasional Transformasi Teknologi Untuk Peningkatan Kualitas Hidup Manusia Universitas Teknologi Yogyakarta. 16 Desember 2006.
3. Langyel, E. *Mathematics for 3D game Programming and Computer Graphisc 2nd*. 2004. Charles River Media, INC. Boston, Massachusett.
4. Andersson, F. *Bezier and B-Splines Technology*. 2003. Thesis report.
5. Arvo, J. *Backward Ray Tracing*. 1986. Apollo Computer, Inc. Chelmsfort, MA didownload dari situs: http://graphics.stanford.edu/ courses/cs348b-97/basics/intersection/slides/.
6. Shirley, P. and Marschner, S. *Fundamentals of Computer Graphics 3rd*. 2010. Taylor and Francis Group, LLC
7. Whitted. *Basics of Ray Tracing*. 1997. Lecture notes for Spring Quarter
8. Lawrence, J. 2011. Barycentric Coordinates (and Some Texture Mapping) [PowerPoint Slide]. Didownload dari: http://www.mpi-inf.mpg.de/departments/irg3/ws0506/cg/slides/12-supersamplingexercise s.pdf.
9. B-spline Surfaces: Construction. Didownload dari: http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/surface/bspline-construct.html, 23Oktober 2012.