PENERAPAN ORDERED GRID SUPER SAMPLING UNTUK FULL SCENE ANTI ALIASING PADA MESIN RENDER RAY TRACING

Febriliyan Samopa dan Budi Kurniawan

Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember (ITS) - Surabaya
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111
Tel. + 62 31 5939214, Fax + 62 31 5939363
Email : iyan@its-sby.edu

ABSTRAK

Metode ray tracing adalah salah satu metode render dalam grafika komputer tiga-dimensi yang mensimulasikan sifat-sifat fisik sinar di alam. Dengan metode ray tracing, akan didapatkan citra hasil render dengan model pencahayaan yang realistis. Salah satu kelemahan ray tracing adalah munculnya alias pada citra hasil rendernya. Alias timbul karena undersampling, yaitu pengambilan sample dengan jumlah yang kurang. Alias dalam ray tracing nampak dalam bentuk "jaggies" pada citra hasil.

Untuk menghilangkan alias, ada berbagai metode anti-aliasing, di antaranya Ordered Grid supersampling (OGSS). Metode ini melakukan supersampling pada citra untuk meningkatkan jumlah sampel, kemudian melakukan downsample kembali pada citra yang diinginkan. Penelitian ini akan menerapkan metode OGSS pada mesin render ray tracing, serta membandingkan citra hasil render dengan OGSS dengan citra hasil render biasa melalui perangkat lunak (mesin render) yang dibuat.

Penelitian ini menunjukkan bahwa OGSS dapat diterapkan pada mesin render ray tracing dengan hasil yang menguntungkan. Karena dengan beban sebesar empat kali lipat waktu komputasi hanya meningkat tiga kali lipat. Sedangkan alias yang terjadi dapat diminimkan atau bahkan dihilangkan.

Kata kunci : *Ray Tracing*, *Ordered Grid Super Sampling*, *Renderer (mesin render)*, *Full Scene Anti Aliasing*

1. PENDAHULUAN

Dunia grafika komputer terdiri dari sejumlah unsur penyusun, dimana salah satu unsur dalam grafika komputer tiga-dimensi adalah *render*. *Render* adalah proses memberi permukaan dan kesan tiga-dimensi pada objek yang ditampilkan pada bidang dua dimensi (layar monitor).

Dari berbagai metode render yang ada, *ray tracing* (penelusuran sinar) termasuk salah satu metode yang banyak mendapat perhatian, karena sudah diakui bahwa citra hasil render menggunakan metode ini adalah yang paling alami (natural) dan presisi dibanding metode yang lain.

Hanya saja citra hasil render apabila menggunakan metode *ray tracing* biasa akan mempunyai *alias*, yang dapat diamati dari timbulnya efek *jaggies* (efek tangga) pada tepian objek yang direnderer. OGSS (*Ordered Grid Super Sampling*) adalah salah satu metode *anti-aliasing* (penghilangan *alias*). Diharapkan dengan metode ini, citra hasil render akan dapat diperbaiki kualitasnya.

2. RAY TRACING

Penelusuran sinar (*ray tracing*) adalah suatu teknik untuk melakukan *render* (memberi permukaan pada objek sehingga objek tampak nyata). Proses *render* terdiri dari tiga langkah, yaitu:

1. Pemodelan.
2. Proses.
3. Penampilan dan penyimpanan hasil render.

2.1. Pemodelan

Pemodelan adalah tahap mendeskripsikan model bagi proses *render*. Pada tahap ini dilakukan pendefinisian adegan (*scene*) yang akan di-*render* dan ditampilkan. Suatu adegan umumnya terdiri dari objek-objek tiga-dimensi, kamera, dan sumber cahaya. Deskripsi sifat-sifat permukaan objek juga disertakan.

Proses render berfungsi melakukan render adegan yang telah didefinisikan pada tahap pemodelan. Hasil dari proses ini adalah suatu citra adegan yang telah di-render (dalam hal ini, data warna piksel pada citra tersebut).

Penampilan hasil *render* adalah proses menampilkan citra pada layar monitor, dapat dilanjutkan dengan penyimpanan citra.

Yang mendasari teknik *ray tracing* adalah apa yang kita lihat tergantung dari sinar yang mencapai mata kita. Sehingga masuk akal untuk memodelkan program untuk me-*render* dengan cara yang sama dengan sifat sinar. *Ray tracing* melakukan hal ini

dengan memodelkan cahaya sebagai sinar-sinar tunggal.

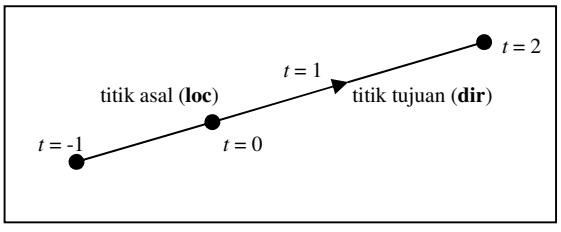
Dalam dunia nyata, cahaya berasal dari sumber cahaya, dan mengenai objek, memantul, dan kadang mencapai mata pengamat. Saat sinar mengenai objek, energinya sebagian diserap dan sebagian dipantulkan. Sinar yang dipantulkan membawa warna objek. Sinar ini pada gilirannya mungkin akan memantul pada objek lain dan mempengaruhi warnanya, atau mencapai mata pengamat sehingga pengamat dapat melihat objek.

Karena kebanyakan sinar tidak mencapai mata, ray tracing diimplementasikan dengan cara terbalik—dimulai dari pengamat dan menelusuri sinar dari belakang untuk menentukan dari mana asal sinar itu dan warna apa yang ditransmisikan

Sinar direpresentasikan dalam bentuk persamaan parametrik garis. Persamaan ini terdiri dari dua vektor: lokasi titik asal sinar dan titik tujuan sinar. Setiap titik p pada sinar dapat dinyatakan dengan

$$\mathbf{p} = \mathbf{loc} + t\,\mathbf{dir} \qquad (1)$$

Untuk $t = 0$, **p** berada di titik asal sinar (**loc**). Apabila nilai t bertambah, **p** bergerak sepanjang sinar, semakin jauh dari titik asal sinar. Jika $t < 0$, titik akan berada pada garis yang sama, tetapi di belakang titik asal sinar. Gambar 1 menunjukkan hal ini.



Gambar 1. Reprensentasi Sinar

Sumber cahaya yang digunakan dalam ray tracing pada penelitian ini adalah sumber cahaya titik, yaitu sumber cahaya yang tidak memiliki dimensi. Dengan kata lain, sumber cahaya ini memancarkan sinar secara radial ke semua arah dari satu titik tetap.

Sumber cahaya titik secara khusus cocok untuk ray tracing, karena hanya satu sinar bayangan yang perlu diproses untuk suatu titik perpotongan untuk tiap sumber cahaya dan perhitungan pencahayaan juga lebih sederhana. Konsekuensinya, sumber cahaya titik menghasilkan bayangan yang tajam dan geometris.

2.2. Proses

Ray tracing bekerja dengan beberapa tahapan, yaitu:

1. Menentukan bidang pandang dan arah pandang.
2. Untuk setiap piksel pada bidang pandang (bidang pandang merepresentasikan citra yang akan dihasilkan), ditembakkan satu sinar dengan titik awal sinar pada mata, dan titik akhir sinar pada titik yang diwakili oleh piksel tersebut.
3. Untuk tiap objek pada adegan, dicari titik perpotongan terdekat dengan sinar tersebut. Apabila ditemukan, maka diterapkanlah model pencahayaan pada objek ini untuk menentukan warna piksel tadi.

Ray tracing digolongkan sebagai algoritma iluminasi global. Iluminasi global dalam grafika komputer adalah istilah yang diberikan untuk model-model untuk me-render suatu adegan dengan mengevaluasi cahaya yang dipantulkan dari sebuah titik x dengan membawa semua iluminasi yang sampai ke titik tersebut [14]. Sehingga nilai cahaya yang diperhitungkan bukan hanya dari sumber cahaya secara langsung, tetapi dari semua iluminasi tak langsung yang mungkin berasal dari sumber cahaya lewat objek lain.

Ray tracing menggabungkan beberapa elemen berikut ke dalam suatu model tunggal :

- Penghilangan permukaan yang tersembunyi (*hidden surface removal*).
- Pemberian warna akibat iluminasi langsung.
- Efek interaksi spekular umum seperti refleksi dan refraksi.
- Komputasi bayangan.
- Proyeksi perspektif.

Penghilangan permukaan yang tersembunyi (yang tidak terlihat oleh mata) secara otomatis dilakukan oleh *ray tracing*, karena sinar yang ditelusuri dari mata akan digunakan untuk menentukan titik perpotongan sinar itu dengan objek. Titik perpotongan ini akan diambil yang terdekat dengan mata.

Setelah titik perpotongan sinar dengan objek diketahui, dengan demikian diketahui pula objek yang akan diperiksa oleh proses *ray tracing*. Kini dilakukan proses penentuan warna piksel yang bersesuaian dengan titik ini.

Cukup sulit bagi kita untuk mendefinisikan warna, meskipun warna memiliki makna naluriah bagi kita. Warna adalah suatu sifat dari zat di mana kita bisa membedakannya dari zat lain dengan warna yang berbeda hanya dengan melihatnya [3]. Secara mendasar, warna adalah sifat, atau indeks dari sifat, dari suatu zat untuk memantulkan, menghasilkan atau membiaskan sinar dengan panjang gelombang yang spesifik. Sesuatu yang memantulkan cahaya dengan panjang gelombang rendah (bagian merah

dari akhir spektrum) secara dominan dikatakan memiliki warna merah. Sesuatu yang lain yang memantulkan cahaya dengan panjang gelombang tinggi adalah ungu, dan seterusnya.

Warna seringkali direpresentasikan dalam grafika komputer dengan tiga komponen – umumnya Merah, Hijau dan Biru (*Red*, *Green*, *Blue*, RGB). Komponen-komponen ini (primer) memiliki panjang gelombang standar tertentu. Dengan menggabungkan jumlah yang sesuai dari komponen ini, setiap warna dapat secara tepat atau secara aproksimasi direpresentasikan.

Apabila titik perpotongan sinar dengan objek telah ditemukan, maka warna pada titik perpotongan tersebut dapat dihitung. Warna titik perpotongan adalah jumlah dari lima komponen warna yang berbeda: ambien, difusi, spekular, pantulan dan pembiasan. Tiap komponen memodelkan cara transportasi cahaya yang berbeda. Pemantulan dan pembiasan adalah yang membedakan ray tracing dengan teknik render lain; objek yang reflektif dan transparan cukup sulit dimodelkan dengan teknik render lain.

Komponen-komponen tersebut dapat dipisahkan menjadi tiga kategori:

1. Aproksimasi refleksi difusi antar objek:
   - Komponen ambien
2. Komponen akibat sumber cahaya:
   - Komponen difusi
   - Komponen spekular
3. Komponen yang dihasilkan lewat rekursi:
   - Komponen pantulan
   - Komponen pembiasan

Komponen ambien mengaproksimasi pantulan difusi cahaya antar objek. Yaitu, cahaya yang memantul tidak hanya dari sumber cahaya pada objek, tetapi juga antar objek. Metode ray tracing yang sederhana (tradisional) tidak cukup baik dalam memodelkan refleksi antar objek ini. Algoritma radiosity memiliki kemampuan memodelkan komponen ambien ini secara lebih realistis.

Kategori yang kedua, cahaya yang berasal dari sumber cahaya yang memantul pada objek, meliputi komponen difusi dan spekular. Seperti komponen ambien, komponen difusi memantulkan cahaya secara sama pada semua arah, tetapi tidak seperti komponen ambien, cahaya yang dipantulkan menjadi lemah menurut jumlah sinar yang tiba dari sumber cahaya pada titik yang akan dihitung warnanya. Komponen spekular memodelkan sorotan (highlights) sumber cahaya pada objek-objek yang mengkilap seperti apel atau bola bilyar. Komponen-komponen ini keduanya disebabkan oleh sumber cahaya dalam dunia, jadi proses ray tracing harus menentukan seberapa banyak sinar yang mencapai titik perpotongan dari sumber cahaya.

Kategori ketiga, komponen yang tiba secara rekursi, meliputi komponen pantulan dan pembiasan. Komponen pantulan dan pembiasan dihitung dengan menembakkan sinar pada titik perpotongan dengan arah pantul atau bias, menemukan objek terdekat sepanjang sinar tersebut, dan mengevaluasi persamaan pewarnaan pada titik potong tersebut. Proses ini dapat disamakan dengan menembakkan sinar primer melalui sebuah piksel. Karena objek terdekat yang dipotong oleh sinar juga bisa berupa objek yang reflektif atau transparan, sinar lain mungkin dihasilkan lagi. Karena alasan inilah, proses ray tracing yang mendukung objek reflektif dan transparan seringkali disebut recursive ray tracers.

Karena proses rekursi, tentu diperlukan suatu faktor batasan kapan rekursi tersebut berhenti. Faktor-faktor ini dapat berupa:

- Ketika sinar memotong permukaan yang bersifat difusi (tidak reflektif atau tidak transparan).
- Ketika faktor kedalaman rekursi (yang telah ditentukan sebelumnya, menentukan berapa banyak sinar pantul atau sinar bias yang akan dihasilkan) telah tercapai.
- Ketika energi dari sinar telah menurun di bawah suatu batasan tertentu.

Dalam perangkat lunak yang akan dibuat pada penelitian ini, warna yang digunakan adalah modus RGB, yaitu dispesifikasikan sebagai 3-tuple, dengan tiap nilai tuple berada dalam wilayah 0 sampai 1. Nilai 0 berarti tidak ada intensitas warna, dan nilai 1 berarti warna berada pada intensitas yang maksimal. Dengan menggunakan skema ini, warna hitam adalah (0, 0, 0) (merupakan warna merah, hijau, dan biru secara berurutan). Putih adalah (1, 1, 1).

Efek refleksi (pantulan) dan refraksi (pembiasan) dapat diimplementasikan karena dengan representasi sinar dalam bentuk vektor, dapat juga dicari sinar pantul dan sinar bias juga dalam bentuk vektor. Untuk menentukan warna yang dibawa oleh sinar pantul dan sinar bias ini, algoritma ray tracing menerapkan proses rekursi.

Komputasi bayangan juga dapat langsung dilakukan oleh ray tracing. Untuk menentukan apakah suatu titik pada objek berada dalam bayangan atau tidak, cukup dengan menembakkan sinar dari titik tersebut ke sumber cahaya, apabila sinar tersebut terhalang oleh objek lain, maka titik tadi berada dalam bayangan.

Proyeksi perspektif juga langsung dilakukan oleh ray tracing. Pemanfaatan sifat sinar yang menyebar ke semua arah mengakibatkan perhitungan titik perpotongan sinar dengan objek juga akan mengikuti prinsip-prinsip perspektif.

Sehingga objek yang jauh akan nampak lebih kecil. Secara logika, hal ini dapat dijelaskan sebagai berikut: sinar akan lebih banyak memotong objek yang lebih dekat dengan mata daripada objek yang lebih jauh. Dengan demikian titik perpotongan yang digunakan untuk membentuk objek juga akan lebih banyak pada objek yang dekat. Hal ini mengakibatkan objek yang lebih dekat akan terlihat lebih besar.

2.3. Implementasi Mesin Render

Algoritma *ray tracing* yang diimplementasi pada mesin render adalah:

```
TraceRay (struktur sinar)
    tes perpotongan (terdekat)
    if sinar memotong sebuah objek
        tentukan normal pada titik
           perpotongan
        hitung intensitas warna lokal
           (I_local)
        kurangi kedalaman trace
        if kedalaman trace > 0
            hitung dan tembakkan sinar
                pantul
            hitung dan tembakkan sinar
                bias
```

Dua baris terakhir dalam algoritma ray tracing di atas menyatakan secara tidak langsung pemanggilan fungsi TraceRay secara rekursif. Detail dari kedua baris tersebut dijelaskan berikut ini :

Hitung dan tembakkan sinar pantul dinyatakan sebagai

```
    if objek bersifat reflektif
        hitung vektor refleksi dan masukkan
           dalam struktur sinar
        titik asal sinar = titik
           perpotongan
        kurangi intensitas sinar
        TraceRay (struktur sinar pantul)
        if sinar pantul memotong sebuah
           objek
              gabungkan warna yang didapat
                 dengan I_local
```

Hitung dan tembakkan sinar bias dinyatakan sebagai

```
    if objek bersifat refraktif (transpa-
       ran)
        if sinar memasuki objek
            kumpulkan indeks refraksi
            naikkan jumlah objek di mana
               sinar berada di dalamnya
            hitung vektor refraksi dan
               masukkan dalam struktur
               sinar
        else
            keluarkan indeks refraksi
            turunkan jumlah objek di mana
               sinar berada di dalamnya
            hitung vektor refraksi dan
               masukkan dalam struktur
               sinar
        titik asal sinar = titik
           perpotongan
        kurangi intensitas sinar
```

```
TraceRay (struktur sinar bias)
if sinar bias memotong sebuah objek
    gabungkan warna yang didapat
       dengan Ilocal
```

Perhitungan komponen-komponen cahaya (ambien, difusi, dan spekular) hanya dilakukan apabila titik perpotongan pada objek tidak berada dalam bayangan. Perhitungan refleksi hanya dilakukan apabila objek memiliki koefisien refleksi yang bernilai lebih dari 0. Proses rekursif perhitungan refleksi akan berakhir apabila variabel jumlah refleksi telah bernilai 0.

3. ANTI-ALIASING DENGAN ORDERED GRID SUPER SAMPLING

Citra yang dihasilkan oleh *ray tracing* memiliki suatu kelemahan yaitu timbulnya *alias*. *Alias* merupakan salah satu artefak dalam citra. Artefak adalah suatu elemen pada citra yang bukan merupakan bagian dari proses pemodelan citra tersebut. Artefak ini muncul di luar proses pemodelan. Artefak merupakan konsekuensi dari operasi pada algoritma render tertentu yang digunakan untuk menghasilkan citra.

3.1. Alias dan Sampling

Dalam teori, istilah '*alias*' merupakan artefak pada citra tertentu yang kebanyakan nampak pada tekstur ketika periodisitas pendekatan tekstur mendekati dimensi piksel [14]. Ini diakibatkan oleh *undersampling*. Istilah *sampling* dalam grafika komputer berasal dari fakta bahwa saat kita menghitung warna tunggal atau nilai setiap piksel; kita melakukan sampling terhadap sebuah solusi pada titik-titik diskrit dalam ruang solusi. Ini adalah ruang yang secara potensial kontinu, karena citra grafika komputer dibuat dari abstraksi, kita dapat menghitung sampel di manapun dalam bidang citra.

3.2. Jaggies

Dalam *ray tracing*, *alias* secara lebih khusus dapat diamati dari timbulnya efek '*jaggies*'. *Jaggies* biasanya nampak pada tepi suatu objek. Tepi objek tersebut menjadi kelihatan bergerigi (*jagged*). Seringkali istilah alias itu sendiri disebut dengan *jaggies* [10].

Penyebab timbulnya *jaggies* ini adalah kenyataan bahwa meskipun piksel bersesuaian dengan wilayah tak-nol yang terbatas dalam bidang pandangan, hanya satu sinar yang ditembakkan melalui wilayah ini (satu sinar pada satu piksel). Padahal dalam kenyataannya, satu piksel tidaklah cukup kecil untuk merepresentasikan satu titik koordinat. Bahkan dalam satu piksel dalam layar komputer terdapat tak terhingga banyaknya titik. Dengan demikian *undersampling* di sini terjadi

karena sampel titik yang kurang, atau dipandang dari sudut lain, jumlah sinar pada satu piksel yang seharusnya lebih dari satu sinar

3.3. Full-Scene Anti-aliasing dengan Ordered Grid Super Sampling

Anti-aliasing adalah istilah umum yang diberikan pada metode yang berhubungan dengan ketidakcocokan yang timbul akibat *undersampling* [14]. Dalam *ray tracing*, ini berarti menghilangkan efek *jaggies*. Salah satu metode *anti-aliasing* adalah *supersampling*, yaitu menambah jumlah sampel. Seperti telah dibahas di bagian sebelumnya bahwa penyebab *alias* adalah *undersampling*, maka dengan memperbanyak jumlah sampel diharapkan dapat menghilangkan *alias*.

Penambahan sampel di sini berupa memperbanyak jumlah sinar yang melewati satu piksel. Seperti penjelasan sebelumnya, satu piksel sebenarnya terdiri dari tak terhingga banyaknya titik koordinat, dengan memperbanyak sinar, maka diharapkan titik yang dapat di-sampel menjadi lebih banyak pula.

Memperbanyak sinar dapat pula dipandang sebagai memperbanyak piksel. Karena dengan jumlah piksel yang bertambah, tentunya sinar yang ditembakkan pada piksel juga semakin banyak

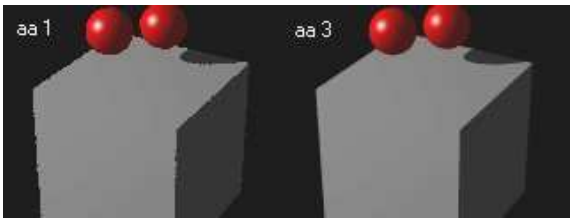
Hasil supersampling ini kemudian di-*downsample* dengan cara menghitung rata-rata intensitas warna sub piksel hasil *supersampling* menjadi warna satu piksel kembali. Dengan demikian, resolusi citra yang dihasilkan akan tetap sama, namun telah melalui proses penambahan sampel. Anti-*aliasing* dilakukan pada keseluruhan citra (satu adegan), sehingga disebut juga *full-scene anti-aliasing*. Contoh citra yang mengalami alias dan yang telah di-anti-alias dapat dilihat pada gambar 2 dan gambar 3.

Ide *supersampling* sangat sederhana, namun cukup memakan sumber daya [10]. Suatu adegan di-*render* ke resolusi yang lebih tinggi dari resolusi yang diinginkan. Sebagai contoh, 4× *supersampling* adalah me-*render* menjadi resolusi empat kali lebih besar daripada resolusi target, yang berarti dua kali jumlah piksel dalam dimensi lebar dan dua kali jumlah piksel dalam dimensi panjang.

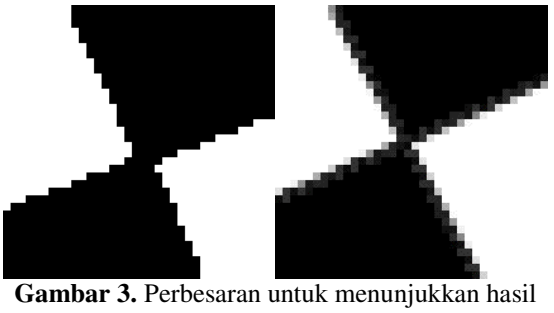
Supersampling 4× ini memiliki akibat tiap piksel pada adegan akan direpresentasikan menjadi empat piksel pada *buffer*, dengan masing-masing adalah seperempat bagian dari piksel pada adegan.

Dengan cara ini tingkat sub-piksel telah tercapai (sampel menjadi lebih banyak) dan melakukan filter (*downsample*) pada empat piksel tersebut menghasilkan piksel yang telah di-*anti-alias*.

Teknik melakukan render pada resolusi yang lebih tinggi, kemudian melakukan *downsample* dengan merata-rata intensitas sub piksel disebut *Ordered Grid Super Sampling* (OGSS) [1]. Sesuai namanya, sub piksel yang terbentuk tersusun dalam bentuk grid sehingga *supersampling* 4× akan menghasilkan grid 2×2 untuk satu piksel asal.



Gambar 2. Citra dengan *alias* (kiri) dan setelah di *anti-aliasing* (kanan)



Gambar 3. Perbesaran untuk menunjukkan hasil *anti aliasing*, citra kiri sebelum di *anti alias* dan kanan setelah di *anti alias*

3.4. Implementasi Ordered Grid Super Sampling pada Mesin Render

Proses *anti-aliasing* berlangsung dengan cara me-*render* citra dengan resolusi dua kali lebih besar, kemudian melakukan rata-rata nilai warna citra *supersample* ini. Algoritma yang digunakan sebenarnya sama dengan algoritma mesin *render*, hanya saja resolusi citra yang diperbesar, sehingga sinar yang ditembakkan juga semakin banyak. Saat warna piksel akhir telah diketahui, maka empat buah piksel yang bertetangga dihitung rata-rata warnanya, untuk dijadikan warna satu piksel citra hasil.

Berikut ini algoritma *anti-aliasing*, dengan proses perata-rataan berlangsung bersamaan dengan jalannya *render* citra *supersample* :

```
startx, starty = 0
limitx, limity = 2
while limity <= tinggi resolusi * 2
    while limitx <= lebar resolusi * 2
       for y = starty to limity
           for x = startx to limitx
                hitung sinar dan masukkan
                   dalam struktur sinar
                TraceRay(struktur sinar)
                simpan warna yang didapat
       limitx = limitx + 2
       startx = startx + 2
       hitung rata-rata warna piksel dan
          simpan
    limitx = 2
    limity = limity + 2
```

```
startx = 0
starty = starty + 2
```

Dengan algoritma ini, maka proses *render* akan berjalan tiap 2 × 2 piksel (grid 2 × 2). Kemudian warna dari 4 piksel ini dihitung rata-ratanya dan disimpan sebagai warna citra hasil.

4. HASIL EKSPERIMEN

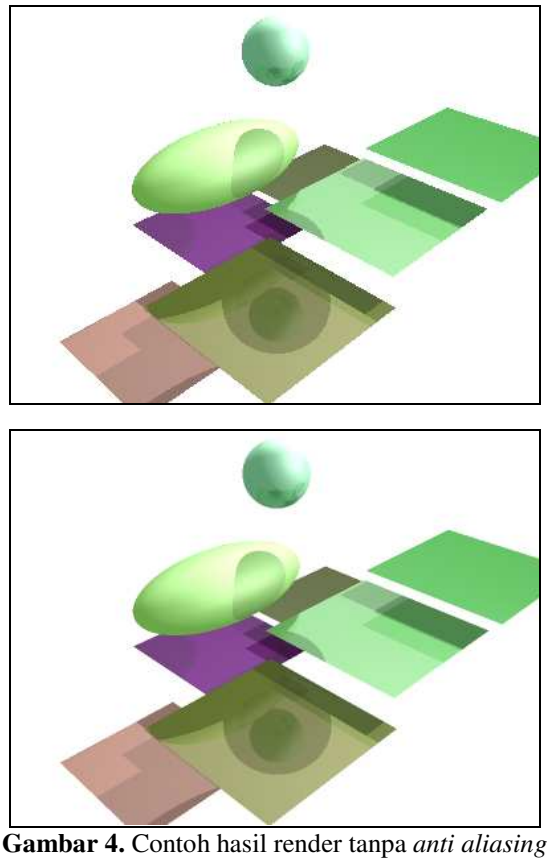
Untuk menghitung kinerja dari metode ini digunakan parameter Peningkatan Waktu Komputasi (T), yaitu waktu komputasi yang dibutuhkan untuk me-*render* sebuah adegan dengan *anti aliasing* dibandingkan dengan waktu komputasi untuk me-*render* adegan yang sama tetapi tanpa *anti aliasing*. Didefinisikan sebagai :

$T = t_A / t_N$

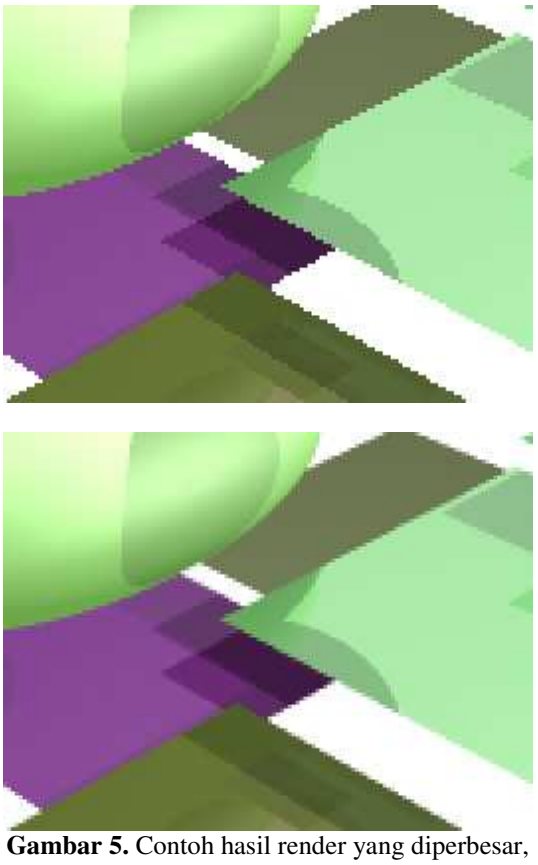
t_A = waktu komputasi dengan anti aliasing

t_N = waktu komputasi tanpa anti aliasing

Untuk eksperimen ini digunakan sebuah adegan yang di -*render* menjadi 21 macam resolusi yang berbeda, mulai dari 300 x 225 sampai 800 x 600, serta 20 macam adegan dengan resolusi ang sama. Contoh hasil render dapat dilihat pada gambar 4 dan gambar 5.



Gambar 4. Contoh hasil render tanpa *anti aliasing* (atas) dan dengan *anti aliasing* (bawah)



Gambar 5. Contoh hasil render yang diperbesar, tanpa anti aliasing (kiri) dan dengan anti aliasing (kanan)

Dari beberapa kali eksperimen, dihasilkan citra hasil render sesuai yang diinginkan, baik yang menggunakan render secara normal, dan yang menggunakan anti-aliasing. Untuk render dengan anti-aliasing, citra yang dihasilkan nampak lebih halus walaupun dalam resolusi yang sama. Dengan demikian, anti-aliasing dengan OGSS berhasil mengurangi efek alias dengan baik.

Sedangkan untuk pengukuran kinerjanya didapat hasil yang terdapat pada tabel 1 dan 2 serta gambar 6 dan 7. Dari tabel 1 diperoleh bahwa untuk adegan yang sama dengan resolusi yang berbeda, waktu render akan meningkat sesuai dengan besarnya resolusi (berbanding lurus dengan besarnya resolusi). Untuk tabel ini rata-rata peningkatan waktu komputasi adalah 3,32.

Sedangkan dari tabel 2 didapat bahwa untuk resolusi yang sama, render tanpa OGSS akan membutuhkan waktu yang relatif sama untuk adegan yang berbeda. Namun untuk render dengan OGSS, membutuhkan waktu yang lebih banyak jika jumlah objek atau jumlah cahaya pada adegan tersebut ditambah. Dengan demikian, untuk render dengan OGSS, waktu render juga berbanding lurus dengan banyaknya jumlah objek dalam adegan (semakin banyak objek semakin lama pula waktu yang

diperlukan). Untuk tabel ini rata-rata peningkatan waktu komputasi adalah 3,37.

Tabel 1. Hasil Eksperimen Adegan yang Sama dengan Resolusi yang Berbeda

Resolusi	Waktu (detik)		Peningkatan Waktu Komputasi (T)
	Normal	OGSS	
300x225	2	8	4,00
325x243	3	9	3,00
350x262	4	10	2,50
375x281	4	12	3,00
400x300	4	13	3,25
425x318	5	15	3,00
450x337	5	16	3,20
475x356	5	19	3,80
500x375	6	20	3,33
525x393	6	23	3,83
550x412	8	24	3,00
575x431	8	27	3,38
600x450	9	29	3,22
625x468	10	32	3,20
650x487	10	35	3,50
675x506	11	37	3,36
700x525	11	40	3,64
725x543	13	43	3,31
750x562	13	45	3,46
775x581	14	49	3,50
800x600	16	52	3,25

Tabel 2. Hasil Eksperimen dengan Adegan yang Berbeda dan Resolusi yang Sama

Indeks adegan	Deskripsi Adegan		Waktu Render (detik)		Peningkatan Waktu Komputasi (T)
	Jumlah Objek	Jumlah sumber cahaya	normal	OGSS	
1	1	1	7	22	3,14
2	2	1	7	24	3,43
3	3	1	8	25	3,13
4	4	1	9	29	3,22
5	5	1	9	29	3,22
6	1	2	8	24	3,00
7	2	2	8	25	3,13
8	3	2	8	28	3,50
9	4	2	9	31	3,44
10	5	2	9	31	3,44
11	1	3	8	26	3,25
12	2	3	9	30	3,33
13	3	3	9	31	3,44
14	4	3	9	34	3,78
15	5	3	9	37	4,11
16	1	4	9	29	3,22
17	2	4	9	31	3,44
18	3	4	10	34	3,40
19	4	4	11	36	3,27
20	5	4	12	42	3,50

Dari tabel 1 dan 2 di atas diperoleh hasil bahwa OGSS 4× membutuhkan waktu sekitar tiga kali lebih banyak daripada render normal. Hal ini cukup menguntungkan, karena mesin *render* harus merender citra yang resolusinya empat kali lebih besar. Waktu yang diperlukan tidak empat kali lebih lama, tetapi hanya sekitar tiga kali lebih lama.

Faktor lain yang mempengaruhi waktu *render* adalah jenis dan jumlah objek, dan jumlah sumber cahaya. Semakin kompleks dan banyak jumlah objek, semakin lama pula waktu yang diperlukan. Semakin banyak sumber cahaya, semakin lama pula waktu render yang diperlukan. Namun faktor objek dan sumber cahaya ini tidak cukup signifikan pengaruhnya apabila *render* dilakukan tanpa *anti-aliasing*.

5. KESIMPULAN

Dari hasil eksperimen di atas, dapat ditarik beberapa kesimpulan, yaitu:

1. Hasil citra yang di-render dengan OGSS memiliki kualitas yang lebih baik daripada citra hasil render tanpa OGSS, karena alias sudah tidak tampak lagi.
2. Waktu yang diperlukan untuk melakukan OGSS 4× adalah sekitar tiga kali lebih lama daripada waktu yang diperlukan untuk melakukan render secara normal. Hal ini tentunya menguntungkan, mengingat sebenarnya citra harus di-render empat kali lebih besar dari resolusi yang diinginkan. Dengan OGSS, waktu yang diperlukan hanya tiga kali lebih lama, bukan empat kali.
3. Dari perbandingan hasil render, maka dapat disimpulkan bahwa OGSS adalah metode anti-aliasing yang baik, baik ditinjau dari citra hasilnya maupun dari waktu yang diperlukan, karena meskipun melakukan render pada resolusi empat kali lebih besar, cukup diperlukan waktu sekitar tiga kali lipat lebih lama.

6. DAFTAR PUSTAKA

[1] Barron, Dave, *Multi-Sampling Anti-Aliasing Explained*, FiringSquad, http://firingsquad.gamers.com, 13 Februari 2001.

[2] Beets, Kristof, *Full Scene Anti Aliasing*, PVR-Net, 24 Mei 1998.

[3] Chaudhuri, Siddhartha., *FuzzyPhoton*, http://fuzzyphoton.tripod.com, 2002.

[4] Foley, James D., et. al., *Computer Graphics, Principles and Practice*, 2nd ed., Addison-Wesley Publishing Company, 1990.

[5] Genetti, Jon & Dan Gordon, *Ray Tracing with Adaptive Supersampling in Object Space*, Graphics Interface, 1993. pp. 70-77.

[6] Hearn, Donald & Pauline Baker, *Computer Graphics*, 2nd ed., Prentice Hall, Inc., 1994.

[7] Heiny, Loren. *Advanced Graphics Programming Using C/C++,* John Wiley & Sons, Inc., 1993.

[8] Hill Jr., Francis, *Computer Graphics*, Prentice-Hall, Inc., 1990.

[10] nVIDIA, *High Resolution Antialiasing (HRAA)*, GeForce3 Features, http://developer.nvidia.com.
[11] nVIDIA, *Frequently Asked GeForce 3 Questions*, FAQs, http://developer.nvidia.com.
[12] Pabst, Thomas, *High-Tech and Vertex Juggling – NVIDIA's New GeForce3 GPU*, Tom's Hardware, http://www.tomshardware.com, 27 Februari 2001.
[13] Rogers, David F. & J. Alan Adams, *Mathematical Elements for Computer Graphics*, 2nd ed., McGraw-Hill Publishing Company, 1990.
[14] Watt, Alan, *3D Computer Graphics*, 3rd ed., Addison-Wesley, 2000.
[15] Wilt, Nicholas, *Object-Oriented Ray Tracing in C++*, John Wiley & Sons, Inc., 1994.