Pengolahan Citra
Pada
Mobil Robot

Tabratas Tharom

tharom@yahoo.com

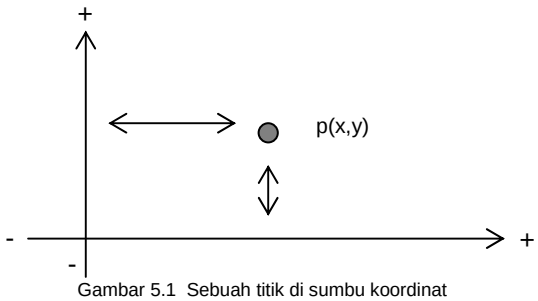
BAB 5

PEMBENTUKAN MODEL 3D VIRTUAL PADA KOMPUTER GRAFIK

5.1. KONSEP MATEMATIKA DALAM PEMBUATAN MODEL VIRTUAL TIGA DIMENSI

Sebelum kita melangkah pada pembentukan virtual tiga dimensi untuk merepresentasi hasil akhir kerja pencitraan ini, ada baiknya kita mengenal beberapa konsep matematika yang akan sangat membantu dalam banyak pengerjaan grafik nantinya. Selain itu, akan disajikan juga dasar grafik dua dimensi.

Pada grafik dua dimensi dimungkinkan sebuah titik pada permukaan bidang dengan bantuan sepasang sumbu horizontal (X) dan vertikal (Y). Pada gambar 5.1 tampak titik p(x,y) yang mempunyai dua koordinat, yakni x dan y, yang ditentukan oleh jarak horizontal dan vertikalnya. Juga diperlihatkan perjanjian tanda untuk koordinat yang disebutkan.



Gambar 5.1 Sebuah titik di sumbu koordinat

Koordinat pada dua dimensi diproses dengan bentukan matriks. Untuk memudahkan perhitungan digunakan pula operasi manipulasi dalam grafik dua dimensi. Operasi manipulasi yang dilakukan dinyatakan dalam bentuk matriks untuk mempercepat dan memudahkan perhitungan. Berikut ini dinyatakan tiga bentuk operasi manipulasi matriks, yaitu :

1. Penyekalaan (*scaling*) :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r & 0 & 0 & 0 \\ 0 & r & 0 & 0 \\ 0 & 0 & r & 0 \\ 0 & 0 & 0 & r \end{bmatrix}$$, r adalah faktor penyekalaan terhadap (*x,y,z*) untuk mendapatkan posisi (*x',y',z'*).

2. *Translasi*

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$, (u,v,w) adalah titik translasi untuk menciptakan (*x',y',z'*) dari titik (*x,y,z*)

3. *Rotasi*

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$, β merupakan sudut rotasi yang memutar titik (*x,y,z*) untuk menciptakan (*x',y',z'*).

5.1.1. KONSEP ALJABAR LINIER, VEKTOR, PERSPEKTIF, DAN TRANSFORMASI

Pada aljabar linier, istilah yang sering dipakai adalah matriks dan vektor. Tujuan aljabar linier dalam grafik tiga dimensi adalah untuk mengimplementasi rotasi, kemiringan, perpindahan, perubahan koordinat. Di lain pihak transformasi merupakan alat untuk mengubah dimensi matriks ataupun vektor dari dan hingga dimensi yang diinginkan.

Matriks merupakan alat untuk mengatasi masalah kombinasi linier dengan cara yang sederhana. Notasi matriks digunakan pada proyeksi geometris untuk memanipulasi operasi yang ada pada grafik komputer.

Pada umumnya, matriks pxq dinyatakan bahwa p merupakan baris dan q adalah kolom (banyak baris disebut dahulu dan selanjutnya banyaknya kolom). Matriks F dinyatakan dengan F = (f_{ij}), dengan (f_{ij}) merupakan unsur matriks. Indeks pertama adalah bilangan baris dan indeks kedua merupakan bilangan kolom. Contohnya diperlihatkan di bawah ini, sebagai matriks bujursangkar :

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Matriks bujur sangkar bersesuaian dengan matriks untuk p = q; artinya, matriks ini memiliki jumlah yang sama dengan jumlah kolom.

Operasi pada matriks didefinisikan untuk memudahkan tugas apa pun yang ingin dilakukan terhadap operasi matematika. Di bawah ini disajikan beberapa operasi matriks dengan penjelasannya, sebagai berikut.

⇒ **Penjumlahan matriks**

Dua buah matriks $\mathbf{A = (a_{ij})}$ dan $\mathbf{B = (b_{ij})}$ memiliki dimensi pxq, sehingga **U = A + B** dan terdefinisi menjadi $\mathbf{(u_{ij}) = (a_{ij}) + (b_{ij})}$. Jadi, secara matematis dapat ditulis:

$$\mathbf{U = A + B}$$

$$\mathbf{(u_{ij}) = (a_{ij} + b_{ij})}$$

⇒ **Perkalian matriks dengan skalar**

Misalkan skalar k dan matriks $M = (m_{ij})$ didapatkan sehingga operasinya sebagai berikut:

$$\mathbf{U = k * M}$$

$$\mathbf{(u_{ij}) = (k * m_{ij})}$$

⇒ **Perkalian matriks**

Diberikan matriks A berdimensi p x q, dan matriks berdimensi q x r, maka C = A x B terdefinisi sebagai berikut:

$$C = A \times B$$

$$\mathbf{(c_{ij}) = \Sigma_{1 \le k \le q} (a_{ik} + b_{kj})}$$

Artinya jumlah $(a_{ik} + b_{kj})$ untuk k berubah dari 1 ke q.

Pada matriks terdapat beberapa hukum matriks yang merupakan sifat matriks yang berlaku pada saat mengadakan operasi matematis. Pada matriks hanya ada dua hukum, yakni hukum distributif dan asosiatif, sedangkan komutatif tidak berlaku.

Di bawah ini diberikan hukum yang berlaku:

- *Distributif* : **A (B+C) = AB + AC**
- *Asosiatif* : **(A x B) x C = A x (B x C)**
- *Tidak komutatif* : **A x B = B x A**

Operasi matriks harus memiliki hukum yang berlaku untuk menghindari kesalahan perhitungan.

Pada matriks juga terdapat beberapa istilah sebaai berikut:

- **Matriks identitas**

  Matriks ini mempunyai keunikan, yaitu untuk matriks A,

  A x I = I x A = A

  I merupakan elemen netral dari matriks perkalian yang disebut matriks identitas. Matriks identitas mempunyai tipe matriks khusus yang dinyatakan sebagai I = f_{ij} dan terdefinisi seperti:

  $f_{ij} = 0$ *jika* $i \neq j$ dan $f_{ij} = 1$ *jika* $i = j$

  Contoh matriks identitas adalah:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Matriks transposisi**

  Matriks A dilambangkan sebagai A^T, mencerminkan matriks A sepanjang diagonal A = (a_{ij}) dan $A^T = (b_{ij})$, dengan $b_{ij} = a_{ji.}$

- **Matriks determinan**

  Diberikan determinan n x n sebagai berikut:

$$D(a_1, a_2, a_3, a_4) = \begin{vmatrix} a_{11} & a_{21} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ a_{41} & a_{42} & \cdots & a_{4n} \end{vmatrix} = \sum_{k=1}^{n} (-1)^{n+k} a_{kn} D^{(k)}$$

  dengan $D^{(k)}$ merupakan determinan (n-1) x (n-1).

- **Matriks inverse**

  Diberikan sebuah matriks A, dan invers matriks tersebut dinyatakan sebagai A^{-1} dengan

  $A \times A^{-1} = A^{-1} \times A = I$,

  Persamaan ini menghasilkan matriks identitas. Selanjutnya, perumusannya dinyatakan dalam:

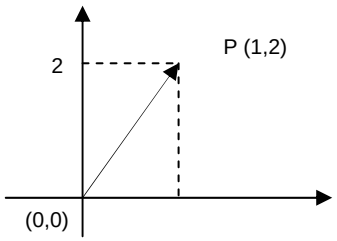
  $A^{-1} = (1/detA) \times B^T$

  dengan B^T merupakan kofaktor matriks B

Vektor adalah segmen garis yang berarah; panjangnya disebut panjang vektor dan arahnya disebut arah vektor. Dua vektor sama jika dan hanya jika mereka memiliki panjang dan arah yang sama. Panjang vektor juga disebut *Euclidian norm* atau *magnitude* vektor.

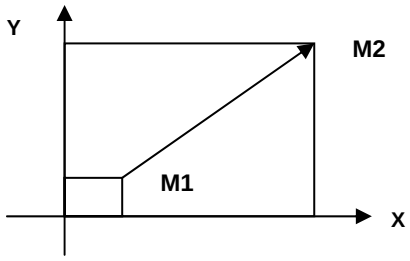
Suatu vektor dalam tiga dimensi (3D) dinyatakan dengan (a,b,c) dengan a,b,c merupakan bilangan *real*. Sama halnya dengan suatu vektor dalam dua dimensi (2D) yang dinyatakan dengan (a,b,) dengan a,b merupakan bilangan *real*. Vektor yang semua komponennya berharga nol memiliki penyebutan yang khusus, biasanya dinyatakan dengan vektor nol (0).

Sebuah vektor dianggap sebagai segmen berarah dari titik asal (0) ke titik P di dalam ruang. Perhatikan ruang dua dimensi (2D) berikut ini, yang juga dapat diterapkan untuk tiga dimensi (3D) atau mungkin dimensi yang lebih tinggi. Vektor **V** = (1,2) dapat dinyatakan sebagai segmen berarah dari (0,0) ke P = (1,2). Gambarannya seperti berikut ini:



Gambar 5.2 Vektor yang mempunyai segmen berarah (1,2)

Dengan cara lain, sebuah vektor dapat digambarkan sebagai arah dari titik 0 ke titik M. Dengan menggunakan pemodelan seperti terlihat pada gambar berikut ini, dapat dibuat suatu vektor dari M1 = (a,b) ke M2 = (c,d) sebagai vektor dari (M1 – M1) ke (M2 – M1), dari titik (0,0) ke (c-a,d-b). Vektor dari titik M1 ke M2 merupakan vektor yang sama seperti vektor dari 0 ke M2 – M1.



Gambar 5.3 Vektor dari Mi ke M2

Operasi vektor adalah operasi matematis yang memiliki arah dan panjang, seperti halnya operasi matematis yang melibatkan penjumlahan, pengurangan, pembagian, dan perkalian. Di bawah ini ditampilkan beberapa operasi pada vektor.

- **Vektor penambahan**

  u + v = (u1+v1, u2+v2, u3+v3)

  a + b = (a1+b1, a2+b2)

  Untuk vektor dalam tiga dimensi, u = (u1,u2,u3) dan v =(v1,v2,v3). Untuk vektor dalam dua dimensi, a =(a1,a2) dan b = (b1,b2).

- **Perkalian vektor terhadap skalar**

  Misalnya diketahui vektor u = (u1,u2,u3) dan skalar a, dengan notasi :

  a x u = (axu1,axu2,axu3)

- **Perkalian dua vektor**

  Perkalian dua vektor didefinisikan sebagai suatu vektor yang melibatkan perkalian bilangan *real* biasa. Operasi ini dibagi ke dalam dua bagian sebagai berikut.

  I. *Dot product* (perkalian dot)

  Operasi ini mengandung notasi u.v dan menghasilkan bilangan *real* (bukan vektor). Komponen u (u1,u2,u3) dan v (v1,v2,v3) dinyatakan dengan notasi sebagai berikut :

  $u.v = u1xv1 + u2xv2 + u3xv3$

  $u.v = /u/x/v/ Cos\ \alpha$

  II. *Cross product* (perkalian silang)

  Operasi ini dinyatakan dengan notasi seperti uxv. Jika vektor-vektor tersebut mewakili perkalian komponennya seperti komponen pada *dot product*, maka notasinya dinyatakan sebagai :

(u1,u2,u3) x (v1,v2,v3) = (u2v3 - u3v2, u3v1 -u1v3, u1v2 - u2v1)

$u \times v = |u|.|v| \sin \theta$

Hasil *cross product* dengan sudut θ merupakan panjang vektor. θ merupakan sudut yang diapit oleh vektor U dan vektor V.

Pada vektor terdapat istilah yang digunakan untuk melambangkan pemakaian vektor untuk definisi tertentu. Berikut ini dikemukakan berapa istilah dalam vektor.

- Panjang atau modul atau *norm* vektor U ditulis $|U|$ dan memiliki nilai $(U,U)^{1/2}$ serta $|U|$ juga disebut *nilai absolut* (mutlak).
- Jika panjang vektor bernilai satu, maka panjang itu disebut panjang unit, atau vektor unit, atau vektor normal.
- Vektor normalisasi adalah mengalikan vektor V dengan skalar $1/|V|$.
- *Ortogonal* digunakan untuk menggambarkan vektor tegak lurus
- *Ortonormal* adalah basis yang dibuat dari vektor unit ortogonal

Perspektif merupakan distorsi kuat yang terjadi saat si pemotret mengambil pemandangan yang benar-benar hidup dan memotretnya. Saat ini diketahui bahwa mungkin berkas cahaya tidak bergerak pada suatu garis lurus. Bahkan pada hampa udara (*vacuum*), berkas cahaya sedikit berosilasi. Saat melewati suatu materi, terjadi deviasi setiap waktu ketika cahaya terbelah, dipantulkan, dan sebagainya.

Proyeksi yang paling sederhana adalah suatu transformasi *affin* dari 3D ke 2D. Contohnya adalah transformasi (x,y,z) $\Rightarrow$ (x,y) yang mentransformasikan titik (x,y,z) dalam tiga dimensi ke titik (x,y) pada dua dimensi. Contoh sederhana lainnya adalah transformasi (x,y,z) $\Rightarrow$ (x+z,y+z). Proyeksi ini paralel karena garis-garis paralel pada tiga dimensi berisi paralel yang diproyeksikan sekali dalam dua dimensi. Contoh ini merupakan bagian dari transformasi perspektif.

Transformasi perspektif merupakan sebutan lain dari proyeksi perspektif. Ada asumsi untuk memudahkan pemahaman proyeksi perspektif. Untuk memahaminya diperlukan beberapa asumsi berikut:

- Diasumsikan cahaya merupakan suatu garis lurus. Cahaya dapat bergerak dari objek ke mata untuk ditangkap sehingga mata melihat sebuah objek. Karena cahaya bergerak dalam sebuah garis lurus, ada dua kemungkinan, yakni cahaya bergerak lurus ke mata atau terpantulkan oleh beberapa permukaan pantulan di dalam lingkungannya. Kemungkinan lain adalah cahaya datang lurus dari objek ke mata. Garis cahaya ini disebut sebuah proyektor.

- Asumsi lain berlawanan dengan keadaan di atas. Bermula dari mata, cahaya mengenai objek. Pada kenyataannya, citra akan dikirim ke layar, kertas, atau media lain. Hal ini berarti bahwa pada pemodelan, cahaya tidak mencapai mata; cahaya terhenti pada layar atau kertas, dan kemudian diperagakan. Dalam kenyataannya, media perantara (seperti udara) membawa sinar-sinar cahaya dari layar ke mata yang sebenarnya. Dalam hal ini kita

menganggap mata sebagai pusat proyeksi dan bidang sebagai permukaan proyeksi.

Transformasi *affin* merupakan transformasi linier yang diikuti oleh suatu perpindahan. Transformasi *affin* berguna untuk menggambarkan grafik tiga dimensi (3D) yang berhubungan dengan perpindahan. Matriks 3 x 3 tidak dapat digunakan untuk memindahkan titik 3D. Berikut ini diperlihatkan matriks A dan titik P. Kedua komponen ini dikalikan silang (*cross product*) dan menghasilkan pengoperasian berikut.

$$A=\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}; P(0,0,0)$$

$$A\times P=\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}\times\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}=\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Dengan demikian, dibutuhkan suatu cara yang dapat mengatasi keadaan ini. Sebuah transformasi linier dalam ruang 4D yang diproyeksikan dengan cara yang khusus pada ruang 3D, yakni dengan suatu transformasi *affin*.

Matriks 4 x 4 dapat dipakai untuk memodelkan sebuah transformasi *affin* pada tiga dimensi; matriks tersebut memiliki bentuk sebagai berikut :

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Submatriks 3 x 3 (m_{ij}), merupakan rotasi atau regangan atau kemiringan biasa. Vektor (T_x,T_y,T_z) ditambahkan terhadap titik setelah transformasi. Titik (x,y,z) ditransformasikan ke dalam (p,q,r) yang dinyatakan sebagai berikut :

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \\ 1 \end{bmatrix}$$

Selanjutnya akan dijelaskan penerapan transformasi linier, yang akan berpusat pada arah dan posisi serta berbagai istilah yang terlibat di dalamnya. Posisi dan arah ini terletak dalam suatu ruang yang berhubungan dengan suatu titik referensi atau acuan. Ruang nyata (dunia) adalah sistem referensi secara umum untuk semua objek. Transformasi *affin* menggambarkan suatu arah dan posisi objek yang biasanya berhubungan dengan ruang dunia.

Suatu matriks A mewakili transformasi *affin* yang mengambil sebuah objek dari ruang M ke ruang N (dalam contoh ini M adalah ruang objek, sedangkan N merupakan ruang dunia) yang dinyatakan dengan $A_{N \leftarrow M}$. Objek yang transformasi *affin*-nya $A_{World \leftarrow objek}$, juga mempunyai posisi dan arah kamera yang dinyatakan dalam $C_{World \leftarrow Kamera}$. Pada kasus ini, yang pertama kali dilakukan adalah

mengubah transformasi $C_{World \leftarrow Kamera}$ untuk menemukan transformasi $C_{Kamera \leftarrow World,}$ lalu akan ditransformasikan titik P_i pada objek dengan

$$C_{Kamera \leftarrow World} \times A_{World \leftarrow Objek} \times P_i = M_{Kamera \leftarrow Objek} \times P_{i.}$$

Berikut ini disajikan penurunan dari awal hingga akhir transformasi titik P_I pada objek:

Kamera ← World
World ← Objek
} Kamera ← World ← Objek = Kamera ←Objek

$$C_{Kamera \leftarrow World} \times A_{World \leftarrow Objek} \times P_i = M_{Kamera \leftarrow Objek} \times P_{i.}$$

5.1.2. KONSEP PROBABILITAS

Dalam pengertian yang sangat umum, probabilitas adalah ukuran kemungkinan bagi suatu *kejadian* untuk terjadi dalam suatu *percobaan*, atau eksperimen, yang dilaksanakan dalam kondisi tertentu. Setiap kemungkinan yang dihasilkan dari percobaan disebut hasil (*outcome*).

Kemampuan untuk meramalkan kemungkinan terjadinya suatu kejadian mempunyai kaitan yang jelas dengan penerapannya, misalnya dalam masalah pengendalian navigasi mobil robot, serta penggunaan luminansi pada objek ataupun model. Probabilitas suatu kejadian akan terjadi biasanya diwakilkan dengan *P(A)*.

Dalam probabilitas, akan banyak dijumpai istilah.

- **Probabilitas empiris**

  Probabilitas jenis ini didasarkan pada hasil-hasil sebelumnya yang diketahui. Frekuensi relatif banyaknya ulangan suatu kejadian telah muncul sebelumnya dan merupakan indikasi mengenai kemungkinan kemunculannya di kemudian hari

- **Ukuran sampel**

  Ukuran sampel asal membuat angka probabilitas dapat dihitung dan mempengaruhi keterandalan hasil. Probabilitas yang diturunkan dari sampel kecil jarang mencerminkan probabilitas yang berhubungan dengan keseluruhan populasi. Semakin besar sampel, semakin terandalkan hasil yang diperoleh.

- **Kejadian terputus (*mutually exclusive*) dan kejadian berbarengan (*mutually non-exclusive*)**

  Kejadian terputus adalah kejadian yang tidak terjadi secara bersamaan, sedangkan kejadian berbarengan adalah beberapa kejadian yang dapat terjadi secara bersamaan.

- **Kejadian bebas dan takbebas**

  Kejadian disebut kejadian bebas (*independent*) bila munculnya tidak mempengaruhi probabilitas munculnya kejadian kedua. Sebaliknya kejadian disebut tak bebas (*dependent*) bila mempengaruhi probabilitas munculnya kejadian kedua.

- **Hukum pada probabilitas**

  a. *Penjumlahan probabilitas*

     Jika kejadian A dan kejadian B adalah kejadian yang terputus, maka:

P (A atau B) = P(A) + P(B),

sedangkan jika kejadian A dan kejadian B adalah kejadian yang berbarengan, maka :

P (A atau B) = P(A) + P(B) – P(A dan B)

b. Perkalian probabilitas

Pada probabilitas, operasi perkalian akan berbentuk:

P (A dan B) = P(A) x P(B)

◎ **Probabilitas bersyarat**

Probabilitas ini diperlukan untuk memprediksi kemungkinan terjadinya kejadian B, bila diketahui bahwa suatu kejadian A telah terjadi. Hal ini dilambangkan dengan menggunakan lambang $P(B \mid A)$.

Jika A dan B adalah kejadian yang bebas, maka kejadian A tidak akan mempengaruhi probabilitas kejadian B. Dalam kasus demikian :

$$P(B \mid A) = P(A \cap B) / P(A)$$

Jika A dan B adalah kejadian yang takbebas, maka kejadian A akan mempengaruhi probabilitas munculnya kejadian B.

◎ **Distribusi pada probabilitas**

Apakah distribusi peluang diskrit disajikan secara grafik dalam bentuk histogram, dalam bentuk tabel, atau melalui rumus tidak menjadi masalah. Yang ingin dilukiskan ialah kelakuan peubah acak tersebut. Seringkali pengamatan yang dihasilkan melalui percobaan staistika yang berbeda memiliki bentuk kelakuan umum yang sama. Karena itu peubah acak diskrit yang berkenaan dengan percobaan tersebut pada dasarnya dapat dilukiskan dengan distribusi peluang yang sama. Malah, kita hanya akan memerlukan beberapa distribusi peluang yang penting untuk

menyatakan banyaknya peubah acak diskrit yang ditemui pada praktek. Beberapa distribusi dan pengertian dalam distribusi ditampilkan sebagai berikut.

- *Distribusi seragam diskrit*

  Distribusi peluang diskrit yang paling sederhana ialah yang peubah acaknya memperoleh semua nilainya dengan peluang yang sama. Yang seperti itu disebut **distribusi seragam diskrit**.

  Bila peubah acak X mendapat nilai x_1, x_2, ……,x_k, dengan peluang yang sama, maka distribusi seragam diskrit diberikan oleh

  $f(x;k) = 1/k$, dengan $x = x_1, x_2, \ldots\ldots, x_k$

  Lambang f(x;k) telah dipakai sebagai pengganti f(x) untuk menunjukkan bahwa distribusi seragam tersebut bergantung pada parameter k.

- *Proses Bernoulli*

  Secara singkat proses Bernoulli harus memenuhi persyaratan sebagai berikut :

  1. Percobaan terdiri atas *n* usaha yang berulang.
  2. Tiap usaha memberi hasil yang dapat dikelompokkan menjadi sukses atau gagal.
  3. Peluang sukses, dinyatakan dengan p, tidak berubah dari usaha yang satu ke yang berikutnya.
  4. Tiap usaha bersifat bebas dengan usaha lainnya.

- *Distribusi binomial*

  Suatu usaha Bernoulli dapat menghasilkan sukses dengan peluang p dan gagal dengan peluang q = 1-p. Maka, distribusi peluang peubah acak binomial X, yaitu banyaknya sukses dalam n usaha bebas, ialah :

$$b(x;n,p) = \binom{n}{x} p^x q^{n-x},\ x = 0,1,2,.....,n$$

  Distribusi binomial b(x;n,p) mempunyai rataan dan variansi :

  $\mu = np$ *dan* $\sigma^2 = npq$

- *Proses Poisson*

  Percobaan yang menghasilkan peubah acak X yang bernilai numerik, yaitu banyaknya hasil selama selang waktu tertentu atau dalam daerah tertentu, disebut proses Poisson. Proses Poisson memiliki sifat-sifat sebagai berikut.

  1. Banyaknya hasil yang terjadi dalam suatu selang waktu atau daerah tertentu tidak terpengaruh oleh apa yang terjadi pada selang waktu atau daerah lain yang terpisah. Dalam hubungan ini proses Poisson dikatakan *un-memoriable*

2. Peluang terjadinya sebuah hasil (tunggal) dalam selang waktu yang amat pendek atau dalam daerah yang kecil sebanding dengan panjang waktu atau besarnya daerah dan tidak tergantung pada banyaknya hasil yang terjadi di luar selang waktu atau daerah tersebut.
3. Peluang terjadinya lebih dari satu hasil dalam selang waktu yang pendek atau daerah yang sempit tersebut dapat diabaikan.

Banyaknya hasil X dalam suatu percobaan Poisson disebut sebagai **peubah acak Poisson** dan selanjutnya distribusi peluangnya disebut **distribusi Poisson**.

- *Distribusi Poisson*

Distribusi peluang peubah acak Poisson X, yang menyatakan banyaknya sukses yang terjadi dalam suatu selang waktu atau daerah tertentu dinyatakan dengan *t*, diberikan oleh

$$p(x;\lambda t) = \frac{e^{-\lambda t}(\lambda t)^x}{x!}$$

dengan *x = 0,1,2, …*

λt menyatakan rata-rata banyaknya sukses yang terjadi per satuan waktu atau daerah tersebut dan ***e*** = 2,71828 … .

5.2. REPRESENTASI OBJEK TIGA DIMENSI

Scene grafik dapat terdiri atas beberapa bentuk objek, yaitu pohon, bunga, awan, bebatuan, air, bata, kayu, kertas, dan banyak lagi. Jadi, besar kemungkinan bahwa tidak ada satu metode pun yang sebenarnya dapat menggambarkan semua objek tersebut beserta bentukan materialnya. Maka, untuk menghasilkan bentuk seperti itu, yakni tampilan yang lebih realistik, dirasakan perlu untuk merepresentasi model objek tersebut dengan akurat dengan menggunakan pendekatan karakteristik.

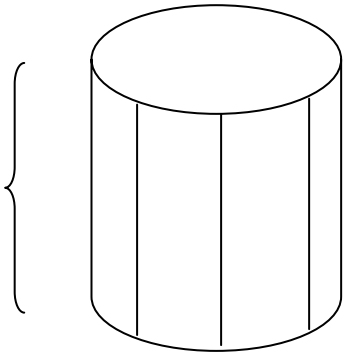
Permukaan dan bentuk poligon menyediakan deskripsi yang tepat untuk objek Euclidian sederhana seperti polihedron dan elipsoid. Permukaan *spline* dan teknik konstruksi sangat berguna untuk mendesain sayap pesawat terbang, *gear*, dan struktur rekayasa yang lainnya dengan permukaan kurva. Metode yang prosedural juga digunakan seperti konstruksi fraktal. Sistem partikel memberi kita representasi yang akurat untuk awan dan beberapa objek alami lainnya.

5.2.1. POLIGON

Cara yang paling umum digunakan untuk merepresentasi sebuah kurva pada objek grafik tiga dimensi adalah satu set permukaan poligon yang menyertakan bagian dalam objek. Beberapa sistem grafik menyimpan semua deskripsi objeknya dalam beberapa set permukaan poligon. Hal tersebut menyederhanakan dan mempercepat proses *rendering* permukaan dan menampilkan objek itu sendiri karena hampir semua permukaan biasanya dapat didekati dengan persamaan linier.

Oleh karena itu, deskripsi poligon biasanya mengacu pada objek grafik standar. Pada beberapa kasus, sebuah representasi poligon hanya satu-satunya yang tersedia. Tapi, beberapa kejadian membolehkan sebuah objek dideskripsikan dengan cara skematik lainnya, misalnya dengan permukaan *spline*, yang selanjutnya juga akan dikonversikan menjadi representasi poligon untuk kemudahan pengolahan.

Representasi poligon pada sebuah polihedron tepatnya mendefinisikan permukaan fitur objek, tapi untuk objek lainnya, permukaan mengalami *tesselated* untuk menghasilkan aproksimasi *poligon-mesh*. Gambar 5.4 berikut ini adalah permukaan sebuah silinder, direpresentasikan sebagai sebuah *poligon-mesh*. Proses *rendering* yang realistik dapat dihasilkan dengan interpolasi bentuk bayangan melewati permukaan poligon untuk mengeliminasi ataupun mereduksi kehadiran sudut pada lintasan poligon. Aproksimasi *poligon mesh* pada permukaan kurva dapat ditingkatkan dengan melakukan pembagian permukaan menjadi beberapa permukaan poligon yang lebih kecil.

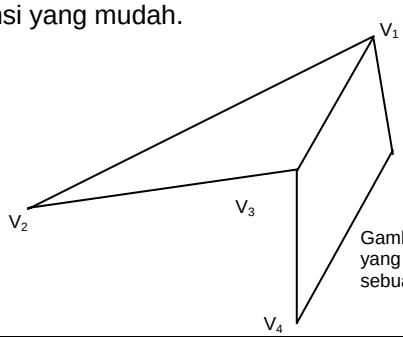


Gambar 5.4 Representasi *wireframe* dari sebuah silinder dengan garis-garis di bagian belakang dihilangkan.

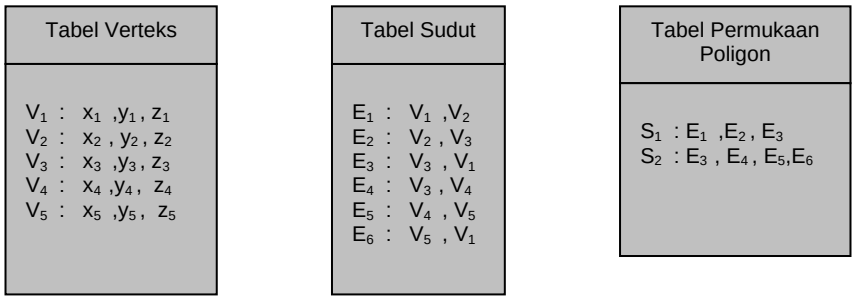
Kita menentukan sebuah permukaan poligon dengan satu set koordinat verteks dan penggabungan atribut-atribut parameter yang dibutuhkan. Sebagai informasi untuk tiap poligon adalah masukan dan data yang ditempatkan pada sejumlah tabel yang digunakan dalam pengolahan *subsequent*, tampilan, dan manipulasi pada objek dalam *scene*. Tabel data poligon dapat diatur ke dalam dua kelompok, yaitu tabel geomterik dan tabel atribut.

Tabel data geometri terdiri atas koordinat verteks dan parameter untuk mengidentifikasi orientasi *spatial* dari permukaan poligon. Informasi atribut pada sebuah objek mengandung parameter yang men-spesifikasi derajat pergerakan objek dan permukaannya serta karakteristik teksturnya.

Pengorganisasian yang baik pada penyimpanan data geometri akan sangat bermanfaat untuk menciptakan tiga daftar, yakni tabel verteks, tabel sudut, dan tabel poligon. Nilai koordinat untuk tiap verteks di dalam objek disimpan pada tabel verteks. Tabel sudut terdiri atas penunjuk (*pointer*) kembali pada tabel verteks untuk mengidentifikasi verteks untuk tiap sudut pada poligon. Yang terakhir adalah tabel poligon yang bertujuan untuk menunjuk balik pada tabel sudut untuk mengidentifikasi sudut untuk tiap poligon. Skema yang diberikan pada gambar 5.5 adalah untuk dua buah poligon pada permukaan objek. Sebagai tambahan, objek individual dan komponen poligonnya dapat mengenali objek dan melakukan identifikasi untuk referensi yang mudah.



Gambar 5.5 Dua buah poligon yang disatukan menjadi sebuah objek



Daftar data geometri dalam ketiga tabel itu, menyediakan referensi yang baik untuk komponen individual (verteks, sudut, dan poligon) dari tiap-tiap objek. Juga, objek dapat ditampilkan secara efisien dengan menggunakan data dari tabel sudut untuk menggambarkan komponen garisnya. Pengaturan alternatif adalah dengan menggunakan hanya dua tabel, yakni tabel verteks dan tabel poligon, namun akan memberikan hasil yang kurang sempurna dalam penggambarannya. Kemungkinan lain adalah dengan hanya menggunakan tabel poligon tapi hal ini akan memberikan informasi koordinat yang terduplikasi, karena nilai koordinat eksplisit didaftarkan untuk tiap verteks pada masing-masing poligon. Informasi sudut seharusnya juga direkonstruksi dari daftar verteks di dalam tabel poligon.

Informasi tambahan pada geometri biasanya disimpan dalam tabel data yang terdiri atas *slope* untuk tiap sudut dan koordinat untuk tiap poligon. Dengan analogi yang hampir sama dengan verteks, kita juga dapat mengkalkulasi *slope* sudut dan melakukan *scan* atas nilai koordinat untuk mengidentifikasi nilai maksimum dan minimum dari x,y, dan z untuk poligon individual. *Slope* sudut dan informasi *bounding-box* pada poligon diperlukan dalam pengolahan *subsequent*, misalnya

rendering. Perluasan koordinat juga digunakan dalam algoritma determinasi *visible-surface*.

5.2.2. REPRESENTASI *SPLINE*

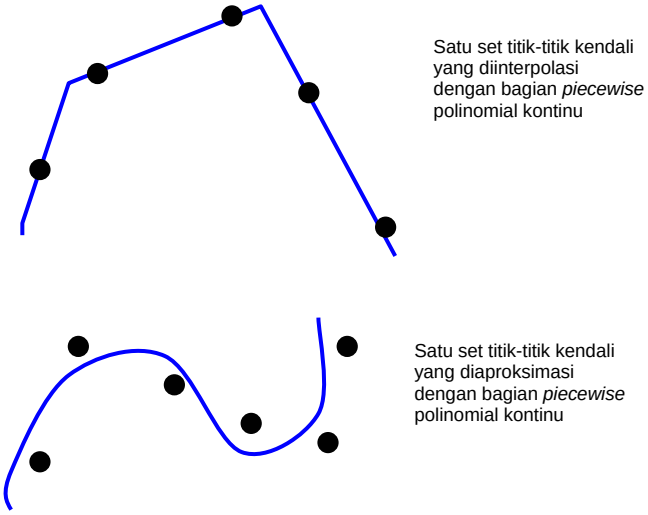
Dalam terminologi *drafting*, sebuah *spline* merupakan bidang yang paling fleksibel yang digunakan untuk membentuk kurva halus yang melalui beberapa set titik yang didesain sedemikian rupa. Beberapa bobot yang kecil terdistribusi sepanjang lebar bidang untuk tetap berada pada posisinya pada tabel *drafting* sebagai kurva untuk digambar. Pada hakikatnya, kondisi kurva *spline* mengacu pada kurva yang digambar dalam tujuan ini. Kita dapat menjelaskan secara matematika – kurva dengan fungsi polinomial kubik dengan mengambil *piecewise* nya yang turunan pertama dan keduanya merupakan bentuk kontinu yang melintasi bagian kurva yang bervariasi.

Dalam grafik komputer, kondisi kurva *spline* dibuat menunjuk pada susunan kurva yang dibentuk dengan bagian polinomial yang dibuat untuk memenuhi persyaratan kondisi ketersambungan dari sejengkal lintasan yang ingin dilihat. Permukaan *spline* dapat dijelaskan dengan dua set kurva ortogonal *spline*. Ada beberapa jenis spesifikasi *spline* yang berbeda yang digunakan dalam aplikasi grafik. Tiap spesifikasi individual yang sederhana mengacu pada bagian tipe dari polinomial dengan kondisi lintasan terspesifikasi dengan pasti.

Kita menentukan kurva *spline* dengan memberikan satu set posisi koordinat, yang disebut titik-titik kendali (*control points*), yang mengindikasikan bentuk umum kurva. Titik-titik kendali itu kemudian dilengkapi dengan fungsi polinomial *piecewise* kontinu yang terparameter dengan dua cara. Ketika bagian polinomial dilengkapi sehingga kurva melewati tiap-tiap titik kendali, resultan kurva dapat

dikatakan mengalami proses interpolasi dari set-set titik kendali tersebut. Di lain pihak, ketika polinomial dilengkapi dengan titik-titik kendali yang umum tanpa melewati titik-titik kendali utama, resultan kurva dikatakan mengalami proses aproksimasi dari set titik kendali tersebut.

Interpolasi kurva biasanya digunakan untuk menentukan jalan animasi. Aproksimasi kurva digunakan sebagai alat untuk mendesain permukaan struktur objek.



Gambar 5.6 Dua buah metode dalam mendekati persamaan sebuah kurva

Kurva *spline* didefinisikan, dimodifikasi, dan dimanipulasi dengan operasi-operasi pada titik-titik kendali. Dengan memilih posisi *spatial* secara interaktif untuk titik kendali, kita dapat membangun sebuah kurva awal. Setelah polinomial yang dipersiapkan selesai dilengkapi, kuva dapat ditampilkan untuk beberapa set titik kendali lain yang diinginkan. Dengan pengetahuan mengenai titik kendali yang lain itu, kita kemudian dapat mereposisi beberapa atau bahkan semua titik

kendali untuk membangun kembali bentuk kurva berdasarkan titik kendali referensi. Untuk tambahan, bahwa kurva dapat ditranslasi, dirotasi, atau dibuat penskalaannya dengan penggunaan aplikasi transformasi untuk titik kendali tersebut. Pada *Computer Aided Design* diikutsertakan titik kendali tambahan untuk memberi tambahan kepada kita untuk mengatur bentuk kurvanya. Lintasan *poligon* konveks yang menyertakan satu set titik kendali disebut *convex hull*.

Pada spesifikasi *spline*, ada tiga metode ekuivalen untuk menentukan representasi bagian-bagiannya, yakni :

a. Kita dapat menentukan satu set kondisi lintasan yang diletakkan pada kurva *spline* itu.
b. Kita dapat mendeterminasi matriks keadaaan yang mengkarakterisasi kurva *spline* itu.
c. Kita dapat menentukan pula satu set fungsi *blending* (atau fungsi basis) yang kemudian menspesifikasi kekuatan geometri pada kurva yang dikombinasikan untuk mengkalkulasi posisi sepanjang lintasan kurva.

Salah satu metode alternatif untuk menyatukan dua bagian sempurna dari partisi kurva adalah dengan menentukan kondisi ketersambungan geometri. Pada kasus ini, kita hanya membutuhkan parameter turunan dari dua bagian untuk dijadikan proporsional untuk tiap-tiap lintasan umum.

5.2.3. TAMPILAN PERMUKAAN DAN KURVA *SPLINE*

Untuk menampilkan permukaan dan kurva *spline*, kita harus menentukan posisi koordinat pada kurva atau permukaan yang menggabungkan posisi piksel pada *device* tampilan. Ini berarti bahwa kita harus mengevaluasi parameter fungsi polinomial *spline* dalam

pertambahannnya yang pasti, yang melebihi jangkauan fungsi. Ada beberapa metode yang dapat kita gunakan untuk menghitung posisi pada jangkauan daerah permukaan dan kurva *spline*.

i. Aturan Horner

Metode yang paling sederhana untuk mengevaluasi sebuah polinomial adalah aturan Horner, yang dirasakan lebih mengacu pada kesuksesan perhitungan yang diinginkan. Aturan ini membutuhkan satu perkalian dan satu penjumlahan untuk tiap langkahnya. Untuk polinomial dengan derajat –n, terdapat pula n buah cara.

Misalnya, kita memiliki representasi *spline* kubik yang posisi koordinatnya dapat dinyatakan sebagai:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

dengan ekspresi yang sama untuk koordinat y dan z. Untuk bagian nilai parameter u, kita dapat melakukan evaluasi polinomial tersebut dengan melakukan pemfaktoran seperti berikut ini:

$$x(u) = [(a_x u + b_{x)} u + c_x] u + d_x$$

Kalkulasi tiap-tiap nilai x membutuhkan tiga perkalian dan tiga penjumlahan sehingga penentuan posisi koordinat tiap x,y, dan z sepanjang kurva *spline* kubik membutuhkan sembilan perkalian dan sembilan penjumlahan.

Kiat penjumlahan dapat diaplikasikan untuk mengurangi sejumlah proses komputasi yang dibutuhkan oleh metode *Horner.* Namun, pengurangan dalam menentukan posisi koordinat pada daerah jangkauan fungsi *spline* dapat dihitung lebih cepat dengan menggunakan *forward difference calculation.*

ii. Kalkulasi *Forward-Difference*

Cara ini adalah cara yang cepat untuk mengevaluasi fungsi polinomial, yaitu dengan membangun nilai untuk sukses secara rekursif dengan melakukan penambahan yang sebelumnya dengan kalkulasi beberapa nilai. Contohnya di bawah ini :

$$x_{k+1} = x_k + \Delta x_k$$

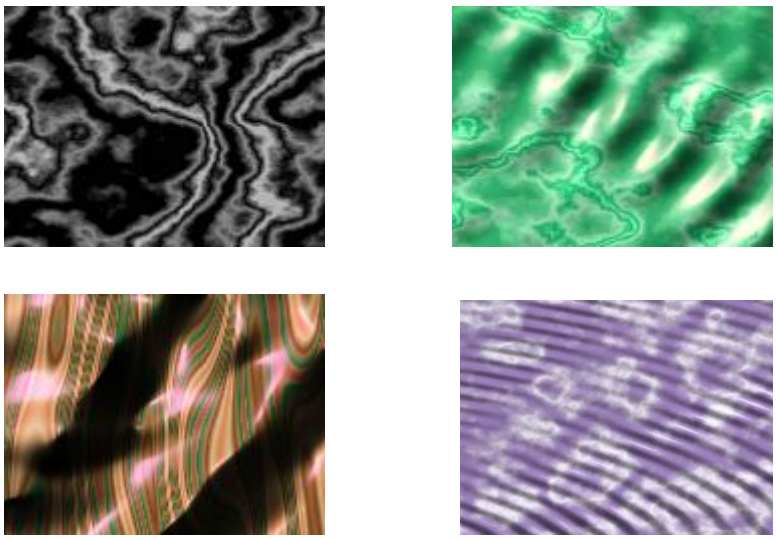
Kemudian, setelah kita mengetahui pertambahan dan nilai x_k untuk sembarang langkah, kita dapat mengetahui nilai berikutnya dengan cara menambahkan ke nilai-nilai yang berikutnya muncul. Pertambahan sebesar Δx_k pada tiap-tiap langkah disebut *forward difference*. Jika kita melakukan pembagian total jangkauan pada u menjadi beberapa subinterval dengan nilai tetap δ, akan muncul dua buah posisi sukses dari x yang ditandai dengan $x_k = x(u_k)$ dan $x_{k+1} = x(u_{k+1})$, yang dapat pula didefinisikan sebagai:

$$u_{k+1} = u_k + \delta, \text{ dengan } k = 0,1,2\ldots.$$

5.2.4. METODE GEOMETRI FRAKTAL

Semua representasi objek dapat dinyatakan dengan metode geometri Euclidian, dan bentuk yang dideskripasikan berasal dari persamaan. Metode itu cukup untuk menjelaskan bentuk yang termanufakturisasi, terutama yang memiliki permukaan halus dan bentuk yang *regular*. Tapi, untuk bentuk yang alami seperti awan dan gunung, yang memiliki bentuk yang ataupun bentuk yang terfragmentasi, dan metode Euclidian dirasakan tidak cukup realistik

untuk merepresentasikan model itu lagi, digunakanlah metode geometri fraktal.



Gambar 5.7 Beberapa contoh tekstur pada geometri fraktal

Representasi geometri fraktal untuk objek biasanya diaplikasikan dalam banyak situasi dan medan untuk menjelaskan dan menerangkan fitur fenomena yang alami. Dalam grafik komputer, kita menggunakan metode fraktal untuk membangun tampilan objek yang alami dan visualisasi variasi sistem fisika dan matematika.

Objek fraktal memiliki dua karakteristik dasar, yakni detail terhingga pada setiap titik dan *self similarity* yang pasti di antara bagian-bagian objek dan keseluruhan fitur pada objek. Sifat karakteristik yang kedua, yakni *self similarity* dari objek, dapat mengambil bentuk yang berbeda dari objek, dan bergantung pada pilihan kita tentang representasi fraktal. Kita mendeskripsikan objek fraktal dengan prosedur yang men-spesifikasi operasi perulangan yang

menghasilkan detail pada subbagian objek itu. Objek yang alamiah direpresentasi dengan prosedur yang secara teoretis berulang selama tidak terhingga kali. Tampilan grafik dari objek yang alamiah tadi, tentu dibangun dengan langkah-langkah yang terhingga banyaknya.

Jika kita melakukan *zoom* dalam bentuk Euclidian yang kontinu, tidak penting bagaimana kompeksnya objek itu, kita tetap mendapat pandangan yang didekati untuk diperhalus. Tapi, jika kita melakukan hal yang sama pada objek yang fraktal, kita akan melihat banyak sekali detail dalam magnifikasi seperti yang kita lihat pada bentuk yang asli.

Proses z*ooming in* pada tampilan grafik dari objek yang fraktal diperoleh dengan memilih *window* yang lebih kecil dan mengulangi prosedur fraktal untuk membuat detail pada *window* yang baru. Konsekuensi *window* yang tak terhingga dari objek fraktal sebenarnya tidak memiliki ukuran yang tetap, melainkan tak hingga. Kita dapat menjelaskan sejumlah variasi pada detail objek dengan beberapa cara yang disebut dimensi fraktal. Tidak seperti dimensi *Euclidian*, angka yang dipakai bukanlah tipe bilangan *integer.*

Klasifikasi geometri fraktal yang diketahui adalah sebagai berikut:

- ***Self similar fractal***

  Klasifikasi ini mempunyai bagian-bagian yang mengalami versi skala bawah untuk keseluruhan objeknya. Dimulai dengan objek inisial, kita mengkonstruksi sub bagian objek dengan memakai parameter penskalaan *s* untuk bentuknya yang lengkap. Kita dapat menggunakan faktor penskalaan yang sama, yaitu s, untuk semua subbagian lainnya atau mungkin kita dapat menggunakan faktor

penskalaan yang berbeda untuk bagian skala bawah yang berbeda pula dari objek tersebut. Seandainya kita juga memakai aplikasi yang bervariasi untuk melakukan skala bawah pada subbagiannya, fraktal dapat dikatakan *statistically self-similar*. Kemudian, bagian-bagian itu akan mempunyai sifat statistik yang sama. *Statistically self similar* pada umumnya digunakan untuk model pohon serta bentuk tumbuhan lainnya.

- ***Self affine***

  Fraktal model ini memiliki bagian-bagian yang dibentuk dengan parameter penskalaan yang berbeda-beda, yakni s_x, s_y, s_z dalam arah koordinat yang berbeda. Kita juga akan memasukkan variasi acak untuk memperoleh fraktal *statistically self affine*. Air dan awan adalah objek tipikal yang biasanya dimodelkan dengan metode konstruksi fraktal *statistically self affine* ini.

- ***Invariant fractal sets***

  Kelas fraktal ini dibentuk dengan transformasi nonlinier, menyertakan fraktal *self squaring*, seperti set *Mandelbrot* yang dibentuk dari fungsi *squaring* pada ruang kompleks, dan fraktal *self inverse* yang dibentuk dengan inversi prosedur.

  Fungsi *self squaring* menghasilkan satu dari ketiga kemungkinan hasil yang mungkin didapat :

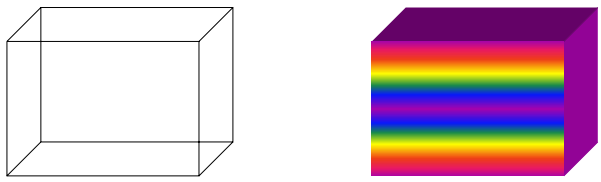
  1. Posisi yang ditransformasi dapat didivergenkan menuju ke tidak terhinggaan.

2. Posisi yang ditransformasi dapat dikonvergenkan menuju titik batas yang terhingga, yang disebut *attractor.*
3. Posisi yang ditransformasi mempunyai sisa pada lintasan untuk beberapa objek.

5.3. *RENDERING* PADA OBJEK TIGA DIMENSI

Rendering merupakan suatu fasa pada saat melakukan penggambaran yang sebenarnya. Pada proses *rendering,* agar hasil penggambaran menunjukkan kualitas baik dan proses tersebut berjalan cukup cepat, dibutuhkan prosesor yang memiliki unjuk kerja yang baik dan memiliki kecepatan tinggi

Oleh karena itu, dibutuhkan pula prosesor *slave* khusus untuk mengerjakan grafik. Suatu citra yang mengalami proses rendering yang baik akan menghasilkan gambar hasil akhir, sedangkan tanpa *rendering* tidak akan dihasilkan gambar, seperti yang terlihat berikut ini.



Tanpa *rendering* Dengan *rendering*

Gambar 5.8 Perbandingan antara objek *wireframe* dan objek yang di-*render*

Pada dasarnya rendering melibatkan penggambaran titik, garis, dan poligon, dan juga berhubungan dengan model pencahayaan dan model pemantulan. Poligon telah dijelaskan pada subbab sebelumnya.

Sebuah titik secara geometrik merupakan objek berdimensi 0 (nol). Pada komputer grafik, titik merupakan entiti paling kecil yang dapat diperadakan pada *device* tampilan. Entiti paling kecil ini dapat berupa lingkaran atau persegi atau di antara keduanya. Pada kajian ini titik dianggap sebagai piksel.

Secara garis besar, piksel adalah suatu bentuk yang berubah-ubah (sering menyerupai persegi panjang ataupun bujur sangkar), dan disusun dengan cara yang sangat terstruktur pada *device* tampilan. Pada setiap piksel diberikan warna dan jumlah warna yang tersedia tiap piksel didefinisikan oleh jumlah bit yang dilokasikan terhadap tiap piksel.

Jika setiap piksel memiliki 6 bit tiap data warna, maka setiap piksel dapat menjadi 64 warna. Perhitungan ini menggunakan formula pemangkatan yang biasa yakni 2^N dengan N merupakan banyaknya bit setiap piksel. Kemudian, satu *byte* berhubungan dengan satu piksel, sehingga satu byte dianggap sebagai 8 bit.

Mengalikan sekali untuk tiap piksel adalah proses yang mahal, dan caranya diuraikan berikut ini.

- **Proses s*traight forward***

  Proses ini sangat bergantung pada kemampuan perangkat keras yang kita gunakan; semakin baik dan bagus perangkat keras yang kita gunakan, hasilnya semakin optimal.

  Proses dekomposisi yang sama dalam pemangkatan oleh dua skalar yang berbeda dapat dicari, tapi hal tersebut membutuhkan

lebar *display* tampilan dalam piksel untuk dikodekan dengan keras (*hard coded*) di dalam program.

- **Asumsi *coherent way***

  Kita mengasumsikan bahwa kita akan melakukan proses bit dalam *coherent way,* yakni tidak keseluruhannya acak.

  Jika kita merencanakan semua piksel pada garis *scan* yang diberikan, dimulai dari kiri ke kanan, maka hasil yang didapat bernilai *true.* Piksel pada lebar lokasi memori A + y* adalah bagian *piksel* paling kiri pada garis yang di-sc*an*. Kedua bagian itu ditambahkan dengan satu, bagian yang ketiga juga ditambahkan satu dari yang kedua, dan seterusnya. Pada hakikatnya proses pengaksesan *piksel* dilakukan pada garis *scan* yang berdekatan, dan dilakukan juga pertambahan satu demi satu hanya pada *piksel*. Mengakses *piksel* pada kolom yang sama juga dapat dilakukan dengan proses yang identik, kecuali jika lebar ditambahkan untuk tiap kali pertambahan lokasi memori.

Garis merupakan kumpulan titik yang terhubung satu sama lain. Persyaratan minimal yang dibutuhkan adalah dua buah titik untuk membentuk sebuah garis. Garis yang paling sederhana untuk digambar adalah garis horizontal dan garis vertikal. Di dalam ruang dua dimensi (2D), garis horizontal diwakilkan sepanjang sumbu x dan vertikal diwakilkan sepanjang sumbu y.

Jika garis tidak vertikal atau pun horizontal, maka kita menggunakan persamaan garis yang biasa dipakai, yakni : y = mx + c. Di sini diperhitungkan sudut garis yang biasa dinyatakan dengan kemiringan garis (m) terhadap perbandingan nilai titik koordinat sumbu y dan sumbu x. Nilai konstan c dapat ditentukan pada saat nilai absis sumbu x adalah 0 (nol).

5.4. METODE PENDETEKSIAN PERMUKAAN VISIBLE

Faktor pertimbangan utama pada generasi tampilan grafik yang realistik adalah bagaimana mengidentifikasi bagian *scene* yang dapat dilihat (*visible*) dari posisi sudut pandang yang dipilih. Banyak pendekatan yang dilakukan dalam bidang ini, dan untuk mendeteksi permukaan yang dapat dilihat ini algoritma yang ada dapat dibagi menjadi dua pendekatan yang berbeda.

a. Metode ruang objek

Metode ini coba membandingkan antara objek dengan bagiannya. Untuk tiap bagian yang mengikuti definisi *scene* atau objek, akan diberi label sebagai tanda bahwa bagian tersebut adalah bagian yang terlihat (*visible*).

b. Metode ruang citra

Metode ini diunakan untuk menentukan titik demi titik pada tiap posisi piksel untuk permukaan yang diproyeksikan.

Selain dua algoritma di atas, banyak didapati metode lain dalam mendeteksi permukaaan yang akan dilihat sebagaimana douraikan berikut ini.

I. ***Back face detection***

Cara yang paling mudah dan cepat pada metode ruang objek adalah dengan mengidentifikasi *back face* (bagian belakang) sebuah polihedron berdasarkan tes luar-dalam ruang yang dimaksudkan.

II. ***Depth buffer method***

Algoritma metode ini dapat dijabarkan sebagai berikut:

1. Inisialisasikan kedalaman penampung (*buffer*) dan kembalikan penampung sesegera mungkin sehingga untuk semua penampung berlaku:

$$depth(x,y) = 0,\ \ refresh(x,y) = I_{backgnd}$$

2. Untuk tiap posisi pada masing-masing permukaan poligon, bandingkan nilai kedalaman dengan nilai yang sebelumnya disimpan pada penampung kedalaman untuk menentukan keterlihatan objek.

III. ***A buffer method***

Pada metode ini, jika kedalaman medan bernilai negatif, maka proses mengindikasikan kontribusi perkalian permukaan untuk intensitas piksel. Intensitas medan kemudian menyimpan penunjuk ke *linked list* data permukaan. Data pada tiap *linked list* mencakup :

1. intensitas komponen RGB
2. parameter opasitas (persentase transparansi)
3. kedalaman

4. persentase area yang terlingkupi
5. pengidentifikasi permukaan
6. parameter *rendering* dari permukaan yang lain
7. penunjuk (*pointer*) ke permukaan berikutnya.

IV. *Depth sorting method*

Dengan menggunakan operasi pada ruang objek dan ruang citra, metode ini menampilkan fungsi dasar berikut:

1. Permukaan diurutkan menurut pengurangan kedalaman.
2. Permukaan mengalami konversi *scan*, yang dimulai dengan kedalaman terbaik pada permukaan.

LATIHAN DAN SOAL
BAB V

1. Apakah yang dimaksud dengan matriks ? Jelaskan pula tiga bentuk operasi manipulasi pada matriks.
2. Buktikan sifat asosiatif dan distributif pada matriks.
3. Buktikan perbedaan antara operasi *cross product* dan operasi *dot product*.
4. Jelaskan asumsi yang terdapat pada transformasi perspektif.
5. Jelaskan transformasi *affin*.
6. Apa yang dimaksud dengan probabilitas empiris ? Sertakan juga tiga contoh yang mendukung.
7. Jelaskan probabilitas bersyarat secara sistematis.
8. Apa yang Anda ketahui tentang :
   - distribusi seragam diskret
   - proses Bernoulli
   - distribusi binomial
   - distibusi Poisson

   Sertakan persamaan matematis yang mendukung.
9. Jelaskan representasi *spline*.
10. Sebutkan perbedaan dan persamaan antara metode Horner dan Kalkulasi *forward-difference*.
11. Sebutkan dan jelaskan klasikasi geometri fraktal. Berikan skema yang memadai.
12. Sebutkan dan jelaskan metode yang terdapat pada teknik *render*.
13. Sebutkan keunggulan dan kelemahan yang terdapat pada teknik pendeteksian permukaan yang terlihat.
14. Jelaskan metode ruang objek dan metode ruang citra.
15. Buatlah dengan menggunakan bahasa C/C++ dan konsep alokasi memori dinamik sebuah program operasi matriks (termasuk perkalian, penjumlahan, pengurangan, mencari determinan) sampai orde 80.

16. Buatlah perbedaan antara *cross product* dan *dot product* berdasarkan algoritma dan koding masing-masing dalam bahasa C/C++.
17. Buatlah algoritma proses interpolasi dan kodekan dengan menggunakan bahasa C/C++.
18. Diketahui data :

    Y[0.1]=0.3222

    Y[0.2]=0.3224

    Y[0.3]=0.3421

    Y[0.4]=0.3425

    Y[0.5]=0.35

    Y[0.51]=0.352

    Y[0.52]=0.3525

    Y[0.61]=0.37.

    Dari data di atas, prediksikan nilai Y untuk nilai

    X= -0.1, -0.002, 0, 0.23, 0.35, 0.515, 0.6, 0.7, 0.8

    Plot data yang diketahui dan ditanyakan itu dalam sebuah grafik. Gunakanlah pemrograman grafik yang Anda ketahui.
19. Dengan pengetahuan VRML yang Anda dapatkan dari Lampiran C, buatlah *grid* dua dimensi yang menunjukkan kolom dan baris dengan ukuran 4x4.
20. Dengan bahasa VRML dan konsep transformasi yang ada, buatlah sebuah gubuk sederhana.

    Catatan:

    Dengan hanya menggunakan bentuk dasar serta pengetahuan tentang warna yang ada.
21. Jelaskan teknik *render* yang Anda ketahui dengan algoritma.