IMPLEMENTASI RAYTRACING PADA MODEL WAJAH MANUSIA MENGGUNAKAN ALGORITMA MONTE CARLO

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Informatika

Oleh:

Nama : Canthes Widyawaty

No. Mahasiswa : 03523127

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2008

DAFTAR GAMBAR

DAFTAR ISI

DAFTAR PUSTAKA

[BER97] Berg, M. de, M. van Kreveld, M. Overmars, & O. Schwarzkopf. *Computational Geometry-Algorithms and Applications*. Berlin : Springer-Verlag, 1997.

[GLA89] Glassner, Andrew S. *An Introduction to Raytracing*. San Diego: Academic, 1989.

[HAL95] Halawa, Edward E & Sakti, Setyawan P. *Pemrograman dengan C/C++ dan Aplikasi Numerik*. Jakarta: Penerbit Erlangga, 1995.

[HAM07] Hamzah, Amir. *7 Objek Realistik 3ds Max*. Palembang: Maxikom, 2007.

[KAD04] Kadir, Abdul. *Panduan Pemograman Visual C++*. Yogyakarta: Penerbit Andi, 2004.

[KEL01] Keller, Alexander. *Monte Carlo and Beyond*. Makalah disampaikan di Universitas Ulm. Jerman, 30 Juli – 3 Agustus, 2001.

[LIL00] Liliana. *Pembuatan Perangkat Lunak untuk Memvisualisasikan Benda Tembus Pandang dengan Metode Raytracing*. Surabaya; Fakultas Teknologi Industri, Jurusan Teknik Informatika, Universitas Kristen Petra, 2000.

[RAH07] Raharjo, Budi. *Pemrograman C++*. Bandung: Penerbit Informatika, 2007.

[RUS98] Rushmeier, Holly. *A Basic Guide to Global Illumination*. Makalah disampaikan di 25th International Conference on Computer Graphics and Interactive Techniques. New York, 19 Juli, 1998.

[THA03] Tharom, Tabratas. *Pengolahan Citra pada Mobil Robot*. Jakarta: Elex Media Komputindo, 2003.

[WAN03] Wann Jensen, Henrik. *Monte Carlo Ray Tracing*. Siggraph Course 44, 2003.

[WAT92] Watt, Alan. *Advanced Animation and Rendering Techniques*. New York: N.Y ACM Press, 1992.

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah, segala puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayahnya. Sholawat dan salam kepada junjungan kita Nabi Muhammad SAW beserta keluarga dan para sahabat, serta orang-orang yang bertaqwa, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul *"Implementasi Raytracing pada Model Wajah Manusia Menggunakan Algoritma Monte Carlo"* sebagaimana mestinya. Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh kelulusan Strata 1 jurusan Teknik Informatika Universitas Islam Indonesia.

Penulis menyampaikan ucapan terimakasih dan penghargaan yang setinggi-tingginya atas bantuan, bimbingan dan dukungan dari berbagai pihak yang ikut serta demi kelancaran pelaksanaan Tugas Akhir kepada :

1. Bapak Fathul Wahid ST, M.Sc selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Yudi Prayudi, S.Si, M.Kom selaku Ketua Jurusan Teknk Informatika Universitas Islam Indonesia dan sebagai dosen pembimbing yang dengan penuh perhatian membimbing penulis dan memberikan masukan serta dorongan sehingga tugas akhir ini dapat terselesaikan.

3. Ayah dan mama tercinta yang telah membesarkan, mendidik, dan tiada hentinya mendoakan dan memberikan dukungan kepada penulis. Semoga kelulusan ini dapat menjadi kado kecil penulis bagi kedua orang tua tercinta.
4. Kakak dan adikku (Mbak Rika dan de Gagang), terima kasih atas persaudaraan yang indah.
5. Sahabat-sahabatku Aya, Kak Deddy, Mas Ai, Nonik, Indri, Fara, Eva, Tanti, Tifa, Ria, Ayok, Hanna, dan Itaxs, terima kasih atas persahabatan yang indah.
6. Teman-teman Informatika'03, terima kasih atas kekompakan dan kebersamaannya selama ini.
7. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah membantu sejak pengumpulan data sampai penyusunan Tugas Akhir ini.

Semoga amal ibadah dan kebaikan yang telah diberikan mendapatkan imbalan yang setimpal dari Allah SWT.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan, untuk itu penulis sangat mengharapkan kritik serta saran yang bersifat membangun untuk perbaikan di masa mendatang. Semoga Tugas Akhir ini bermanfaat untuk kita semua. Amin.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 17 Januari 2008

penyusun

LAMPIRAN

SINTAKS SCENE FILE

Sintaks *scene* file yang dijadikan input untuk aplikasi yaitu:

```
camera {                                    # Kamera
   background <r> <g> <b>                   # Warna latar, default hitam
   position <x> <y> <z>                     # Posisi dalam koordinat
   up <x> <y> <z>                           # Vektor up
   fov <angle>                              # Sudut bidang pandang
   target <x> <y> <z>                       # Koordinat target.
   depth <int>                              # Nilai rekursif, default 3
   diffusereflections <true |               # Mengaktifkan pemantulan diffuse
   false>
   adaptivesupersampling <true |            # Dilakukan untuk mengetahui apakah
   false>                                   dibutuhkan sampling secara penuh.
   samplesperpixel <int>                    # Jumlah sample per piksel.
}
material <name> {                           # Memberikan nama pada material yang
                                            digunakan untuk objek
   colour <r> <g> <b>                       # Warna material
   diffuse <double>                         # Koefisien diffuse
   diffusemap <texture filename>            # Diffuse texture map
   specular <double>                        # Koefisien specular
   reflective <double>                      # Koefisien sinar pantul
   reflectivescatter <double>               # Mengatur tingkat kekilapan (glossy)
   reflectivefeelers <double>               # Jumlah pemantulan
```

```
   refractive <double>          # Index pembiasan
}
plane {
   material <name>              # Material yang digunakan
   normal <x> <y> <z>           # Normal plane, yaitu bidang yang
                                mempunyai panjang dan lebar tanpa batas,
                                tanpa ketebalan, maupun lengkungan.
   displacement <double>        # Jarak terhadap bidang normal
}
sphere {
   material <name>              # Material yang digunakan
   position <x> <y> <z>         # Posisi dalam titik koordinat
   radius <double>              # Radius dalam titik koordinat
}
light {
   type <ambient | point |      # Tipe sinar. Radius akan diabaikan jika
   <area>                       diset ke <area>
   colour <r> <g> <b>           # Warna sinar
   position <x> <y> <z>         # Posisi dalam titik koordinat (untuk cahaya
                                non-ambient)
   nofalloff <true | false>     # Ketergantungan sinar terhadap jarak titik
                                cahaya dengan objek (hanya digunakan
                                untuk cahaya non-ambient)
   radius <double>              # Nilai radius (diisi jika tipe sinar yang
                                dipilih adalah area)
   numshadowfeelers <double>    # Bayangan yang dibentuk
}
triangle {                      # Poligon
```

```
   material <name>                  # Material yang digunakan
   point1 0.5 -0.5 0.5              # Vertices (Kumpulan vektor)
   point2 0.5 -0.5 -0.5
   point3 0.5 0.5 -0.5
   numemitters                      # Jumlah sinar yang dipancarkan
}
mesh {                              # Mesh objek
   file <filename>                  # File .obj yang dibaca
   overridematerial <name>          # Bersifat opsional, digunakan jika akan
                                    mengganti material awal pada objek
   position <x> <y> <z>             # Posisi dalam titik koordinat
   rotation <x> <y> <z>             # Rotasi objek
   scale <x> <y> <z>                # Skala objek
   interpolatenormals <true |       # Nilai true berarti objek akan dikenai
   false>                           interpolasi untuk menghasilkan citra yang
                                    lebih halus
}
```

Ada beberapa catatan yang harus diperhatikan dalam membuat file input untuk scene ini, diantaranya adalah:

- Pemberian nama material tidak lebih dari 255 karakter
- Tidak ada batasan dalam jumlah objek atau sinar
- Semua komponen warna memiliki nilai antara 0 sampai dengan 1
- Urutan penginisialisasian setiap komponen tidak harus berurutan. Misalnya dengan meletakkan objek di atas plane, kemudian sinar, lalu objek lagi.

LEMBAR PENGESAHAN PEMBIMBING

IMPLEMENTASI RAYTRACING PADA MODEL WAJAH MANUSIA MENGGUNAKAN ALGORITMA MONTE CARLO

TUGAS AKHIR

Oleh :

Nama : Canthes Widyawaty

No. Mahasiswa : 03 523 127

Yogyakarta, 23 Januari 2008

Pembimbing

(Yudi Prayudi, S. Si, M. Kom)

LEMBAR PENGESAHAN PENGUJI

IMPLEMENTASI RAYTRACING
PADA MODEL WAJAH MANUSIA
MENGGUNAKAN ALGORITMA MONTE CARLO

TUGAS AKHIR

Oleh :

Nama : Canthes Widyawaty

No. Mahasiswa : 03 523 127

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 31 Januari 2008

Tim Penguji

Yudi Prayudi, S.Si, M.Kom _____
Ketua

Syarif Hidayat, S. Kom _____
Anggota I

Lizda Iswari, ST _____
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

(Yudi Prayudi, S.Si, M.Kom)

tiada kata-kata yang patut diucapkan
kecuali terima kasih yang sedalam-dalamnya
kepada orang-orang yang aku sayangi
dan tiada pernah aku lupa,
aku persembahkan tugas akhir ini kepada mereka,
Ayah dan Bunda tersayang,
Saudara-saudaraku mb Rika dan Gagang
yang selalu aku kasihi,
Serta calon keponakan yang kutunggu

MOTTO

- " Ilmu adalah teman dalam kesendirian, dalil bagi agama dan penolong dalam suka dan duka. Ia adalah sahabat karib di kala sepi, teman yang paling baik dan paling dekat serta cahaya menuju surga. Dengan ilmu Allah akan mengangkat martabat ummat, lalu dijadikan-Nya mereka pimpinan dalam kebaikan ".

  (Mu'adz bin jabal RA)

- " Reach for the highest strive for the best live day by day and to God lave the rest ".

- " Orang terkuat bukan mereka yang selalu menang melainkan mereka yang tetap tegar ketika mereka jatuh ".

SARI

Telah banyak teori dan cara untuk menghasilkan suatu objek tiga dimensi yang divisualisasikan ke dalam layar dua dimensi dengan efek bayangan terhadap berbagai jenis material objek, diantaranya adalah dengan teknik proyeksi atau perspektif tiga dimensi ke dua dimensi. *Rendering* sebagai bagian dari komputer grafik merupakan satu cara untuk mendapatkan citra gambar yang bersifat realistis.

Setiap objek di alam yang dinyatakan ke dalam bentuk poligon apabila di-*render* dengan model penerangan sederhana akan menghasilkan citra gambar yang kurang baik, sehingga dapat menimbulkan gejala dimana intensitas permukaan di suatu poligon sama semua. Perbaikan yang dilakukan oleh Monte Carlo *Raytracing* telah menghasilkan citra yang lebih baik. Dengan perunutan sinar (*raytracing*) dari pengamat ke objek yang dipandang akan menghasilkan citra yang jauh lebih realistik walaupun menghabiskan waktu komputasi yang lama, karena teknik ini akan memperhitungkan nilai warna sinar dan nilai koefisien pantul dari benda dalam penentuan warna penggambaran pada layar.

Pada tugas akhir ini akan dirancang sebuah aplikasi yang dapat melakukan proses *rendering* suatu objek tiga dimensi ke dalam citra dua dimensi beserta efek bayangan yang terjadi dengan memperhitungkan beberapa macam unsur seperti pencahayaan, titik pandangan atau kamera, dan pemantulan sinar terhadap benda. Objek dibuat dengan memanfaatkan software 3D Studio Max 8 sedangkan bahasa pemrograman yang digunakan adalah C++.

Kata kunci : *Rendering*, Monte Carlo *raytracing*, komputer grafik, pencahayaan

TAKARIR

Ambience	Tingkat intensitas cahaya
Batch File	File teks untuk mengeksekusi perintah
Compiler	Penterjemah bahasa pemrograman
Computer Graphics	Grafika komputer
Export	Menyimpan file ke dalam bentuk lain
Flow Chart	Diagram alir
Geometri	Pengukuran berhubungan dengan ruang
Hardware	Perangkat keras
High-Poly	Menggunakan banyak polygon
Hue	Pendefinisian warna
Image	Citra
Input	Masukan
Interface	Antarmuka
Mesh	Kumpulan titik 3D yang terhubung
Objek	Obyek
Output	Keluaran
Pattern Recognition	Pengenalan pola
Pixel	Penentuan warna berdasarkan titik
Polygon	Suatu bentuk bangun
Rasterisasi	Pengisian piksel untuk membentuk titik
Raytracing	Penelusuran sinar
Render	Membangkitkan citra dari sebuah model
Resolusi	Tingkat detil suatu gambar
Scene	Bidang tempat objek berada
Shading	Pembentukan bayangan

Syntax	Notasi
Software	Perangkat lunak
Texel	Warna berdasarkan permukaan global
Tool	Alat
Vector	Titik
Vertex	Titik yang berjumlah banyak
View Plane	Bidang pandang
View Point	Titik pandang
Wireframe	Benda 2D yang dibentuk skalanya

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di masa sekarang ini, grafika komputer bukan sekedar suatu alat penghubung antara manusia dan komputer tetapi telah menjadi suatu alat visualisasi, mengungkapkan ekspresi dan komunikasi. Sudah banyak bidang yang memanfaatkan grafika komputer, misalnya untuk memudahkan antarmuka pada komputer, memungkinkan manipulasi dan visualisasi data dengan cara yang lebih berarti, untuk memudahkan eksplorasi medis pada tubuh manusia, dan menyediakan kecepatan grafik pada aplikasi yang bersifat *real-time* seperti simulator dan game. Dengan adanya grafika komputer para aktor digital dapat diciptakan untuk menggantikan berbagai adegan berbahaya dan sulit yang biasa dilakukan oleh aktor sungguhan. Selain itu menghasilkan produk desain berkualitas memiliki tingkat keakuratan grafis yang tinggi.

Penggambaran bentuk objek tiga dimensi sering dibutuhkan untuk keperluan visualisasi suatu objek beserta atribut-atributnya. Visualisasi ini digunakan untuk mempresentasikan bentuk suatu objek dengan pewarnaan yang mencakup beberapa faktor penting dalam pencahayaan. Salah satu teknik presentasi citra tersebut adalah dengan menggunakan teknik *raytracing*, yaitu teknik visualisasi yang dipakai untuk membuat sebuah citra yang realistis dengan cara mengkalkulasi jalur yang dilewati

oleh cahaya yang masuk ke dalam pandangan pengamat pada beberapa sudut. Jalur ini ditelusuri mulai dari titik pandang pengamat kembali ke objek yang akan digambar untuk menentukan apakah titik pandang pengamat mengenai benda atau tidak. Pada saat titik pandang pengamat mengenai benda, maka akan dihitung nilai warna piksel pada titik tabrak tersebut dengan memperhitungkan warna cahaya, posisi cahaya dan koefisien pantul benda.

Cara kerja dari aplikasi ini adalah dengan mengambil *input* dari file teks yang dibuat sebelumnya kemudian akan dibaca dan diproses oleh program. Proses ini akan menghasilkan *output* berupa citra dua dimensi sesuai dengan *input* yang telah ditentukan sebelumnya.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah bagaimana membuat sebuah aplikasi yang mampu mengimplementasikan *raytracing* pada model wajah manusia menggunakan algoritma Monte Carlo.

1.3 Batasan Masalah

Dalam melaksanakan suatu penelitian diperlukan adanya batasan-batasan agar tidak menyimpang dari yang telah direncanakan sehingga tujuan yang sebenarnya dapat tercapai. Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Objek wajah manusia dibuat dengan menggunakan *software* 3D Studio Max 8.0 dengan memanfaatkan *tool* yang ada di dalamnya.

2. Objek bersifat statis.
3. Bahasa pemrograman yang digunakan untuk membuat aplikasi adalah C++ dan *software* yang digunakan untuk membangun aplikasi yaitu DevCpp 5 Beta 9 dari BloodShed Software.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk menerapkan *raytracing* pada model wajah manusia menggunakan algoritma Monte Carlo.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah untuk membuat sebuah aplikasi yang mampu me-*render* model tiga dimensi menjadi citra dua dimensi yang terlihat realistis beserta efek bayangan yang dihasilkan.

1.6 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam penelitian ini adalah studi literatur. Studi literatur merupakan penelitian yang dilakukan dengan cara mempelajari materi-materi yang diperoleh berkaitan dengan topik tugas akhir seperti buku-buku, artikel-artikel, tulisan di internet, dan media informasi lainnya.

1.7 Sistematika Penulisan

Dalam penyusunan laporan tugas akhir disusun per bab yang berurutan untuk mempermudah pembahasan. Secara garis besar sistematika penulisan dari pokok permasalahan terdiri dari lima bab dengan masing-masing adalah sebagai berikut:

BAB I Pendahuluan

Bab ini menguraikan latar belakang yang mendasari pelaksanaan penelitian. Adapun sub-sub bab bagian ini adalah latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II Landasan Teori

Bab ini merupakan penjelasan yang memberikan informasi tentang teori-teori yang berhubungan dengan penelitian dan digunakan sebagai acuan di dalam pembahasan dan penyelesaian masalah.

BAB III Metodologi

Bab ini memuat uraian tentang langkah-langkah penyelesaian masalah yang digunakan selama melakukan penelitian.

BAB IV Hasil dan Pembahasan

Bab ini memuat uraian mengenai hasil penelitian yang dilakukan. Pembahasan dari setiap aktifitas dan bagian-bagian yang dilakukan dalam penelitian.

BAB V Simpulan dan Saran

Bab ini berisi kesimpulan-kesimpulan yang merupakan rangkuman dari hasil analisis dan saran yang perlu diperhatikan berdasarkan keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama penelitian.

BAB II

LANDASAN TEORI

2.1 Grafika Komputer

Grafika komputer (*Computer graphics*) adalah bagian dari ilmu komputer yang berkaitan dengan pembuatan dan manipulasi gambar (visual) secara digital. Bentuk sederhana dari grafika komputer adalah grafika komputer dua dimensi yang kemudian berkembang menjadi grafika komputer tiga dimensi, pemrosesan citra *(image processing),* dan pengenalan pola *(pattern recognition).* Grafika komputer sering dikenal juga dengan istilah visualisasi data.

Grafika komputer tiga dimensi (*3D Computer graphics*) dapat diartikan sebagai representasi dari data geometrik tiga dimensi sebagai hasil dari pemrosesan dan pemberian efek cahaya terhadap grafika komputer dua dimensi. Hasil ini kadang ditampilkan secara nyata (*real time*) untuk keperluan simulasi. Secara umum prinsip yang dipakai mirip dengan grafika komputer dua dimensi dalam hal penggunaan algoritma, grafika vektor, *wireframe model*, dan grafika rasternya.

Grafika komputer tiga dimensi sering disebut sebagai model tiga dimensi. Namun, model tiga dimensi ini lebih menekankan pada representasi matematis untuk objek tiga dimensi. Data matematis ini belum bisa dikatakan sebagai gambar grafis hingga saat ditampilkan secara visual pada layar komputer atau printer. Proses

penampilan suatu model matematis ke bentuk citra dua dimensi biasanya dikenal dengan proses *rendering* tiga dimensi.

2.2 Rendering

2.2.1 Definisi Rendering

Rendering adalah proses pembangkitan citra dari suatu model dengan bantuan program komputer, sedangkan model adalah uraian dari objek tiga dimensi dalam suatu struktur data atau bahasa tertentu. Di dalam proses *rendering* berisi geometri, *viewpoint*, tekstur, pencahayaan, dan *shading*.

Rendering merupakan salah satu topik utama dalam grafika komputer tiga dimensi yang dalam prakteknya juga berhubungan dengan bidang-bidang lainnya. Di dalam urutan langkah untuk pengolahan grafik *rendering* adalah langkah utama terakhir, fungsinya memberikan penampilan akhir pada model dan animasi.

Rendering banyak digunakan dalam bidang arsitektur, *video game*, simulator, film atau efek khusus dalam berbagai acara televisi, dan desain visual dengan impelementasi teknik yang berbeda-beda untuk masing-masing bidang. *Rendering* sering dipandang sebagai teknik yang berdasarkan pada campuran beberapa disiplin seperti ilmu fisika, persepsi visual, matematika, pengembangan perangkat lunak.

Ketika masih berupa *pre-image*, yang pada umumnya berupa sketsa *wireframe*, telah selesai dikerjakan, *rendering* digunakan untuk menambahkan tekstur, pencahayaan, *bump mapping*, atau posisi relatif terhadap objek lainnya. Hasil akhir yang diharapkan berupa sebuah citra utuh untuk diperlihatkan kepada penonton

atau konsumen. Untuk film animasi, beberapa citra harus menjalani proses *rendering* dalam *frame-frame* terpisah dan kemudian disatukan kembali menjadi suatu animasi yang lengkap.

2.2.2 Teknik-Teknik Rendering

Banyak algoritma rendering yang telah diteliti dan dikembangkan, dan software yang digunakan untuk me-*render* telah mengadopsi teknik-teknik yang berbeda untuk mendapatkan sebuah citra sebagai hasil akhir. Merunut setiap sinar pada *scene* dapat menjadi sesuatu yang tidak praktis dan membuang banyak waktu. Untuk mengatasi masalah tersebut, para peneliti telah berusaha mengembangkan teknik-teknik baru agar rendering menjadi lebih efisien. Teknik-teknik tersebut seperti rasterisasi, *ray casting*, *radiosity*, dan *raytracing* yang merupakan topik pembahasan pada penelitian ini.

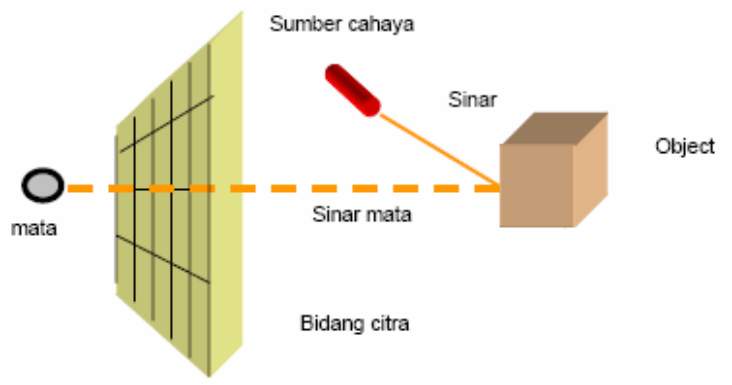
2.3 Raytracing

2.3.1 Definisi Raytracing

Raytracing adalah teknik pencahayaan global yang mengenali dan me-*render* suatu gambar pada basis piksel demi piksel. Ide *raytracing* adalah menelusuri cahaya untuk tiap piksel, dari mata atau titik pandang lewat piksel dan menuju ke tempat kejadian atau *scene*. Ada sejumlah cahaya yang tidak hanya berasal dari sumber cahaya dan ini menimbulkan masalah. Oleh karena itu, dengan merunut cahaya yang arahnya berlawanan dengan penyebaran cahaya, saat suatu objek ditemukan,

dihitunglah kontribusi tiap sumber cahaya yang tampak dari garis perpotongan. Garis cahaya yang merambat ke dalam lingkungan sesuai dengan model pemantulan dan perpotongan hingga diabaikan kontribusi yang mungkin berasal dari objek yang lebih jauh. Nilai akhir intensitas dihitung sebagai jumlah seluruh kontribusi perpotongan. Untuk menciptakan suatu citra layar dari titik pandang yang berbeda dibutuhkan pengulangan seluruh algoritma, tak ada hasil dari tempat kejadian atau *scene* yang dapat digunakan.

Pada gambar 2.2 di bawah, merupakan gambaran sederhana untuk mengilustrasikan suatu proses yang menggunakan *raytracing* untuk menentukan permukaan yang diinginkan. Suatu titik mata (*eye*) dan bidang citra (*view plane*) yang terpikselkan, suatu sinar (*eye ray*) ditembakkan dari titik mata, lewat sebuah piksel, dan menuju *scene*. Sinar yang memotong objek adalah objek yang terlihat pada piksel bidang citra (*view plane*) dan piksel itu ditandai dengan warna objek [THA03].



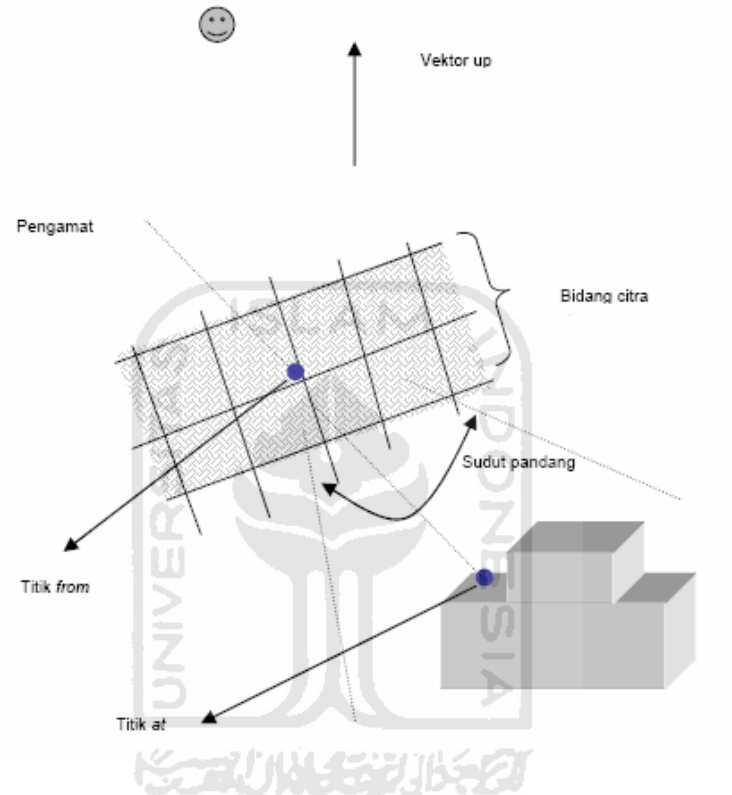
Gambar 2.1 Diskripsi sederhana ray tracing

Sinar yang ditembakkan pada tiap sumber cahaya dari titik tempat sinar mata dan objek perpotongan. Sinar bayangan membantu dalam menentukan intensitas dan warna dari titik perpotongan objek atau sinar mata. Oleh karena itu, warna dan intensitas diberikan ke piksel. Proses ini dilakukan terhadap piksel dalam bidang citra. Saat terpenuhi, bidang citra mendapatkan gambar dari tempat kejadian dengan resolusi yang ditentukan oleh pikselisasi bidang citra. Energi cahaya yang datang ke permukaan yang bukan kaca sempurna dipantulkan secara acak (pemantulan yang tidak sederhana). Untuk permukaan spekular, arah penyebaran adalah variabel acak sebuah distribusi yang bergantung pada sudut datang dan sifat permukaan. Pada tiap-tiap perpotongan, suatu sinar pantul ditembakkan dalam arah yang ditentukan dengan menyampling distribusi yang digunakan untuk memodelkan penyebaran secara fisik.

2.3.2 Menggunakan Suatu Model Kamera

Raytracing menggunakan pemodelan kamera yang ditujukan untuk menentukan begaimana memproyeksikan dengan inisialisasi sinar berarah mundur terhadap *interface* komputer, seperti yang akan diperlihatkan pada gambar 2.3. Komponen terdiri atas titik *from*, titik pengamat, titik *at*, bidang citra, sudut pengamat, kamera objek, dan vektor *up*. Titik *from* merupakan lokasi kamera atau pengamat, titik *at* adalah tempat kamera yang mengarah di depan objek. Bidang citra, secara konsep terletak antara pengamat dan model dunia, yang berisi citra yang dibuat. Vektor *up* menentukan bagaimana bidang citra diarahkan mengenai kamera.

Yang terakhir adalah dunia yang diambil dalam satu gambar yang tergantung pada sudut pengamat [THA03].



Gambar 2.2 Pemodelan ray tracing dengan sebuah kamera

Aspek penting dari model kamera adalah memberikan arsiran objek yang menggunakan proyeksi perspektif dan memperoleh rasa kedalaman pada citra dua dimensi. Proyeksi perspektif membuat objek yang muncul terputus-putus (terdistorsi) saat mendekati pengamat, dan garis paralel berkumpul (konvergen) saat menjauh. Sejumlah distorsi berhubungan dengan sudut pandang. Umumnya semakin besar sudut pandang, semakin besar pula distorsi.

Raytracing memproyeksikan sinar cahaya dari pengamat lewat tiap piksel di dalam citra yang menggunakan kekuatan model kamera. *Raytracer* (pengusut sinar) menerapkan piksel terhadap warna objek yang sinarnya mengenai objek tersebut. Suatu *raytracer* me-*render* atau memberi arsiran pada layar komputer yang realistik, harus diperhitungkan pemantulan, bayangan, pencahayaan, dan bermacam-macam keberadaan permukaan objek. Kumpulan konstanta menggambarkan tiap objek. Konstanta menetapkan ukuran, lokasi, dan permukaan objek seperti halus dan bersinar, warna dan *hue*-nya, dan sebagainya. Konstanta ini dimasukkan ke dalam persamaan yang menentukan tipe khusus objek.

2.3.3 Model-Model Pencahayaan

Model iluminasi mengacu pada berbagai tipe sumber cahaya yang dapat disimulasi oleh grafik komputer. Grafik komputer berusaha memodelkan bermacam-macam tipe sumber cahaya yang dapat ditemukan pada dunia nyata. Sumber cahaya dapat dikelompokkan ke dalam beberapa tipe, yaitu:

a. Sumber cahaya titik
b. Sumber cahaya berarah (*directional*)
c. Sumber cahaya setempat

Hal lain yang dianggap penting pada model pencahayaan adalah:

a. Intensitas menurun dengan jarak
b. Keseimbangan level iluminasi

c. Bayangan

Sumber cahaya titik merupakan sumber penerangan yang paling sederhana untuk pemodelan, karena itu kita harus meletakkan posisinya dalam ruang bersamaan dengan intensitas dan warnanya. Sumber cahaya yang berarah dapat dikenali dari warna, intensitas, dan arahnya. Contohnya sinar cahaya yang tiba pada sebidang kecil tanah di atas permukaan bumi akan kelihatan paralel dan memiliki intensitas yang sama. Sudut antara vektor dan permukaan normal digunakan untuk menentukan kuantitas energi cahaya. Sumber cahaya *spot* (setempat) mensimulasikan tingkah lakunya yang menciptakan pancaran cahaya yang dikendalikan dalam bentuk *cone* (kerucut). Kisaran minimum parameter dibutuhkan untuk memodelkan sumber cahaya ini, terdiri atas posisi, intensitas, warna, dan sudut setempat.

2.3.4 Model Pemantulan

Tiga tipe pemantulan cahaya terdiri atas *ambient, diffuse*, dan *specular*. Cahaya *ambient* mensimulasikan level cahaya konstan yang dapat menyebabkan banyak pantulan. Cahaya *diffuse* diciptakan oleh permukaan yang memiliki kekasaran sehingga menyebabkan cahaya yang datang dipantulkan sama ke semua arah. Cahaya *specular* mengacu pada cahaya yang dipantulkan oleh permukaan licin dan menciptakan suatu sinar terang dari sumber cahaya yang teriluminasi dan menerangi lingkungan sekitarnya. Warna objek ditentukan oleh radiasi penerangan (iluminasi) bagian-bagian yang diserap serta dipantulkan.

2.3.5 Perpotongan Sinar dan Objek

Untuk menentukan jatuhnya sinar *raytracing* pada objek yang terdekat, diperkirakan akan diperlukan waktu yang relatif tidak sedikit. Pengujian pada *raytracing* dilakukan dengan mendapatkan objek yang tepat berpotongan dengan sinar. *Raytracer* pada beberapa fungsi berusaha memecahkan masalah ini dengan memanfaatkan perhitungan matematis dan pemilihan objek yang tepat. Contoh yang paling sederhana adalah bidang. Setiap titik objek dapat ditentukan dari persamaannya. *Raytracer* mensubtitusi persamaan sinar ke dalam persamaan objek dan mencari pemecahan permasalahannya. Jika ada, maka koordinatnya merupakan tempat sinar memotong objek. Jika tidak, maka sinar tidak mengenai objek dan sisa objek lainnya harus diuji lagi.

2.4 Integrasi Algoritma Monte Carlo

2.4.1 Dasar Algoritma Monte Carlo

Algoritma Monte Carlo adalah algoritma komputasi yang termasuk dalam kategori algoritma numerik yang banyak digunakan untuk menemukan solusi problem matematis yang terdiri dari banyak variabel yang sulit dipecahkan, misalnya dengan kalkulus integral, atau metode numerik lainnya. Istilah Monte Carlo *Sampling Technique* digunakan untuk menggambarkan kemungkinan penggunaan data sampel dalam metode Monte Carlo yang dapat diketahui atau diperkirakan distribusinya. Simulasi ini menggunakan data yang sudah ada (*historical data*) yang sebenarnya dipakai dalam simulasi untuk tujuan lain [KAK03].

Penggunaan klasik metode ini adalah untuk mengevaluasi integral definit, terutama integral multidimensi dengan syarat dan batasan yang rumit. Algoritma ini memerlukan pengulangan dan perhitungan yang amat kompleks, namun banyak digunakan dalam fisika komputasi dan bidang terapan lainnya karena terbukti efisien dalam memecahkan persamaan diferensial integral medan radians, sehingga digunakan dalam perhitungan iluminasi global yang menghasilkan gambar-gambar fotorealistik untuk model tiga dimensi.

Algoritma Monte Carlo dianggap sebagai penemuan dari Stanislaw Ulam, seorang ahli matematika yang bekerja untuk John Von Neumann di proyek *United State's Manhattan* selama perang dunia ke dua. Ulam adalah orang utama yang diketahui merancang bom hidrogen dengan Edward Teller pada tahun 1951. Dia menemukan algoritma Monte Carlo pada tahun 1946 saat memikirkan peluang memenangkan suatu permainan kartu soliter.

Algoritma Monte Carlo, sebagaimana yang dipahami saat ini, melingkupi sampling statistik yang digunakan untuk memperkirakan solusi permasalahan kuantitatif. Ulam tidak menciptakan sampling statistik. Metode ini sebelumnya digunakan untuk menyelesaikan permasalahan kuantitatif dengan proses fisik, seperti pelemparan dadu atau pengocokan kartu untuk menurunkan sampel. W.S. Gosset, yang mempublikasikan karyanya dengan nama "*Student*", secara acak menarik sampel ukuran jari tengah dari 3000 kriminal untuk mensimulasikan dua distribusi normal berhubungan. Dia mendiskusikan algoritma Monte Carlo dalam dua publikasinya pada tahun 1908. Kontribusi Ulam diakui dalam potensi penemuan baru

komputer elektronik untuk mengotomasi penarikan sampel. Bekerja dengan John von Neuman dan Nicholas Metropolis, Ulam mengembangkan algoritma untuk implementasi komputer, juga mengeksplor alat transformasi permasalahan tidak acak ke dalam bentuk acak yang akan memfasilitasi solusinya melalui penarikan sampel acak.

Nama Monte Carlo diberikan oleh Metropolis dan dipublikasikan pertama kali pada tahun 1949. Nama algoritma Monte Carlo diberikan sesuai dengan nama salah satu kota di Monaco, yaitu Monte Carlo, kota yang menjadi pusat kasino-kasino besar di dunia. Permainan peluang seperti roda rolet, dadu dan mesin slot yang ada di kasino menunjukkan perilaku acak. Perilaku acak dalam permainan peluang adalah sama dengan bagaimana simulasi memilih nilai variabel secara acak untuk mensimulasikan model. Ketika sebuah dadu dilemparkan, beberapa peluang kemunculannya adalah 1, 2, 3, 4, 5 atau 6, tapi nilai pasti untuk sebuah lemparan tertentu tidak dapat diketahui. Hal itu sama dengan variabel yang mempunyai kisaran nilai diketahui tapi tidak diketahui nilai pasti untuk waktu atau kejadian tertentu. Pemahaman metode Monte Carlo dapat dilakukan dengan memikirkan bahwa hal tersebut merupakan teknik umum integrasi numeris. Setiap aplikasi metode Monte Carlo dapat direpresentasikan sebagai integral terbatas.

2.4.2 Karakteristik Algoritma Monte Carlo

Ada beberapa istilah penting dalam mempelajari algoritma Monte Carlo, seperti variabel acak (*continuous random variable*), fungsi densitas peluang

(*probability density function*), nilai ekspektasi (*expected value*), dan variansi (*variance)*. Variabel acak *x* adalah jumlah vektor yang diambil secara acak dari ruang *S*. Nilai *x* dapat digambarkan dari distribusi nilai yang diambil, yang ditentukan dari fungsi densitas peluang. Sehingga hubungan antara variabel acak *x* dan fungsi densitas peluang *p* dapat dinotasikan $x \sim p$. Jika harga *x* melebihi *S*, maka peluang *x* pada $S_i \subset S$ adalah:

$$\Pr ob(x \in S_i) = \int_{Si} p(x) d\mu \quad (p : S \to R^1) \qquad \textbf{(2.1)}$$

Pada grafika komputer, *S* biasanya merupakan area ($d\mu = dA = dxdy$) atau satuan arah (titik pada satu unit bola: $d\mu = d\omega = \sin\theta d\theta d\phi$). Fungsi densitas peluang menggambarkan hubungan variabel acak yang memiliki nilai tertentu: jika $p(x_1) = 6.0$ dan $p(x_2) = 3.0$, maka variabel acak dengan fungsi densitas *p* terlihat seperti memiliki nilai yang lebih mendekati x_1 daripada x_2. Fungsi densitas *p* memiliki dua karakteristik, yaitu:

$p(x) \geq 0$ (probabilitas non-negatif) **(2.2)**

dan

$$\int_S p(x) d\mu = 1 \quad (\Pr ob(x \in S) = 1 \qquad \textbf{(2.3)}$$

Sebagai contoh, variabel random ξ memiliki nilai diantara 0 (inklusif) dan 1 (*non*-inklusif) dengan peluang yang sama, maksudnya setiap nilai ξ adalah sama. Hal ini menunjukkan bahwa:

$$f(\xi) = \begin{cases} 1 \\ 0 \end{cases} \text{ if } 0 \leq \xi \leq 1 \qquad \textbf{(2.4)}$$

Menurut persamaan tersebut, ruang yang ditempati ξ terletak pada interval [0,1]. Maka peluang ξ pada interval [a,b] $\in$ [0,1] adalah:

$$\Pr ob(a \le \xi \le b) = \int_a^b 1 dx = b - a \qquad \textbf{(2.5)}$$

Sebagai contoh,variabel acak dua dimensional α mendistribusikan variabel acak yang sama pada radius R. Karena sama, maka $p(\alpha)$ bersifat tetap. Dari persamaan 2.3 dan dasar bahwa area merupakan ukuran yang sesuai, dapat ditarik kesimpulan bahwa $p(\alpha) = 1/(\pi R^2)$. Ini berarti peluang α berada pada subset S_1 tertentu adalah:

$$\Pr ob(\alpha \in S_1) = \int_{S1} \frac{1}{\pi R^2} dA \qquad \textbf{(2.6)}$$

Misalnya S_i adalah ruang yang lebih dekat ke titik pusat. Jika diubah ke sistem koordinat, kemudian α ditunjukkan sebagai pasangan (r,θ), dan S_i adalah ruang dimana $r < R/2$. Namun karena α sama tidak berarti bahwa *theta* atau r juga harus sama. Area diferensial dA menjadi $r\, dr\, d\theta$, sehingga:

$$\Pr ob(r < \frac{R}{2}) = \int_0^{2\pi} \int_0^{\frac{R}{2}} \frac{1}{\pi R^2} r dr d\theta = 0.25 \qquad \textbf{(2.7)}$$

Harga rata-rata yang merupakan fungsi f dari variabel acak satu dimensi disebut sebagai nilai ekspektasi yang dinotasikan sebagai $E(f(x))$:

$$E(f(x)) = \int_S f(x) p(x) d\mu \qquad \textbf{(2.8)}$$

Nilai ekspektasi dari sebuah variabel acak satu dimensi dapat dihitung dengan menggunakan $f(x) = x$. Nilai ekspektasi dari penjumlahan dua buah variabel acak adalah jumlah dari nilai ekspektasi masing-masing variabel x dan y, yaitu:

$$E(x + y) = E(x) + E(y) \quad \textbf{(2.9)}$$

Karena fungsi variabel acak adalah variabel acak itu sendiri, maka ekspektasi linier yang dapat diterapkan adalah:

$$E(f(x) + g(y)) = E(f(x)) + E(g(y)) \quad \textbf{(2.10)}$$

Hasil dari penjumlahan dua variabel acak adalah variabel acak juga. Hasil ini tergantung pada titik acak pada radius R. Jarak yang diinginkan r pada titik pusat dari radius R yaitu:

$$E(r) = \int_0^{2\pi} \int_0^R \left(\frac{1}{\pi R^2} r \right) r dr d\theta = \frac{2R}{3} \quad \textbf{(2.11)}$$

Variansi yang dinotasikan dengan $var(x)$ dari sebuah variabel acak satu dimensi merupakan nilai ekspektasi dari kuadrat antara x dan $E(x)$, maka:

$$var(x) = E([x - E(x)]^2) \quad \textbf{(2.12)}$$

Dari hasil manipulasi aljabar didapatkan pernyataan:

$$var(x) = = E(x^2) - [E(x)]^2 \quad \textbf{(2.13)}$$

Pernyataan $E([x - E(x)]^2)$ digunakan untuk pencarian variansi, sedangkan persamaan $E(x^2) - [E(x)]^2$ digunakan dalam perhitungan. Variansi dari penjumlahan variabel-variabel acak adalah jumlah variansi jika variabel bersifat independen. Penjumlahan ini adalah alasan mengapa banyak digunakan dalam analisa model peluang. Akar

kuadrat dari variansi disebut standar deviasi, dinotasikan dengan σ, yang memberikan indikasi mutlak dari nilai ekspektasi.

Banyak kasus yang melibatkan penjumlahan variabel acak independen x_i, dimana variabel-variabel berbagi fungsi densitas *f*. Variabel seperti itu disebut variabel acak terdistribusi kembar independen. Apabila hasil penjumlahan dibagi dengan sejumlah variabel, akan didapatkan perkiraan *E(x)*:

$$E(x) \approx \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \textbf{(2.14)}$$

Apabila nilai *N* semakin besar, variansi perkiraan *E(x)* akan menurun. Nilai *n* harus besar untuk memastikan estimasi yang benar-benar mendekati. Bagaimanapun juga, tidak ada hal yang pasti di dalam Monte Carlo, karena hanya membangkitkan estimasi yang bagus secara statistik. Untuk memastikan dipakai $n = \infty$. Estimasi ini dinyatakan dengan *Law of Large Numbers*:

$$\Pr ob\left[E(x) = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} x_i \right] = 1 \qquad \textbf{(2.15)}$$

2.5 DevCpp Versi 5 Beta

DevCpp adalah sebuah aplikasi IDE (*Integrated Development Environment*) yang dikembangkan oleh Bloodshed Software dan ditujukan untuk pengembangan C atau C++ di atas sistem operasi *Windows.* DevCpp merupakan sebuah aplikasi yang bersifat gratis atau *freeware*. DevCpp menggunakan *compiler MinGW* untuk mengkompilasi kode.

Salah satu keunggulan DevCpp adalah memiliki *library* sendiri serta mendukung penggunaan *template* yang dapat digunakan untuk membangun aplikasi tertentu. *Template* ini berisi seperangkat *source code* dan file *project* yang bisa dipakai untuk membuat sebuah aplikasi. *Template* ini bisa ditambahkan dengan menginstal *pack* khusus sesuai dengan kebutuhan user.

Ada tiga macam pilihan jenis *project* yang bisa dibangun menggunakan DevCpp, yaitu basic, introduction, dan multimedia. Pada *basic project*, user dapat memilih empat macam kategori, seperti Windows Application, Console Aplication, Static Library, dan DLL. Introduction digunakan untuk membuat aplikasi "Hello World" klasik. Sedangkan pada multimedia ada tiga pilihan utama, yaitu OpenGL, SDL dan GL, serta SDL.

Selain menggunakan Windows API, DevCpp juga memiliki *library* sendiri yang dapat dipanggil untuk membangun aplikasi. *Library* ini disebut SDL API, yang terbagi menjadi delapam macam subsistem terdiri dari Audio, CDROM, Event Handling, File I/O, Joystick Handling, Threading, Timers and Video.

Sebelum salah satu dari subsistem tersebut dapat dipakai, masing-masing harus diinisialisasikan terlebih dahulu dengan memanggil `SDL_Init` atau `SDL_InitSubSystem`. `SDL_Init` harus dipanggil sebelum semua fungsi SDL lainnya. Pada pemangilan tersebut, secara otomatis juga akan menginisialisasikan subsistem Event Handling, File I/O dan Threading serta akan mengambil parameter yang

menentukan subsistem lainnya yang akan dipanggil. Sebagai contohnya, untuk inisialisasi subsistem default dan video akan dipanggil:

```
SDL_Init ( SDL_INIT_VIDEO );
```

Kemudian untuk menginisialisasikan subsistem default, video dan Timers akan dilakukan pemanggilan:

```
SDL_Init ( SDL_INIT_VIDEO | SDL_INIT_TIMER );
```

`SDL_Init` dilengkapi oleh `SDL_Quit` atau `SDL_QuitSubSystem`. `SDL_Quit` akan menutup semua subsistem, termasuk subsistem default, dan harus dipanggil sebelum keluar dari aplikasi SDL.

Dengan menggunakan `SDL_Init` dan `SDL_Quit`, *programmer* dapat membangun hampir semua apliaksi dasar SDL. Namun, hal yang tetap tidak boleh dilakukan adalah penanganan kesalahan (*error handling*). Banyak fungsi SDL mengembalikan nilai yang menandakan kesuksesan atau bahkan kegagalan dalam kompilasi. Misalnya, `SDL_Init` mengembalikan nilai -1 jika tidak dapat menginisialisasikan sebuah subsistem. SDL mendukung banyak fasilitas yang berguna yang mengizinkan *programmer* untuk menentukan secara tepat masalah yang dihadapi, setiap saat terjadi kesalahan di dalam SDL sebuah pesan kesalahan akan dimunculkan dengan cara pemanggilan `SDL_GetError.`

BAB III

METODOLOGI

3.1 Metode Analisis

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh ke dalam beberapa bagian untuk mendapatkan dan menganalisis data yang diperlukan untuk melakukan perancangan perangkat lunak. Analisis sistem dilakukan untuk mengetahui semua permasalahan serta kebutuhan yang diperlukan dalam pengembangan aplikasi. Analisis dilakukan dengan mencari dan menentukan permasalahan yang dihadapi serta semua kebutuhan seperti analisis kebutuhan proses, analisis kebutuhan *input*, analisis kebutuhan *output*, kebutuhan antarmuka, analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan perancangan perangkat lunak.

Aplikasi yang dibangun dalam penelitian ini dirancang menggunakan algoritma Monte Carlo Raytracing. Metode analisis yang digunakan adalah dengan mencari data dan teori mengenai pengimplementasian raytracing menggunakan algoritma Monte Carlo. Sedangkan untuk melihat proses yang mencakup proses *input* maupun *output* dalam aplikasi ini dinyatakan dengan diagram alir *(Flow Chart)*. Pada tahap ini digunakan notasi-notasi untuk menggambarkan arus data yang akan sangat membantu dalam proses komunikasi dengan pemakai.

3.2 Hasil Analisis

Dari data yang diperoleh selama penelitian dan setelah dilakukan proses analisis, kebutuhan yang diperlukan dalam membangun aplikasi meliputi kebutuhan masukan (*input*), kebutuhan proses, serta kebutuhan keluaran (*output*), kebutuhan perangkat lunak (*software*), kebutuhan perangkat keras (*hardware*), dan kebutuhan antarmuka (*interface*).

3.3 Analisis Kebutuhan Sistem

3.3.1 Analisis Kebutuhan Input

Input atau masukan yang dibutuhkan untuk aplikasi ini adalah file model yang telah dikonversi ke dalam format yang berbasis teks. Tujuan dari konversi ini adalah agar *input* dapat terbaca oleh aplikasi.

3.3.2 Analisis Kebutuhan Proses

Proses rendering objek pada aplikasi ini menggunakan metode raytracing dengan mengimplementasikan algoritma Monte Carlo.

3.3.3 Analisis Kebutuhan Output

Output atau keluaran yang diharapkan dari penelitian ini adalah sebuah aplikasi yang mampu me-*render* input objek wajah manusia tiga dimensi menjadi sebuah gambar wajah manusia dua dimensi.

3.3.4 Kebutuhan Antarmuka

Kebutuhan antar muka *(interface)* yang dibuat mempertimbangkan kondisi supaya mudah digunakan oleh pemakai *(user).* Pembuatan interface ini dibuat atas dasar observasi dari literature dan *software-software* yang sudah ada.

Perancangan antarmuka aplikasi ini dibuat dengan menggunakan DevCpp 5 Beta 9. Kemudian untuk pembuatan model *software* yang digunakan adalah 3D Studio Max 8.

3.3.5 Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan dalam pemodelan wajah manusia ini antara lain:

1. Microsoft Windows XP sebagai sistem operasi yang digunakan dalam mengimplementasikan perangkat lunak.
2. 3D Studio Max 8 sebagai *software* yang digunakan untuk membuat objek wajah manusia.
3. DevCpp 5 Beta 9 sebagai *software* yang digunakan untuk membangun aplikasi.
4. 3DWin5 versi 5.6 sebagai *software* untuk mengkoversi objek yang akan dijadikan sebagai masukan sistem.

3.3.6 Kebutuhan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk membuat dan menjalankan aplikasi Monte Carlo *raytracer* dengan objek model wajah manusia ini adalah:

1. Komputer dengan prosesor Intel Centrino Duo
2. RAM 1 GB
3. Harddisk 40 GB
4. Monitor
5. Keyboard
6. Mouse

3.4 Perancangan Perangkat Lunak

3.4.1 Metode Perancangan

Metode perancangan yang digunakan untuk mengembangkan sistem ini berupa metode berarah aliran data dengan menggunakan *flowchart* atau diagram alir proses.

Perancangan ini dimulai dengan perancangan secara umum dengan tujuan untuk membuat suatu sistem baru yang logik dan konseptual. Hasil dari tahap ini akan membentuk model esensial yaitu apa yang harus dilakukan untuk memenuhi kebutuhan dari pemakai dan menggambarkan bagaimana nantinya sistem akan diimplementasikan.

3.4.2 Hasil Perancangan

Berdasarkan hasil analisis yang telah dilakukan, maka dapat diketahui apa saja yang menjadi masukan sistem, keluaran sistem, metode yang digunakan sistem, serta antar muka sistem yang dibuat, sehingga sistem yang dibuat nantinya sesuai dengan apa yang diharapkan. Pengembangan aplikasi ini digambarkan dengan menggunakan *flowchart* sehingga dapat terlihat urutan proses yang dilakukan dalam pembuatan aplikasi, dan selanjutnya dari *flowchart* ini akan dibuat *pseudocode* algoritma Monte Carlo untuk lebih memudahkan pembuatan aplikasi.

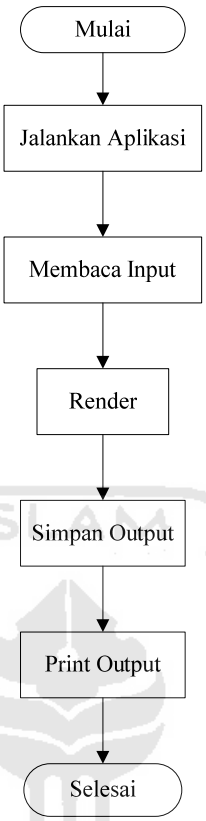
Perancangan sistem ini akan dibagi menjadi beberapa subsistem yaitu :

1. Perancangan *Flowchart*
2. Perancangan *Pseudocode*
3. Perancangan Antarmuka

3.4.2.1 Perancangan Flowchart

Diagram alir sistem merupakan visualisasi langkah kerja dalam wujud *flowchart*. Penggunaan *flowchart* ini akan memberikan keuntungan, yaitu seluruh bagian sistem dapat ditampilkan mulai dari bagian input, proses-proses yang terlibat di dalam sistem sampai dengan bagian output sistem. Penggunaan *flowchart* juga memberikan kemudahan dalam menilai alur kerja serta efektivitas dari kerja sistem.

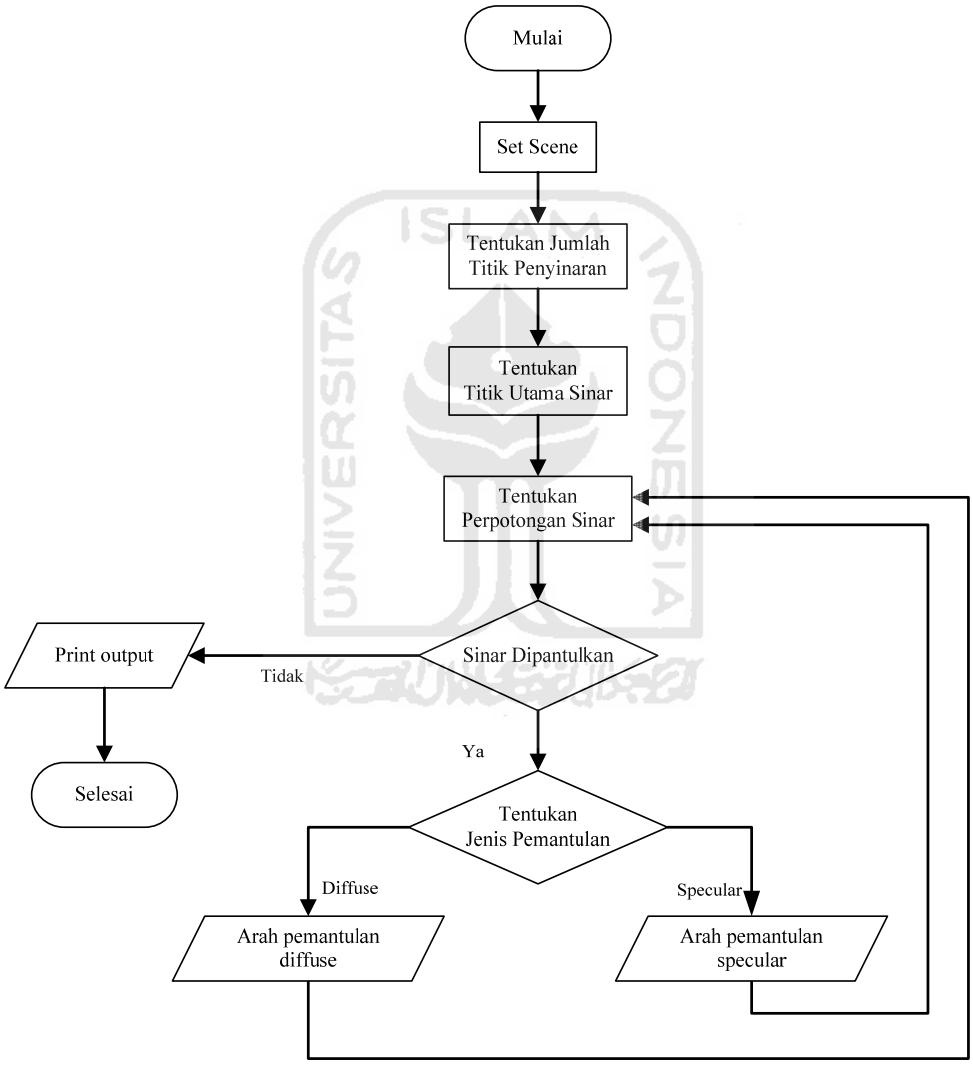
Flowchart atau diagram alir proses berikut merupakan mekanisme kerja aplikasi secara umum.



Gambar 3.1 Flowchart Cara Kerja Aplikasi

Pada gambar 3.1, kerja aplikasi dimulai dengan membuka aplikasi, dinotasikan dengan Jalankan Aplikasi. Aplikasi ini dijalankan lewat *Command Prompt* Windows. Sedangkan untuk membukanya adalah lewat file yang berisi sintaks khusus terdiri dari nama file yang akan dijalankan dan resolusi *output* yang diinginkan. Setelah aplikasi berhasil dijalankan, maka sistem kemudian membuka file *input* yang akan diproses, dinotasikan dengan Membaca Input. Apabila pembacaan *input* berhasil dilakukan dan tidak terjadi *error* maka aktivitas selanjutnya adalah memulai proses *rendering* yang dinotasikan dengan Render. Hasil dari proses

rendering ini akan disimpan oleh sistem, dinotasikan dengan Simpan Output kemudian sistem akan segera menampilkan *output* berupa citra gambar dua dimensi sesuai dengan resolusi yang diinginkan, dinotasikan dengan Print Output.



Gambar 3.2 Flowchart Monte Carlo Raytracing

Pada gambar 3.2 terlihat aktivitas dimulai dengan melakukan pengaturan *scene* untuk objek, dinotasikan dengan Set Scene. Kemudian notasi Tentukan Jumlah Titik Penyinaran menunjukkan proses selanjutnya adalah menentukan berapa jumlah titik sinar akan dipancarkan terhadap objek. Setelah jumlah sinar ditentukan, aktivitas berikutnya adalah mencari titik utama sinar, dinotasikan dengan Tentukan Titik Utama Sinar. Berikutnya aktivitas yang dilakukan adalah menentukan koordinat perpotongan sinar dengan objek, dinotasikan dengan Tentukan Perpotongan Sinar. Langkah selanjutnya adalah mencari tahu apakah sinar dipantulkan atau tidak, dinotasikan dengan Sinar Dipantulkan. Jika sinar tidak dipantulkan, langkah Print Output akan dijalankan dan aktivitas berhenti. Namun jika sinar dipantulkan, notasi Tentukan Jenis Pemantulan akan dikerjakan, yaitu menentukan jenis pemantulan sinar, apakah *diffuse* atau *specular*. Setelah jenis pemantulan diketahui, masing-masing arah pemantulan juga akan ditentukan, dinotasikan dengan Arah Pemantulan Diffuse dan Arah Pemantulan Specular. Berikutnya aktivitas akan menuju kembali ke notasi Tentukan Arah Sinar untuk memulai lagi aktivitas secara berulang sampai titik tertentu.

3.4.2.2 Perancangan Pseudocode

Setelah algoritma Monte Carlo *raytracing* diubah ke dalam bentuk flowchart, maka langkah selanjutnya adalah merancang *pseudocode* dari algoritma. *Pseudocode* merupakan suatu cara penulisan algoritma agar ide dan logika dari algoritma dapat diekspresikan menggunakan gaya bahasa pemrograman tertentu. *Pseudocode* berguna

untuk membantu dalam memahami struktur algoritma sehingga akan memudahkan dalam pembuatan kode program nantinya. *Pseudocode* algoritma Monte Carlo raytracing yang dipakai dalam pembuatan aplikasi ini yaitu:

1. Set scene untuk objek
2. Tentukan jumlah sinar = jumlah sample per piksel
3. Temukan titik utama sinar
    - Titik x = x * VektorUp.x + y * Dv.x + Vp.x,
    - Titik y = x * VektorUp.y + y * Dv.y + Vp.y,
    - Titik z = x * VektorUp.z + y * Dv.z + Vp.z;
4. Temukan perpotongan sinar

    Cari titik terdekat sinar dengan objek:
    - Perpotongan dengan sinar
    - Titik potong
    - Permukaan normal pada perpotongan
5. Hitung sinar yang dipancarkan
    - Reflection
    - Refraction
6. Hitung pemantulan sinar
    - Diffuse
    - Shadow
    - Specular
7. Hitung koefisien bayangan
8. Perunutan sinar dari titik potong

9. `Get material`
10. `End`

3.4.2.3 Perancangan Antarmuka

Antarmuka (*interface*) merupakan bagian dimana pengguna akan berhadapan langsung dengan aplikasi yang dibuat. Antarmuka seharusnya dirancang sebaik mungkin agar pengguna atau user tidak bingung dan tertarik untuk menggunakan aplikasi yang ada. Antarmuka yang sulit untuk dipahami akan menyebabkan kebingungan dari pemakai untuk memahami aplikasi.

Pada aplikasi ini tidak disediakan menu khusus baik untuk membuka, me-*render*, maupun menyimpan file *output* karena aplikasi akan segera me-*render* file *input* menjadi *ouput* berupa gambar dua dimensi yang berformat Windows Bitmap secara langsung setelah dijalankan.

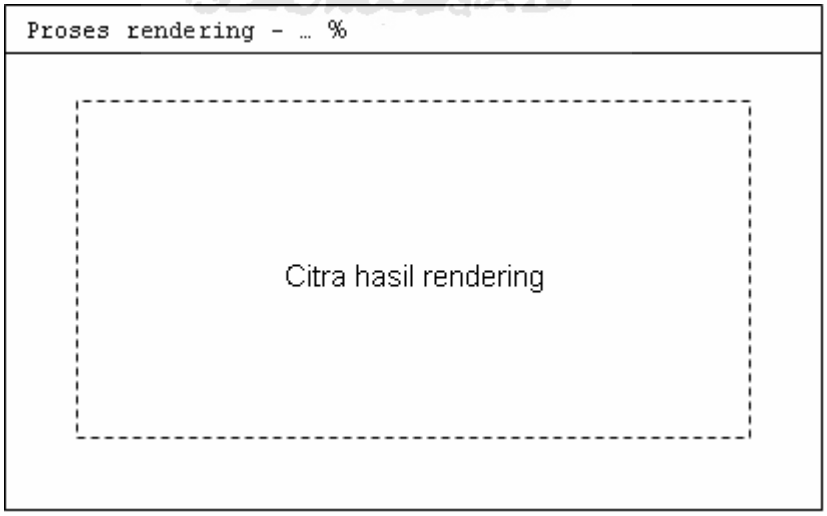
Pada gambar 3.3 menunjukkan rancangan untuk tampilan pada layar *Command Prompt*. Dalam tampilan tersebut terdapat informasi mengenai objek dan lama proses *rendering*.

```
<Direktori file> <input.art> <nama_output> <lebar> <panjang>
Menjalankan proses rendering ... Loading file ...
         <objek.obj>:        OK
Loading file OK
Rendering ...                OK (<waktu rendering>, <jumlah sinar>
Menyimpan ...                OK

Selesai
```

Gambar 3.3 Rancangan Tampilan pada Command Prompt

Apabila objek berhasil dibuka, maka aplikasi akan segera memulai proses rendering, dan sebuah layar baru akan terbuka. Layar ini berisi *ouput* hasil *rendering* sesuai dengan persentasi proses yang sudah dikerjakan. Layar tersebut ditunjukkan dengan gambar 3.4.



Gambar 3.4 Rancangan Layar Berisi Citra Hasil Rendering

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi Perangkat Lunak

Implementasi merupakan tahap dimana sistem siap dioperasikan pada tahap yang sebenarnya, sehingga akan diketahui apakah sistem yang telah dibuat benar-benar sesuai dengan yang direncanakan. Pada implementasi perangkat lunak ini akan dijelaskan bagaimana aplikasi *raytracer* yang menerapkan algoritma Monte Carlo ini bekerja.

4.1.1 Batasan Implementasi

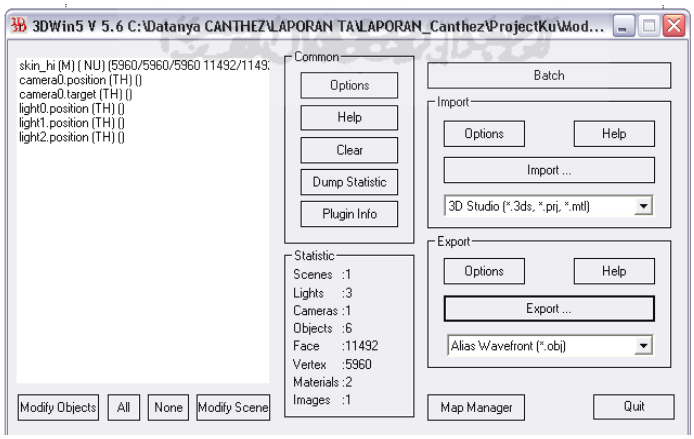
Software yang digunakan dalam membangun aplikasi adalah DevCpp 5 Beta 9 dan untuk membuat model wajah manusia digunakan 3D Studio Max 8. 3D Studio Max 8 dipilih karena sudah banyak dikenal mudah dioperasikan serta dipahami. Dalam proses pembuatan aplikasi dipilih DevCpp 5 Beta 9 sebagai *compiler*. Untuk proses konversi, *software* yang digunakan adalah 3DWin5 versi 5.6 produksi TBSoftware. *Software* ini memiliki keunggulan karena ringan dan mendukung konversi beragam format file yang sering dipakai dalam dunia tiga dimensi, seperti Alias Wavefront (*.obj), DirectX (*.x), Quake (*.mdl, *.mdl2), 3D Studio (*.3ds, *.prj, *.mtl, *.shp), VRML (*.wrl), Autocad (*.dxf), Lightwave (*.lwo, *.lws),

POVRay (*.pov, *.inc), dan sebagainya. *Input* yang digunakan dalam aplikasi adalah file *mesh* objek yang berformat .obj.

4.1.2 Implementasi Proses Input

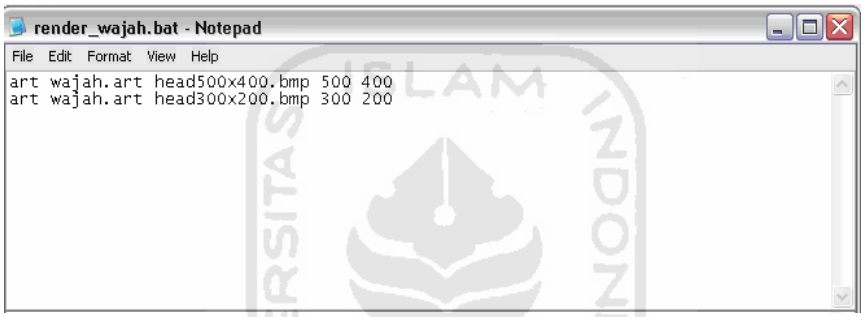
Objek yang digunakan untuk menguji aplikasi adalah model wajah manusia tiga dimensi, model teko, dan model naga. Masing-masing objek merupakan model yang dibuat menggunakan 3D Studio Max.

Objek tiga dimensi dari 3D Studio Max tidak dapat langsung dipakai sebagai masukkan dalam sistem. Sehingga objek harus di-*export* dan disimpan ke dalam bentuk *3D Studio Binary Mesh File*. Kemudian objek yang telah berubah format menjadi teks dikonversi menjadi format obj. Contoh konversi file menggunakan *software* 3Dwin5 V 5.6 dapat dilihat pada gambar 4.1.



Gambar 4.1 Konversi File Menggunakan 3Dwin5 V 5.6

Aplikasi dijalankan lewat *Command Prompt Windows* yaitu lewat sebuah file *batch* berisi sintaks khusus untuk membaca input. Sintaks untuk menjalankan aplikasi yaitu: `art <input_scene> <output_bmp> <lebar_citra> <panjang_citra>`. Sebagai contoh, sintaks yang disimpan dalam file render_wajah.bat dapat dilihat pada gambar 4.2



Gambar 4.2 Sintaks Dalam render_wajah.bat

Kedua sintaks tersebut disimpan ke dalam satu file bernama render_wajah.bat. Untuk membuka aplikasi dan memulai proses *rendering*, user tinggal membuka file render_wajah.bat ini dan kemudian setelah rendering selesai, sistem akan menghasilkan citra bitmap baru sesuai dengan masukan yang telah ditentukan.

4.1.3 Implementasi Proses Rendering

Apabila semua sintaks telah benar dan file berhasil dibaca oleh sistem, pada layar di *command prompt* akan terlihat pemberitahuan bahwa file telah dapat dibaca dengan sukses. Kemudian secara otomatis, sistem akan segera memulai proses

rendering, setelah itu akan muncul sebuah layar baru yang menunjukkan tampilan *output* citra sesuai prosentasi proses rendering yang sedang berjalan. Pada gambar 4.3 dtunjukkan ketika sistem akan memulai proses *rendering*.

Gambar 4.3 Layar Command Prompt Saat Memulai Rendering

Setelah semua proses selesai dijalankan, sistem akan membuat sebuah citra baru sesuai dengan masukan yang telah ditentukan sebelumnya pada *command prompt* pun akan tertampil informasi yang menandakan proses *rendering* berhasil dilakukan. Informasi yang tertampil misalnya lama proses *rendering* dan jumlah sinar yang dihasilkan seperti yang terlihat pada gambar 4.4.

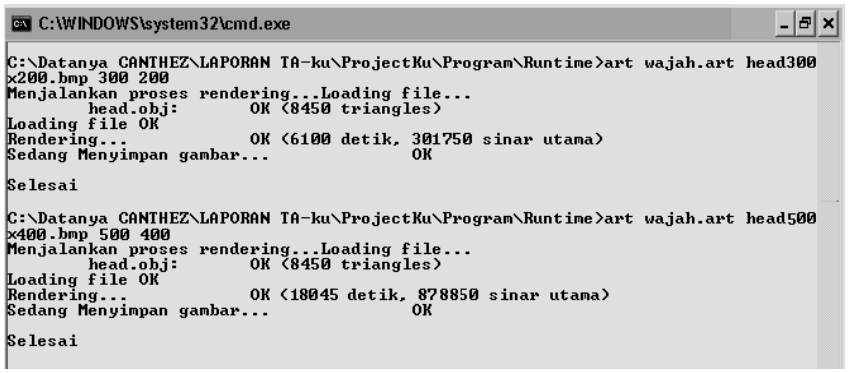
Gambar 4.4 Layar Command Prompt Saat Proses Selesai

4.2 Pengujian Sistem

Pada tahap ini menjelaskan mengenai pengujian perangkat lunak yang telah selesai dibangun. Sebelum program diterapkan, terlebih dahulu program harus bebas dari kesalahan. Untuk itu program harus diuji untuk menemukan kesalahan-kesalahan yang mungkin dapat terjadi. Pengujian dilakukan secara menyeluruh untuk mengetahui kekurangan pada aplikasi yang telah dibuat.

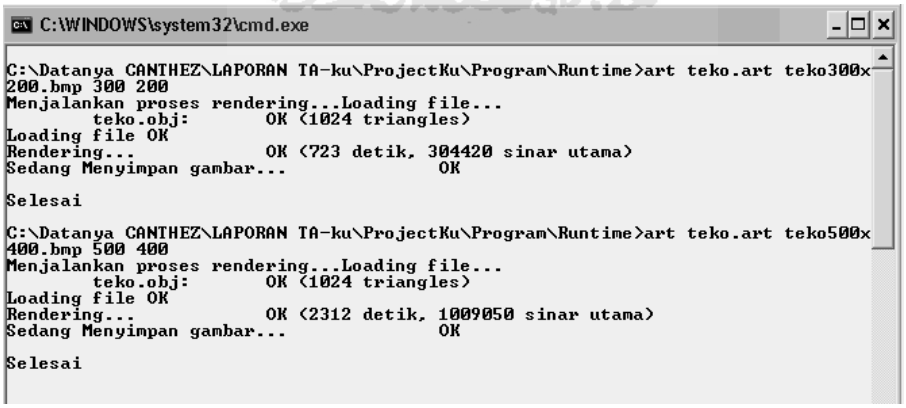
Pengujian yang dilakukan terhadap aplikasi ini berupa pengujian kecepatan *rendering* aplikasi dengan tiga objek masukkan yang berbeda. Pengujian kecepatan diwujudkan dengan lama proses *rendering* untuk masukkan dengan resolusi *output* sebesar 300 x 200 piksel dan 500 x 400 piksel untuk masing-masing objek. Tidak ada material khusus yang diterapkan terhadap objek. Dalam pengujian, peneliti hanya menggunakan material warna yang akan dibuat oleh sistem sesuai *input* yang ditentukan.

Pada pengujian dengan objek wajah manusia, sistem memerlukan waktu *rendering* selama 6.100 detik dan jumlah sinar 301.750 untuk *output* dengan resolusi 300 x 200 piksel. Sedangkan untuk menghasilkan *output* dengan resolusi sebesar 500 x 400 piksel, sistem memerlukan waktu 18.045 detik dengan sinar berjumlah 878.850 buah. Layar *command prompt* pada proses *rendering* objek wajah manusia ditunjukkan dengan gambar 4.5.



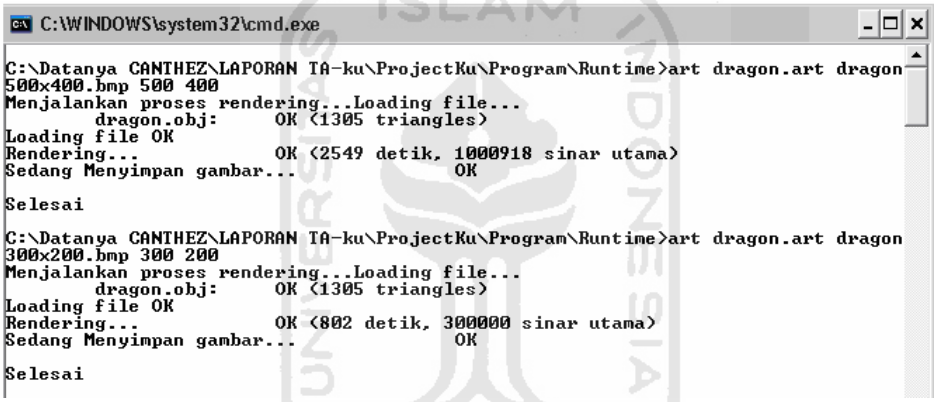
Gambar 4.5 Layar Command Prompt pada Rendering Objek Wajah

Objek kedua yang diuji adalah objek sederhana berupa sebuah teko air. Seperti yang terlihat pada gambar 4.6, waktu yang diperlukan dalam *rendering* objek teko adalah 723 detik dengan jumlah sinar utama sebanyak 304.420 untuk ukuran *output* 300 x 200 piksel. Sedangkan untuk menghasilkan *output* 500 x 400 piksel diperlukan waktu 2312 detik dan 1.009.050 sinar.



Gambar 4.6 Layar Command Prompt Pengujian Kedua

Objek uji terakhir merupakan objek berupa naga. Sumber objek ini diambil dari situs *http://www.turbosquid.com/*. Pada gambar 4.7 menunjukkan layar *command prompt* setelah proses *rendering* selesai dilakukan. Untuk *output* dengan resolusi 300 x 200 piksel, waktu *rendering* yang diperlukan adalah 802 detik dengan 1.000.918 buah sinar. Sedangkan untuk *output* berukuran 500 x 400 memerlukan waktu *rendering* 2549 detik dan jumlah sinar 300.000.



Gambar 4.7 Layar Command Prompt Pengujian Ketiga

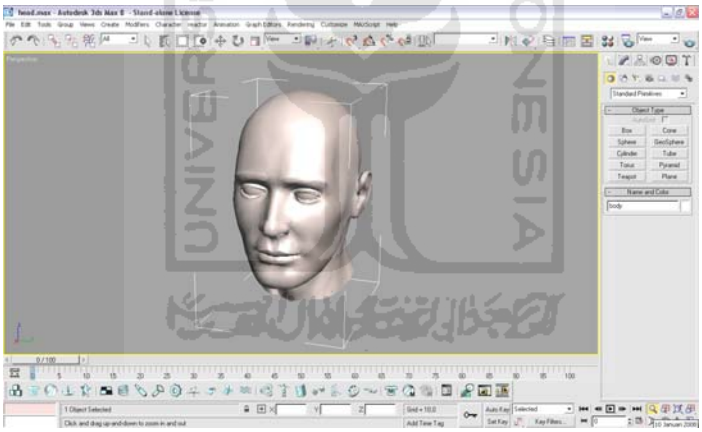
4.3 Perbandingan Sistem

Perbandingan sistem merupakan salah satu hal yang perlu dilakukan dalam tahap analisis, karena disinilah penulis bisa membandingkan sistem yang telah dibuat dengan sistem yang telah ada sebelumnya. Sehingga penulis dapat mengetahui apa saja kelebihan dan kekurangan dari sistem yang telah dibuat.

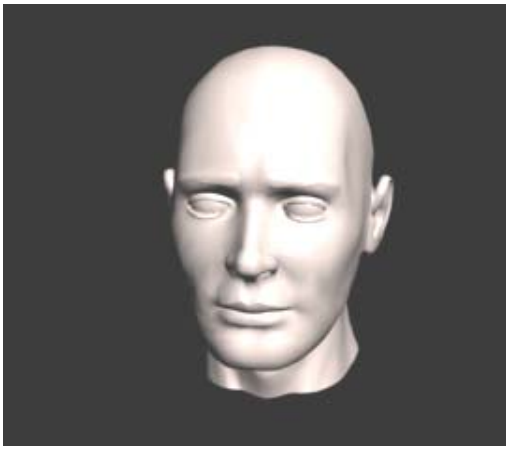
Dalam hal ini, penulis belum menemukan program yang sama persis untuk dibandingkan, sehingga perbandingan yang diuji adalah antara hasil *rendering* pada

aplikasi ini dengan hasil *rendering* menggunakan 3D Studio Max 8. Beberapa masukan diatur agar sama dengan aplikasi yang diuji sehingga dapat diketahui perbandingannya secara jelas.

Pada gambar 4.8, dapat dilihat model wajah manusia yang dibuat menggunakan 3D Studio Max 8. Hasil ketika objek wajah ini di-*render* menggunakan 3D Studio Max tanpa Monte Carlo *raytracing* ditunjukkan pada gambar 4.9. Sebagai perbandingannya, citra yang di-*render* menggunakan Monte Carlo *raytracing* dapat dilihat pada gambar 4.10.



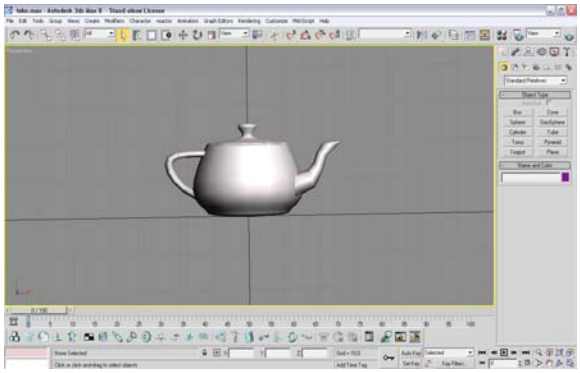
Gambar 4.8 Model Wajah Manusia Sebelum Rendering



Gambar 4.9 Rendering Wajah di 3D Studio Max 8

Gambar 4.10 Rendering Wajah dengan Monte Carlo raytracing

Model teko sederhana sebelum mengalami proses *rendering* terlihat dari gambar 4.11. Proses *rendering* pertama dilakukan dengan menggunakan 3D Studio Max biasa tanpa menerapkan Monte Carlo *raytracing*. *Output* dapat dilihat dari gambar 4.12. Setelah di-*render* dengan Monte Carlo *raytracing*, maka citra dua dimensi yang terbentuk seperti ditunjukkan oleh gambar 4.13.

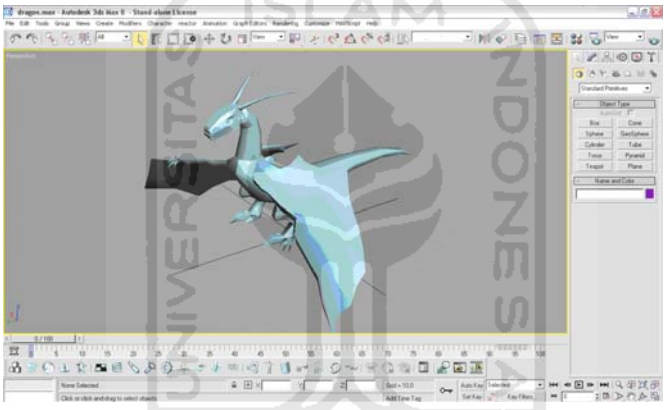


Gambar 4.11 Model Teko Sebelum Rendering

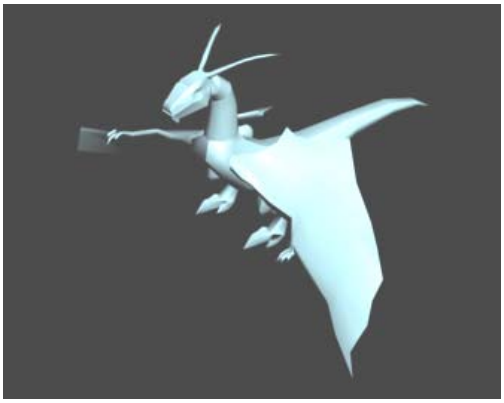
Gambar 4.12 Rendering Teko di 3D Studio Max 8

Gambar 4.13 Rendering Teko dengan Monte Carlo raytracing

Objek terakhir dalam pengujian sistem adalah model naga. Sebelum dilakukan proses *rendering* model naga terlihat seperti pada gambar 4.14. Setelah dilakukan proses *rendering* biasa menggunakan 3D Studio Max 8, tanpa menerapkan Monte Carlo *raytracing* menghasilkan citra yang ditunjukkan pada gambar 4.15. Sedangkan pada proses *rendering* setelah menggunakan Monte Carlo *raytracing*, *output* yang dihasilkan seperti ditunjukkan oleh gambar 4.16.



Gambar 4.14 Model Naga Sebelum Rendering



Gambar 4.15 Rendering Naga di 3D Studio Max

Gambar 4.16 Rendering Naga dengan Monte Carlo raytracing

4.4 Kelebihan dan Kekurangan Sistem

Dari hasil pengujian pada sistem ditemukan banyak kekurangan dan kelebihannya.

4.4.1 Kelebihan

Adapun kelebihan-kelebihan dari aplikasi ini, antara lain:

a. Aplikasi berjalan dengan baik dan dapat menghasilkan *output* yang diinginkan beserta efek bayangan pada citra.
b. File *input* berupa teks sehingga mampu mendukung berbagai format objek tiga dimensi baik yang dibuat dengan 3D Studio Max 8, Maya, Swish, dan sebagainya.

4.4.2 Kekurangan

Adapun kekurangan dari sistem yang telah dibuat adalah sebagai berikut.

a. Proses rendering berjalan lambat, apalagi untuk objek yang bersifat *high-poly* dan *ouput* dengan resolusi tinggi. Untuk memperbaikinya, diperlukan penemuan metode baru yang mampu mempercepat jalannya proses rendering.
b. Material yang bisa diterapkan hanya terbatas pada material warna.
c. *Input* file yang berupa teks sedikit menyulitkan, karena user tidak dapat langsung melihat hasil dari perubahan nilai yang dibuat, seperti posisi koordinat kamera, skala objek, dan sebagainya.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah dilakukan serangkaian pengujian dan memperhatikan seluruh proses yang terjadi didalam perancangan sistem, maka dapat disimpulkan bahwa aplikasi ini sudah berjalan cukup baik. Sesuai dengan perancangan awal, maka dapat ditarik kesimpulan sebagai berikut :

1. Setelah dilakukan pengujian terhadap sistem, maka dapat dikatakan bahwa aplikasi dapat berjalan sesuai dengan apa yang diharapkan.
2. Aplikasi *raytracer* ini dapat menjadi salah satu alternatif *tool* yang digunakan untuk *rendering* objek tiga dimensi ke dalam citra dua dimensi berdasarkan efek pencahayaan yang terjadi terhadap objek.
3. Waktu komputasi yang diperlukan untuk me-*render* sebuah objek sangat bergantung terhadap tingkat kerumitan model, seperti jumlah poligon dan material yang diterapakan serta tingkat resolusi citra output. Semakin tinggi tingkat resolusi, maka komputasi akan berjalan semakin lambat.

5.2 Saran

Penulis berharap pengembangan aplikasi *raytracer* ini tidak hanya berhenti sampai disini saja, namun dapat dilakukan pengembangan yang lebih jauh lagi

sehingga benar-benar dapat bermanfaat bagi masyarakat, dalam pengembangan selanjutnya penulis berharap :

1. Aplikasi *raytracer* dengan algoritma Monte Carlo ini masih memerlukan banyak pengembangan seperti antarmuka yang lebih *user-friendly* misalnya dengan penambahan beberapa menu untuk menjalankan aplikasi secara lebih mudah dan sederhana.
2. Proses input sedikit menyulitkan karena harus dilakukan beberapa pengaturan berupa beberapa masukan nilai untuk setiap objek yang akan di-*render*.
3. Program yang dibuat ini terbatas dalam hal waktu rendering yang memakan waktu cukup lama, bahkan untuk resolusi yang paling kecil. Sehingga diharapkan dapat dilakukan pengembangan lebih lanjut untuk menemukan metode agar proses *rendering* bisa berjalan lebih cepat.

DAFTAR PUSTAKA

[BER97] Berg, M. de, M. van Kreveld, M. Overmars, & O. Schwarzkopf. *Computational Geometry-Algorithms and Applications*. Berlin : Springer-Verlag, 1997.

[GLA89] Glassner, Andrew S. *An Introduction to Raytracing*. San Diego: Academic, 1989.

[HAL95] Halawa, Edward E & Sakti, Setyawan P. *Pemrograman dengan C/C++ dan Aplikasi Numerik*. Jakarta: Penerbit Erlangga, 1995.

[HAM07] Hamzah, Amir. *7 Objek Realistik 3ds Max*. Palembang: Maxikom, 2007.

[KAD04] Kadir, Abdul. *Panduan Pemograman Visual C++*. Yogyakarta: Penerbit Andi, 2004.

[KEL01] Keller, Alexander. *Monte Carlo and Beyond*. Makalah disampaikan di Universitas Ulm. Jerman, 30 Juli – 3 Agustus, 2001.

[LIL00] Liliana. *Pembuatan Perangkat Lunak untuk Memvisualisasikan Benda Tembus Pandang dengan Metode Raytracing*. Surabaya; Fakultas Teknologi Industri, Jurusan Teknik Informatika, Universitas Kristen Petra, 2000.

[RAH07] Raharjo, Budi. *Pemrograman C++*. Bandung: Penerbit Informatika, 2007.

[RUS98] Rushmeier, Holly. *A Basic Guide to Global Illumination*. Makalah disampaikan di 25th International Conference on Computer Graphics and Interactive Techniques. New York, 19 Juli, 1998.

[THA03] Tharom, Tabratas. *Pengolahan Citra pada Mobil Robot*. Jakarta: Elex Media Komputindo, 2003.

[WAN03] Wann Jensen, Henrik. *Monte Carlo Ray Tracing*. Siggraph Course 44, 2003.

[WAT92] Watt, Alan. *Advanced Animation and Rendering Techniques*. New York: N.Y ACM Press, 1992.