ALGORITMA PENCARIAN MELEBAR (BFS) DALAM *RAY TRACING RENDERING*

Magdalena Marlin Amanda

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jln. Ganesha 10, Bandung
e-mail: if16042@students.if.itb.ac.id

ABSTRAK

***Rendering* adalah proses pembuatan sebuah gambar dengan menggunakan model dengan bantuan program komputer. Model yang dimaksud merupakan sebuah struktur data dari objek yang akan digambar, di antaranya berisi informasi sudut pandang, tekstur, pencahayaan dan bayang-bayang. Salah satu metode rendering yang dinamakan *ray tracing* memiliki algoritma dasar yang bersifat rekursif dan dapat digambarkan dengan sebuah pohon. Metode ini menggunakan algoritma pencarian melebar dalam prosesnya. Karena proses *rendering* tidak bersifat mencari sebuah solusi, pencarian akan berhenti apabila telah mencapai kondisi tertentu. Algoritma akan dilakukan hingga setiap pixel yang membentuk gambar terbentuk dan ditampilkan oleh layar.**

Kata kunci: *rendering*, *ray tracing*, pohon, rekursif, pencarian melebar, pixel.

1. PENDAHULUAN

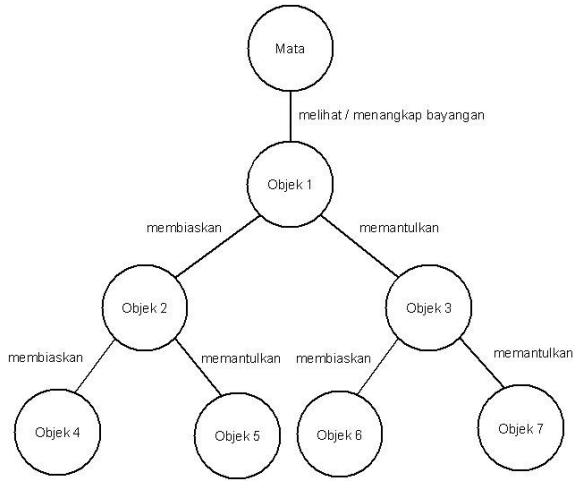
Ray tracing sebagai sebuah metode *rendering* pertama kali digunakan pada tahun 1980 untuk pembuatan gambar tiga dimensi. Ide dari metode *rendering* ini sendiri berasal dari percobaan Rene Descartes, di mana ia menunjukkan pembentukan pelangi dengan menggunakan bola kaca berisi air dan kemudian merunut kembali arah datangnya cahaya dengan memanfaatkan teori pemantulan dan pembiasan cahaya yang telah ada saat itu.

Metode *rendering* ini diyakini sebagai salah satu metode yang menghasilkan gambar bersifat paling fotorealistik, meskipun gambar yang dihasilkan kadang-kadang justru menghasilkan tampilan kurang nyata karena adanya pengabaian fakta bahwa tidak ada benda yang berpermukaan sangat halus, menyebabkan benda-benda tidak transparan menyerupai cermin.

Beberapa tahun yang lalu, metode ini hanya diterapkan pada gambar statis karena komputer belum mencapai kemampuan perhitungan seperti saat ini. Sekarang, metode ini mulai dipertimbangkan sebagai salah satu cara menampilkan tampilan realistis pada gambar bergerak.

Konsep dasar dari metode ini adalah merunut proses yang dialami oleh sebuah cahaya dalam perjalanannya dari sumber cahaya hingga layar dan memperkirakan warna macam apa yang ditampilkan pada pixel tempat jatuhnya cahaya. Proses tersebut akan diulang hingga seluruh pixel yang dibutuhkan terbentuk.

Karena selama dalam perjalanannya, cahaya akan mengalami pemantulan, pembiasan dan penyerapan oleh objek-objek yang ada, proses pencarian dilakukan secara rekursif. Jika digambarkan dengan sebuah pohon, proses yang terjadi akan nampak seperti gambar di bawah:



Gambar 1. Pohon yang Menunjukkan Gambaran *Raytracing*

Setiap simpul menyatakan objek yang ada dalam model yang telah ditentukan sebelumnya. Jumlah objek dapat berbeda-beda, tergantung dari deklarasi dalam kode program. Meskipun setiap objek memiliki sifat permukaan

yang berbeda-beda, ketiga sifat dasar cahaya tetap berpengaruh pada benda dengan kadar yang berbeda-beda.

Untuk menentukan warna apa yang akan ditampilkan oleh setiap pixel yang membentuk gambar, digunakan algoritma pencarian melebar.

2. PEMBAHASAN

Pada bab ini akan dijelaskan lebih lanjut tentang konsep *ray tracing* dan bagaimana menyelesaikan metode ini bekerja menggunakan algoritma pencarian melebar.

2.1 Deskripsi Masalah

Salah satu hal yang membuat sebuah objek nampak bersifat tiga dimensi adalah efek cahaya yang timbul saat benda tersebut terkena sebuah cahaya dengan intensitas tertentu dari suatu arah, seperti yang diperlihatkan oleh gambar di bawah ini:

Gambar 2. Contoh Gambar Hasil *Ray Tracing* Menggunakan Aplikasi POV-Ray

Sebuah objek, sekalipun bersifat transparan, masih dapat memantulkan cahaya. Hal ini terbukti dari mata manusia yang dapat menangkap wujudnya dan dari bayangan samar yang nampak pada permukaan objek tersebut. Bayangan pada permukaan objek transparan itu sendiri muncul karena permukaannya menerima pantulan dari objek lain dan objek transparan tersebut memantulkan kembali cahaya sehingga nampak bayangan pada permukaannya. Di sinilah algoritma rekursif berperan.

2.2 Algoritma Pencarian Melebar

Algoritma pencarian melebar yang juga dikenal dengan nama BFS *(Broadth First Search)* adalah salah satu algoritma traversal untuk graf selain pencarian mendalam atau DFS (*Depth First Search*).

Selain dapat diterapkan pada graf statis, algoritma ini juga dapat diterapkan pada graf dinamis yang dibentuk selama pencarian berlangsung.

Algoritma pencarian melebar dalam graf statis adalah sebagai berikut: kunjungi salah satu simpul, kemudian semua simpul yang bertetangga dengan simpul tersebut dikunjungi Setelah semua simpul yang bertetangga dikunjungi, kunjungi simpul yang terhubung dengan simpul pertama namun tidak bertetangga dengan simpul tersebut, dan ulangi mengunjungi tetangga dari simpul yang tengah dikunjungi.

Jika algoritma pencarian melebar diterapkan pada graf dinamis, simpul akar dibangkitkan terlebih dahulu, kemudian semua simpul anaknya, lalu simpul anak dari setiap simpul anak dan seterusnya hingga ditemukan solusi.

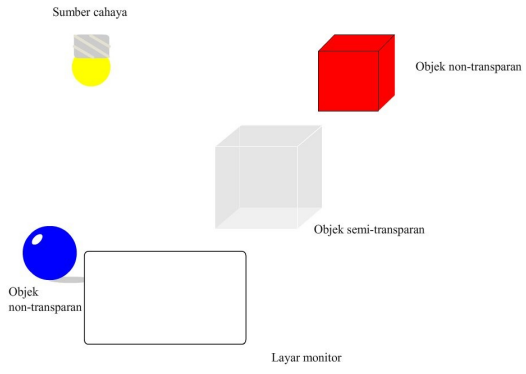
2.3 Penerapan Algoritma Pencarian Melebar

Langkah 1

Mula-mula, hasilkan warna dari objek pertama yang terkena cahaya secara langsung dan simpan warna akibat cahaya tersebut pada simpul. Objek yang terkena cahaya secara langsung, tidak selamanya berjumlah satu, tapi dapat lebih banyak.

Dalam contoh yang diambil, diasumsikan objek yang akan kita bangkitkan sebagai simpul pertama adalah objek transparan yang berada dalam cakupan layar monitor dan berada di depan objek tidak transparan berwarna merah.

Karena cahaya bersifat merambat ke segala arah, objek tidak transparan yang berada di belakang objek transparan dan objek tidak transparan yang berada di luar cakupan layar juga memantulkan cahaya yang berasal dari sumber cahaya, tapi untuk mempermudah penjelasan, difokuskan pada satu objek terlebih dahulu.



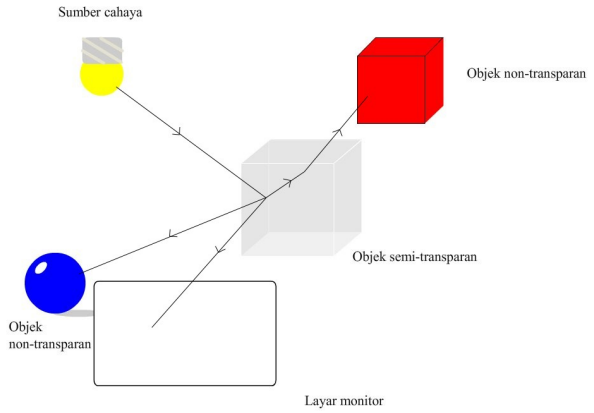
Gambar 3. Sketsa Contoh Kasus

Gambar 4. Pohon Ray Tracing 1

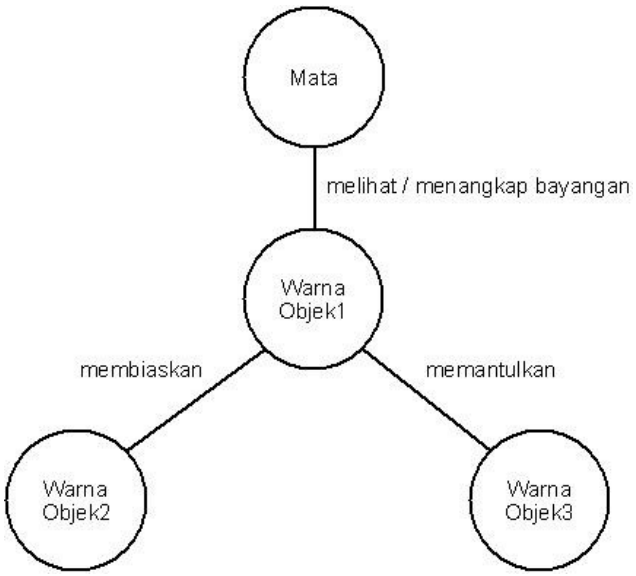
Langkah 2

Lakukan pengecekan apakah cahaya yang dipantulkan atau dibiaskan oleh objek pertama akan mengenai benda lain atau tidak, jika ya, bangkitkan simpul anak.

Pada contoh, cahaya yang dibiaskan dan dipantulkan oleh permukaan objek transparan kemungkinan akan mengenai permukaan objek tidak transparan berwarna biru yang berada di luar cakupan layar dan objek tidak transparan berwarna merah yang ada di belakang objek transparan.



Gambar 5. Jalannya Cahaya Yang Mungkin Terjadi



Gambar 6. Pohon *Ray Tracing* 2

Langkah 3

Dari simpul-simpul anak, lakukan pengecekan kembali apakah cahaya yang mengenai objek dipantulkan atau dibiaskan hingga mengenai benda lain dan simpan warna lokal objek yang kini tengah dianalisis.

Pemantulan dan pembiasan cahaya yang mungkin terjadi pada simpul anak, berbeda-beda. Hal ini disebabkan, intensitas cahaya yang merupakan hasil pemantulan, tidak sebesar intensitas cahaya langsung yang berasal dari sumber. Diperlukan perhitungan dengan rumus fisika untuk menentukan hal tersebut.

Langkah 4

Setelah cahaya tidak lagi dipantulkan atau dibiaskan hingga mengenai objek lain, gabungkan warna-warna akibat pemantulan dan pembiasan yang jatuh pada titik yang sama untuk dibentuk menjadi sebuah pixel.

2.4 *Pseudo-code*

Contoh pseudo-code dari metode *ray tracing* adalah sebagai berikut:

```
void TraceRay(point start, point direction, int
depth, colours *colour)
{
//Kamus lokal
  point hitPoint, reflectedDir, transmittedDir;
  colours localCol, reflectedCol,
transmittedCol;
  object hitObj;
//Algoritma
  if (depth > MAXDEPTH)
  //jika kedalaman sudah melebihi batas
```

```
  {
    *colour = BLACK;
    //warna benda menjadi hitam
  }
  else
  {
    //jika cahaya mengenai objek
   if    (rayHit(start,    direction,    &hitObj,
&hitPoint))
    {
          //prosedur   antara   untuk   pembuatan
"bayangan"
      shade(hitObj, hitPoint, &localCol);
       //prosedur antara untuk menghitung arah
pantulan cahaya
               calReflection(hitObj,  hitPoint,
&reflectedDir);
        //prosedur antara untuk menghitung arah
cahaya yang diteruskan
        atau dibiaskan
             calTransmission(hitObj,  hitPoint,
&transmittedDir);
           //Rekursif  dengan  penggunaan  hasil
prosedur antara
      TraceRay(hitPoint, reflectedDir, depth+1,
&reflectedCol);
       TraceRay(hitPoint, transmittedDir, depth
+1, &transmittedCol);
        //Prosedur  antara  untuk  menggabungkan
kembali warna-warna
      yang dihasilkan
        Combine(hitObj, localCol, reflectedCol,
transmittedCol, colour);
    }
    else
    {
       //jika tidak terkena cahaya, warna benda
tetap sama
      *colour = BACKGROUND;
    }
  }
  }
```

4. KESIMPULAN

Proses *ray tracing* yang prosesnya dapat digambarkan sebagai sebuah pohon sebenarnya dapat diselesaikan dengan algoritma yang lain, yakni algoritma pencarian runut balik. Tetapi, jika dianalisis lebih lanjut, penggunaan algoritma runut balik justru kurang efisien untuk kasus ini, karena setiap kali simpul dimatikan, perlu proses lain untuk naik ke simpul di atasnya dan membangkitkan anak-anak simpul lain hingga seluruh anak simpul dibangkitkan. Proses naik ke simpul sebelumnya membuat proses pencarian menjadi lama.

Algoritma *branch and bound* yang merupakan perbaikan dari algoritma pencarian melebar pun tidak bisa digunakan karena algoritma tersebut menerapkan suatu batasan berupa *cost*. Pada proses *rendering* dengan metode *ray tracing*, *cost* bukanlah hal yang paling utama karena pada akhirnya, untuk mencapai hasil yang paling mendekati kenyataan, semua simpul pada pohon perlu dibangkitkan untuk kemudian digabungkan kembali.

Algoritma pencarian melebar dirasa paling tepat untuk metode *rendering* ini, karena dalam satu waktu, algoritma ini dapat membangkitkan banyak simpul.

REFERENSI

[1] Munir, Rinaldi, "Strategi Algoritmik", Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2008.

[2] Watt, Alan, "*Advanced Animation and Rendering Techniques: Theory and Practice",* Addison-Wesley, 1992.