LAPORAN PENELITIAN PAK

Pengembangan Aplikasi Untuk Memperhalus Penampilan Permukaan Obyek *Mesh* 3D pada Metode *Rendering Ray Tracing*

Oleh:

1. **Liliana, M.Eng**
2. **Gregorius Satia Budhi**

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS KRISTEN PETRA
JULI 2012

HALAMAN PENGESAHAN LAPORAN AKHIR

1	Judul Penelitian	Pengembangan Aplikasi Untuk Memperhalus Penampilan Permukaan Obyek *Mesh* 3D pada Metode *Rendering Ray Tracing*
2	Ketua Peneliti	
	a. Nama Lengkap	Liliana
	b. Jenis Kelamin	Perempuan
	c. NIP	03-024
	d. Jabatan Fungsional	Lektor
	e. Prodi/fakultas/pusat studi	Teknik Informatika/ teknologi Industri
3	Alamat Ketua Peneliti	
	a. Alamat kantor (telp/fax/E-mail)	Siwalankerto 121-131 Surabaya (62+312983455)
	b. Alamat rumah (telp/fax/E-mail)	Wiguna timur II /16 Surabaya (62+8123149023/lilian@petra.ac.id)
4	Jumlah Anggota Peneliti	1
	a. Nama Anggota Peneliti 1	Gregorius Satia Budhi
5	Lokasi Penelitian	UK Petra
6	Kerjasama dengan institusi lain	-
7	Jangka Waktu Penelitian	1 tahun
8	Biaya Penelitian	
	a. Sumber dari UK Petra	Rp 9.000.000,00
	b. Sumber lainnya	-
	Total	Rp 9.000.000,00

Surabaya, 8 Juli 2013

Mengetahui,

Ketua Peneliti,

Dekan Fakultas
(Djoni Haryadi Setiabudi, M.Eng)
NIP: 85009

(Liliana, M.Eng)
NIP: 03024

Menyetujui:
Kepala LPPM-UK Petra

(Prof. Ir. Lilianny Sigit Arifin, M.Sc., Ph.D.)
NIP: 84011

RINGKASAN DAN SUMMARY

Seiring berkembangnya teknologi, hampir semua *game* dan animasi berpindah dari berbasis 2D menjadi 3D. *Ray tracing* merupakan metode *rendering* untuk menghasilkan simulasi tampilan 2D dari objek 3D. metode rendering *Ray tracing* memperhitungkan efek pencahayaan dan efek optik pada setiap *face* objek *mesh.* Perhitungan warna pada mesh ditentukan oleh posisi normal bidang terhadap sumber cahaya*.* Detail perhitungan yang ada menyebabkan proses perhitungan warna pada sebuah *face* berlangsung lama. Lamanya proses rendering dipengaruhi oleh banyaknya jumlah face pada sebuah obyek mesh. Untuk mempercepat proses rendering seringkali digunakan obyek mesh dengan jumlah face sesedikit mungkin. Hal ini menghasilkan gradasi warna permukaan obyek mesh yang kasar karena setiap face menghasilkan warna permukaan yang sama.

Untuk menyelesaikan permasalahan tersebut, digunakan perhitungan normal secara proporsi yang mengadopsi konsep *barycentric coordinate* dan kurva b-splines. *Barycentric normal* memanfaatkan interpolasi vektor normal dari masing-masing vector titik pembentuk segitiga. Hal ini bertujuan untuk mendapatkan vektor normal yang bervariasi di dalam sebuah *face*. Proses perhitungan normal ini dilakukan sebelum proses rendering dimulai. Dengan demikian, lama proses rendering tidak meningkat banyak.

Pengujian dilakukan dengan membandingkan hasil penggunaan barycentric normal, dan normal b-splines terhadap metode *simple subdivision* dan *ray tracing* biasa*.* Dari hasil percobaan, dapat disimpulkan bahwa hasil ray tracing dengan menggunakan *barycentric normal* lebih halus dibanding dengan normal b-splines. Sementara jika dibandingkan dengan subdivision, 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.

Kata kunci: *ray tracing, rendering, b-spines surface, smooth surface, mesh object, barycentric normal*

PRAKATA

Merupakan suatu kasih karunia dari Tuhan Yesus Kristus jika kami berhasil menyelesaikan penelitian ini. Ditengah begitu banyak kesibukan dan keterbatasan kami, penyertaan dan pengetahuan yang dikaruniakanNya kepada kami yang memungkinkan kami mewujudkan penelitian ini.

Kami juga berterima kasih kepada Universitas Kristen Petra yang sudah memberi kami kesempatan dalam bentuk dukungan dana untuk bisa menyelesaikan penelitian ini. Kepada segenap rekan di Lembaga Penelitian dan Pengabdia kepada Masyarakat yang telah mendorong dan menyemangati serta membantu dalam segala proses yang ada.

Kami juga berterima ksih kepada mahasiswa-mahasiswa dari program studi teknik informatika yang sudah membantu kami dalam membuat program implementasi, Ferdi Atmaja Wong Susilo, Erick Leonardo dan Evans Sanjaya. mereka yang tak kenal lelah saat percobaan-percobaan dan implementasi persamaan kami gagal.

Terima kasih juga kepada segenap rekan dosen di program studi informatika yang bersama dengan kami membuat penelitian mereka masing-masing. Kehadiran mereka dan semangat mereka yang terus memotivasi kami dan kebersamaan saat menyelesaikan penelitian ini.

Terima kasih kepada pihak-pihak yang lain, yang tidak dapat kami sebutkan satu persatu.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1. PENDAHULUAN

1.1. Latar Belakang

Dalam grafika komputer, sebuah obyek 3D dimodelkan dengan menggunakan persamaan matematika 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.

Salah satu metode rendering yang banyak digunakan pada aplikasi komersial adalah *ray tracing* karena mampu 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.

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 kaku maka biasanya digunakan ukuran segitiga yang lebih 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 segitiga akan diwarnai dengan teknik blending berdasarkan warna pada ketiga titik tersebut. Namun hasil dari penelitian ini masih belum memuaskan.

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 hibah penelitian yang diajukan ini, akan dilakukan implementasi dari kedua metode yang diusulkan, serta membandingkan hasil keduanya dengan penelitian

sebelumnya. Perbandingan akan dilakukan pada aspek kecepatan proses rendering dan kehalusan gradasi warna yang terbentuk.

1.2. Tujuan

Tujuan penelitian ini adalah mengimplementasikan dan menguji dua buah metode yang diusulkan untuk menampilkan permukaan obyek mesh 3D dengan gradasi warna yang halus namun mempunyai proses rendering yang relatif cepat. Dua metode yang akan diimplementasikan dan diuji adalah perhitungan normal bidang secara proporsi dari tiga titik pembentuk segitiga dan perhitungan normal dengan permukaan b-spline virtual.

1.3. Ruang Lingkup

Ruang lingkup dari penelitian ini adalah sebagai berikut:

- Input berupa obyek 3D yang digunakan adalah obyek mesh.
- Metode yang digunakan untuk menampilkan permukaan obyek dengan gradasi warna yang halus adalah dengan melakukan perhitungan normal pada setiap titik permukaan obyek secara proporsi dari tiga titik pembentuk segitiga dan perhitungan normal dengan permukaan b-spline virtual.
- Output berupa tampilan dari obyek mesh 3D dengan adanya gradasi warna yang halus pada permukaannya.

1.4. Urgensi Penelitian

Proses rendering merupakan proses utama dalam mensimulasikan obyek 3D. salah satu metode yang diunggulkan adalah ray tracing. Ray tracing mampu menghasilkan obyek yang mendekati aslinya namun membutuhkan waktu proses yang lama. Untuk mengurangi waktu proses, salah satunya adalah dengan mengurangi jumlah mesh pada obyek 3D. mengurangi jumlah mesh akan mengakibatkan permukaan yang lebih luas dan mengakibatkan pewarnaan yang tidak halus pada permukaan tersebut. Pewarnaan yang tidak halus akan menyebabkan obyek tidak terlihat alami lagi.

Dengan keberhasilan metode yang mampu menghasilkan gradasi warna yang halus pada permukaan mesh, maka kealamian obyek bisa didapatkan dengan tidak menambah jumlah mesh.

Luaran dari penelitian ini adalah potensi pengajuan HAKI jika metode terbukti cukup efektif dari segi waktu proses rendering dan hasilnya memuaskan dalam arti, gradasi warna tampak alami.

BAB 2. STUDI PUSTAKA

4.

2.1. Ray Tracing

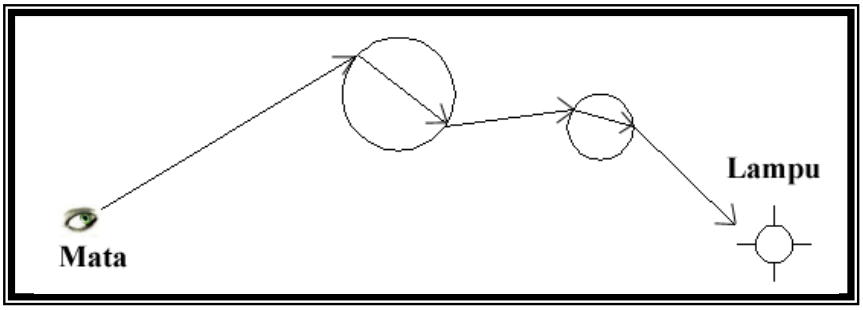
Ray tracing adalah suatu metode untuk menggambar objek 3D yang hasilnya realistik, sama seperti foto[6]. Metode ini dilakukan dengan cara menelusuri sinar, kemudian dilakukan pengecekan apakah sinar tersebut mengenai obyek atau tidak. Jika ternyata sinar yang ditelusuri tersebut mengenai suatu obyek maka selanjutnya diperhitungkan intensitas pada obyek tersebut. Hasil dari perhitungan intensitas inilah yang terlihat oleh mata.

Metode ray tracing dibagi menjadi dua jenis, yaitu forward ray tracing dan backward ray tracing. Kelemahan dalam penggunaan metode forward ray tracing adalah waktu yang dibutuhkan sangat banyak karena jumlah sinar yang dipancarkan oleh lampu juga sangat banyak, padahal tidak semua sinar mengenai mata. Namun keunggulan dari metode ini adalah kemampuan untuk mendapatkan intensitas obyek yang banyak dikenai sinar dari lampu, yang jika ditelusuri dari mata, hanya satu sinar yang mampu ditangkap oleh mata, sementara sinar – sinar yang lainnya tidak dideteksi. Dengan metode ini bisa didapatkan efek caustik dari obyek – obyek yang transparan. Sedangkan pada backward ray tracing, sinar yang ditelusuri adalah sinar yang menuju ke mata, dengan cara ini waktu yang dibutuhkan relatif lebih sedikit karena hanya sinar – sinar yang pasti mengenai mata saja yang ditelusuri.

Selain itu ada beberapa hal penting dalam ray tracing yang harus diperhatikan, seperti mencari waktu tabrakan dengan obyek, efek - efek pencahayaan dan efek - efek optik yang dihasilkan.

2.1.1. Backward Ray Tracing

Backward ray tracing mengunakan penelusuran sinar dari mata[6]. Sinar dipancarkan dari mata ke arah setiap pixel yang membentuk layar gambar dan kemudian diteruskan ke obyek – obyek yang akan digambar, seperti terlihat pada gambar 1. Jika sinar yang melalui suatu pixel tersebut mengenai sutu obyek maka dilakukan perhitungan intensitas pada titik tabrak obyek tersebut. Intensitas hasil perhitungan tersebut digunakan untuk memberi warna pada pixel tersebut. Perhitungan intensitas yang dilakukan adalah dengan memperhitungkan efek pencahayaan dan efek optik. Jika sinar yang dipancarkan tersebut tidak mengenai obyek sama sekali maka pixel diberi warna sama dengan warna latar belakangnya.



Gambar 1. Proses alur sinar pada backward ray tracing

Untuk memulai perhitungan dengan backward ray tracing maka pertama-tama adalah dengan mencari arah dari mata ke pixel – pixel pembentuk layar gambar. Untuk mencari vektor tersebut digunakan P0 sebagai vektor acuan, yaitu vektor dari mata ke pixel pertama dari kiri atas. Baru setelah itu dapat dicari vektor untuk pixel – pixel yang lain dengan menambahkan atau mengurangkan dengan vektor acuan ke arah horisontal dan arah vertikalnya, sesuai dengan urutan pixel tersebut. Untuk mendapatkan arah vektor ke pixel pertama digunakan persamaan 1[6].

$$P0 = c + (S / (2 * pix\ x)) * x – (S / (2 * pix\ y)) * u \quad (1)$$

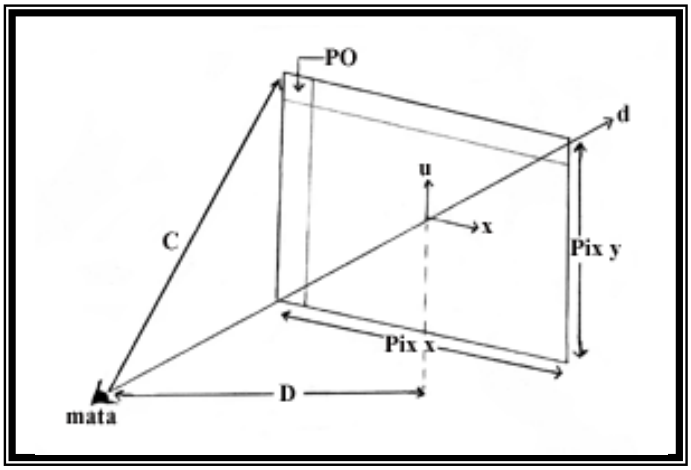
Dimana pix x adalah lebar layar gambar sedangkan pix y adalah tinggi layar gambar. S adalah ukuran layar kamera. Vektor u adalah vektor yang sejajar dengan arah atas kamera. Vektor x adalah hasil cross antara vektor u dengan vektor arah pandang mata (vektor d , merupakan unit vektor).dari hasil cross tersebut akan didapatkan vektor acuan untuk arah horisontal. Vektor c merupakan vektor arah ke ujung kiri atas layar gambar, didapatkan dengan persamaan 2[6]:

$$c = D * d – (S / 2) * x + (S / 2) * u \quad (2)$$

dimana D adalah jarak antara mata dengan layar gambar, yang mana hal ini ditentukan oleh sudut pandang mata, yang dapat dicari dengan menggunakan persamaan 3[6].

$$D = S / 2 / \tan (FOV * \pi / 180) \quad (3)$$

Dimana FOV adalah sudut pandang mata yang dinyatakan dalam satuan derajat. Untuk lebih jelas dalam mencari vektor – vektor tersebut, perhatikan gambar 2



Gambar 2. Menentukan vektor pada setiap pixel

2.1.2. Mencari Waktu Tabrakan Sinar dengan Obyek

Selanjutnya untuk mencari titik tabrak antara sinar dengan suatu obyek maka perlu dicari terlebih dahulu waktu yang diperlukan sinar dari titik asal sampai menabrak obyek tersebut. Untuk mendapatkan waktu tabrakan tersebut adalah dengan cara mensubstitusikan persamaan sinar dengan persamaan yang membentuk obyek tersebut. Maka cara mencari titik tabrak untuk setiap obyek berbeda. Persamaan sinar adalah persamaan 4 berikut[6]:

$$P(t) = S + dir * t \quad (4)$$

Dimana :
- S = posisi asal sinar
- dir = arah sinar
- t = waktu yang dibutuhkan sinar untuk mencapai titik tabrak
- P(t) = titik tabrak

- Bola

Objek bola dengan titik pusat di C (C_x, C_y, C_z) dan radius = r satuan panjang memiliki persamaan geometri pada persaman 5 [6] sebagai berikut :

$$(Px – C_x)^2 + (Py – C_y)^2 + (Pz – C_z)^2 – r^2 = 0 \quad (5)$$

dimana P adalah semua titik yang berada di permukaan bola. Untuk mencari waktu tabrakan, subtitusikan persamaan (5) tersebut dengan persamaan sinar (4) untuk mendapatkan persamaan kuadrad sebagai berikut :

$$(Sx + dir\ x * t – Cx)^2 + (Sy + dir\ y * t – Cy)^2 + (Sz + dir\ z * t – Cz)^2 – R^2 = 0$$

$(dir\ x^2 + dir\ y^2 + dir\ z^2)\ t^2 + (2.(Sx – Cx).dir\ x + 2.(Sy – Cy).dir\ y + 2.(Sz – Cz).\quad dir\ z)t + ((Sx – Cx)^2 + ((Sy – Cy)^2 + (Sz - Cz)^2 – R^2) = 0$

Selanjutnya dari persamaan hasil subtitusi tersebut dapat dirumuskan menjadi bentuk umum:

$$Ax^2 + Bx + C = 0$$

di mana :

$A = dir \bullet dir$

4.1. $B = 2 * (S – C) * dir$

$C = (S – C)^2 - r^2$

Dari nilai ketiga komponen A, B dan C tersebut selanjutnya dapat dicari akar kuadrat t dengan persamaan 6[6]

$$t_{1,2} = \frac{-B \pm \sqrt{D}}{2A} \qquad (6)$$

di mana nilai D adalah: $D = B^2 – 4AC$

Dengan ketentuan :

$D < 0 \rightarrow$ sinar tidak menabrak bola.

$D = 0 \rightarrow$ sinar menabrak bola di satu titik

$D > 0 \rightarrow$ sinar menembus bola di dua titik (pilih t positif yang terkecil)

Arah normal bola dapat diperoleh dengan mengurangkan titik tabrakan pada bola dengan titik pusat bola. Sedangkan untuk mencari titik tabrakan (P(t)), substitusikan waktu tabrakan (t) yang telah telah diperoleh ke persamaan sinar (4).

- Bidang Datar

  Bidang datar adalah suatu bentuk geometri dengan persamaan[6] :

$$ax + by + cz = d \qquad (7)$$

dimana a , b , c merupakan bilangan arahnya yang diambil dari nilai x , y , z dari vektor normal bidang dan x, y, z adalah komponen vektor dari sutu titik yang terletak pada bidang.

persamaan 7 dapat juga dituliskan :

$$N \bullet P = d$$

Dengan mensubsitusikan persamaan diatas dengan persamaan 4 maka didapat persamaan baru untuk mencari waktu tabrakan sebagai berikut:

$$N * (S + t * Dir) = d$$

$$N * S + t * N \bullet Dir = d$$

Sehingga $t = (d – N * S) / (N \bullet Dir)$ (8)

Dimana :

t = waktu tabrakan

d = jarak bidang dengan bidang koordinat

N = normal bidang

S = titik asal sinar yang menabrak

Dir = arah sinar

2.1.3. Pencahayaan pada Ray Tracing

Untuk memperoleh gambar yang semirip mungkin dengan aslinya, perlu ditambahkan pencahayaan, karena pada dunia nyata pun semua benda dapat terlihat karena adanya cahaya. Pencahayaan dapat dibedakan menjadi tiga, yaitu ambient, diffuse dan specular[6]. Di bawah ini akan dijelaskan secara lebih rinci tentang ketiga efek pencahayaan tersebut.

- Ambient

Ambient adalah efek pencahayaan yang telah membaur dengan lingkungan sehingga arah cahaya tidak dapat diketahui, seakan-akan cahaya datang dari segala arah. Efek ini akan mempengaruhi terang atau tidaknya suatu lingkungan yang terlihat oleh mata. Contoh dari ambient adalah matahari yang bersinar pada saat langit berawan.

Intensitas ambient pada suatu objek dapat dicari dengan persamaan 9[6] :

$$I = Ia * Ka \qquad (9)$$

dimana, I = Intensitas yang dihasilkan
Ia = Intensitas ambient
Ka = Koefisien ambient

- Diffuse

Jenis pencahayaan yang kedua ialah diffuse. Diffuse adalah pencahayaan yang tergantung dari besarnya sudut yang dibentuk antara sinar dari lampu ke titik tabrak pada obyek dengan normal obyek, seperti terlihat pada. Sehingga posisi lampu sangat mempengaruhi efek diffuse ini. Intensitas diffuse dapat dicari dengan hukum Lambertian sebagai berikut[6] :

$$I = \frac{Ip * Kd (\cos \varnothing)}{d} \quad (10)$$

Dari persamaan intensitas diffuse tersebut cos∅ dapat dihitung dengan melakukan dot product antara sinar dari lampu ke titik tabrak obyek dengan normal obyek itu, masing – masing merupakan unit vektor. Sehingga didapat persamaan baru yaitu persaman 11[6].

$$I = \frac{Id * Kd * (L \bullet N)}{d} \quad (11)$$

dimana, I = Intensitas yang dihasilkan
Id = Intensitas diffuse dari sumber cahaya ‘x’
Kd = Koefisien diffuse
N = Vektor normal dari objek
L = Vektor dari titik tabrak ke sumber cahaya
∅ = Sudut antara N dengan L
d = Jarak antara sumber cahaya dengan titik tabrak

- Specular

Specular adalah efek pencahayaan dimana bayangan sumber cahaya terlihat pada permukaan obyek[6]. Efek specular terlihat pada obyek yang mengkilap. Semakin mengkilap permukaan suatu obyek maka makin jelas bayangan sumber cahaya yang terlihat pada permukaan obyek tersebut. Untuk mencari intensitas specular maka dapat digunakan persamaan 12 sebagai berikut:

$$I = \frac{Ip * Ks (\cos \varnothing)^n}{d} \quad (12)$$

Dari persamaan intensitas specular tersebut cos∅ dapat dihitung dengan melakukan dot product antara arah pantulan dengan negasi dari arah sinar.

$$I = \frac{Ip * Ks * (R \bullet V)^n}{d} \quad (13)$$

dimana I = Intensitas yang dihasilkan
Ip = Intensitas specular dari sumber cahaya ‘x’
Ks = Koefisien specular
n = Variabel yang menentukan luas area yang berkilau jika terkena cahaya yang dipancarkan oleh sumber cahaya (bila n semakin besar maka cahaya semakin terfokus atau area yang berkilau menjadi lebih kecil)
R = arah pantulan
V = negasi dari arah sinar
d = Jarak antara sumber cahaya dengan titik tabrak pada obyek

Sedangkan vektor R diperoleh dari -S + 2 * (S • N) * N, dimana :
S = Vektor dari titik tabrak ke sumber cahaya
N = Vektor normal dari objek

2.1.4. Efek – Efek Visual pada Ray Tracing

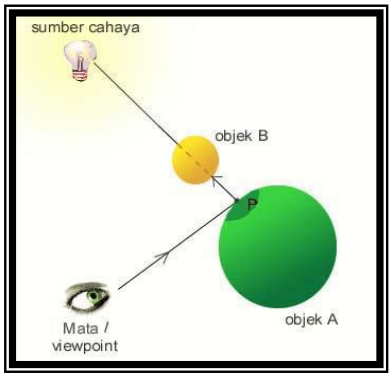
Efek – efek visual yang terjadi pada ray tracing dihasilkan dari sinar generasi kedua, dalam arti sinar yang dipancarkan untuk kemudian ditelusuri kembali sampai akhirnya mendapatkan efek – efek tersebut bukan sinar yang dipancarkan dari sumber cahaya atau dari mata. Yang termasuk sinar generasi kedua antara lain sinar hasil pembiasan dan pemantulan. Dengan menggunakan perhitungan dari sinar – sinar generasi kedua ini maka akan didapatkan efek bayangan suatu obyek, reflektifitas dan transparansi. Efek – efek tersebut akan dijelaskan lebih lanjut di bawah ini:

- Bayangan

Bayangan terjadi jika ada benda lain yang menghalangi sinar langsung dari sumber cahaya. Untuk mengecek apakah suatu obyek terkena bayangan dari obyek lain, maka telusuri kembali sinar dari titik tabrak pada obyek dengan setiap sumber cahaya yang ada. Jika sinar tersebut mengenai obyek lain maka obyek yang bersangkutan (pada titik tersebut) berada di daerah bayangan. Untuk mengetahui lebih jelas lagi mengenai posisi suatu titik yang berada pada daerah bayangan dapat melihat pada gambar 2.8 Sedangkan untuk mencari sinar yang akan dipancarkan kembali dapat digunakan rumus sebagai berikut[6]:

L = posisi sumber cahaya – titik tabrak pada obyek (14)

Dimana L adalah sinar yang akan dipantulkan kembali.



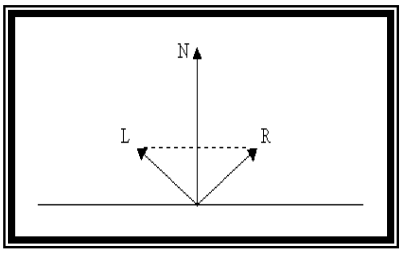
Gambar 3. Bayangan objek B pada objek A

Bila arah langsung suatu titik ke suatu sumber cahaya terhalang obyek lain maka dari sumber cahaya tersebut tidak diperhitungkan efek pencahayaan diffuse dan specularnya kerena untuk menghitung efek ini posisi lampu sangat berperan.

- Reflektifitas

Reflektifitas terjadi pada obyek yang mempunyai kemampuan untuk memantulkan sinar (reflektif). Efek yang terlihat adalah tampaknya obyek lain pada obyek yang reflektif. Untuk mendapatkan efek reflektifitas maka sinar yang mengenai suatu obyek reflektif akan dipantulkan untuk kemudian ditelusuri kembali apakah sinar pantul tersebut mengenai obyek yang lain. Jika sinar tersebut mengenai obyek lain, maka tambahkan intensitas hasil perhitungan efek pencahayaan pada titik tabrak tersebut, yang sudah dikalikan dengan koefisien reflektifitasnya ke intensitas pada obyek reflektif.

Proses yang terjadi adalah sebagai berikut, sinar yang datang mengenai suatu permukaan obyek akan dipantulkan dengan sudut yang sama besar terhadap normal pada bidang yang dikenai. Gambaran dari sinar dating dan sinar pantul dapat dilihat pada gambar 1. Secara sederhana dapat dikatakan sudut antara $\vec{L}$ dengan $\vec{N}$ sama besar dengan sudut antara $\vec{N}$ dengan $\vec{R}$.



Gambar 4. Pembentukan sinar pantul

Untuk mencari arah sinar pantul dapat dirumuskan [7] seperti pada persamaan 15.

$$\vec{R} = 2 * N * (N \bullet L) - L \quad (15)$$

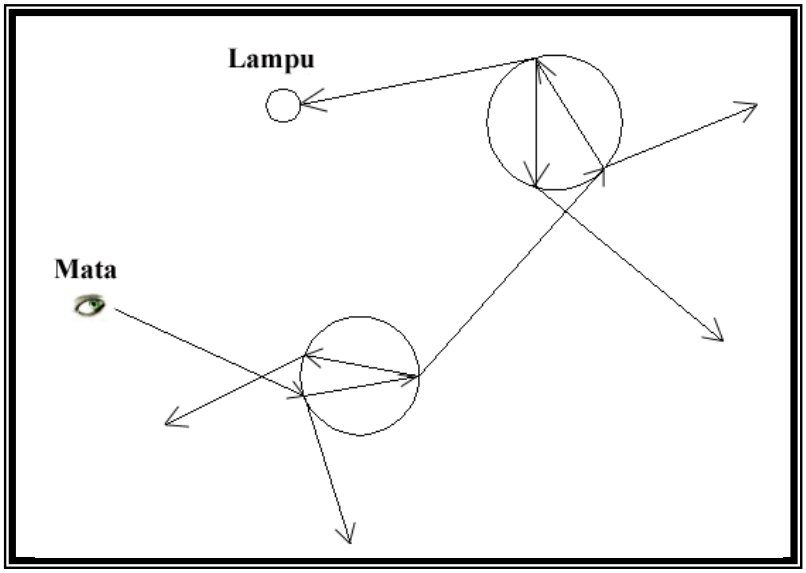
Dinama :

$\vec{R}$ **= sinar pantul**

$\vec{N}$ **= normal**

$\vec{L}$ **= negasi dari sinar datang**

Proses akan terulang kembali jika obyek yang terkena sinar pantulan tersebut juga reflektif. Proses akan berhenti jika sinar tidak lagi mengenai obyek reflektif atau tidak mengenai obyek sama sekali. Sedangkan untuk memperjelas proses pemantulan dapat melihat pada gambar 5.



Gambar 5. Gambaran mengenai reflektifitas

2.2. B-Splines

Kurva *B-Splines* yang merupakan perkembangan dari kurva *Bezier.* Kurva *B-splines* terbentuk dari gabungan segmen/bagian kurva dari sekumpulan knot. Kurva *B-Splines* lebih fleksibel karena perubahan yang dilakukan pada knot tertentu hanya akan mempengaruhi bentuk kurva pada segmen didekat knot tersebut, tergantung dari input user. Hal ini dikarenakan segmen kurva pada *B-Splines* hanya dipengaruhi oleh beberapa titik kontrol saja, bukan keseluruhan titik knot seperti halnya pada kurva *Bezier*.

Untuk membatasi jumlah knot yang akan mempengaruhi pembentukan kurva, maka akan ditentukan derajat dan untuk menentukan seberapa besar pengaruh knot tertentu pada bentuk kurva yang dihasilkan, maka setiap knot/titik diberi waktu. Knot-knot yang waktunya saling intersection saja yang akan mempengaruhi bentuk kurva pada segmen tersebut.

Bila terdapat n buah knot, maka C_t dapat dihasilkan melalui fungsi yang kontinu C(t) pada persamaan 16.

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P_i}\, N_{i,p}(t) \tag{16}$$

dimana:

P_i = Titik control ke i
p = derajat
N = *blending function* atau basis.

N(t) disebut sebagai *blending functions* yang dijabarkan sebagaimana pada persamaan 17.

$$N_{i,0}(t) = \begin{cases} 1 & \text{bila } t_i \le t < t_{i+1} \\ 0 & \text{sebaliknya} \end{cases}$$

$$N_{i,p}(t) = \frac{t - u_i}{u_{i-p} - u_i} N_{i,p-1}(t) + \frac{u_{i+p+1} - t}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(t) \tag{17}$$

dimana :

u_i = knot vektor (akan dijelaskan pada subbab selanjutnya)

Beberapa elemen penting pada pembentukan kurva *B-Splines* adalah :

- Derajat (p) - mengatur seberapa dekat kurva tersebut melewati titik kontrol dari kurva *B-Splines*. Semakin kecil derajat dari kurva *B-Splines* tersebut, maka semakin dekat pula kurva tersebut akan melewati titik-titik kontrol pembentuknya, dan sebaliknya apabila derajat kurva tersebut semakin besar, maka jarak titik kontrol kurva dengan kurva akan semakin jauh.
- *Blending function* atau *basis function* (N) – merupakan fungsi yang menentukan seberapa besar lengkungan dari kurva *B-Splines*, yang dipengaruhi oleh besarnya derajat, knot vektor dan t.

2.3. Barycentric Normal

Barycentric normal dihitung berdasarkan interpolasi barycentric. Interpolasi barycentric sangat efektif digunakan untuk mencari semua titik pada permukaan sebuah grid. Bentuk grid yang paling sederhana adalah segitiga. Interpolasi barycentric mencari proporsi bobot dari sembarang titik pada sebuah grid berdasarkan titik-titik pembentuk grid. Persamaan untuk mencari interpolasi barycentric dapat dilihat pada persamaan 18. dari persamaan 18, dikembangkan sebuah interpolasi untuk mencari normal secara proporsi di seluruh permukaan grid. Mengadopsi persamaan 18, maka digunakan interpolasi barycentric dari normal pada setiap titik pembentuk grid untuk mencari normal di seluruh permukaan grid. Persamaan untuk mencari barycentric normal dapat dilihat pada persamaan 19.

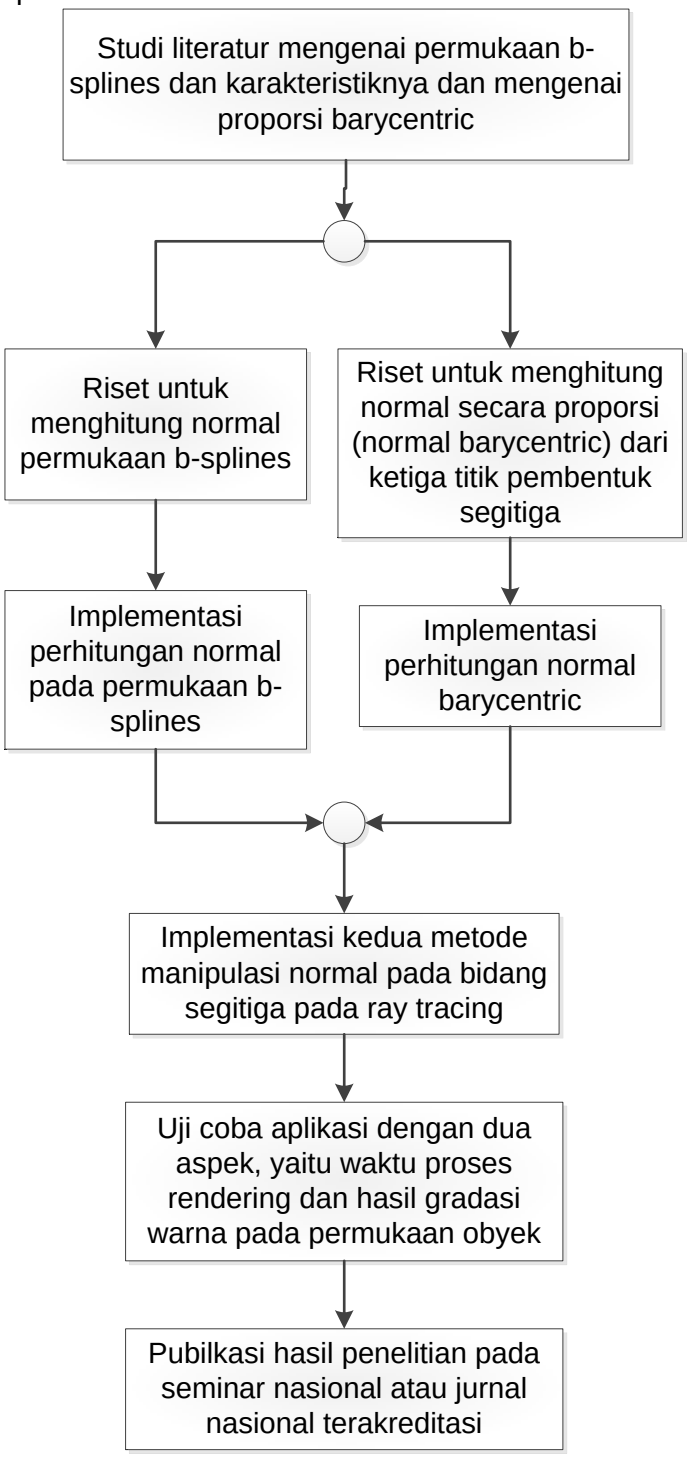
$$x = \sum_{i=1}^{n} \alpha_i \times x_i \quad , \alpha_i > 0 \text{ dan } \sum_{i=0}^{n} \alpha_i = 1 \tag{18}$$

$$N = \sum_{i=1}^{n} \alpha_i \times N_i \quad , \alpha_i > 0 \text{ dan } \sum_{i=0}^{n} \alpha_i = 1 \tag{19}$$

Dimana x adalah sembarang titik pada permukaan grid, α adalah bobot dari masing-masing x atau N. N adalah normal pada sebuah titik. Normal pada titik adalah rata-rata dari normal bidang yang bertumpu pada titik tersebut. n adalah banyaknya titik dari sebuah grid. Semakin dekat sebuah titik sembarang dari sebuah titik pembentuk grid,α akan bernilai semakin besar.

BAB 3. METODE PENELITIAN

Langkah-langkah yang akan dilakukan pada penelitian ini adalah seperti terlihat pada bagan alir penelitian pada Gambar 6.



Gambar 6. Bagan Alir Penelitian

Penjelasan lebih rinci dari bagan alir pada Gambar 5 adalah sebagai berikut:

- Langkah pertama adalah studi literatur untuk mempelajari pembentukan permukaan b-splines dan karakteristiknya. Metode b-splines, adalah metode untuk menghasilkan kurva ataupun permukaan. Permukaan yang terbentuk akan dibuat menjadi mesh. Pada literatur mengenai b-splines pada umumnya, tidak dibahas mengenai perhitungan normal bidang yang terbentuk.
- Langkah kedua adalah melakukan riset untuk menghitung normal pada permukaan b-splines dan perhitungan normal dengan proporsi dari normal titik dari ketiga titik pembentuk segitiga.
- Langkah ketiga adalah mengimplementasikan perhitungan normal pada permukaan obyek dan perhitungan normal dengan proporsi. Implementasi akan dilakukan setelah riset untuk menghitung normal seesai dilakukan. Diimplementasikan satu persatu. Perhitungan normal secara proporsi akan dilakukan paralel dengan implementasi perhitungan normal pada permukaan b-splines.
- Langkah keempat adalah menggabungkan implementasi perhitungan normal bidang pada ray tracing. Perhitungan normal pada langkah ketiga akan disubstitusikan ke bagian penghitungan warna yang melibatkan normal bidang.
- Langkah kelima adalah melakukan uji aplikasi dalam dua aspek, yaitu waktu yang dibutuhkan untuk proses rendering dibandingkan dengan jika dilakukan subdivision dan aspek keberhasilan menamplkan gradasi warna pada permukaan obyek dibandingkan dengan hasil rendering tanpa rekayasa normal bidang.
- Langkah terakhir adalah melakukan dokumentasi hasil penelitian serta publikasi pada seminar / jurnal nasional.

BAB 4. HASIL 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

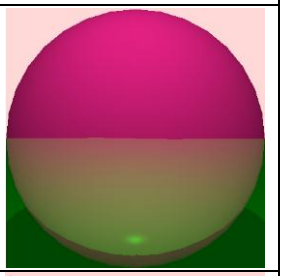
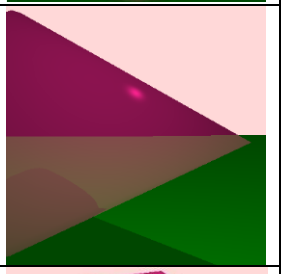
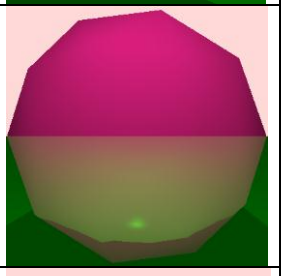
Pengujian dilakukan dengan menggunakan obyek primitif (obyek sederhana) dan obyek kompleks. Perbandingan dilakukan terhadap proses raytracing tanpa metode penghalusan, dengan menggunakan kurva b-splines dan dengan menggunakan perhitungan barycentric normal. Aspek yang diperbandingkan adalah lama proses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

Aspek yang mempengaruhi lama proses rendering pada ray tracing adalah resolusi screen yang akan dihasilkan, jumlah obyek yang direnderr, dalah hal ini adalah jumlah face, posisi kamera yang akan mempengaruhi presentase visualisasi obyek terhadap resolusi screen. Berdasarkan aspek-aspek tersebut maka uji coba Ray Tracing dilakukan dengan konfigurasi sebagai berikut:

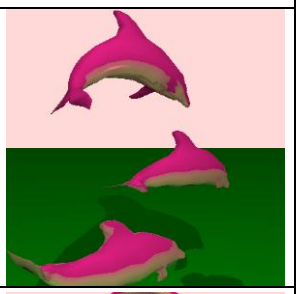
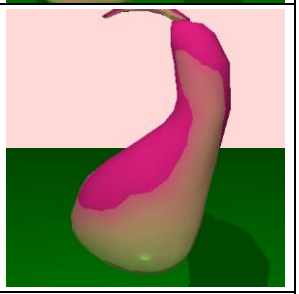
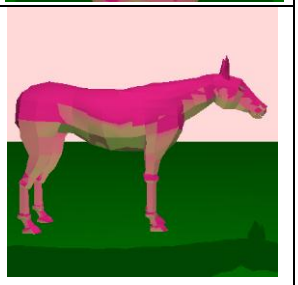
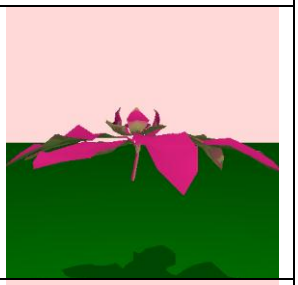
Screen size: 400 x 400 pixel
Camera
0 1 0
0 0 300
0 0 -1
FOV = 45
Screen
s = 1
TotalLight = 1
Ambient & Diffuse & Specular
Ia = 0.8
Id = 0.8
Is = 0.8
LightPosition = 0 150 200
TotalObject = 2
M
Ka = 0.8
Kd = 0.2
Ks = 0.8
P
Normal = 0 1 0
d = -400
Ka = 0.7
Kd = 0.2
Ks = 0.8

Pengujian pertama dilakukan dengan menggunakan perhitungan *barycentric normal* terhadap obyek primitif. 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 1.

Tabel1. Ray Tracing dengan perhitungan Barycentric Normal pada obyek primitif

No	Objek	Jumlah Vertex	Jumlah Face	Lama proses Rendering	Memory	Hasil
1	Ball.3ds	482	960	0(h) : 3(m) : 31(s) : 65(ms)	110 KB	
2	Cone.obj	33	62	0(h) : 0(m) : 22(s) : 29(ms)	80 KB	
3	Icosphere.obj	42	80	0(h) : 0(m) : 30(s) : 33(ms)	154 KB	
4	Torus.obj	576	1152	0(h) : 3(m) : 38(s) : 73(ms)	174 KB	

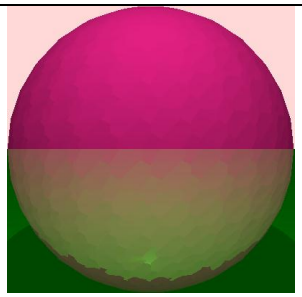
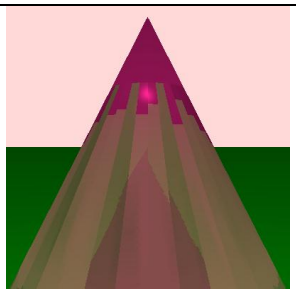
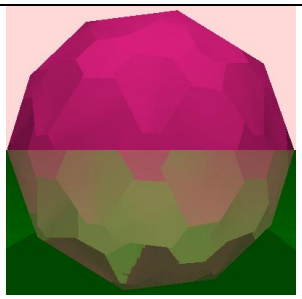
Tabel 2. Ray Tracing dengan perhitungan Barycentric Normal pada obyek kompleks

No	Objek	Jumlah Vertex	Jumlah Face	Lama proses Rendering	Memory	Hasil
1	Dolphin.obj	855	1692	0(h) : 5(m) : 55(s) : 14(ms)	138 KB	
2	Gourd.obj	326	648	0(h) : 2(m) : 21(s) : 10(ms)	102 KB	
3	Skull.obj	6624	7120	0(h) : 29(m) : 40(s) : 10(ms)	183 KB	
4	Unicorn.obj	2356	1572	0(h) : 5(m) : 56(s) : 84(ms)	134 KB	
5	Magnolia.obj	1184	1372	0(h) : 4(m) : 50(s) : 36(ms)	146 KB	
6	Monkey.obj	505	968	0(h) : 3(m) : 41(s) : 40(ms)	162 KB	

Dari hasil pada tabel 2, sebagian besar obyek berhasil direnderer dengan halus. Hanya pada obyek “unicorn.obj” hasil kurang maksimal. Hal ini disebabkan oleh jumlah face yang relatif sedikit dibandingkan dengan ukuran obyek.

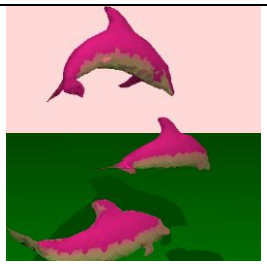
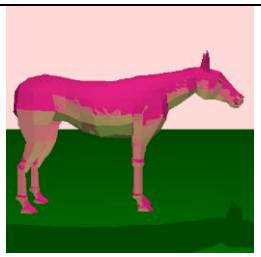
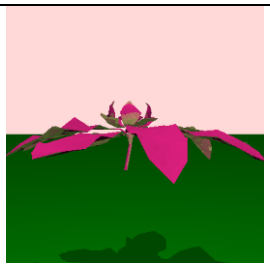
Percobaan berikutnya dilakukan dengan menggunakan perhitungan normal yang dihitung dengan menggunakan kurva b-splines.

Tabel 3. Ray Tracing dengan perhitungan normal B-spline

No	Objek	Jumlah Vertex	Jumlah Face	Waktu	Memory	Hasil
1	Ball.3ds	482	960	0(h) : 1(m) : 56(s) : 77(ms)	99 KB	
2	Cone.obj	33	62	0(h) : 0(m) : 15(s) : 67(ms)	32 KB	
3	Icosphere.obj	42	80	0(h) : 0(m) : 17(s) : 32(ms)	33 KB	
4	Torus.obj	576	1152	0(h) : 2(m) : 1(s) : 39(ms)	69 KB	

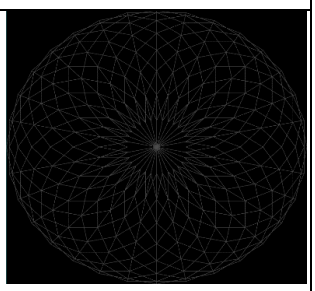
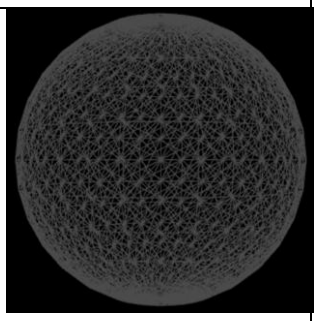
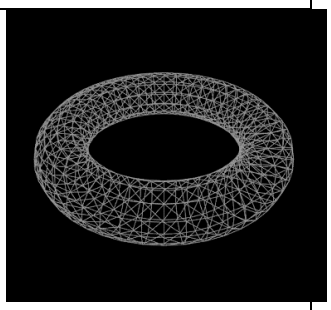
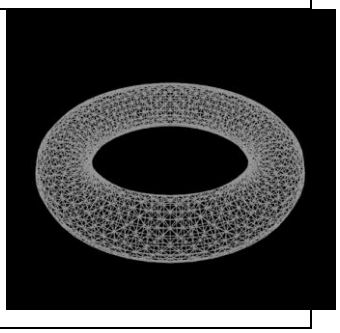
Pada tabel 3 dapat dilihat bahwa hasil rendering tidak halus. Hal ini disebabkan karena adanya kesulitan dalam menerapkan perhitungan permukaan b-splines pada mesh yang bentuk facenya segitiga. Hal ini mudah didapatkan pada obyek mesh yang bentuk face nya adalah segiempat, karena membutuhkan 16 verteks tetangga.

Tabel 4. Ray Tracing dengan perhitungan normal B-splines pada obyek kompleks

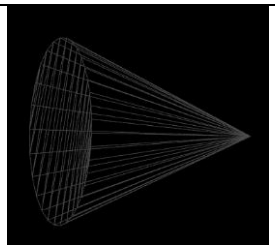
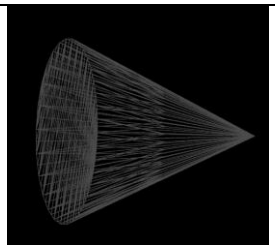
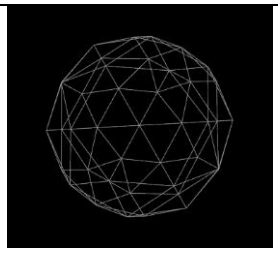
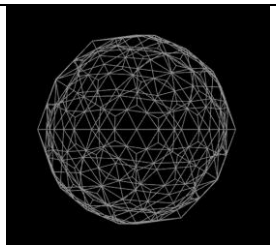
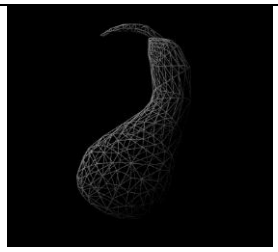
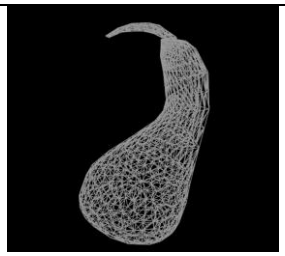
No	Objek	Jumlah Vertex	Jumlah Face	Waktu	Memory	Hasil
1	Dolphin.obj	855	1692	0(h) : 3(m) : 8(s) : 8(ms)	89 KB	
2	Gourd.obj	326	648	0(h) : 1(m) : 16(s) : 88(ms)	51 KB	
3	Skull.obj	6624	7120	0(h) : 12(m) : 38(s) : 51(ms)	248 KB	
4	Unicorn.obj	2356	1572	0(h) : 3(m) : 10(s) : 92(ms)	79 KB	
5	Magnolia.obj	1184	1372	0(h) : 2(m) : 29(s) : 53(ms)	73 KB	
6	Monkey.obj	505	968	0(h) : 2(m) : 1(s) : 47(ms)	60 KB	

Untuk membandingkan hasil dan lama proses rendering, digunakan obyek yang permukaannya diperhalus (diperkecil) dengan menggunakan metode *subdivision*. Pembuatan obyek dengan subdivision menggunakan blender. Perbandingan jumlah face dapat dilihat pada tabel 5-tabel 8.

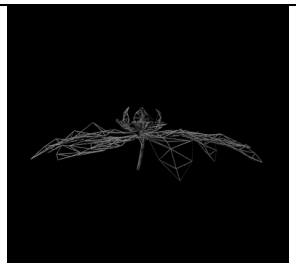
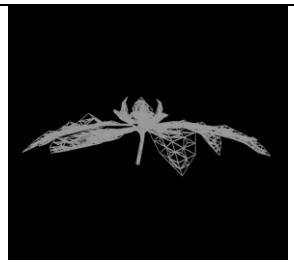
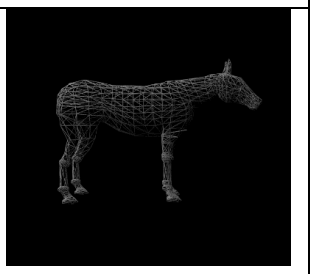
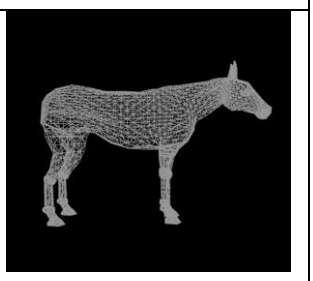
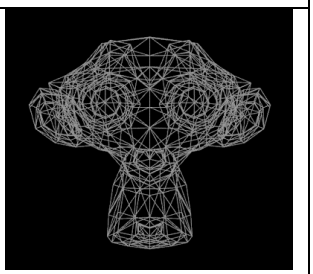
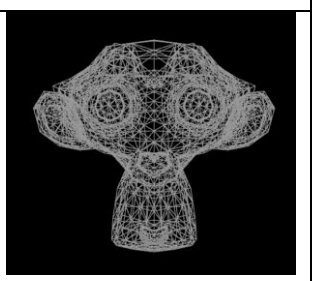
Tabel 5. Obyek asli dan perhalusan permukaan dengan menggunakan subdivision serta jumlah facenya.

No.	*Nama obyek*	*metode*	Jumlah *Face*	*Wireframe*
1	*Ball.3ds*	*Normal*	960	
		Simple subdivision	5760	
2	*Torus.obj*	*Normal*	968	
		Simple subdivision	4608	

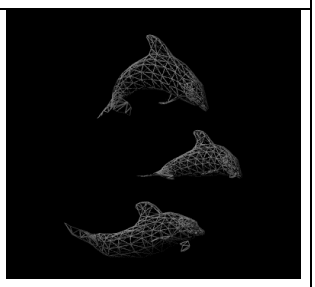
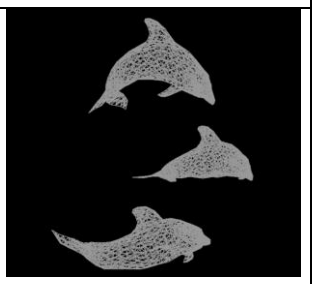
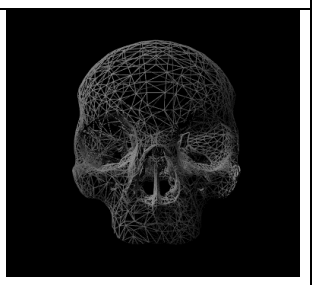
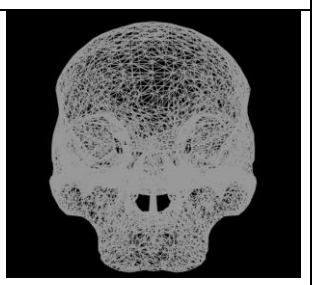
Tabel 6. Obyek asli dan perhalusan permukaan dengan menggunakan subdivision serta jumlah facenya(lanjutan)

No.	***Nama obyek***	***metode***	**Jumlah *Face***	***Wireframe***
3	*Cone.3ds*	*Normal*	62	
		Simple subdivision	372	
4	*Icosphere.obj*	*Normal*	80	
		Simple subdivision	480	
5	*Gourd.obj*	*Normal*	648	
		Simple subdivision	3888	

Tabel 7. Obyek asli dan perhalusan permukaan dengan menggunakan subdivision serta jumlah facenya(lanjutan)

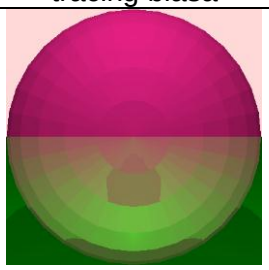
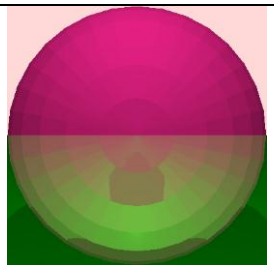
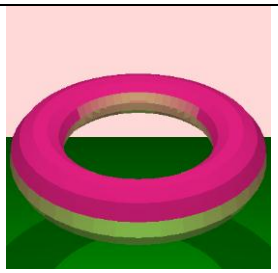
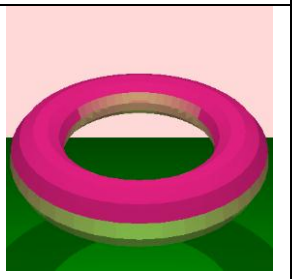
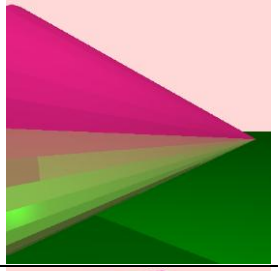
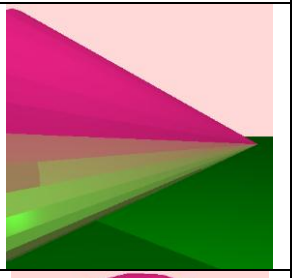
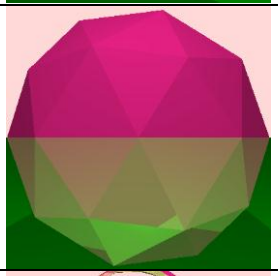
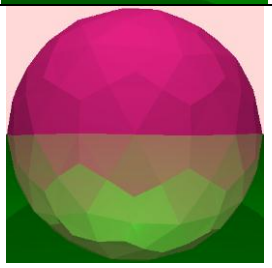
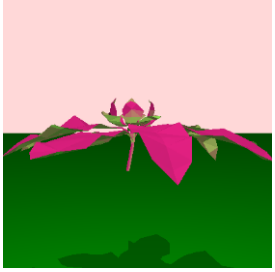
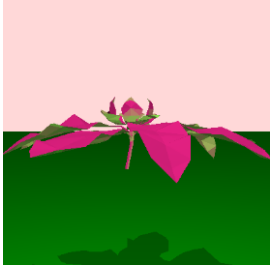
No.	***Nama obyek***	***metode***	**Jumlah *Face***	***Wireframe***
6	*Magnolia.obj*	*Normal*	1372	
		Simple subdivision	7732	
7	*Unicorn.obj*	*Normal*	1572	
		Simple subdivision	6244	
8	*Monkey.obj*	*Normal*	968	
		Simple subdivision	3936	

Tabel 8. Obyek asli dan perhalusan permukaan dengan menggunakan subdivision serta jumlah facenya(lanjutan)

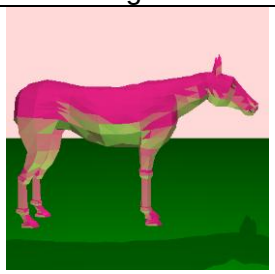
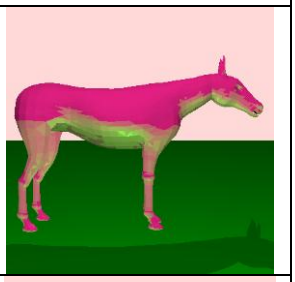
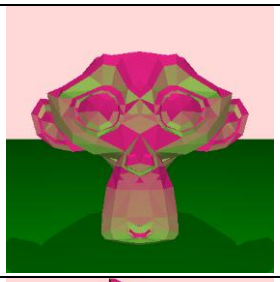
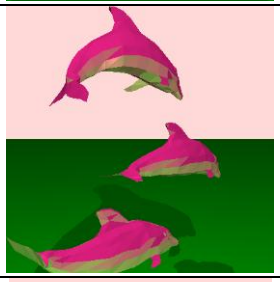
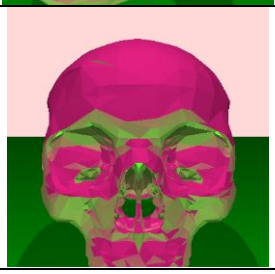
No.	***Nama obyek***	***metode***	**Jumlah *Face***	***Wireframe***
9	*Dolphin.obj*	*Normal*	1692	
		Simple subdivision	10134	
10	*Skull.obj*	*Normal*	7120	
		Simple subdivision	41506	

Dari 10 buah objek *mesh* pada Tabel 5-tabel 8 menunjukkan bahwa *subdivision* akan meningkatkan jumlah *face* pada sebuah *mesh* dan berikut adalah hasil pengujian untuk membandingkan hasil *ray tracing* objek tersebut. Hasil perbandingan antara proses ray tracing secara biasa, dengan obyek subdivision, dengan perhitungan barycentric normal dan dengan perhitungan normal menggunakan kurva b-splines, dapat dilihat pada tabel 9 dan tabel 10.

Tabel 9. Hasil proses rendering dibandingkan dengan subdivison

No	Nama Objek	Lama proses rendering dari Ray tracing biasa	Hasil proses ray tracing biasa	Hasil proses ray tracing dengan subdivision
1	Ball.3ds	0:1:43:90		
2	Torus.obj	0:1:56:90		
3	Cone.obj	0:0:12:1		
4	Icosphere.obj	0:0:22:9		
5	Gourd.obj	0:1:8:54		
6	Magnolia.obj	0:2:18:77		

Tabel 10. Hasil proses rendering dibandingkan dengan subdivison (lanjutan)

No	Nama Objek	Lama proses rendering dari Ray tracing biasa	Hasil proses ray tracing biasa	Hasil proses ray tracing dengan subdivision
7	Unicorn.obj	0:2:50:93		
8	Monkey.obj	0:1:53:73		
9	Dolphin.obj	0:2:52:38		
10	Skull.obj	0:10:54:36		

Dari semua percobaan yang sudah dilakukan, dengan dua aspek pengamatan, yaitu lama proses dan penggunaan memory, maka perbandingan antara ray tracing biasa, ray tracing dengan perhitungan *barycentric normal* dan ray tracing dengan perhitungan normal menggunakan kurva b-splines, dapat dlihat pada tabel 11 dan tabel 12. Tabel 11 menampilkan perbandingan dalam aspek lama proses rendering, sedangkan tabel 12 menampilkan perbandingan dalam aspek penggunaan memory.

Tabel 11. Peningkatan lama proses ray tracing dengan *barycentric normal* dan normal b-splines terhadap ray tracing biasa.

No.	***Nama Object***	***Ray tracing* dengan *barycentric normal***	***Ray tracing* dengan normal b-splines**
1	*Ball.3ds*	203,7%	232,33%
2	*Torus.obj*	180.26%	288,62%
3	*Cone.3ds*	184.21%	190,87%
4	*Icosphere.obj*	132.44%	144,98%
5	*Gourd.obj*	205.86%	215,94%
6	*Magnolia.obj*	209.24%	219,55%
7	*Unicorn.obj*	208.76%	213,18%
8	*Monkey.obj*	194.67%	310,49%
9	*Dolphin.obj*	204.83%	218,96%
10	*Skull.obj*	272.04%	291,37%

Tabel 12. Peningkatan penggunaan *memory* dari proses ray tracing dengan *barycentric normal* dan normal b-splines terhadap ray tracing biasa

No.	***Nama Object***	***Ray tracing* dengan *barycentric normal***	***Ray tracing* dengan normal b-splines**
1	*Ball.3ds*	100%	100,1%
2	*Torus.obj*	100,96%	100 %
3	*Cone.3ds*	100,33%	100,35%
4	*Icosphere.obj*	100%	100%
5	*Gourd.obj*	100,32%	100,35%
6	*Magnolia.obj*	100%	100,41%
7	*Unicorn.obj*	100%	100%
8	*Monkey.obj*	100,32%	100%
9	*Dolphin.obj*	100,32%	100%
10	*Skull.obj*	100,71%	100%

Dari tabel 11 dapat dilihat bahwa waktu proses raytracing dengan menggunakan barycentric normal lebih cepat dibandingkan dengan menggunakan normal b-splines. Sedangkan dari tabel 12 dapat dilihat bahwa penggunaan memory tidak meningkat secara kentara karena dilakukan secara terpisah. Ada bagian perhitungan yang dilakukan terlebih dahulu.

BAB 5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil percobaan, dapat disimpulkan bahwa hasil ray tracing dengan menggunakan *barycentric normal* lebih halus dibanding dengan normal b-splines. Sementara jika dibandingkan dengan subdivision, 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.

5.2. Saran

Mencoba menerapkan normal b-splines khusus untuk meh dengan face berbentuk segi empat.

Lampiran 1. Laporan Anggaran Penelitian

1. Honorarium

Tim Peneliti	Jumlah Anggota	Bulan Kerja	Minggu / Bulan	Jam / Minggu	Total (Rp)
Ketua	1	11	4	2	1.500.000
Anggota 1	1	11	4	2	1.155.000
Sub Total					2.655.000

2. Komponen Peralatan

Upgrade Processor Intel Core i7-3770 3,4 GHz	Rp.	2.940.000
Upgrade Motherboard Gigabyte z 77M-D3H	Rp.	1.380.000
Upgrade Power SupplyAzza Dynamo 650W	Rp.	650.000
Upgrade Casing Middle ATX Simbadda 5819	Rp.	325.000
Sub total	Rp.	5.295.000

3. Biaya Pembuatan Program (coding program)

modul perhitungan normal pada permukaan virtual dengan menggunakan kurva B-Splines	Rp.	350.000
modul perhitungan normal dengan proporsi normal dari tiga titik pembentuk segitiga	Rp.	350.000
implementasi kedua perhitungan normal bidang kedalam metode ray tracing	Rp	350.000
Sub total	Rp.	1.050.000