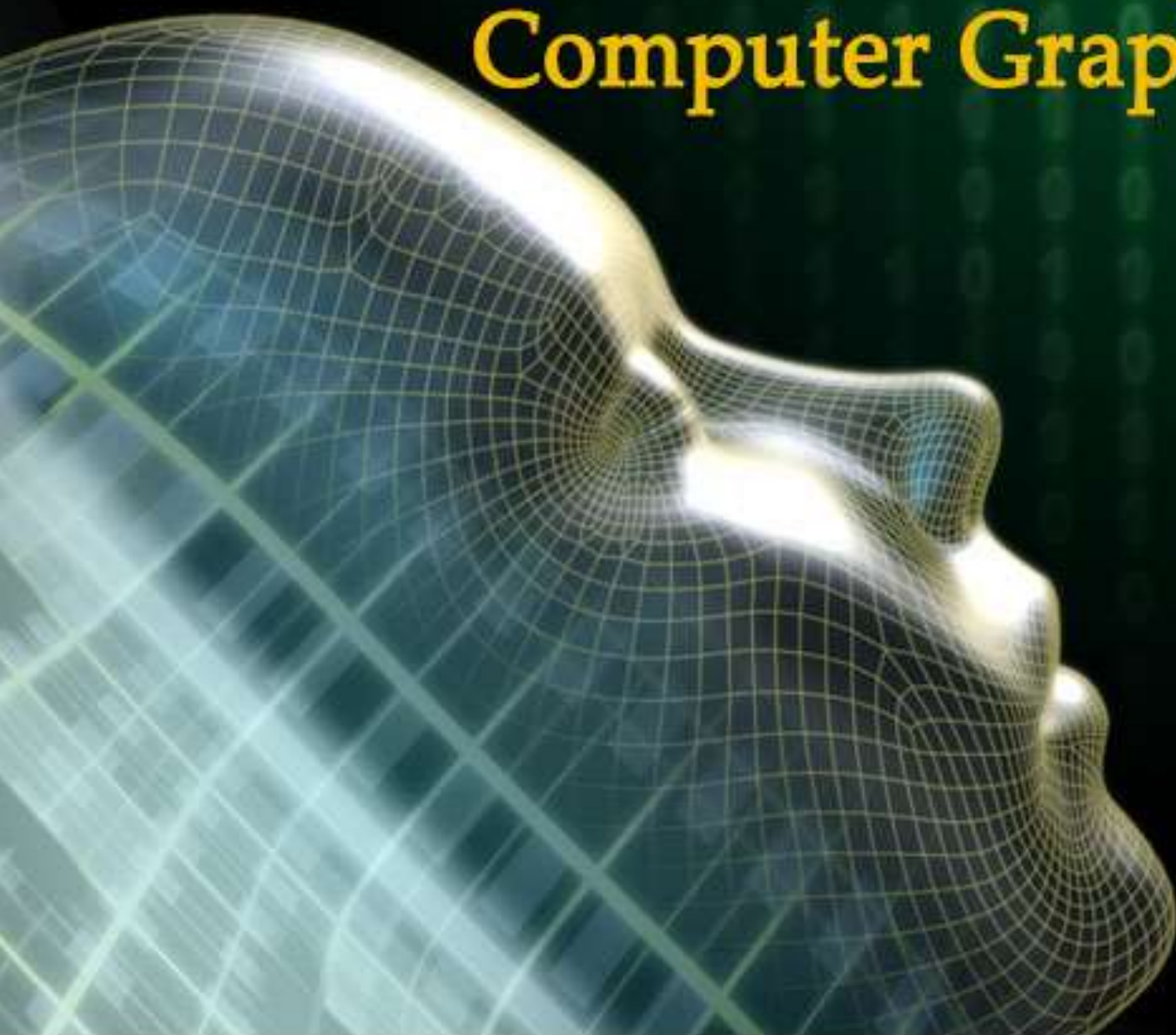




Computer Graphics



tutorialspoint
SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

Tentang Tutorial

Untuk menampilkan gambar ukuran apa pun di layar komputer adalah proses yang sulit. Grafik komputer digunakan untuk menyederhanakan proses ini. Berbagai algoritma dan teknik digunakan untuk menghasilkan grafik di komputer. Tutorial ini akan membantu Anda memahami bagaimana semua ini diproses oleh komputer untuk memberikan pengalaman visual yang kaya kepada pengguna.

Hadirin

Tutorial ini telah disiapkan untuk siswa yang tidak tahu bagaimana grafik digunakan di komputer. Ini menjelaskan dasar-dasar grafik dan bagaimana mereka diimplementasikan dalam komputer untuk menghasilkan berbagai visual.

Prasyarat

Sebelum Anda mulai melanjutkan dengan tutorial ini, kami berasumsi bahwa Anda sudah mengetahui konsep dasar bahasa pemrograman C dan matematika dasar.

Hak Cipta & Penafian

- Hak Cipta 2015 oleh Tutorials Point (I) Pvt. Ltd.

Semua konten dan gambar yang diterbitkan dalam e-book ini adalah milik Tutorials Point (I) Pvt. Ltd. Pengguna e-book ini dilarang untuk menggunakan kembali, menyimpan, menyalin, mendistribusikan atau menerbitkan ulang konten atau bagian dari isi e-book ini dengan cara apa pun tanpa persetujuan tertulis dari penerbit.

Kami berusaha untuk memperbarui konten situs web dan tutorial kami secepat dan setepat mungkin, namun, konten tersebut mungkin mengandung ketidakakuratan atau kesalahan. Poin Tutorial (I) Pvt. Ltd. tidak memberikan jaminan mengenai keakuratan, ketepatan waktu atau kelengkapan situs web kami atau isinya termasuk tutorial ini. Jika Anda menemukan kesalahan di situs web kami atau dalam tutorial ini, harap beri tahu kami di

contact@tutorialspoint.com

Daftar Isi

Tentang Tutorial	i
Pemirsa	i
Prasyarat	i
Hak Cipta & Penafian	i
Daftar Isi	ii
1. GRAFIS KOMPUTER - DASAR	1
Cathode Ray Tube	1
Pindai Raster	2
Aplikasi Grafik Komputer	3
2. ALGORITMA GENERASI LINE	5
Algoritma DDA	5
Generasi Garis Bresenham	6
Algoritma Titik Tengah	9
3. ALGORITMA GENERASI LINGKARAN	11
Algoritma Bresenham	11
Algoritma Titik Tengah	13
4. FILLING POLYGON	16
Algoritma Pindai Garis	16
Algoritma Isi Banjir	17
Algoritma Isi Batas	18
Poligon 4-Terhubung	18
Poligon 8-Terhubung	19
Tes dalam-luar	21

5.	MELIHAT DAN MENCARI CLIPPING	24
	Kliping Poin	24
	Kliping Baris	24
	Kliping Garis Cohen-Sutherland	25
	Algoritma Klipping Garis Cyrus-Beck	27
	Kliping Poligon (Algoritma Sutherland Hodgman)	28
	Kliping Teks	29
	Grafik Bitmap	31
6.	TRANSFORMASI 2D	33
	Koordinat Homogen	33
	Terjemahan	33
	Rotasi	34
	Penskalaan	36
	Refleksi	37
	Geser	38
	Transformasi Komposit	39
7.	GAMBAR 3D	41
	Proyeksi Paralel	41
	Proyeksi Orthografis	42
	Proyeksi Miring	43
	Proyeksi Isometrik	43
	Proyeksi Perspektif	44
	Terjemahan	45
	Rotasi	46
	Penskalaan	47
	Geser	48

Matriks Transformasi	49
8. KURVA	51
Jenis-jenis Kurva	51
Kurva Bezier	52
Sifat-sifat Kurva Bezier	52
Kurva B-Spline	53
Properti dari B-spline Curve	54
9. SURFASI	55
Permukaan Polygon	55
Tabel Polygon	55
Persamaan Pesawat	57
Jerat Poligon	57
10. DETEKSI PERMUKAAN YANG DAPAT DILIHAT	59
Metode Penyangga Kedalaman (Penyangga Z)	59
Metode Pemindaian	61
Metode Pembagian Area	61
Deteksi Wajah Belakang	62
Metode A-Buffer	64
Metode Penyortiran Kedalaman	65
Pohon Partisi Ruang Biner (BSP)	66
11. FRACTALS	68
Apa itu Fraktal?	68
Generasi Fraktal	68
Fraktal Geometris	69
12. ANIMASI KOMPUTER	71

Teknik Animasi	71
Framing Kunci	72
Perubahan.....	73

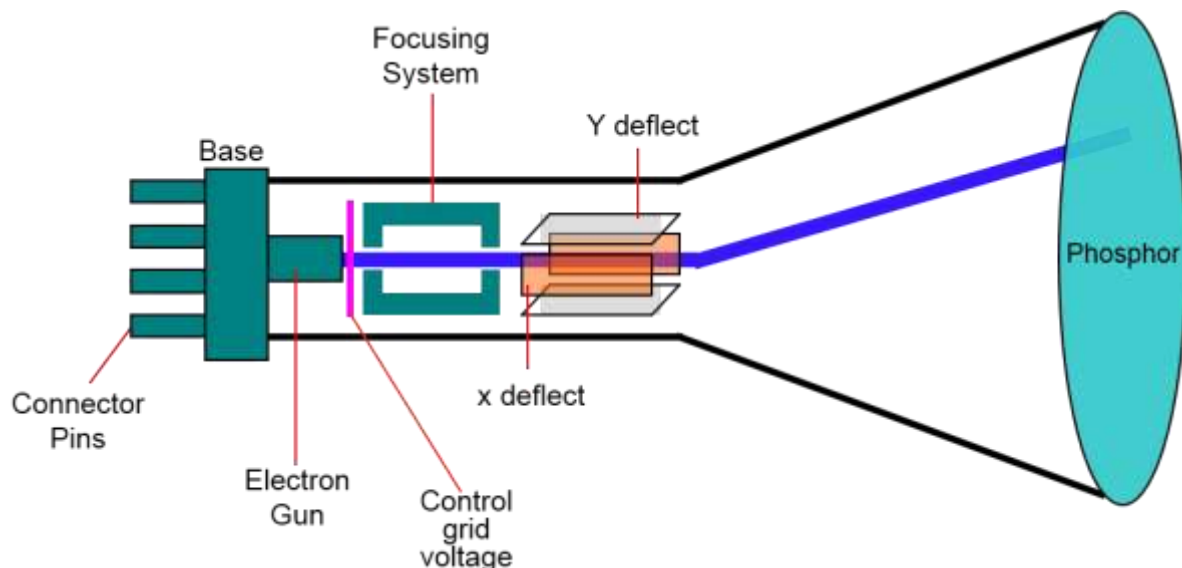
Grafik komputer adalah seni menggambar gambar di layar komputer dengan bantuan pemrograman. Ini melibatkan perhitungan, pembuatan, dan manipulasi data. Dengan kata lain, kita dapat mengatakan bahwa grafik komputer adalah alat rendering untuk pembuatan dan manipulasi gambar.

Cathode Ray Tube

Perangkat keluaran utama dalam sistem grafis adalah monitor video. Elemen utama monitor video adalah **Cathode Ray Tube (CRT)**, ditunjukkan dalam ilustrasi berikut.

Pengoperasian CRT sangat sederhana:

1. Pistol elektron memancarkan seberkas elektron (sinar katoda).
2. Sinar elektron melewati sistem fokus dan defleksi yang mengarahkannya menuju posisi yang ditentukan pada layar berlapis fosfor.
3. Ketika balok menyentuh layar, fosfor memancarkan satu titik kecil cahaya pada masing-masingnya posisi dikontak oleh berkas elektron.
4. Ini menggambar ulang gambar dengan mengarahkan berkas elektron kembali ke layar yang sama poin cepat.



Gambar: Cathode Ray Tube

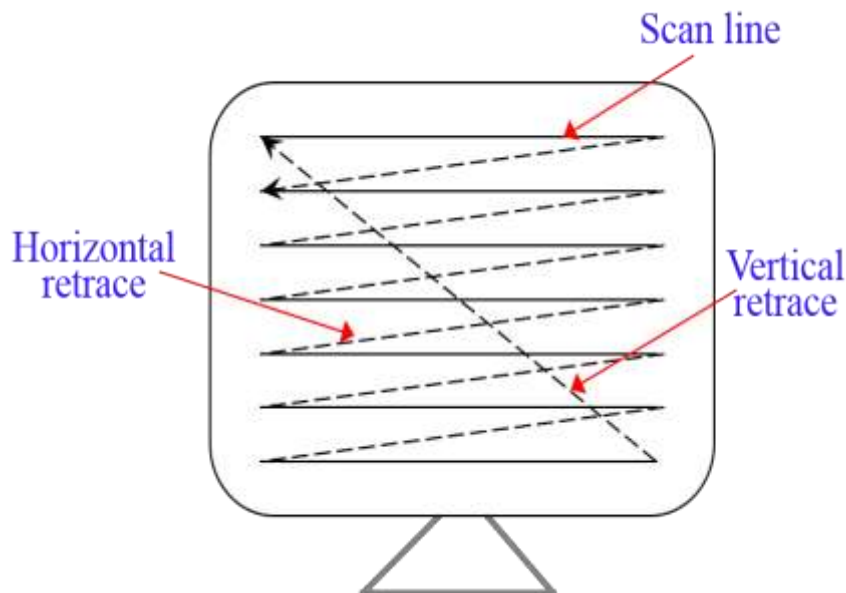
Ada dua cara (Pemindaian acak dan Pemindaian raster) yang dengannya kita dapat menampilkan objek di layar.

Pindai Raster

Dalam sistem pemindaian raster, berkas elektron disapu melintasi layar, satu demi satu baris dari atas ke bawah. Ketika berkas elektron bergerak melintasi setiap baris, intensitas sinar dihidupkan dan dimatikan untuk membuat pola bintik-bintik yang menyala.

Definisi gambar disimpan di area memori yang disebut **Segarkan Buffer** atau **Penyangga Bingkai**. Area memori ini menampung set nilai intensitas untuk semua titik layar. Nilai intensitas yang tersimpan kemudian diambil dari buffer refresh dan “dicat” pada layar satu baris (garis pindai) pada waktu seperti yang ditunjukkan pada ilustrasi berikut.

Setiap titik layar disebut sebagai a **pixel (elemen gambar)** atau **pel**. Pada akhir setiap garis pindai, berkas elektron kembali ke sisi kiri layar untuk mulai menampilkan garis pindai berikutnya.



Gambar: Pindai Raster

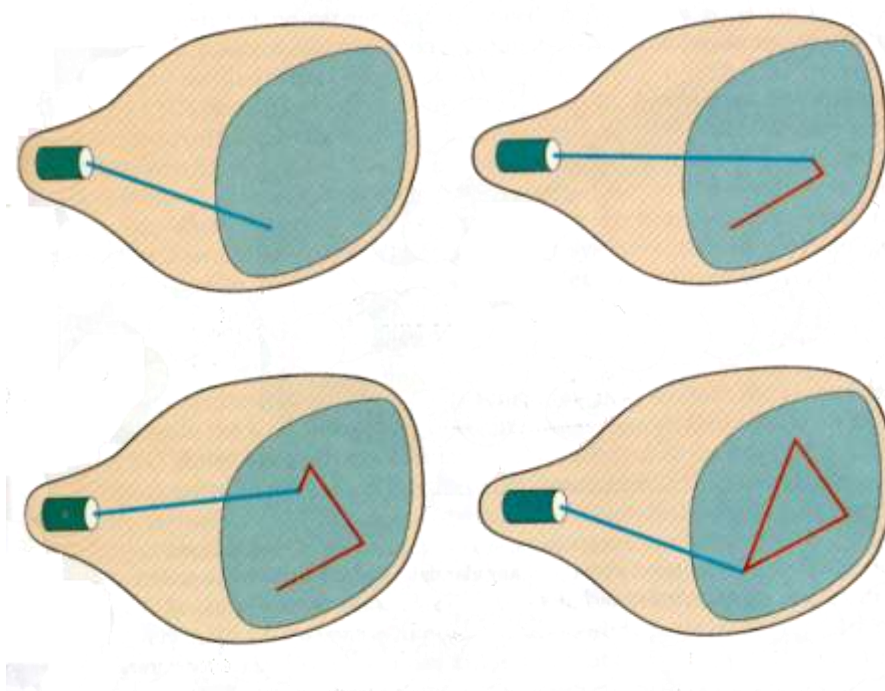
Pemindaian Acak (Pemindaian Vektor)

Dalam teknik ini, berkas elektron diarahkan hanya ke bagian layar di mana gambar harus diambil daripada pemindaian dari kiri ke kanan dan atas ke bawah seperti dalam pemindaian raster. Ini juga disebut **tampilan vektor**, **tampilan penulisan stroke**, atau **tampilan kaligrafi**.

Definisi gambar disimpan sebagai satu set perintah penggambaran baris di area memori yang disebut sebagai **menyegarkan file tampilan**. Untuk menampilkan gambar yang ditentukan, sistem

menggilir serangkaian perintah dalam file tampilan, menggambar setiap baris komponen secara bergantian. Setelah semua perintah menggambar garis diproses, sistem akan kembali ke perintah baris pertama dalam daftar.

Tampilan pindai acak dirancang untuk menggambar semua garis komponen gambar 30 hingga 60 kali setiap detik.



Gambar: Pemindaian Acak

Aplikasi Grafik Komputer

Komputer Grafik memiliki banyak aplikasi, beberapa di antaranya tercantum di bawah ini:

- **Antarmuka pengguna grafis komputer (GUIs)** - Paradigma grafis berorientasi mouse yang memungkinkan pengguna untuk berinteraksi dengan komputer.
- **Grafik presentasi bisnis** - " Sebuah gambar bernilai ribuan kata ".
- **Pemetaan** - Menggambar peta.
- **Peta Cuaca** - Pemetaan waktu nyata, representasi simbolik.
- **Pencitraan Satelit** - Gambar geodesik.
- **Peningkatan Foto** - Mempertajam foto buram.
- **Pencitraan medis** - MRI, pemindaian CAT, dll. - Pemeriksaan internal non-invasif.

- **gambar teknik** - mekanik, listrik, sipil, dll. - Mengganti cetak biru masa lalu.
- **Tipografi** - Penggunaan gambar karakter dalam penerbitan - menggantikan jenis masa lalu yang sulit.
- **Arsitektur** - Rencana konstruksi, sketsa eksterior - menggantikan cetak biru dan gambar tangan dari masa lalu.
- **Seni** - Komputer menyediakan media baru bagi para seniman.
- **Latihan** - Simulator penerbangan, instruksi yang dibantu komputer, dll.
- **Hiburan** - Film dan game.
- **Simulasi dan pemodelan** - Mengganti pemodelan dan pengesahan fisik

Garis menghubungkan dua titik. Ini adalah elemen dasar dalam grafik. Untuk menggambar garis, Anda perlu dua titik untuk menggambar garis. Dalam tiga algoritma berikut, kami merujuk satu titik garis sebagai X_0, Y_0 dan titik kedua garis sebagai X_1, Y_1 .

Algoritma DDA

Algoritma Diferensial Digital (DDA) adalah algoritma generasi garis sederhana yang dijelaskan langkah demi langkah di sini.

Langkah 1: Dapatkan input dari dua titik akhir (X_0, Y_0) dan (X_1, Y_1).

Langkah 2: Hitung perbedaan antara dua titik akhir.

$$\begin{aligned}dx &= X_1 - X_0 \\dy &= Y_1 - Y_0\end{aligned}$$

Langkah 3: Berdasarkan perbedaan yang dihitung pada langkah-2, Anda perlu mengidentifikasi jumlah langkah untuk menempatkan piksel. Jika $dx > dy$, maka Anda memerlukan lebih banyak langkah dalam koordinat x; jika tidak dalam koordinat y.

```
if (dx > dy)
    Langkah = absolut (dx);
else
    Langkah = absolut (dy);
```

Langkah 4: Hitung kenaikan dalam koordinat x dan koordinat y.

```
Langkah Xincrement = dx / (float);
Yincrement = langkah dy / (float);
```

Langkah 5: Letakkan pixel dengan berhasil menambah koordinat x dan y sesuai dan menyelesaikan gambar garis.

```

untuk (int v = 0; v <Langkah; v++)
{
    x = x + Xincrement;
    y = y + Yincrement;
    putpixel (x, y); }

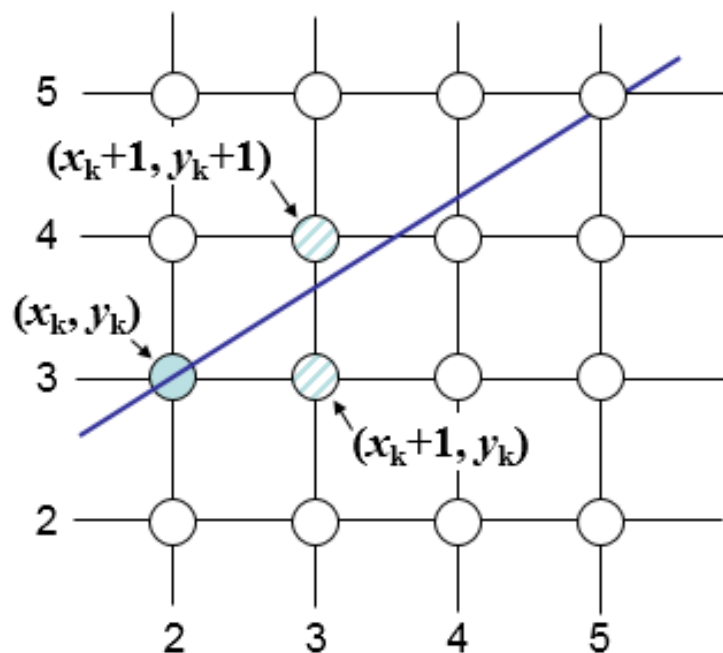
```

Generasi Garis Bresenham

Algoritma Bresenham adalah algoritma konversi pindai tambahan lainnya. Keuntungan besar dari algoritma ini adalah, hanya menggunakan perhitungan integer. Bergerak melintasi

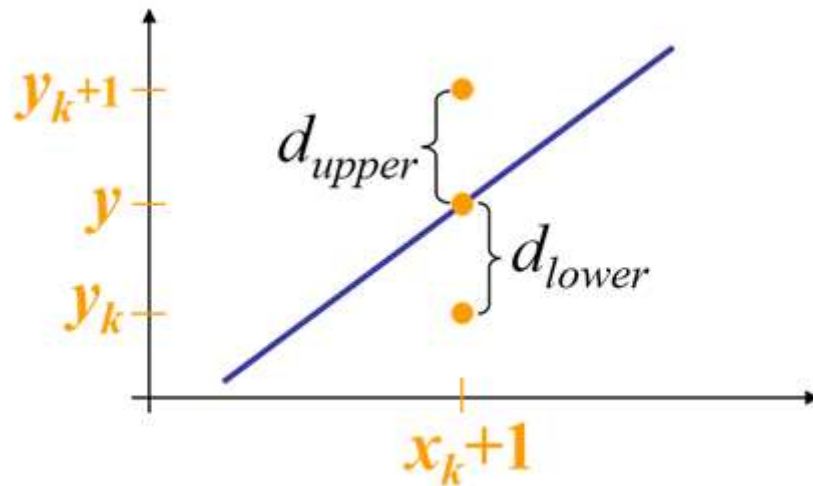
x sumbu dalam satuan interval dan pada setiap langkah memilih antara dua yang berbeda **y** koordinat.

Misalnya, seperti yang ditunjukkan pada ilustrasi berikut, dari posisi (2, 3) Anda harus memilih antara (3, 3) dan (3, 4). Anda ingin titik yang lebih dekat dengan garis aslinya.



Pada posisi sampel $x_k + 1$, pemisahan vertikal dari garis matematika adalah

dilabeli sebagai d atas dan d menurun.



Dari ilustrasi di atas, y berkoordinasi pada garis matematika di x_{k+1} adalah:

$$y = (x_{k+1} - x_k) \cdot m + y_k$$

Begitu, d atas dan d lebih rendah diberikan sebagai berikut:

$$d_{upper} = y_{k+1} - y$$

$$= (x_{k+1} - x_k) \cdot m + y_k - y$$

dan

$$d_{lower} = y - y_k$$

$$= (x_{k+1} - x_k) \cdot m + y_k - y_k$$

Anda dapat menggunakan ini untuk membuat keputusan sederhana tentang piksel mana yang lebih dekat dengan garis matematika. Keputusan sederhana ini didasarkan pada perbedaan antara dua posisi piksel.

$$d_{upper} - d_{lower} = 2 \cdot (x_{k+1} - x_k) \cdot m + 2 \cdot y_k - 1$$

Mari kita gantikan m dengan $\Delta y / \Delta x$ dimana Δx dan Δy perbedaan antara titik akhir.

$$\begin{aligned} 2 \cdot (d_{upper} - d_{lower}) &= 2 \cdot \left((x_{k+1} - x_k) \cdot \frac{\Delta y}{\Delta x} + y_k - y_k \right) \\ &= 2 \cdot (x_{k+1} - x_k) \cdot \Delta y + 2 \cdot y_k - 2 \cdot y_k \\ &= 2 \cdot (x_{k+1} - x_k) \cdot \Delta y \end{aligned}$$

Jadi, parameter keputusan hal_k Untuk k Langkah sepanjang garis diberikan oleh:

$$\begin{aligned} hal_k &= 2 \cdot (d_{upper} - d_{lower}) \\ &= 2 \cdot (x_{k+1} - x_k) \cdot \Delta y \end{aligned}$$

Tanda parameter keputusan hal_k sama dengan $d_{menurunkan} - d_{atas}$.

Jika hal_k negatif, lalu pilih piksel bawah, jika tidak pilih piksel atas.

Ingat, perubahan koordinat terjadi di sepanjang x sumbu dalam satuan langkah, sehingga Anda dapat melakukan segalanya dengan perhitungan bilangan bulat. Pada langkah $k + 1$, parameter keputusan diberikan sebagai:

$$p_{k+1} = 2p_k + y_{k+1} - 2y_k + 1 + p_k$$

Mengurangkan hal_k dari ini kita dapatkan:

$$p_{k+1} - p_k = 2p_k (y_{k+1} - y_k) - 2p_k (y_{k+1} - y_k)$$

Tapi, p_{k+1} sama dengan $p_k + 1$. Jadi:

$$p_{k+1} = p_k + 2p_k - 2p_k (y_{k+1} - y_k)$$

Dimana, $y_{k+1} - y_k$ adalah 0 atau 1 tergantung pada tanda hal_k .

Parameter keputusan pertama p_0 dievaluasi pada (x_0, y_0) diberikan sebagai:

$$p_0 = 2p_k - p_k$$

Sekarang, dengan mengingat semua poin dan perhitungan di atas, berikut adalah algoritma Bresenham untuk slope $m < 1$:

Langkah 1: Masukkan dua titik akhir garis, simpan titik akhir kiri di (x_0, y_0) .

Langkah 2: Plot titik (x_0, y_0) .

Langkah 3: Hitung konstanta Δx , Δy , $2\Delta y$, dan $(2\Delta y - \Delta x)$ dan dapatkan nilai pertama untuk parameter keputusan sebagai:

$$p_0 = 2p_k - p_k$$

Langkah 4: Setiap X_k di sepanjang garis, mulai dari $k = 0$, lakukan tes berikut:

Jika $hal_k < 0$, titik selanjutnya untuk plot adalah (x_{k+1}, y_k) dan

$$p_{k+1} = p_k + 2\Delta y \text{ Jika tidak,}$$

$$p_{k+1} = p_k + 2\Delta y - \Delta x$$

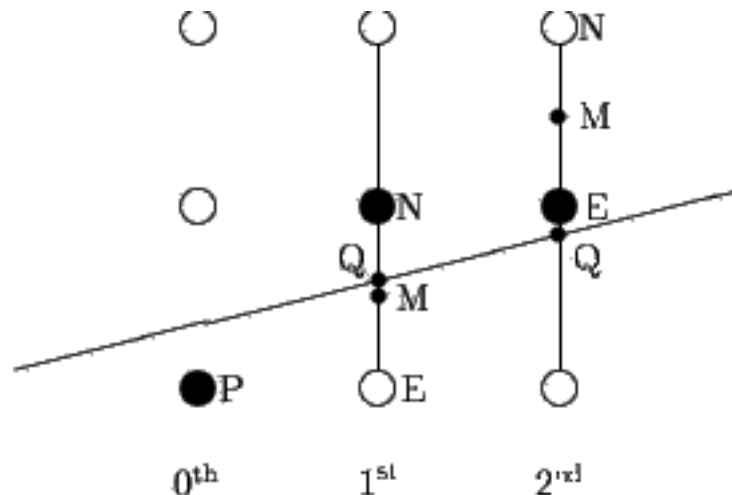
Langkah 5: Ulangi langkah 4 ($\Delta x - 1$ kali).

Untuk $m > 1$, cari tahu apakah Anda perlu menambah x sambil menambah y setiap kali. Setelah menyelesaikan, persamaan untuk parameter keputusan hal_k akan sangat mirip, hanya x dan y dalam persamaan yang dipertukarkan.

Algoritma Titik Tengah

Algoritma titik tengah disebabkan oleh Bresenham yang telah dimodifikasi oleh Pitteway dan Van Aken. Asumsikan bahwa Anda telah meletakkan titik P pada koordinat (x, y) dan kemiringan garis adalah $0 \leq k \leq 1$ seperti yang ditunjukkan pada ilustrasi berikut.

Sekarang Anda perlu memutuskan apakah akan meletakkan titik berikutnya di E atau N. Ini dapat dipilih dengan mengidentifikasi titik persimpangan Q paling dekat dengan titik N atau E. Jika titik persimpangan Q paling dekat dengan titik N maka N dianggap sebagai titik selanjutnya; jika tidak E.



Gambar: Algoritma Titik-Menengah

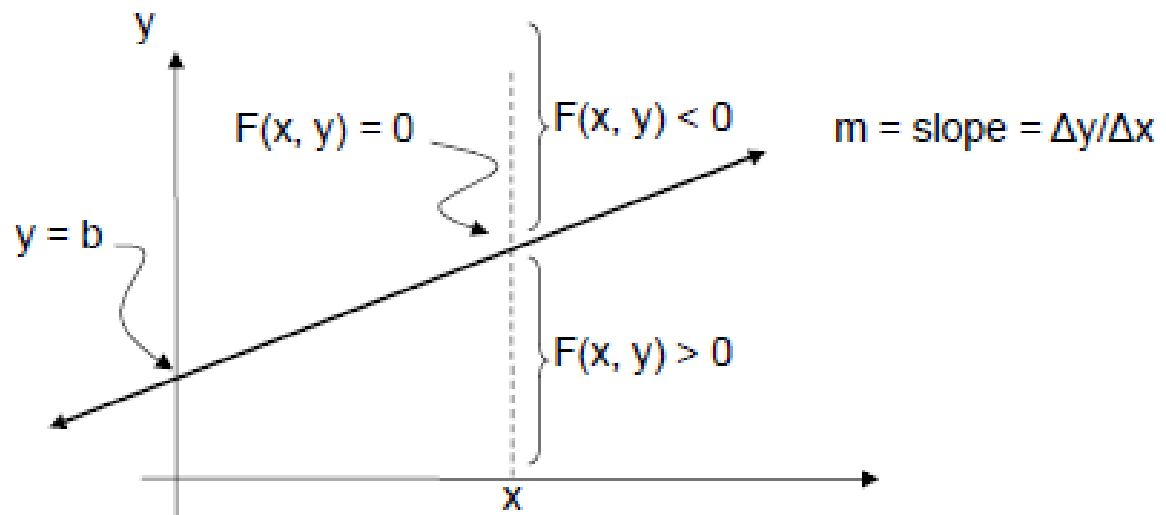
Untuk menentukan itu, pertama hitung titik tengah M ($x + 1, y + \frac{1}{2}$). Jika titik persimpangan Q dari garis dengan garis vertikal yang menghubungkan E dan N di bawah M, maka ambil E sebagai titik berikutnya; jika tidak, ambil N sebagai poin berikutnya.

Untuk memeriksa ini, kita perlu mempertimbangkan persamaan implisit:

$$F(x, y) = mx + b - y$$

Untuk m positif pada setiap X yang diberikan,

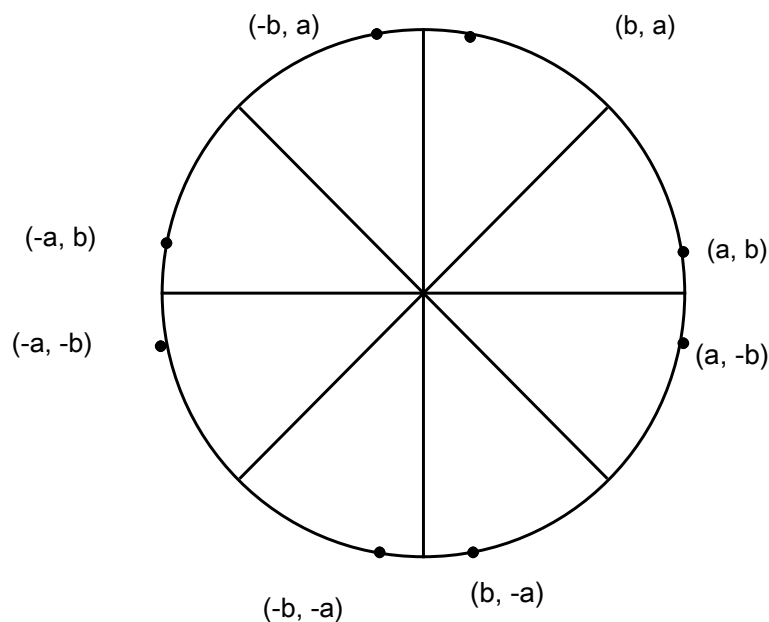
- Jika y ada di telepon, maka $F(x, y) = 0$
- Jika y berada di atas garis, maka $F(x, y) < 0$
- Jika y di bawah garis, maka $F(x, y) > 0$



3. ALGORITMA GENERASI LINGKARAN

Menggambar lingkaran di layar sedikit rumit daripada menggambar garis. Ada dua algoritma populer untuk menghasilkan lingkaran: **Algoritma Bresenham** dan **Algoritma Lingkaran Midpoint**. Algoritma ini didasarkan pada ide menentukan titik-titik berikutnya yang diperlukan untuk menggambar lingkaran. Mari kita bahas algoritma secara rinci:

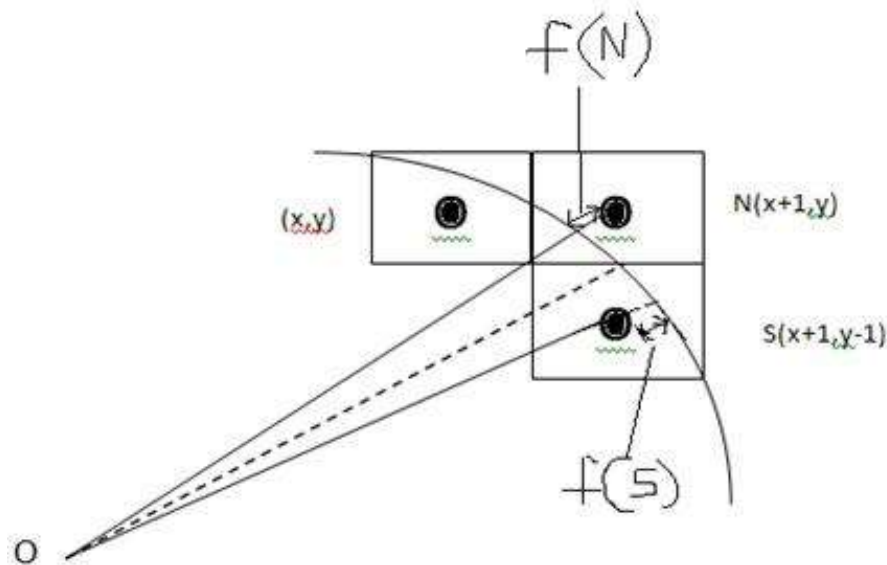
Persamaan lingkaran adalah $X^2 + Y^2 = r^2$, di mana r adalah jari-jari.



Algoritma Bresenham

Kami tidak dapat menampilkan lengkungan kontinu pada tampilan raster. Sebaliknya, kita harus memilih posisi piksel terdekat untuk menyelesaikan busur.

Dari ilustrasi berikut, Anda dapat melihat bahwa kami telah meletakkan piksel pada lokasi (X, Y) dan sekarang perlu memutuskan di mana menempatkan piksel berikutnya: pada $N(X + 1, Y)$ atau di $S(X + 1, Y - 1)$.



Ini dapat ditentukan oleh parameter keputusan **d**.

- Jika $d \leq 0$, maka $N(X + 1, Y)$ harus dipilih sebagai piksel berikutnya.
- Jika $d > 0$, maka $S(X + 1, Y - 1)$ harus dipilih sebagai piksel berikutnya.

Algoritma

Langkah 1: Dapatkan koordinat pusat lingkaran dan jari-jari, dan simpan di x , y , dan R masing-masing. Set $P = 0$ dan $Q = R$.

Langkah 2: Tetapkan parameter keputusan $D = 3 - 2R$.

Langkah 3: Ulangi melalui langkah-8 sementara $X < Y$.

Langkah 4: Sebut Draw Draw (X, Y, P, Q) .

Langkah 5: Menambah nilai P .

Langkah 6: Jika $D < 0$ maka $D = D + 4x + 6$.

Langkah 7: Set $Y = Y + 1$, $D = D + 4(XY) + 10$.

Langkah 8: Sebut Draw Draw (X, Y, P, Q) .

Gambarkan Metode Lingkaran (X, Y, P, Q) .

Panggil Putpixel $(X + P, Y + Q)$.

Panggil Putpixel $(X - P, Y + Q)$.

Panggil Putpixel $(X + P, Y - Q)$. Panggil

Putpixel $(X - P, Y - Q)$.

Panggil Putpixel $(X + Q, Y + X)$.

Panggil Putpixel (X - Q, Y + X).

Panggil Putpixel (X + Q, Y - X).

Panggil Putpixel (X - Q, Y - X).

Algoritma Titik Tengah

Langkah 1: Jari-jari input r dan pusat lingkaran (x_c, y_c) dan dapatkan titik pertama pada keliling lingkaran yang berpusat pada titik asal sebagai

$$(x_0, y_0) = (0, r)$$

Langkah 2: Hitung nilai awal parameter keputusan sebagai

$P_0 = 5/4 - r$ (Lihat uraian berikut untuk penyederhanaan persamaan ini.)

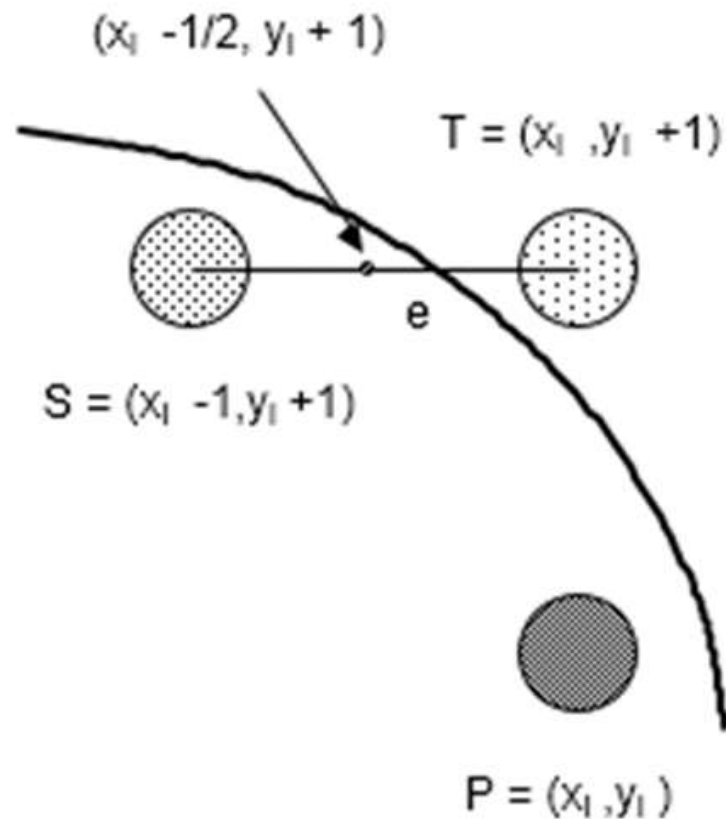
$$f(x, y) = x^2 + y^2 - r^2 = 0$$

$$f(x_{saya} - 1/2 + e, y_{saya} + 1)$$

$$= (x_{saya} - 1/2 + e)^2 + (y_{saya} + 1)^2 - r^2$$

$$= (x_{saya} - 1/2)^2 + (y_{saya} + 1)^2 - r^2 + 2(x_{saya} - 1/2)e + e^2$$

$$= f(x_{saya} - 1/2, y_{saya} + 1) + 2(x_{saya} - 1/2)e + e^2 = 0$$



Misalkan $di = f(x_{\text{saya}} - 1/2, y_{\text{saya}} + 1) = -2(x_{\text{saya}} - 1/2)e - e_2$

Jadi,

Jika $e < 0$ maka $di > 0$ jadi pilih titik $S = (x_{\text{saya}} - 1, y_{\text{saya}} + 1)$.

$$\begin{aligned} d_{i+1} &= f(x_{saya} - 1 - 1/2, y_{saya} + 1 + 1) = ((x_{saya} - 1/2) - 1)_{2+}((y_{saya} + 1) + 1)_{2-} r_2 \\ &= d_{saya} - 2(x_{saya} - 1) + 2(y_{saya} + 1) + 1 \\ &= d_{saya} + 2(y_{i+1} - x_{i+1}) + 1 \end{aligned}$$

Jika $e > 0$ maka $d_i \leq 0$ jadi pilih titik $T = (x_{\text{saya}}, y_{\text{saya}} + 1)$

$$\begin{aligned} d_{i+1} &= f(x_{\text{saya}} - 1/2, y_{\text{saya}} + 1 + 1) \\ &= d_{\text{saya} + 2 \text{ tahun}_{i+1} + 1} \end{aligned}$$

Nilai awal d_{saya} adalah

$$d_0 = f(r - 1/2, 0 + 1) = (r - 1/2)_{2+1} \cdot r_2$$

$$= 5/4 - r \{1 - r \text{ dapat digunakan jika } r \text{ adalah bilangan bulat}\}$$

Ketika titik $S = (x_{\text{saya}} - 1, y_{\text{saya}} + 1)$ dipilih

$$d_{i+1} = d_{\text{saye}} + 2x_{i+1} + 2 \text{ tahun}_{i+1} + 1$$

Ketika titik $T = (x_{\text{saye}}, y_{\text{saye}} + 1)$ dipilih

$$d_{i+1} = d_{\text{saye}} + 2 \text{ tahun}_{i+1} + 1$$

Langkah 3: Di setiap X_K posisi mulai dari $K = 0$, lakukan tes berikut:

Jika $P_K < 0$ maka titik selanjutnya pada lingkaran $(0,0)$ adalah (X_{K+1}, Y_K) dan

$$P_{K+1} = P_K + 2X_{K+1} + 1$$

Lain

$$P_{K+1} = P_K + 2X_{K+1} + 1 - 2Y_{K+1}$$

Di mana, $2X_{K+1} = 2X_{K+2}$ dan $2Y_{K+1} = 2Y_{K-2}$.

Langkah 4: Tentukan titik simetri pada tujuh oktan lainnya.

Langkah 5: Pindahkan setiap menghitung posisi piksel (X, Y) ke jalur melingkar yang berpusat pada (X_c, Y_c) dan plot nilai koordinat.

$$X = X + X_c,$$

$$Y = Y + Y_c$$

Langkah 6: Ulangi langkah-3 hingga 5 hingga $X > Y$.

4. MENGISI POLYGON

Polygon adalah daftar simpul yang diperintahkan seperti yang ditunjukkan pada gambar berikut. Untuk mengisi polygon dengan warna tertentu, Anda perlu menentukan piksel yang jatuh di perbatasan polygon dan yang jatuh di dalam polygon. Dalam bab ini, kita akan melihat bagaimana kita dapat mengisi polygon menggunakan teknik yang berbeda.

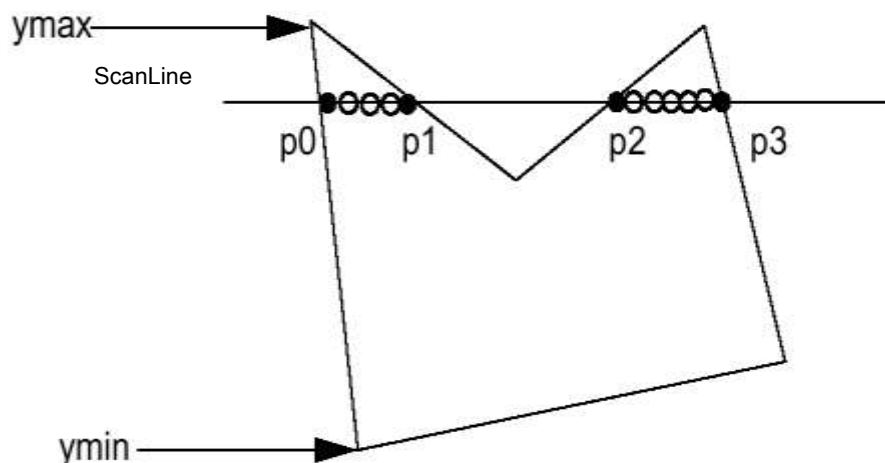


Gambar: A Polygon

Algoritma Pindai Jalur

Algoritma ini bekerja dengan memotong garis pemindaian dengan tepi polygon dan mengisi polygon di antara pasangan persimpangan. Langkah-langkah berikut menggambarkan cara kerja algoritma ini.

Langkah 1: Cari tahu Ymin dan Ymax dari polygon yang diberikan.



Langkah 2: ScanLine berpotongan dengan setiap tepi polygon dari Ymin ke Ymax. Beri nama setiap titik persimpangan polygon. Sesuai gambar yang ditunjukkan di atas, mereka dinamai p0, p1, p2, p3.

Langkah 3: Urutkan titik persimpangan dalam urutan peningkatan koordinat X yaitu (p_0, p_1) , (p_1, p_2) , dan (p_2, p_3) .

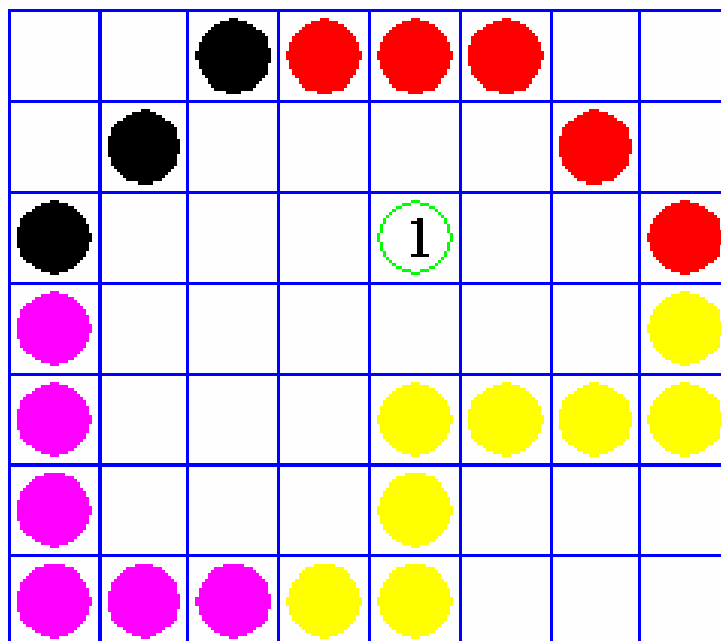
Langkah 4: Isi semua pasangan koordinat yang ada di dalam poligon dan abaikan pasangan alternatif.

Algoritma Isi Banjir

Terkadang kita menemukan sebuah objek di mana kita ingin mengisi area dan batasnya dengan warna yang berbeda. Kita dapat melukis objek seperti itu dengan warna interior yang ditentukan alih-alih mencari warna batas tertentu seperti pada algoritma batas pengisian.

Alih-alih mengandalkan batas objek, ia bergantung pada warna isi. Dengan kata lain, itu menggantikan warna interior objek dengan warna isi. Ketika tidak ada lagi piksel warna interior asli, algoritme selesai.

Sekali lagi, algoritma ini bergantung pada metode Four-connect atau Eight-connect untuk mengisi piksel. Tetapi alih-alih mencari warna batas, ia mencari semua piksel yang berdekatan yang merupakan bagian dari interior.



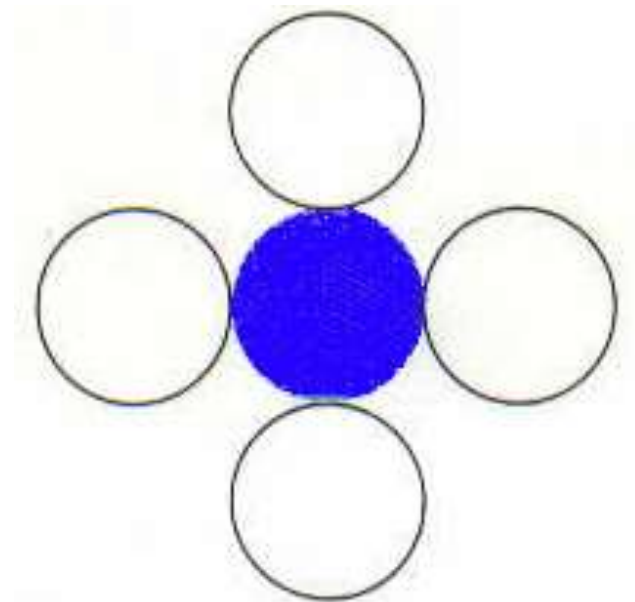
Algoritma Isi Batas

Algoritma batas mengisi berfungsi sebagai namanya. Algoritma ini mengambil titik di dalam objek dan mulai mengisi sampai menyentuh batas objek. Warna batas dan warna yang kita isi harus berbeda agar algoritma ini berfungsi.

Dalam algoritma ini, kita mengasumsikan bahwa warna batas sama untuk seluruh objek. Algoritma batas mengisi dapat diimplementasikan oleh 4-terkoneksi piksel atau 8connected piksel.

Poligon 4-Terhubung

Dalam teknik ini piksel yang terhubung 4 digunakan seperti yang ditunjukkan pada gambar. Kami menempatkan piksel di atas, di bawah, ke kanan, dan ke sisi kiri piksel saat ini dan proses ini akan berlanjut hingga kami menemukan batas dengan warna yang berbeda.



Algoritma

Langkah 1: Inisialisasi nilai titik benih (seedx, seedy), fcolor dan dcol.

Langkah 2: Tentukan nilai batas poligon.

Langkah 3: Periksa apakah titik unggulan saat ini adalah warna default, lalu ulangi langkah 4 dan 5 hingga piksel batas tercapai.

Jika getpixel (x, y) = dcol maka ulangi langkah 4 dan 5

Langkah 4: Ubah warna default dengan warna isian di titik seed.

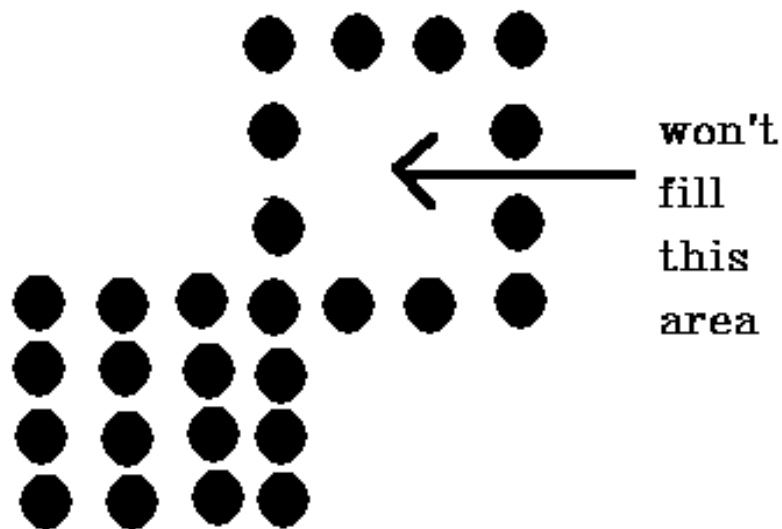
setPixel (seedx, seedy, fcol)

Langkah 5: Secara rekursif ikuti prosedur dengan empat titik lingkungan.

```
FloodFill (seedx - 1, seedy, fcol, dcol)
FloodFill (seedx + 1, seedy, fcol, dcol)
FloodFill (seedx, seedy - 1, fcol, dcol)
FloodFill (seedx, seedy + 1, fcol, dcol)
```

Langkah 6: Keluar

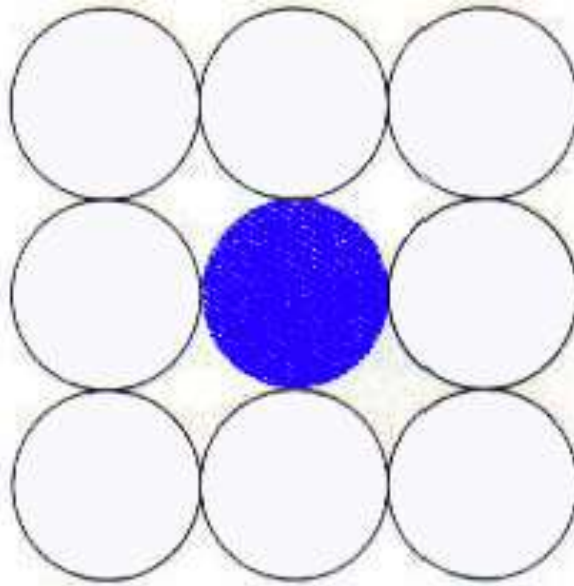
Ada masalah dengan teknik ini. Pertimbangkan kasus seperti yang ditunjukkan di bawah ini di mana kami mencoba mengisi seluruh wilayah. Di sini, gambar hanya diisi sebagian. Dalam kasus seperti itu, teknik piksel 4connected tidak dapat digunakan.



Poligon 8-Terhubung

Dalam teknik ini, 8-terhubung piksel digunakan seperti yang ditunjukkan pada gambar. Kami menempatkan piksel di atas, di bawah, sisi kanan dan kiri dari piksel saat ini seperti yang kami lakukan dalam teknik 4connected.

Selain itu, kami juga menempatkan piksel dalam diagonal sehingga seluruh area piksel saat ini tertutup. Proses ini akan berlanjut hingga kita menemukan batas dengan warna yang berbeda.



Algoritma

Langkah 1: Inisialisasi nilai titik benih (seedx, seedy), fcolor dan dcol.

Langkah 2: Tentukan nilai batas poligon.

Langkah 3: Periksa apakah titik unggulan saat ini adalah warna default, lalu ulangi langkah 4 dan 5 hingga piksel batas tercapai

Jika getpixel (x, y) = dcol maka ulangi langkah 4 dan 5

Langkah 4: Ubah warna default dengan warna isian di titik seed.

setPixel (seedx, seedy, fcol)

Langkah 5: Secara rekursif ikuti prosedur dengan empat titik lingkungan.

FloodFill (seedx - 1, seedy, fcol, dcol)

FloodFill (seedx + 1, seedy, fcol, dcol)

FloodFill (seedx, seedy - 1, fcol, dcol)

FloodFill (seedx, seedy + 1, fcol, dcol) FloodFill (seedx - 1, seedy + 1, fcol, dcol)

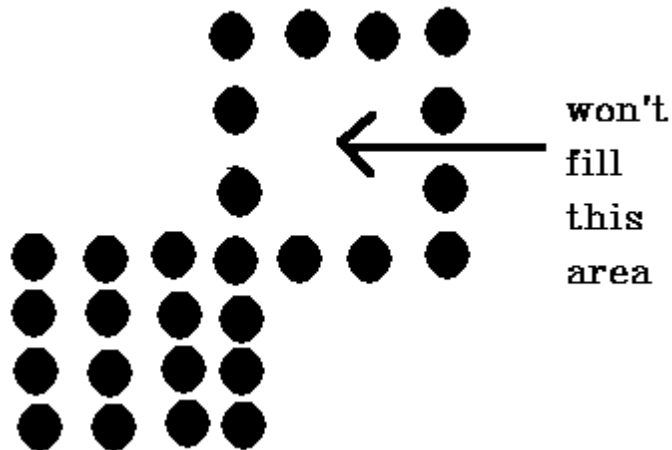
FloodFill (seedx + 1, seedy + 1, fcol, dcol)

FloodFill (seedx + 1, seedy - 1, fcol, dcol)

FloodFill (seedx - 1, seedy - 1, fcol, dcol)

Langkah 6: Keluar

Teknik piksel 4-terhubung gagal mengisi area seperti ditandai pada gambar berikut yang tidak akan terjadi dengan teknik 8-terhubung.



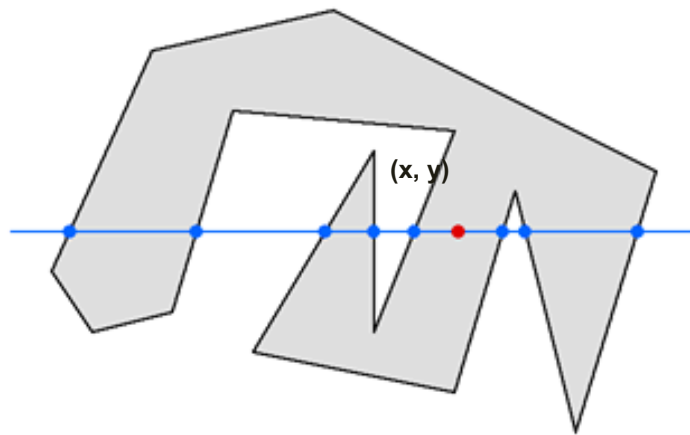
Tes dalam-luar

Metode ini juga dikenal sebagai **menghitung metode angka**. Saat mengisi objek, kita sering perlu mengidentifikasi apakah titik tertentu ada di dalam objek atau di luarnya. Ada dua metode yang dengannya kita dapat mengidentifikasi apakah titik tertentu berada di dalam suatu objek atau di luar.

- Aturan Ganjil-Genap
- Aturan angka berliku bukan nol

Aturan Ganjil-Genap

Dalam teknik ini, kita akan menghitung garis yang melintasi garis dari titik mana saja (x, y) hingga tak terbatas. Jika jumlah interaksi aneh, maka titik (x, y) adalah titik interior; dan jika jumlah interaksi genap, maka titik (x, y) adalah titik eksterior. Contoh berikut menggambarkan konsep ini.

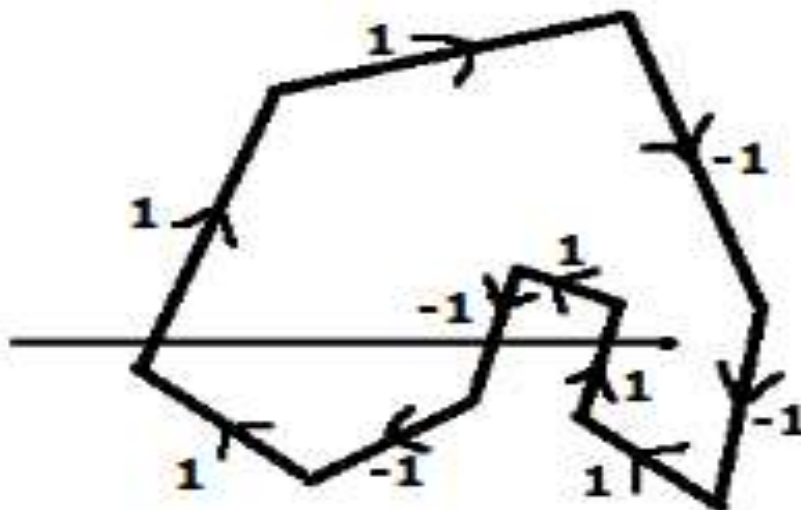


Dari gambar di atas, kita dapat melihat bahwa dari titik (x, y) , jumlah titik interaksi di sisi kiri adalah 5 dan di sisi kanan adalah 3. Dari kedua ujungnya, jumlah titik interaksi adalah ganjil, jadi titik dianggap dalam objek.

Aturan Angka Berliku Nol

Metode ini juga digunakan dengan poligon sederhana untuk menguji titik yang diberikan adalah interior atau tidak. Ini dapat dipahami dengan bantuan pin dan karet gelang. Pasang pin pada salah satu ujung poligon dan ikat karet gelang di dalamnya dan kemudian regangkan karet gelang di sepanjang tepi poligon.

Ketika semua tepi poligon ditutupi oleh karet gelang, periksa pin yang telah diperbaiki pada titik yang akan diuji. Jika kita menemukan setidaknya satu angin pada titik yang mempertimbangkannya dalam poligon, kita dapat mengatakan bahwa titik tersebut tidak berada di dalam poligon.



Dalam metode alternatif lain, berikan arah ke semua tepi poligon. Gambar garis pindai dari titik yang akan diuji ke arah paling kiri X.

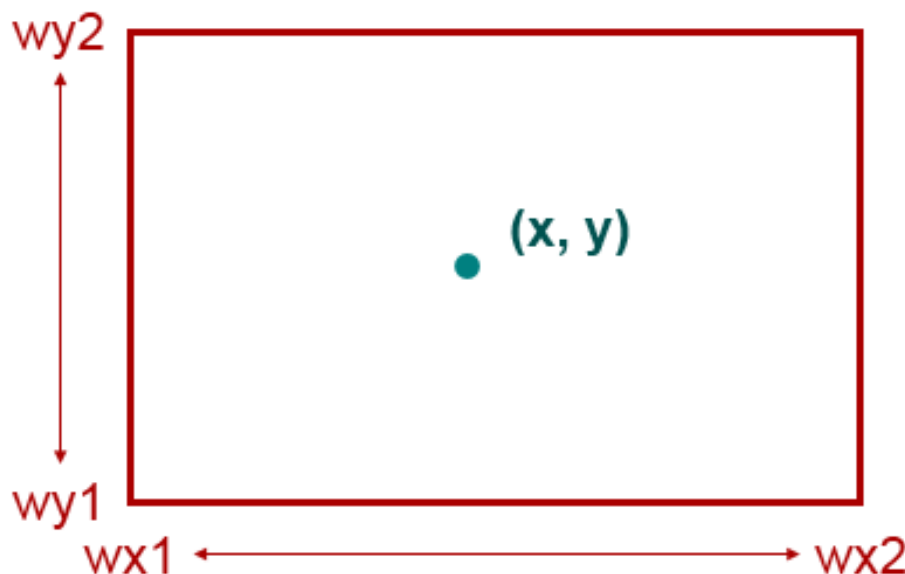
- Berikan nilai 1 ke semua tepi yang mengarah ke atas dan semua -1 lainnya sebagai nilai arah.
- Periksa nilai arah tepi dari mana garis pindai melewati dan jumlahkan mereka.
- Jika jumlah total nilai arah ini adalah nol, maka titik yang akan diuji adalah a **titik interior**, kalau tidak itu adalah **titik eksterior**.
- Pada gambar di atas, kami meringkas nilai-nilai arah dari mana garis pindai dilewati maka totalnya adalah $1 - 1 + 1 = 1$; yang bukan nol. Jadi intinya dikatakan sebagai titik interior.

Penggunaan utama kliping dalam grafik komputer adalah untuk menghapus objek, garis, atau segmen garis yang berada di luar panel tampilan. Transformasi tampilan tidak sensitif terhadap posisi titik relatif terhadap volume tampilan - terutama titik-titik di belakang penonton - dan perlu untuk menghapus titik-titik ini sebelum menghasilkan tampilan.

Klipping Poin

Memotong titik dari jendela tertentu sangat mudah. Pertimbangkan gambar berikut, di mana persegi panjang menunjukkan jendela. Klipping titik memberitahu kita apakah titik yang diberikan (X, Y) ada di dalam jendela yang diberikan atau tidak; dan memutuskan apakah kita akan menggunakan koordinat minimum dan maksimum jendela.

Koordinat X dari titik yang diberikan ada di dalam jendela, jika X terletak di antara $Wx1 \leq X \leq Wx2$. Cara yang sama, koordinat Y dari titik yang diberikan ada di dalam jendela, jika Y terletak di antara $Wy1 \leq Y \leq Wy2$.

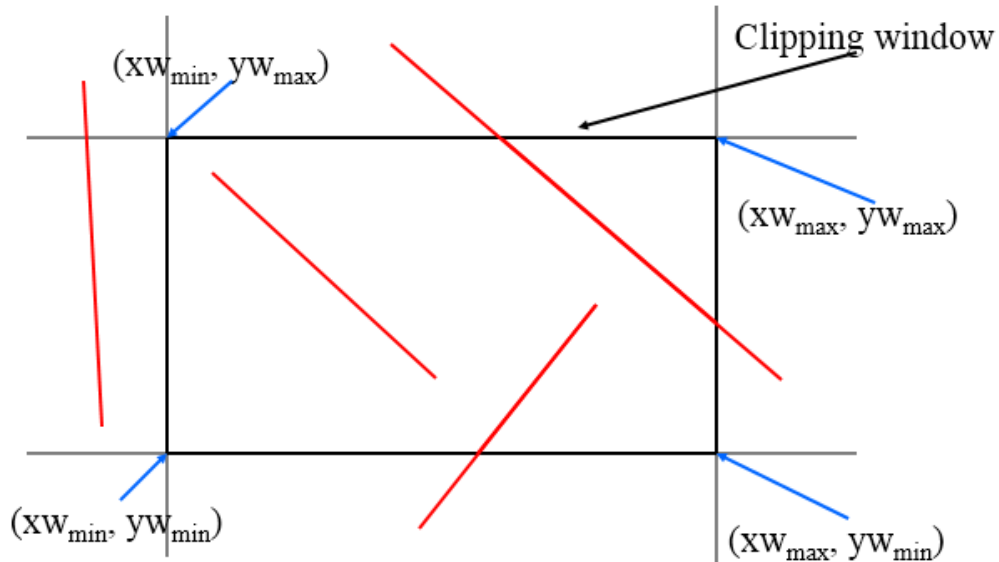


Klipping Baris

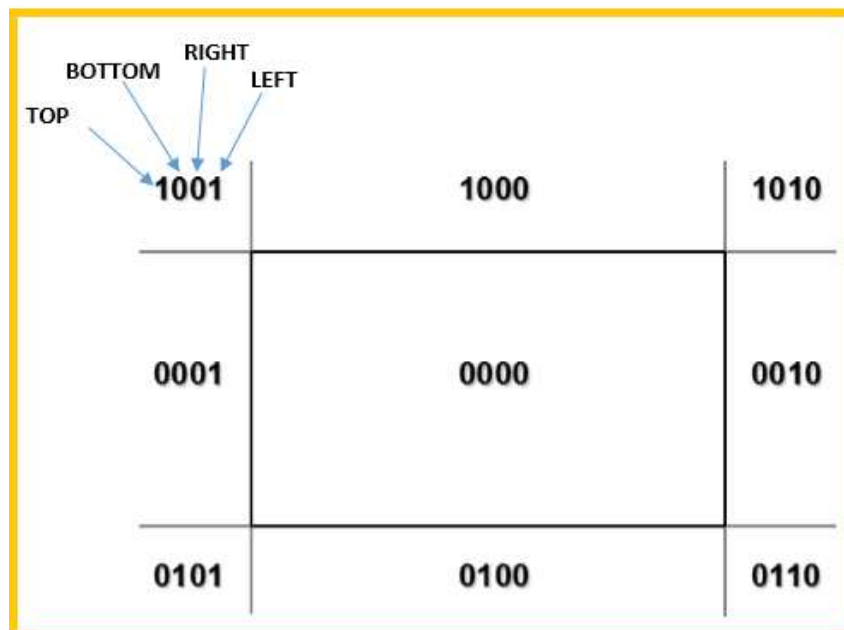
Konsep klipping garis sama dengan klipping titik. Dalam klipping garis, kita akan memotong bagian garis yang berada di luar jendela dan hanya menyimpan bagian yang ada di dalam jendela.

Kliping Garis Cohen-Sutherland

Algoritma ini menggunakan jendela kliping seperti yang ditunjukkan pada gambar berikut. Koordinat minimum untuk wilayah kliping adalah (XW_{min}, YW_{min}) dan koordinat maksimum untuk wilayah kliping adalah (XW_{maks}, YW_{maks}) .



Kami akan menggunakan 4-bit untuk membagi seluruh wilayah. Keempat bit ini mewakili bagian Atas, Bawah, Kanan, dan Kiri dari wilayah seperti yang ditunjukkan pada gambar berikut. Di sini, itu **TERATAS** dan **KIRI** bit diatur ke 1 karena itu adalah **KIRI ATAS** sudut.



Ada 3 kemungkinan untuk saluran:

1. Baris dapat sepenuhnya di dalam jendela (Baris ini harus diterima).
2. Baris dapat sepenuhnya di luar jendela (Baris ini akan sepenuhnya dihapus dari wilayah).
3. Garis dapat sebagian di dalam jendela (Kami akan menemukan titik persimpangan dan menggambar hanya bagian garis yang ada di dalam wilayah).

Algoritma

Langkah 1: Tetapkan kode wilayah untuk setiap titik akhir.

Langkah 2: Jika kedua titik akhir memiliki kode wilayah **0000** kemudian terima baris ini.

Langkah 3: Lain, lakukan yang logis **DAN** operasi untuk kedua kode wilayah.

Langkah 3.1: Jika hasilnya tidak **0000**, lalu tolak garis itu.

Langkah 3.2: Lain Anda perlu klip.

Langkah 3.2.1: Pilih titik akhir dari garis yang ada di luar jendela.

Langkah 3.2.2: Temukan titik persimpangan di batas jendela (berdasarkan kode wilayah).

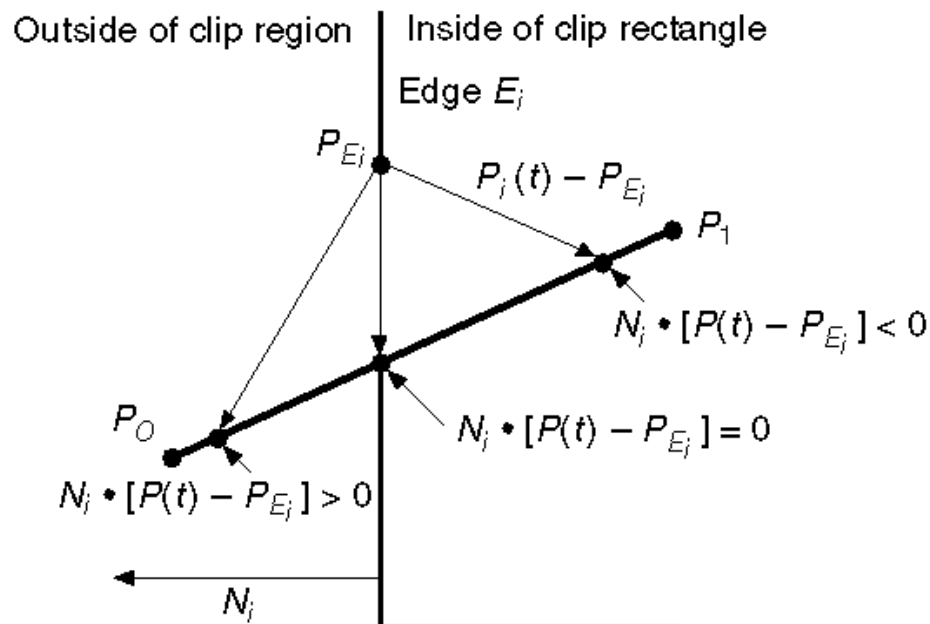
Langkah 3.2.3: Ganti titik akhir dengan titik persimpangan dan perbarui kode wilayah.

Langkah 3.2.4: Ulangi langkah 2 hingga kami menemukan garis yang terpotong diterima baik oleh sepele atau sepele.

Langkah-4: Ulangi langkah 1 untuk baris lain.

Algoritma Kliping Garis Cyrus-Beck

Algoritma ini lebih efisien daripada algoritma Cohen-Sutherland. Ini mempekerjakan representasi garis parametrik dan produk titik sederhana.



Persamaan parametrik garis adalah:

$$P_0 P_1: P(t) = P_0 + t(P_1 - P_0)$$

Biarkan N_i menjadi tepi normal luar E_{saya} . Sekarang pilih sembarang titik P_{E_i} di tepi E_{saya} maka produk titik $N_{saya} \cdot [P(t) - P_{E_i}]$ menentukan apakah titik $P(t)$ adalah "di dalam tepi klip" atau "di luar" tepi klip atau "di" tepi klip.

Titik $P(t)$ ada di dalam jika $N_{saya} \cdot [P(t) - P_{E_i}] < 0$

Titik $P(t)$ di luar jika $N_{saya} \cdot [P(t) - P_{E_i}] > 0$

Titik $P(t)$ ada di tepi jika $N_{saya} \cdot [P(t) - P_{E_i}] = 0$ (titik persimpangan)

$$N_{saya} \cdot [P(t) - P_{E_i}] = 0$$

$$N_{saya} \cdot [P_0 + t(P_1 - P_0) - P_{E_i}] = 0 \text{ (Mengganti } P(t) \text{ dengan } P_0 + t(P_1 - P_0))$$

$$N_{saya} \cdot [P_0 - P_{E_i} + N_{saya} \cdot t(P_1 - P_0)] = 0$$

$$N_{saya} \cdot [P_0 - P_{E_i} + N_{saya} \cdot tD] = 0 \text{ (mengganti } D \text{ untuk } [P_1 - P_0])$$

$$N_{saya} \cdot [P_0 - P_{E_i}] = -N_{saya} \cdot tD$$

Persamaan untuk t menjadi,

$$t = N \cdot \frac{P_1 \cdot 0 - P_0 \cdot E}{-N \cdot D}$$

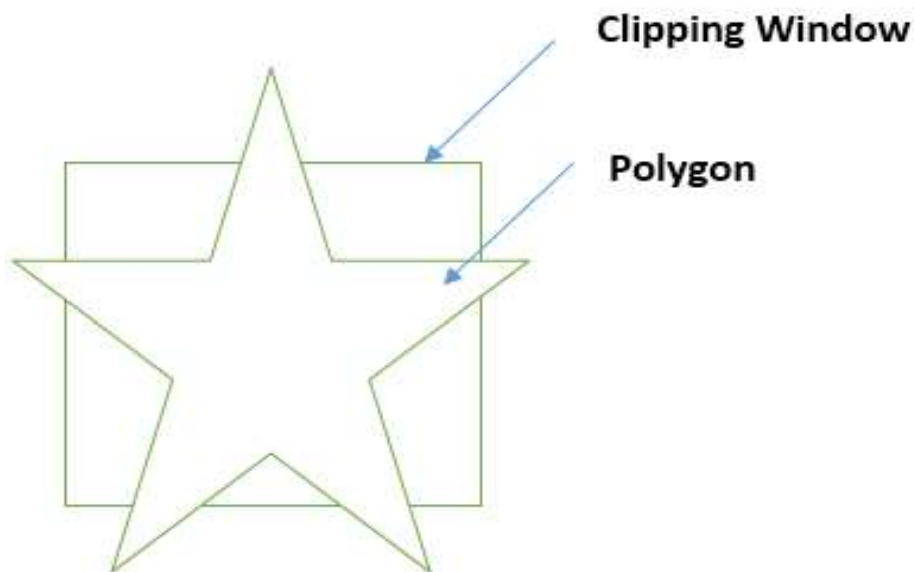
Itu berlaku untuk kondisi berikut:

1. $N_{sayu} \neq 0$ (kesalahan tidak dapat terjadi)
2. $D \neq 0$ ($P_1 \neq P_0$)
3. $N_{sayu} \cdot D \neq 0$ ($P_0 P_1$ tidak sejajar dengan E_{sayu})

Kliping Polygon (Algoritma Sutherland Hodgman)

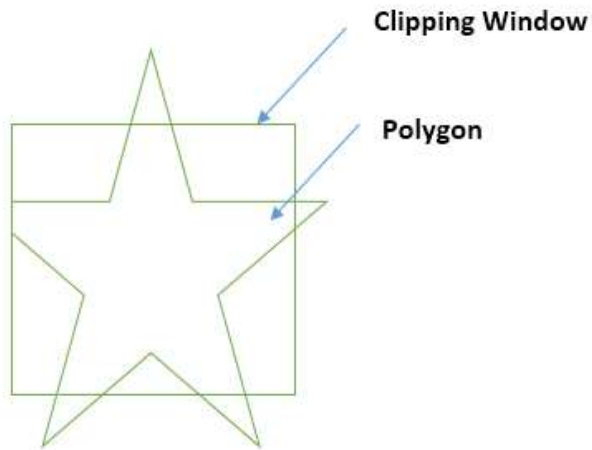
Polygon juga dapat dipotong dengan menentukan jendela kliping. Algoritma pemangkasan poligon Sutherland Hodgeman digunakan untuk pemangkasan poligon. Dalam algoritme ini, semua simpul poligon terpotong pada setiap tepi jendela kliping.

Pertama poligon terpotong di tepi kiri jendela poligon untuk mendapatkan simpul baru poligon. Verteks baru ini digunakan untuk klip poligon terhadap tepi kanan, tepi atas, tepi bawah, dari jendela kliping seperti yang ditunjukkan pada gambar berikut.

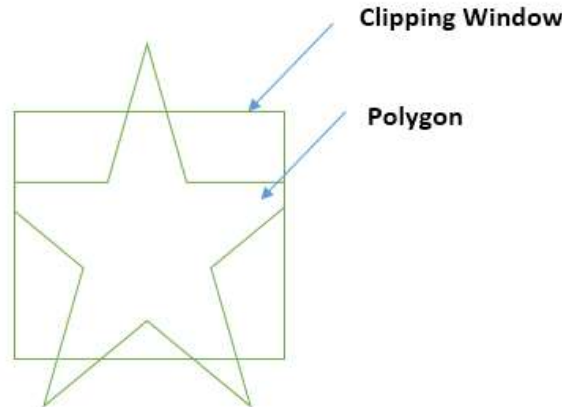


Gambar: Polygon sebelum Mengisi

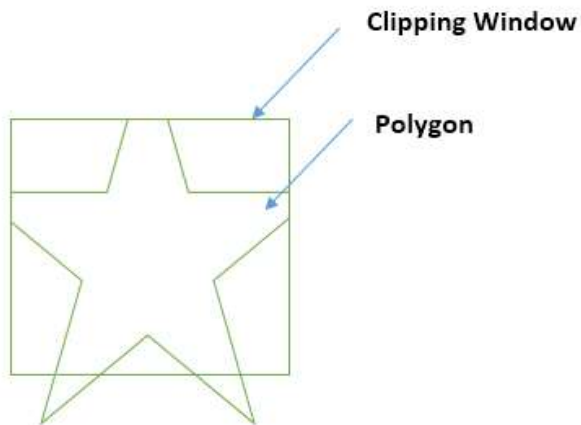
Saat memproses tepi poligon dengan jendela kliping, titik persimpangan ditemukan jika tepi tidak sepenuhnya di dalam jendela kliping dan tepi parsial dari titik persimpangan ke tepi luar terpotong. Gambar-gambar berikut menunjukkan kliping tepi kiri, kanan, atas dan bawah:



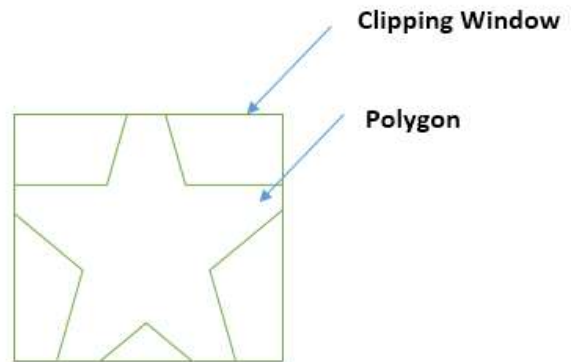
Gambar: Memotong Tepi Kiri



Gambar: Memotong Tepi Kanan



Gambar: Memotong Tepi Atas



Gambar: Memotong Tepi Bawah

Klipping Teks

Berbagai teknik digunakan untuk menyediakan klipping teks dalam grafik komputer. Itu tergantung pada metode yang digunakan untuk menghasilkan karakter dan persyaratan aplikasi tertentu. Ada tiga metode untuk klipping teks yang tercantum di bawah ini:

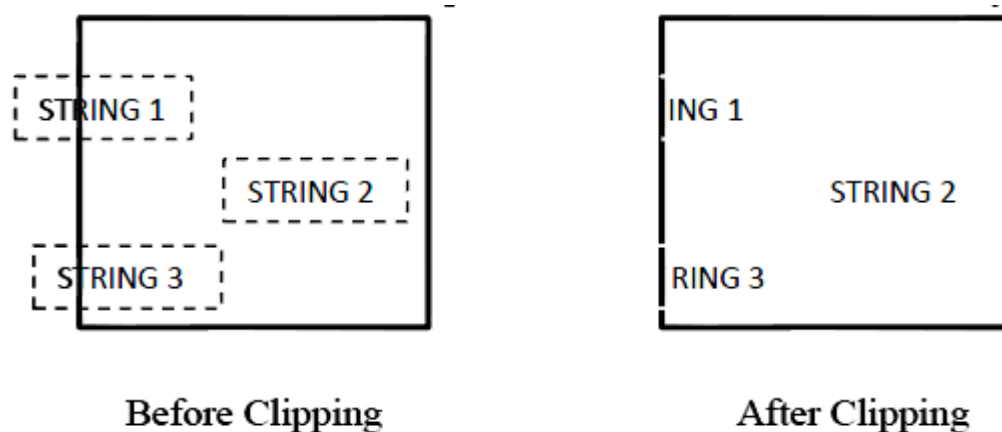
1. Semua atau tidak ada klipping string
2. Semua atau tidak ada klipping karakter
3. Klipping teks

Gambar berikut menunjukkan semua atau tidak ada kliping string:



Dalam semua atau tidak ada metode kliping string, baik kami menyimpan seluruh string atau kami menolak seluruh string berdasarkan jendela kliping. Seperti yang ditunjukkan pada gambar di atas, STRING2 sepenuhnya berada di dalam jendela kliping jadi kami menyimpannya dan STRING1 hanya sebagian di dalam jendela, kami menolak.

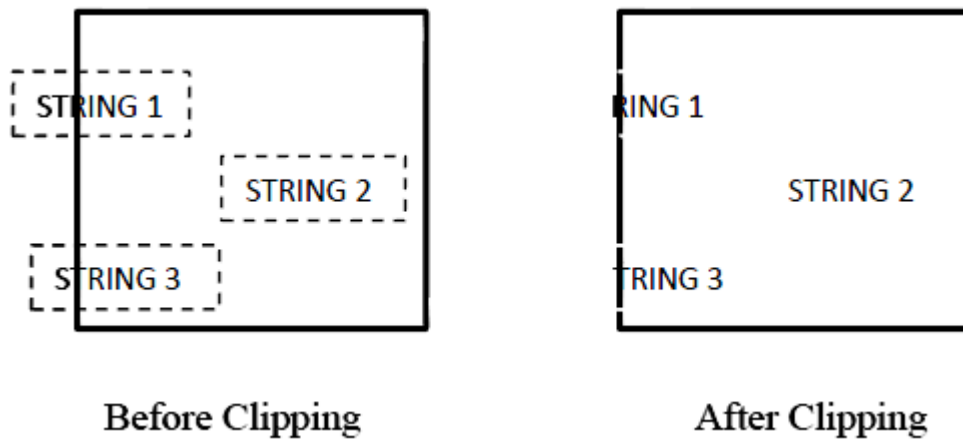
Gambar berikut menunjukkan semua atau tidak ada kliping karakter:



Metode kliping ini didasarkan pada karakter daripada seluruh string. Dalam metode ini jika string sepenuhnya berada di dalam jendela kliping, maka kita menyimpannya. Jika sebagian di luar jendela, maka:

- Anda menolak hanya bagian dari string yang berada di luar
- Jika karakter berada pada batas jendela kliping, maka kami membuang seluruh karakter dan menyimpan string lainnya.

Gambar berikut menunjukkan kliping teks:

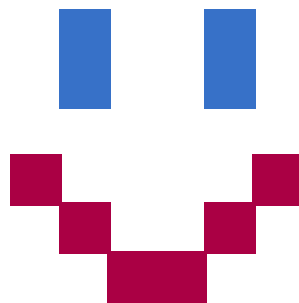


Metode kliping ini didasarkan pada karakter daripada seluruh string. Dalam metode ini jika string sepenuhnya berada di dalam jendela kliping, maka kita menyimpannya. Jika sebagian di luar jendela, maka

- Anda hanya menolak sebagian string yang berada di luar.
- Jika karakter berada di batas jendela kliping, maka kami hanya membuang bagian karakter yang berada di luar jendela kliping.

Grafik Bitmap

Bitmap adalah kumpulan piksel yang menggambarkan suatu gambar. Ini adalah jenis grafik komputer yang digunakan komputer untuk menyimpan dan menampilkan gambar. Dalam jenis grafik ini, gambar disimpan sedikit demi sedikit dan karenanya dinamai Bit-map graphics. Untuk pemahaman yang lebih baik mari kita perhatikan contoh berikut ini di mana kita menggambar wajah tersenyum menggunakan grafik bit-map.



Gambar: Wajah Smiley Asli

Sekarang kita akan melihat bagaimana wajah tersenyum ini disimpan sedikit demi sedikit dalam grafik komputer.

	A	B	C	D	E	F
1		B1			E1	
2		B2			E2	
3						
4	A4					F4
5		B5			E5	
6			C6	D6		

Gambar: Penyimpanan bitmap wajah tersenyum

Dengan mengamati wajah asli yang tersenyum dengan cermat, kita dapat melihat bahwa ada dua garis biru yang direpresentasikan sebagai B1, B2 dan E1, E2 pada gambar di atas.

Dengan cara yang sama, smiley diwakili menggunakan bit kombinasi A4, B5, C6, D6, E5, dan F4 masing-masing.

Kerugian utama dari grafik bitmap adalah:

- Kami tidak dapat mengubah ukuran gambar bitmap. Jika Anda mencoba untuk mengubah ukuran, piksel menjadi kabur.
- Bitmap berwarna bisa sangat besar.

Transformasi berarti mengubah beberapa gambar menjadi sesuatu yang lain dengan menerapkan aturan. Kita dapat memiliki berbagai jenis transformasi seperti terjemahan, penskalaan naik atau turun, rotasi, geser, dll. Ketika transformasi terjadi pada bidang 2D, itu disebut transformasi 2D.

Transformasi memainkan peran penting dalam grafik komputer untuk memposisikan ulang grafik pada layar dan mengubah ukuran atau orientasinya.

Koordinat Homogen

Untuk melakukan urutan transformasi seperti terjemahan diikuti oleh rotasi dan penskalaan, kita perlu mengikuti proses berurutan:

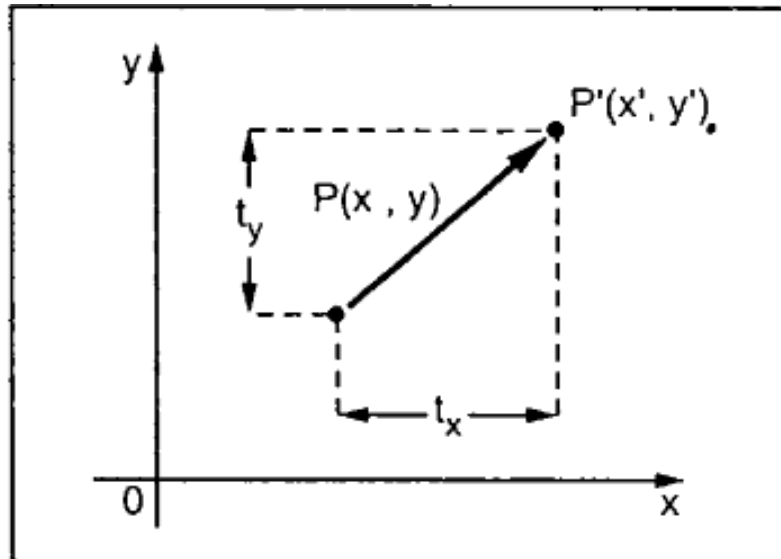
1. Terjemahkan koordinatnya,
2. Putar koordinat yang diterjemahkan, lalu
3. Skala koordinat yang diputar untuk menyelesaikan transformasi komposit.

Untuk mempersingkat proses ini, kita harus menggunakan matriks transformasi 3×3 daripada matriks transformasi 2×2 . Untuk mengubah 2×2 matriks menjadi 3×3 matriks, kita harus menambahkan koordinat dummy ekstra W .

Dengan cara ini, kita dapat mewakili titik dengan 3 angka, bukan 2 angka, yang disebut **Koordinat Homogen** sistem. Dalam sistem ini, kita dapat mewakili semua persamaan transformasi dalam perkalian matriks. Setiap titik Cartesian $P (X, Y)$ dapat dikonversi ke koordinat homogen oleh $P'(X_h, Y_h, h)$.

Terjemahan

Terjemahan memindahkan objek ke posisi berbeda di layar. Anda dapat menerjemahkan suatu titik dalam 2D dengan menambahkan koordinat terjemahan (t_x, t_y) ke koordinat awal (X, Y) untuk mendapatkan koordinat baru (X', Y') .



Dari gambar di atas, Anda dapat menulis bahwa:

$$X' = X + t_x$$

$$Y' = Y + t_y$$

Pasangan (t_x , t_y) disebut vektor terjemahan atau vektor pergeseran. Persamaan di atas juga dapat direpresentasikan menggunakan vektor kolom.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

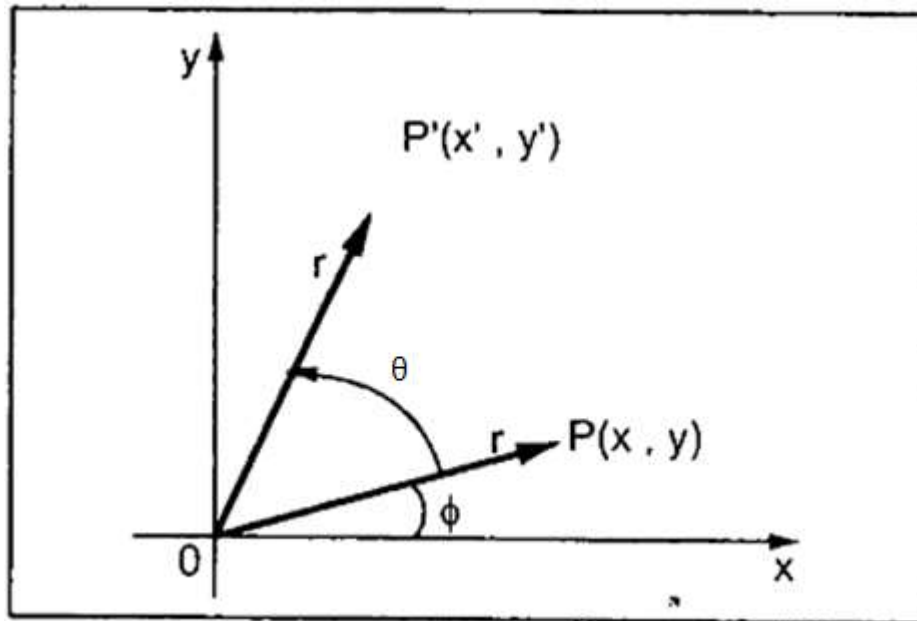
Kita dapat menuliskannya sebagai:

$$P' = P + T$$

Rotasi

Dalam rotasi, kita memutar objek pada sudut tertentu θ (theta) dari asalnya. Dari gambar berikut, kita dapat melihat bahwa titik P (X, Y) terletak di sudut Φ dari koordinat X horizontal dengan jarak r dari titik asal.

Biarkan kami mengira Anda ingin memutarnya pada sudut θ . Setelah memutarnya ke lokasi baru, Anda akan mendapatkan titik baru P '(X', Y ').



Dengan menggunakan trigonometri standar, koordinat awal titik P (X, Y) dapat direpresentasikan sebagai:

$$X = r \cos \Phi \dots\dots\dots (1)$$

$$Y = r \sin \Phi \dots\dots\dots (2)$$

Cara yang sama kita dapat mewakili titik P '(X', Y ') sebagai:

$$X' = r \cos (\Phi + \theta) = r \cos \Phi \cos \theta - r \sin \Phi \sin \theta \dots\dots\dots (3)$$

$$Y' = r \sin (\Phi + \theta) = r \cos \Phi \sin \theta + r \sin \Phi \cos \theta \dots\dots\dots (4)$$

Mengganti persamaan (1) & (2) dalam (3) & (4) masing-masing, akan kita dapatkan

$$X' = X \cos \theta - Y \sin \theta$$

$$Y' = X \sin \theta + Y \cos \theta$$

Mewakili persamaan di atas dalam bentuk matriks,

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

ATAU

$$P' = P \cdot R$$

Di mana R adalah matriks rotasi

$$R = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

Sudut rotasi bisa positif dan negatif.

Untuk sudut rotasi positif, kita dapat menggunakan matriks rotasi di atas. Namun, untuk rotasi sudut negatif, matriks akan berubah seperti yang ditunjukkan di bawah ini:

$$\begin{aligned}
 &= \begin{bmatrix} \cos(-\theta) & \sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
 &= \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (\because \cos(-\theta) = \cos \theta \text{ dan } \sin(-\theta) = -\sin \theta)
 \end{aligned}$$

Scaling

Untuk mengubah ukuran objek, transformasi skala digunakan. Dalam proses penskalaan, Anda memperluas atau memampatkan dimensi objek. Penskalaan dapat dicapai dengan mengalikan koordinat asli objek dengan faktor penskalaan untuk mendapatkan hasil yang diinginkan.

Mari kita asumsikan bahwa koordinat aslinya adalah (X, Y), faktor penskalaannya adalah (S_x, S_y),

dan koordinat yang dihasilkan adalah (X', Y'). Ini dapat direpresentasikan secara matematis seperti yang ditunjukkan di bawah ini:

$$X' = X \cdot S_x \quad \text{dan} \quad Y' = Y \cdot S_y$$

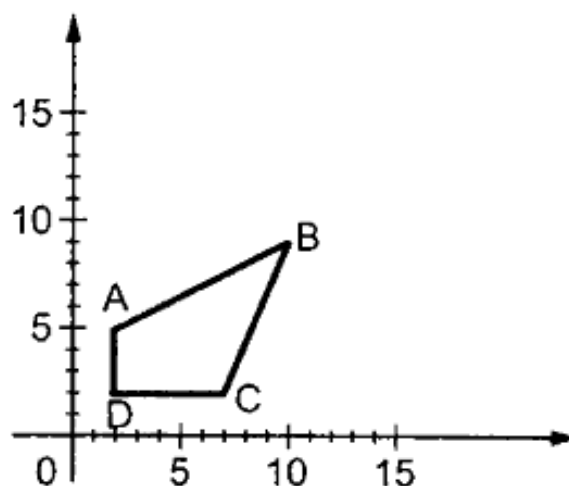
Faktor penskalaan S_x, S_y skala objek dalam arah X dan Y masing-masing. Persamaan di atas juga dapat direpresentasikan dalam bentuk matriks seperti di bawah ini:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

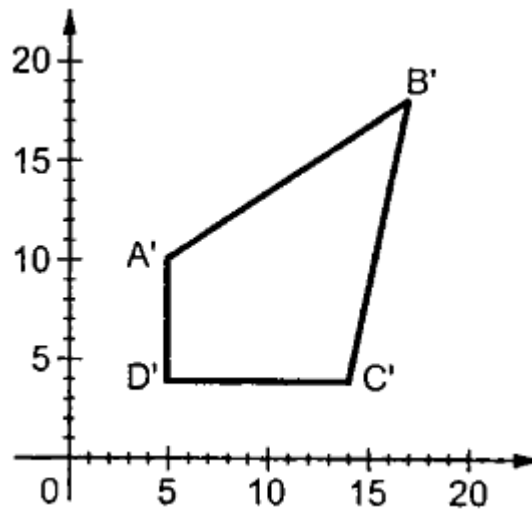
ATAU

$$P' = P \cdot S$$

Di mana S adalah matriks penskalaan. Proses penskalaan ditunjukkan pada gambar berikut.



Gambar: Sebelum proses penskalaan



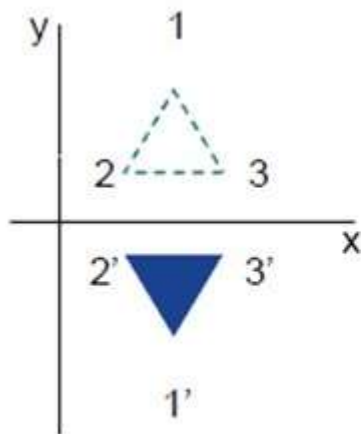
Gambar: Setelah Proses Penskalaan

Jika kami memberikan nilai kurang dari 1 ke faktor penskalaan S , maka kami dapat mengurangi ukuran objek. Jika kami memberikan nilai lebih besar dari 1, maka kami dapat meningkatkan ukuran objek.

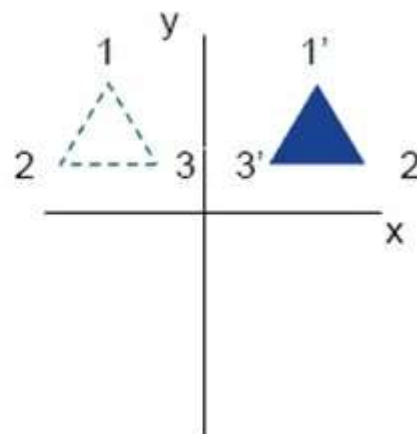
Refleksi

Refleksi adalah bayangan cermin dari objek asli. Dengan kata lain, kita dapat mengatakan bahwa ini adalah operasi rotasi dengan 180° . Dalam transformasi refleksi, ukuran objek tidak berubah.

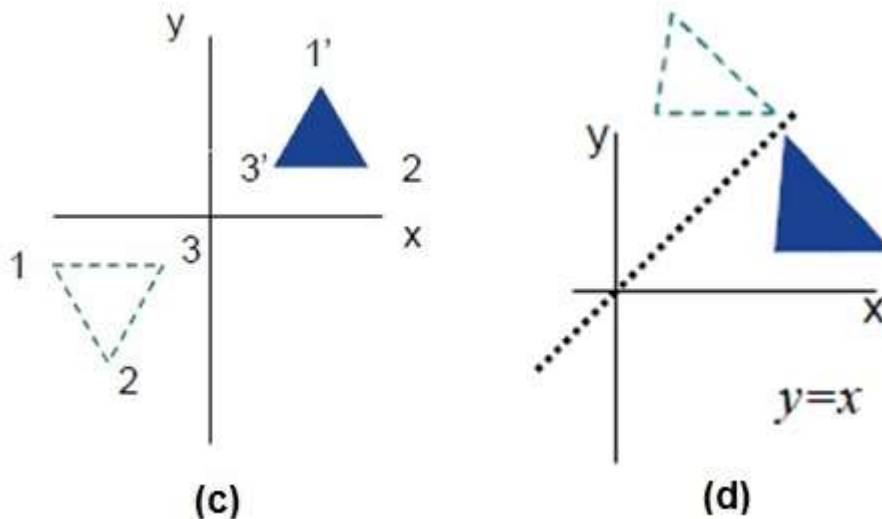
Gambar-gambar berikut menunjukkan refleksi sehubungan dengan sumbu X dan Y , dan tentang asal masing-masing.



(a)



(b)

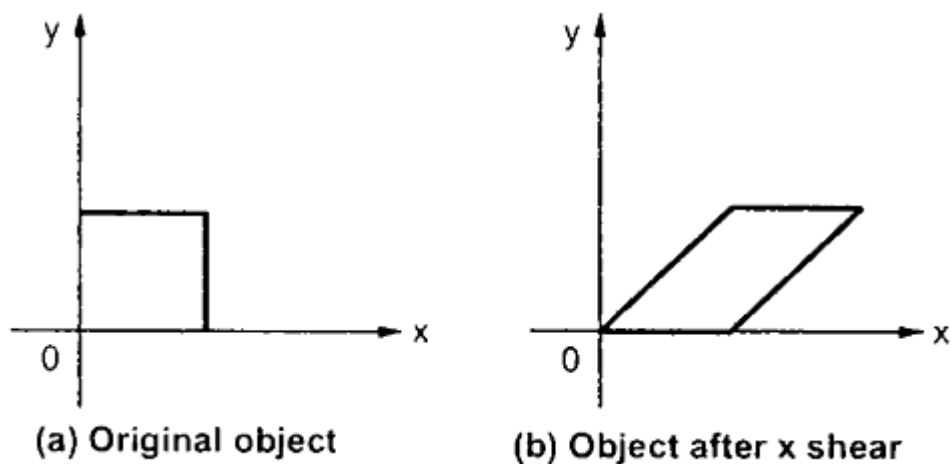
Gambar: Refleksi tentang garis $y = x$

Mencukur

Transformasi yang miring bentuk objek disebut transformasi geser. Ada dua transformasi geser **X-Shear** dan **Y-Shear**. Satu bergeser nilai koordinat X dan bergeser nilai koordinat Y lainnya. Namun, dalam kedua kasus, hanya satu koordinat yang mengubah koordinatnya dan yang lainnya mempertahankan nilainya. Geser juga disebut sebagai **Miring**.

X-Shear

X-Shear mempertahankan koordinat Y dan perubahan dilakukan pada koordinat X, yang menyebabkan garis vertikal miring ke kanan atau kiri seperti yang ditunjukkan pada gambar di bawah ini.



Matriks transformasi untuk X-Shear dapat direpresentasikan sebagai:

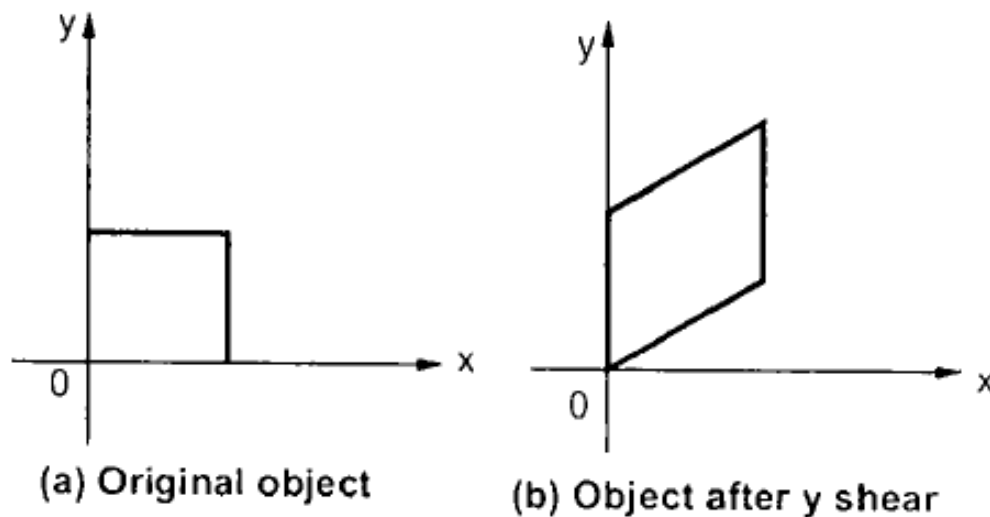
$$T_x = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X' = X + Sh_x \cdot Y$$

$$Y' = Y$$

Y-Shear

Y-Shear mempertahankan koordinat X dan mengubah koordinat Y yang menyebabkan garis horizontal berubah menjadi garis yang miring ke atas atau ke bawah seperti yang ditunjukkan pada gambar berikut.



Y-Shear dapat direpresentasikan dalam matriks dari sebagai:

$$T_y = \begin{bmatrix} 1 & 0 & Sh_y \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Y' = Y + Sh_y \cdot X$$

$$X' = X$$

Transformasi Komposit

Jika transformasi bidang T1 diikuti oleh transformasi bidang kedua T2, maka hasilnya sendiri dapat diwakili oleh transformasi T tunggal yang merupakan komposisi T1 dan T2 yang diambil dalam urutan itu. Ini ditulis sebagai $T = T1 \cdot T2$.

Transformasi komposit dapat dicapai dengan menggabungkan matriks transformasi untuk mendapatkan matriks transformasi gabungan.

Matriks gabungan:

$$[T] [X] = [X] [T1] [T2] [T3] [T4] \dots [Tn]$$

Di mana $[Ti]$ adalah kombinasi dari

- Terjemahan
- Scaling
- Pencukuran
- Rotasi
- Refleksi

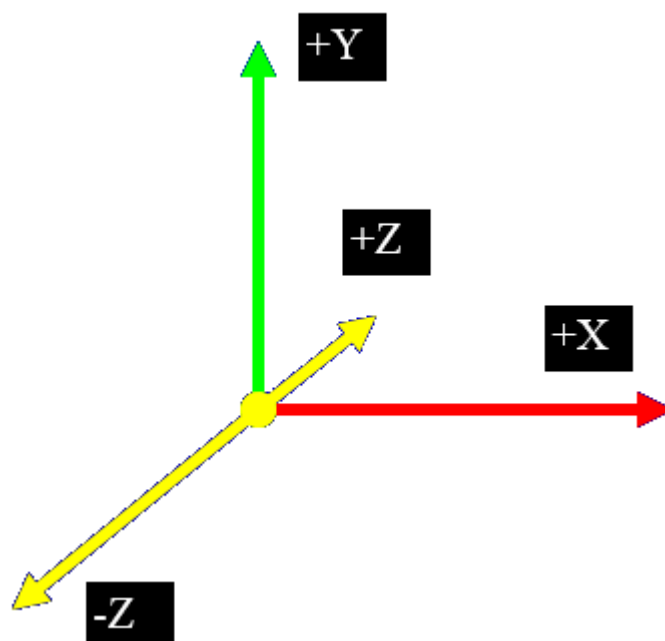
Perubahan dalam urutan transformasi akan menghasilkan hasil yang berbeda, seperti dalam perkalian matriks umum tidak kumulatif, yaitu $[A] \cdot [B] \neq [B] \cdot [A]$ dan urutan perkalian. Tujuan dasar penyusunan transformasi adalah untuk mendapatkan efisiensi dengan menerapkan satu transformasi tersusun ke satu titik, daripada menerapkan serangkaian transformasi, satu demi satu.

Misalnya, untuk memutar objek tentang titik arbitrer (Xp, Yp) , kita harus melakukan tiga langkah:

1. Titik terjemahan (Xp, Yp) ke titik asal.
2. Putar tentang asal.
3. Akhirnya, terjemahkan pusat rotasi kembali ke tempatnya.

Dalam sistem 2D, kami hanya menggunakan dua koordinat X dan Y tetapi dalam 3D, koordinat Z tambahan ditambahkan. Teknik grafis 3D dan aplikasinya sangat penting bagi industri hiburan, permainan, dan desain berbantuan komputer. Ini adalah bidang penelitian berkelanjutan dalam visualisasi ilmiah.

Selain itu, komponen grafik 3D sekarang menjadi bagian dari hampir setiap komputer pribadi dan, meskipun secara tradisional ditujukan untuk perangkat lunak intensif-grafis seperti game, mereka semakin banyak digunakan oleh aplikasi lain.

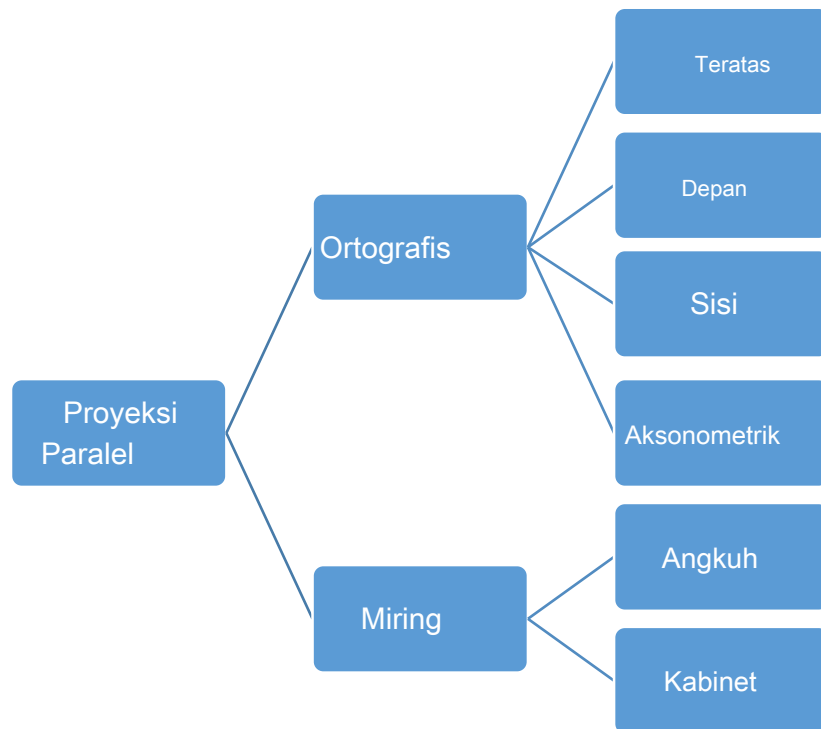


Proyeksi Paralel

Proyeksi paralel membuang koordinat-z dan garis-garis paralel dari setiap titik pada objek diperpanjang sampai mereka memotong bidang pandangan. Dalam proyeksi paralel, kami menentukan arah proyeksi alih-alih pusat proyeksi.

Dalam proyeksi paralel, jarak dari pusat proyeksi ke bidang proyek tidak terbatas. Dalam jenis proyeksi ini, kami menghubungkan simpul yang diproyeksikan berdasarkan segmen garis yang sesuai dengan koneksi pada objek asli.

Proyeksi paralel kurang realistis, tetapi bagus untuk pengukuran yang tepat. Dalam jenis proyeksi ini, garis paralel tetap paralel dan sudut tidak dipertahankan. Berbagai jenis proyeksi paralel ditampilkan dalam hierarki berikut.

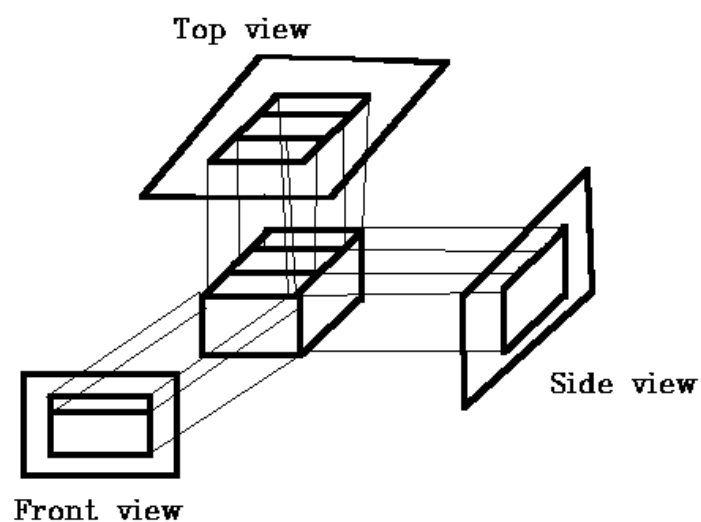


Gambar: Jenis Proyeksi Paralel

Proyeksi Orthografis

Dalam proyeksi ortografis, arah proyeksi adalah normal terhadap proyeksi bidang. Ada tiga jenis proyeksi ortografis:

- Proyeksi depan
- Proyeksi Atas
- Proyeksi Samping



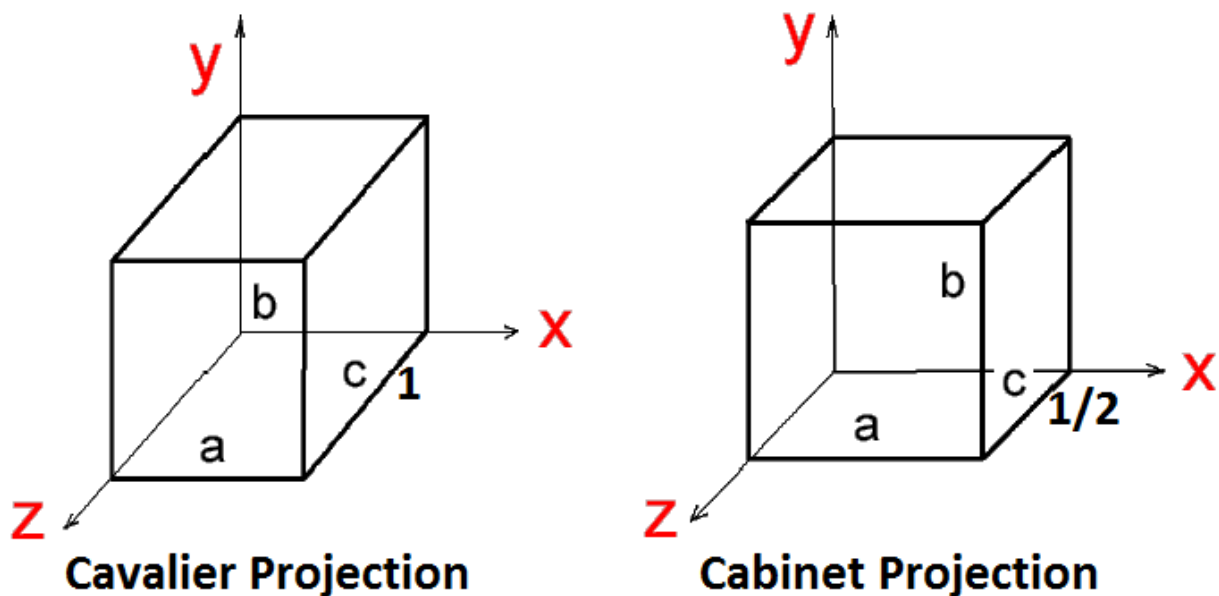
Proyeksi Miring

Dalam proyeksi ortografis, arah proyeksi tidak normal terhadap proyeksi bidang. Dalam proyeksi miring, kita dapat melihat objek lebih baik daripada proyeksi ortografis.

Ada dua jenis proyeksi miring: **Angkuh** dan **Kabinet**. Proyeksi Cavalier membuat sudut 45° dengan bidang proyeksi.

Proyeksi garis yang tegak lurus terhadap bidang pandangan memiliki panjang yang sama dengan garis itu sendiri dalam proyeksi Cavalier. Dalam proyeksi angkuh, faktor foreshortening untuk ketiga arah utama adalah sama.

Proyeksi kabinet menghasilkan sudut $63,4^\circ$ dengan bidang proyeksi. Dalam proyeksi kabinet, garis-garis yang tegak lurus terhadap permukaan tampilan diproyeksikan pada $\frac{1}{2}$ panjang sebenarnya. Kedua proyeksi ditunjukkan pada gambar berikut:

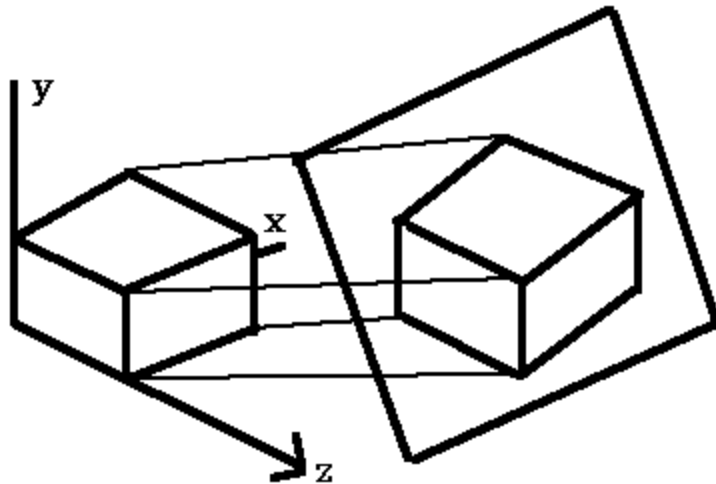


Gambar: Proyeksi Cavalier & Kabinet

Proyeksi Isometrik

Proyeksi ortografis yang menunjukkan lebih dari satu sisi objek disebut

proyeksi ortografi aksonometrik. Proyeksi aksonometrik yang paling umum adalah **proyeksi isometrik** di mana bidang proyeksi memotong setiap sumbu koordinat dalam sistem koordinat model pada jarak yang sama. Dalam proyeksi ini paralelisme garis dipertahankan tetapi sudut tidak dipertahankan. Gambar berikut menunjukkan proyeksi isometrik:



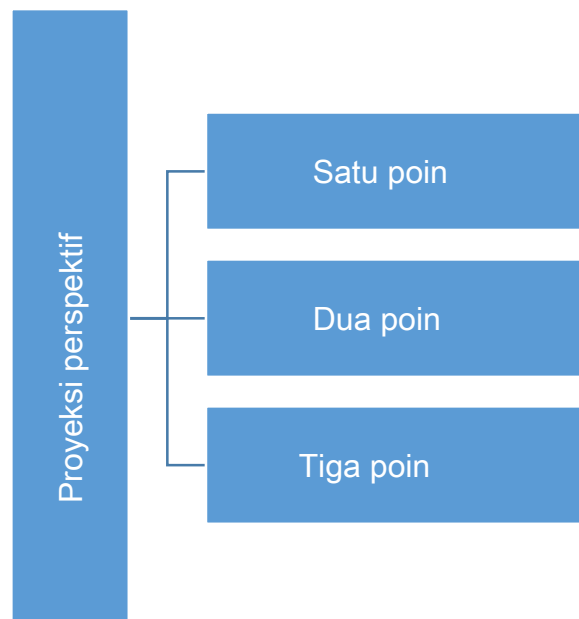
Gambar: Proyeksi Isometrik

Proyeksi Perspektif

Dalam proyeksi perspektif, jarak dari pusat proyeksi ke bidang proyek terbatas dan ukuran objek bervariasi berbanding terbalik dengan jarak yang terlihat lebih realistis.

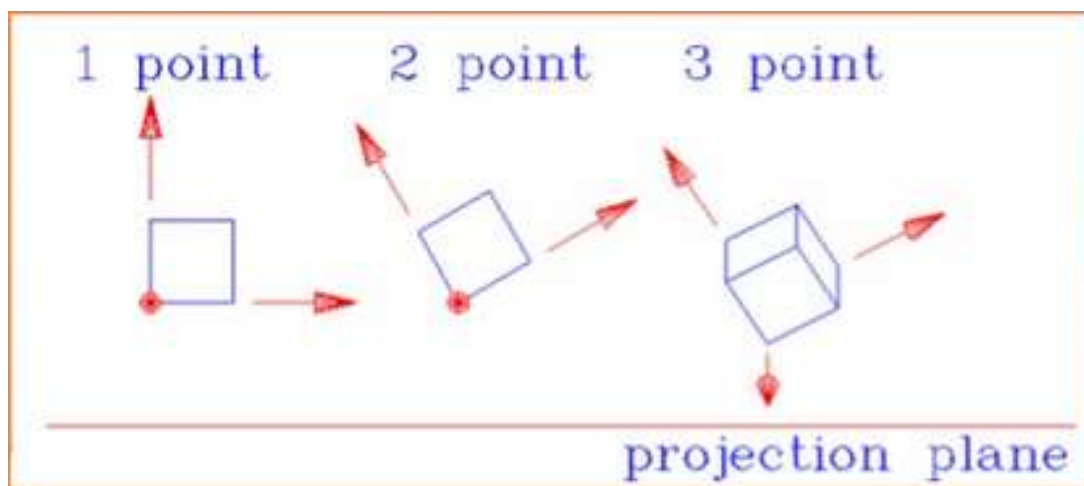
Jarak dan sudut tidak dipertahankan dan garis paralel tidak tetap paralel. Sebaliknya, mereka semua bertemu pada satu titik yang disebut **pusat proyeksi** atau **titik referensi proyeksi**. Ada 3 jenis proyeksi perspektif yang ditunjukkan pada bagan berikut.

- **Satu poin** proyeksi perspektif mudah untuk menggambar.
- **Dua poin** proyeksi perspektif memberi kesan kedalaman yang lebih baik.
- **Tiga poin** proyeksi perspektif paling sulit untuk digambarkan.



Gambar: Jenis Proyeksi Perspektif

Gambar berikut menunjukkan ketiga jenis proyeksi perspektif:

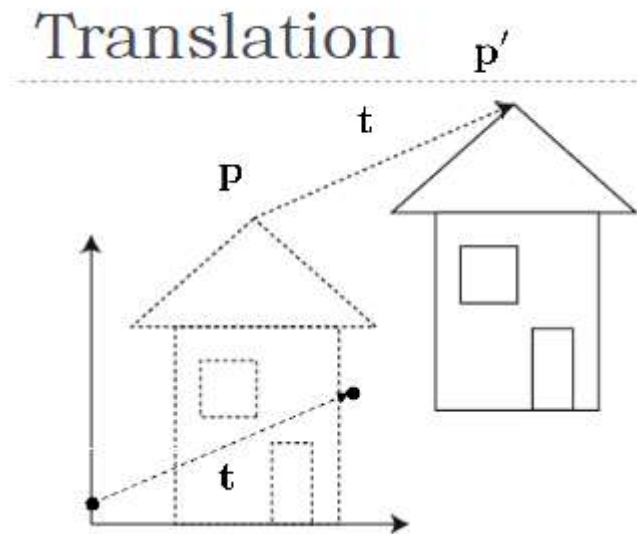


Gambar: Proyeksi perspektif 1 poin, 2 poin, 3 poin

Terjemahan

Dalam terjemahan 3D, kami mentransfer koordinat Z bersama dengan koordinat X dan Y. Proses terjemahan dalam 3D mirip dengan terjemahan 2D. Terjemahan memindahkan objek ke posisi berbeda di layar.

Gambar berikut menunjukkan efek terjemahan:



Gambar: Terjemahan 3D

Suatu titik dapat diterjemahkan dalam 3D dengan menambahkan koordinat terjemahan (t_x, t_y, t_z) ke koordinat awal (X, Y, Z) untuk mendapatkan koordinat baru (X', Y', Z').

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

$$P' = P \cdot T$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

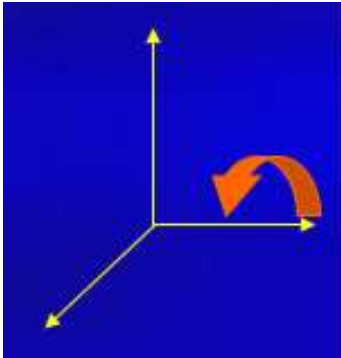
$$= \begin{bmatrix} X + t_x & Y + t_y & Z + t_z & 1 \end{bmatrix}$$

Rotasi

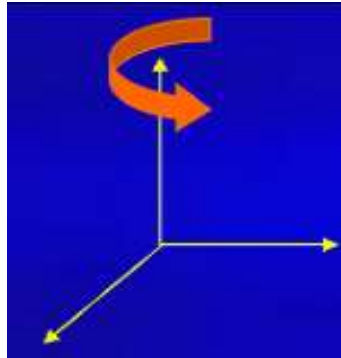
Rotasi 3D tidak sama dengan rotasi 2D. Dalam rotasi 3D, kita harus menentukan sudut rotasi bersama dengan sumbu rotasi. Kita dapat melakukan rotasi 3D tentang sumbu X, Y, dan Z. Mereka diwakili dalam bentuk matriks seperti di bawah ini:

$$\begin{aligned}
 \cdot(\cdot) = & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \cdot(\cdot) = & \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \cdot(\cdot) = & \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

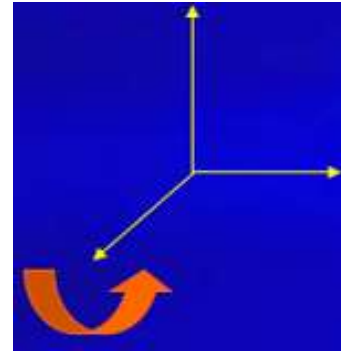
Gambar berikut menjelaskan rotasi tentang berbagai sumbu:



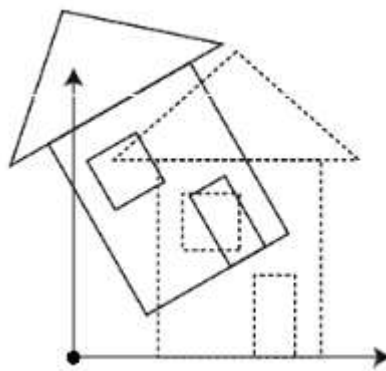
Rotasi tentang sumbu x



Rotasi tentang sumbu-y



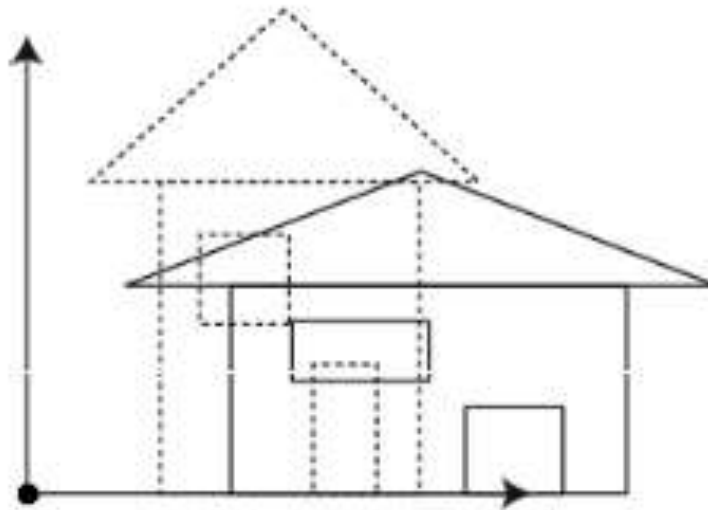
Rotasi tentang sumbu z



Gambar: Rotasi 3D

Scaling

Anda dapat mengubah ukuran objek menggunakan transformasi penskalaan. Dalam proses penskalaan, Anda memperluas atau memampatkan dimensi objek. Penskalaan dapat dicapai dengan mengalikan koordinat asli objek dengan faktor penskalaan untuk mendapatkan hasil yang diinginkan. Gambar berikut menunjukkan efek penskalaan 3D:



Gambar: Penskalaan 3D

Dalam operasi penskalaan 3D, tiga koordinat digunakan. Mari kita asumsikan bahwa koordinat aslinya adalah (X, Y, Z) , faktor penskalaan adalah (S_x, S_y, S_z) masing-masing, dan koordinat yang dihasilkan adalah (X', Y', Z') . Ini dapat direpresentasikan secara matematis seperti yang ditunjukkan di bawah ini:

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = P \cdot S$$

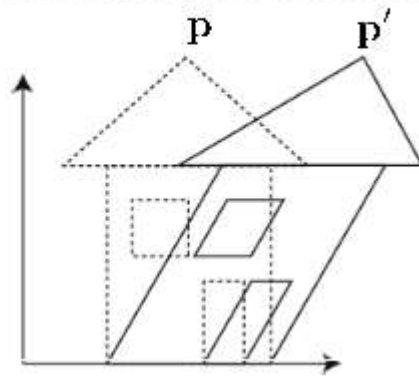
$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} X \cdot S_x & Y \cdot S_y & Z \cdot S_z & 1 \end{bmatrix}$$

Mencukur

Transformasi yang miring bentuk objek disebut **transformasi geser**. Seperti pada geser 2D, kita dapat menggeser objek di sepanjang sumbu X, sumbu Y, atau sumbu Z dalam 3D.

Shear



Seperti yang ditunjukkan pada gambar di atas, ada koordinat P. Anda dapat geser untuk mendapatkan koordinat baru P' , yang dapat direpresentasikan dalam bentuk matriks 3D seperti di bawah ini:

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

$$P' = P \cdot S_h$$

$$X' = X + 0 \cdot Y + 0 \cdot Z$$

$$Y' = 0 \cdot X + Y + 0 \cdot Z$$

$$Z' = 0 \cdot X + 0 \cdot Y + Z$$

Matriks Transformasi

Matriks transformasi adalah alat dasar untuk transformasi. Matriks dengan dimensi $n \times m$ dikalikan dengan koordinat objek. Biasanya matriks 3×3 atau 4×4 digunakan untuk transformasi. Sebagai contoh, perhatikan matriks berikut untuk berbagai operasi.

$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$S = \begin{bmatrix} s_x & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_z & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$S_h = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Matriks Terjemahan	Scaling Matrix	Matriks Geser

$\bullet \cdot (\bullet) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\bullet \cdot (\bullet) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\bullet \cdot (\bullet) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ 0 & \cos \theta & 0 & 0 \\ 0 & \sin \theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Matriks rotasi		

Dalam grafik komputer, kita sering perlu menggambar berbagai jenis objek ke layar. Objek tidak rata sepanjang waktu dan kita perlu menggambar kurva berkali-kali untuk menggambar objek.

Jenis-jenis Kurva

Kurva adalah seperangkat poin yang sangat besar. Setiap titik memiliki dua tetangga kecuali titik akhir. Kurva dapat secara luas diklasifikasikan menjadi tiga kategori: **eksplisit**, **implisit**, dan **kurva parametrik**.

Kurva Tersirat

Representasi kurva implisit mendefinisikan himpunan titik pada kurva dengan menggunakan prosedur yang dapat menguji untuk melihat apakah suatu titik masuk pada kurva. Biasanya, kurva implisit didefinisikan oleh fungsi implisit dari bentuk:

$$f(x, y) = 0$$

Itu bisa mewakili kurva multivalai (beberapa nilai y untuk nilai x). Contoh umum adalah lingkaran, yang representasi implisitnya adalah

$$x^2 + y^2 - R^2 = 0$$

Kurva Eksplisit

Fungsi matematika $y = f(x)$ dapat diplot sebagai kurva. Fungsi seperti itu adalah representasi eksplisit dari kurva. Representasi eksplisit tidak umum, karena tidak dapat mewakili garis vertikal dan juga bernilai tunggal. Untuk setiap nilai x, hanya nilai tunggal y yang biasanya dihitung oleh fungsi.

Kurva Parametrik

Kurva yang memiliki bentuk parametrik disebut kurva parametrik. Representasi kurva eksplisit dan implisit dapat digunakan hanya ketika fungsi diketahui. Dalam praktiknya kurva parametrik digunakan. Kurva parametrik dua dimensi memiliki bentuk berikut:

$$P(t) = f(t), g(t) \text{ atau } P(t) = x(t), y(t)$$

Fungsi f dan g menjadi koordinat (x, y) dari titik mana pun pada kurva, dan titik tersebut diperoleh ketika parameter t divariasikan pada interval tertentu [a, b], biasanya [0, 1].

Kurva Bezier

Kurva Bezier ditemukan oleh insinyur Perancis **Pierre Bézier**. Kurva ini dapat dihasilkan di bawah kendali titik lain. Perkiraan garis singgung dengan menggunakan titik kontrol digunakan untuk menghasilkan kurva. Kurva Bezier dapat direpresentasikan secara matematis sebagai:

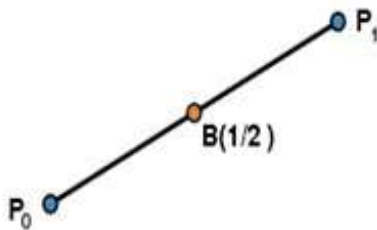
$$\sum_{i=0}^n \binom{n}{i} t^i (1-t)^{n-i} P_i$$

Dimana P_i adalah himpunan poin dan $\binom{n}{i}$ mewakili polinomial Bernstein yang diberikan oleh:

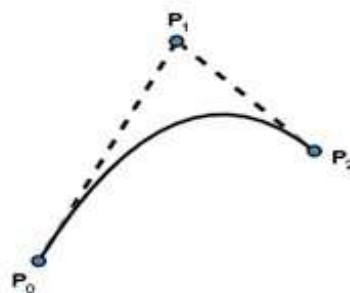
$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

Dimana n adalah derajat polinomial, i adalah indeks, dan t adalah variabelnya.

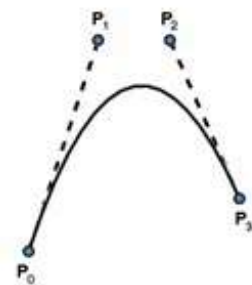
Kurva Bézier yang paling sederhana adalah garis lurus dari titik P_0 ke P_1 . Kurva Bezier kuadrat ditentukan oleh tiga titik kontrol. Kurva Bezier kubik ditentukan oleh empat titik kontrol.



Simple Bezier Curve



Quadratic Bezier Curve



Cubic Bezier Curve

Properti dari Bezier Curves

Kurva Bezier memiliki properti berikut:

- Mereka umumnya mengikuti bentuk poligon kontrol, yang terdiri dari segmen yang bergabung dengan titik kontrol.
- Mereka selalu melewati titik kontrol pertama dan terakhir.
- Mereka terkandung dalam cembung poin kontrol menentukan mereka.
- Derajat polinom menentukan segmen kurva adalah satu kurang dari jumlah titik poligon yang menentukan. Oleh karena itu, untuk 4 titik kontrol, derajat polinomial adalah 3, yaitu polinomial kubik.

- Kurva Bezier umumnya mengikuti bentuk poligon yang menentukan.
- Arah vektor tangen pada titik akhir sama dengan vektor yang ditentukan oleh segmen pertama dan terakhir.
- Properti lambung cembung untuk kurva Bezier memastikan bahwa polinomial dengan lancar mengikuti titik kontrol.
- Tidak ada garis lurus memotong kurva Bezier lebih dari itu memotong poligon kontrolnya.
- Mereka tidak berubah di bawah transformasi affine.
- Kurva Bezier menunjukkan kontrol global berarti memindahkan titik kontrol mengubah bentuk seluruh kurva.
- Kurva Bezier yang diberikan dapat dibagi lagi pada titik $t = t_0$ menjadi dua segmen Bezier yang bergabung bersama pada titik yang sesuai dengan nilai parameter $t = t_0$.

Kurva B-Spline

Kurva Bezier yang dihasilkan oleh fungsi basis Bernstein memiliki fleksibilitas terbatas.

- Pertama, jumlah simpul poligon yang ditentukan memperbaiki urutan polinomial yang dihasilkan yang menentukan kurva.
- Karakteristik pembatas kedua adalah bahwa nilai fungsi blending adalah nol untuk semua nilai parameter di seluruh kurva.

B-spline basis mengandung basis Bernstein sebagai kasus khusus. Dasar B-spline adalah non-global.

Kurva B-spline didefinisikan sebagai kombinasi linier dari titik kontrol P_i dan fungsi basis B-spline $N_{i,k}(t)$ diberikan oleh

$$C(t) = \sum_{i=0}^n P_i \cdot N_{i,k}(t), \quad n \geq k-1, \quad t \in [t_{k-1}, t_{n+1}]$$

Dimana,

- $\{P_i : i = 0, 1, 2, \dots, n\}$ adalah titik kontrol
- k adalah urutan segmen polinomial dari kurva B-spline. Memesan k berarti bahwa kurva terdiri dari segmen derajat polinomial piecewise $k-1$,
- $N_{i,k}(t)$ adalah "fungsi pencampuran B-spline yang dinormalisasi". Mereka dijelaskan oleh perintah k dan dengan urutan bilangan real yang tidak berkurang biasanya disebut "urutan simpul".

$$\{t_{i+k} : i = 0, \dots, n+k\}$$

Fungsi N_i, k dijelaskan sebagai berikut:

$$N_{i,k}(t) = \begin{cases} 1, & \text{if } t \in [t_i, t_{i+1}) \\ 0, & \text{Jika tidak} \end{cases}$$

dan jika $k > 1$,

$$N_{i,k}(t) = \frac{N_{i,k-1}(t)}{t - t_{i-1}} + \frac{N_{i+1,k-1}(t)}{t_{i+1} - t}$$

dan

$$t \in [t_{k-1}, t_{n+1})$$

Properti dari B-spline Curve

Kurva B-spline memiliki properti berikut:

- Jumlah fungsi basis B-spline untuk nilai parameter apa pun adalah 1.
- Setiap fungsi basis bernilai positif atau nol untuk semua nilai parameter.
- Setiap fungsi basis memiliki tepat satu nilai maksimum, kecuali untuk $k = 1$.
- Urutan maksimum kurva adalah sama dengan jumlah simpul dari mendefinisikan poligon.
- Tingkat B-spline polinomial tidak tergantung pada jumlah simpul dari mendefinisikan poligon.
- B-spline memungkinkan kontrol lokal atas permukaan kurva karena setiap vertex mempengaruhi bentuk kurva hanya pada rentang nilai parameter di mana fungsi basis terkait adalah nol.
- Kurva menunjukkan sifat berkurang variasi.
- Kurva umumnya mengikuti bentuk mendefinisikan poligon.
- Setiap transformasi affine dapat diterapkan pada kurva dengan menerapkannya pada simpul dari mendefinisikan poligon.
- Garis kurva di dalam cembung cembung dari poligon penentu.

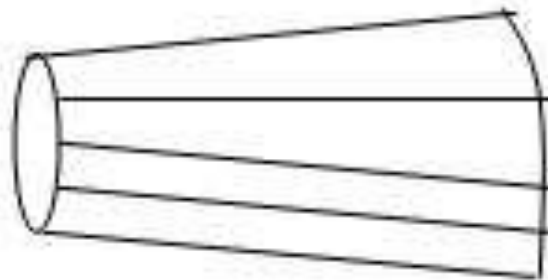
Permukaan Polygon

Objek direpresentasikan sebagai kumpulan permukaan. Representasi objek 3D dibagi menjadi dua kategori.

- **Representasi Batas (B-perwakilan):** Ini menggambarkan objek 3D sebagai satu set permukaan yang memisahkan interior objek dari lingkungan.
- **Representasi ruang-partisi:** Itu digunakan untuk menggambarkan interior properti, dengan mempartisi wilayah spasial yang mengandung objek ke dalam kumpulan kecil, non-tumpang tindih, padatan yang berdekatan (biasanya kubus).

Representasi batas yang paling umum digunakan untuk objek grafis 3D adalah seperangkat poligon permukaan yang melampirkan interior objek. Banyak sistem grafis menggunakan metode ini. Set poligon disimpan untuk deskripsi objek. Ini menyederhanakan dan mempercepat rendering permukaan dan tampilan objek karena semua permukaan dapat dijelaskan dengan persamaan linear.

Permukaan poligon umum digunakan dalam desain dan aplikasi pemodelan padat **bingkai foto tampilan** dapat dilakukan dengan cepat untuk memberikan indikasi umum struktur permukaan. Kemudian adegan realistis dihasilkan dengan menyisipkan pola bayangan melintasi permukaan poligon untuk menerangi.

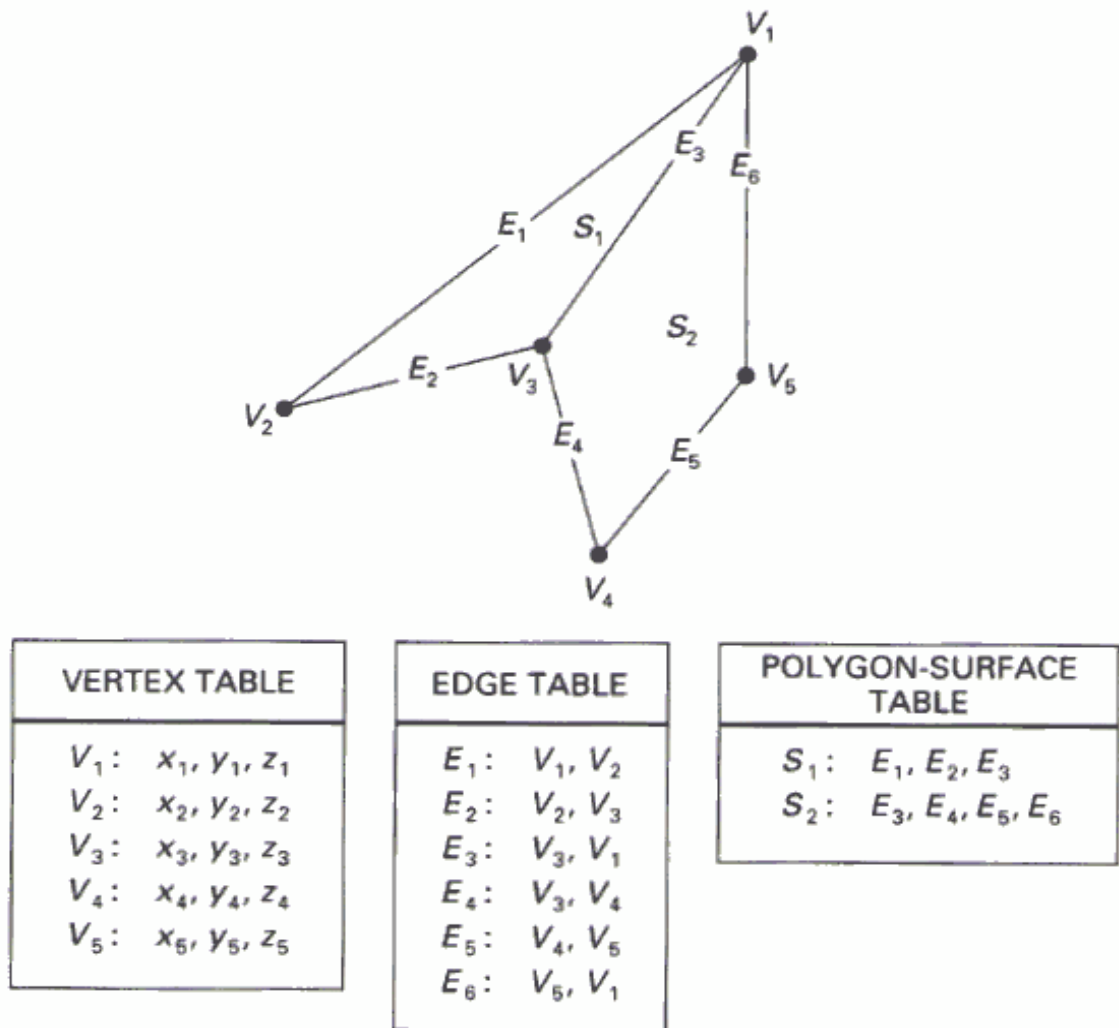


A 3D object represented by polygons

Tabel Polygon

Dalam metode ini, permukaan ditentukan oleh set koordinat titik dan atribut yang terkait. Seperti yang ditunjukkan pada gambar berikut, ada lima simpul, dari v1 ke v5.

- Setiap vertex menyimpan informasi koordinat x , y , dan z yang direpresentasikan dalam tabel sebagai $v_i: x_i, y_i, z_i$.
- Tabel Edge digunakan untuk menyimpan informasi tepi poligon. Pada gambar berikut, tepi E_1 terletak di antara simpul v_1 dan v_2 yang direpresentasikan dalam tabel sebagai $E_i: v_1, v_2$.
- Tabel permukaan poligon menyimpan jumlah permukaan yang ada dalam poligon. Dari gambar berikut, permukaan S_1 ditutupi oleh tepi E_1 , E_2 dan E_3 yang dapat direpresentasikan dalam tabel permukaan poligon sebagai $S_i: E_1, E_2, \text{ dan } E_3$.



Gambar: Tabel Polygon

Persamaan Pesawat

Persamaan untuk permukaan bidang dapat dinyatakan sebagai:

$$Ax + By + Cz + D = 0$$

Di mana (x, y, z) adalah titik di pesawat, dan koefisien A, B, C , dan D adalah konstanta yang menggambarkan sifat spasial pesawat. Kita bisa mendapatkan nilai-nilai A, B, C , dan D dengan menyelesaikan satu set tiga persamaan bidang menggunakan nilai koordinat untuk tiga titik non

collinear pada bidang. Mari kita asumsikan bahwa tiga simpul pesawat adalah (x_1, y_1, z_1) , (x_2, y_2, z_2) dan (x_3, y_3, z_3) .

Mari kita memecahkan persamaan simultan berikut untuk rasio $A/D, B/D$, dan C/D . Anda mendapatkan nilai A, B, C , dan D .

$$(A/D)x_1 + (B/D)y_1 + (C/D)z_1 = -1$$

$$(A/D)x_2 + (B/D)y_2 + (C/D)z_2 = -1$$

$$(A/D)x_3 + (B/D)y_3 + (C/D)z_3 = -1$$

Untuk mendapatkan persamaan di atas dalam bentuk determinan, terapkan aturan Cramer ke persamaan di atas.

$$A/D = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B/D = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C/D = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

Untuk setiap titik (x, y, z) dengan parameter A, B, C , dan D , kita dapat mengatakan bahwa -

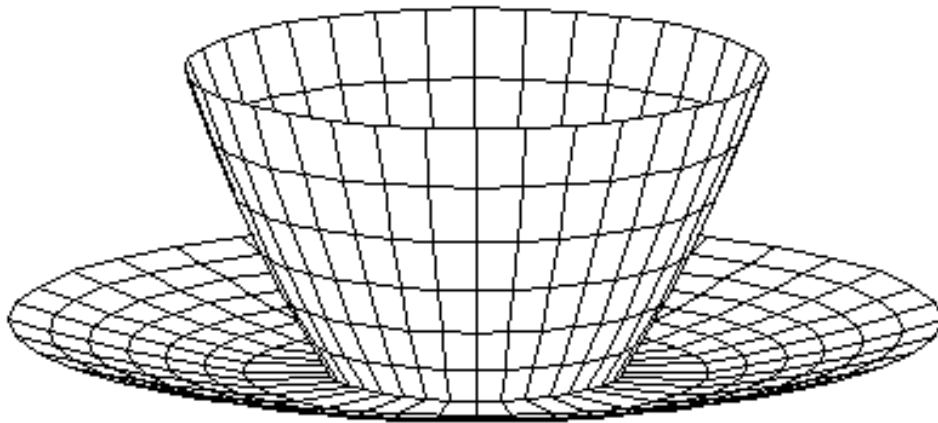
- $Ax + By + Cz + D \neq 0$ berarti titik tidak pada bidang.
- $Ax + By + Cz + D < 0$ berarti titik berada di dalam permukaan.
- $Ax + By + Cz + D > 0$ berarti titik berada di luar permukaan.

Jerat Poligon

Permukaan dan padatan 3D dapat diperkirakan dengan seperangkat elemen garis poligonal dan garis. Permukaan seperti itu disebut **jerat poligonal**. Dalam poligon mesh, setiap sisi dibagi oleh paling banyak dua poligon. Himpunan poligon atau wajah, bersama-sama membentuk "kulit" objek.

Metode ini dapat digunakan untuk mewakili kelas padat / permukaan dalam grafis. Mesh poligonal dapat dirender menggunakan algoritma pelepasan permukaan tersembunyi. Mesh poligon dapat diwakili oleh tiga cara:

- Representasi eksplisit
- Pointer ke daftar titik
- Pointer ke daftar tepi



Gambar: Polygon Mesh

Keuntungan

- Dapat digunakan untuk memodelkan hampir semua objek.
- Mereka mudah direpresentasikan sebagai kumpulan simpul.
- Mereka mudah diubah.
- Mereka mudah menggambar di layar komputer.

Kekurangan

- Permukaan melengkung hanya bisa dijelaskan.
- Sulit untuk mensimulasikan beberapa jenis benda seperti rambut atau cairan.

10. DETEKSI PERMUKAAN YANG TAMPAK

Ketika kita melihat gambar yang mengandung objek dan permukaan yang tidak transparan, maka kita tidak dapat melihat objek-objek itu dari pandangan yang berada di belakang dari objek yang lebih dekat ke mata. Kita harus menghapus permukaan tersembunyi ini untuk mendapatkan gambar layar yang realistis. Identifikasi dan penghapusan permukaan ini disebut **Masalah permukaan tersembunyi**.

Ada dua pendekatan untuk menghilangkan masalah permukaan tersembunyi: **Metode Object-Space** dan **Metode ruang-gambar**. Metode Object-space diimplementasikan dalam sistem koordinat fisik dan metode image-space diimplementasikan dalam sistem koordinat layar.

Ketika kita ingin menampilkan objek 3D pada layar 2D, kita perlu mengidentifikasi bagian-bagian layar yang terlihat dari posisi tampilan yang dipilih.

Metode Penyangga Kedalaman (Penyangga Z)

Metode ini dikembangkan oleh Cutmull. Ini adalah pendekatan ruang-gambar. Ide dasarnya adalah untuk menguji kedalaman Z dari setiap permukaan untuk menentukan permukaan terdekat (terlihat).

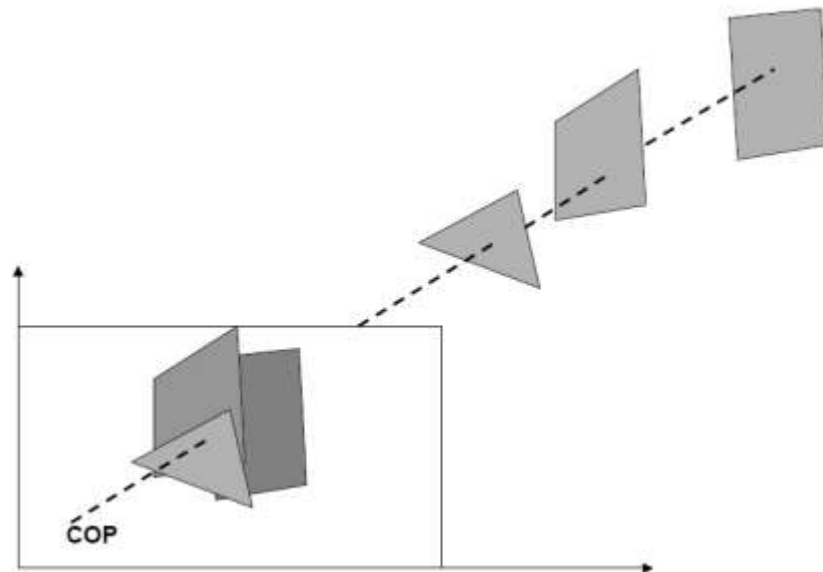
Dalam metode ini setiap permukaan diproses secara terpisah satu posisi piksel pada satu waktu melintasi permukaan. Nilai kedalaman untuk piksel dibandingkan dan permukaan terdekat (terkecil z) menentukan warna yang akan ditampilkan dalam buffer bingkai.

Ini diterapkan dengan sangat efisien pada permukaan poligon. Permukaan dapat diproses dalam urutan apa pun. Untuk mengganti poligon yang lebih dekat dari yang jauh, dua buffer dinamai **bingkai buffer** dan **buffer kedalaman**, digunakan.

Buffer kedalaman digunakan untuk menyimpan nilai kedalaman untuk posisi (x, y), karena permukaan diproses ($0 \leq \text{kedalaman} \leq 1$).

Itu **bingkai buffer** digunakan untuk menyimpan nilai intensitas nilai warna di setiap posisi (x, y).

Koordinat z biasanya dinormalisasi ke kisaran [0, 1]. Nilai 0 untuk zcoordinate menunjukkan panel kliping belakang dan 1 nilai untuk koordinat z menunjukkan panel kliping depan.



Algoritma

Langkah 1: Tetapkan nilai buffer:

Depthbuffer (x, y) = 0

Framebuffer (x, y) = warna latar belakang

Langkah 2: Memproses setiap poligon (Satu per satu)

Untuk setiap posisi piksel yang diproyeksikan (x, y) dari poligon, hitung kedalaman z.

Jika $Z > \text{depthbuffer}(x, y)$

Hitung warna permukaan,

atur $\text{depthbuffer}(x, y) = z$,

$\text{framebuffer}(x, y) = \text{surfacecolor}(x, y)$

Keuntungan

- Mudah diimplementasikan.
- Ini mengurangi masalah kecepatan jika diterapkan pada perangkat keras.
- Ini memproses satu objek pada satu waktu.

Kekurangan

- Itu membutuhkan memori yang besar.
- Ini adalah proses yang memakan waktu.

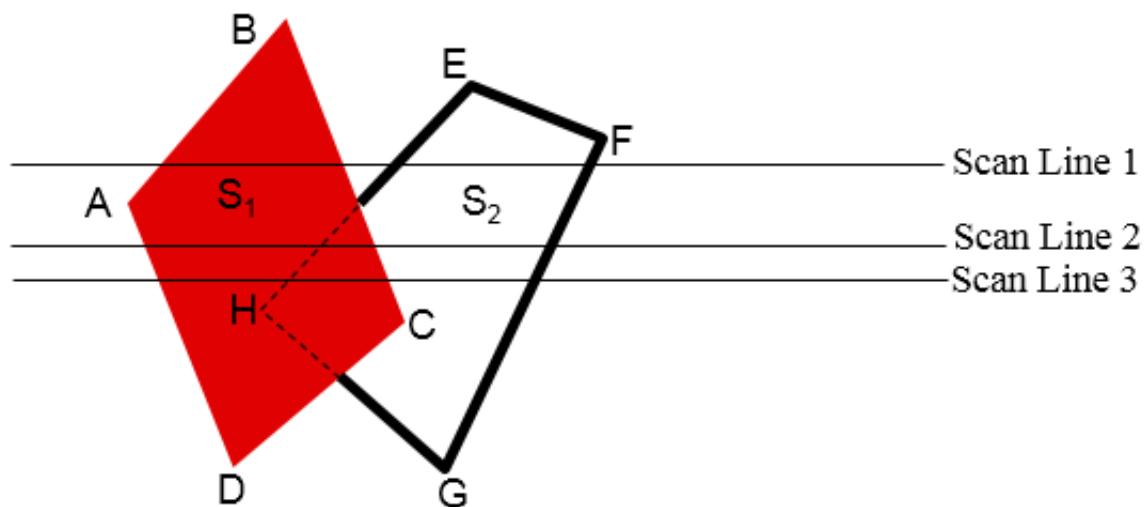
Metode Scan-Line

Ini adalah metode ruang-gambar untuk mengidentifikasi permukaan yang terlihat. Metode ini memiliki informasi mendalam untuk hanya satu garis pemindaian. Untuk memerlukan satu garis pindai nilai kedalaman, kita harus mengelompokkan dan memproses semua poligon yang memotong garis pindai yang diberikan pada saat yang sama sebelum memproses garis pindai berikutnya. Dua tabel penting, **meja tepi** dan

meja poligon, dipertahankan untuk ini.

The Edge Table: Ini berisi titik akhir koordinat setiap garis dalam adegan, kemiringan terbalik dari setiap garis, dan penunjuk ke dalam tabel poligon untuk menghubungkan tepi ke permukaan.

Meja Polygon: Ini berisi koefisien bidang, sifat material permukaan, data permukaan lainnya, dan mungkin pointer ke tabel tepi.



Untuk memudahkan pencarian permukaan yang melintasi garis pindaian yang diberikan, daftar tepi aktif dibentuk. Daftar aktif hanya menyimpan tepi-tepi yang melewati garis pindai dalam rangka meningkatkan x . Bendera juga ditetapkan untuk setiap permukaan untuk mengindikasikan apakah posisi sepanjang garis pindaian berada di dalam atau di luar permukaan.

Posisi piksel di setiap garis pindai diproses dari kiri ke kanan. Di persimpangan kiri dengan permukaan, bendera permukaan dihidupkan dan di sebelah kanan, bendera dimatikan. Anda hanya perlu melakukan perhitungan kedalaman ketika beberapa permukaan bendera dinyalakan pada posisi garis pindai tertentu.

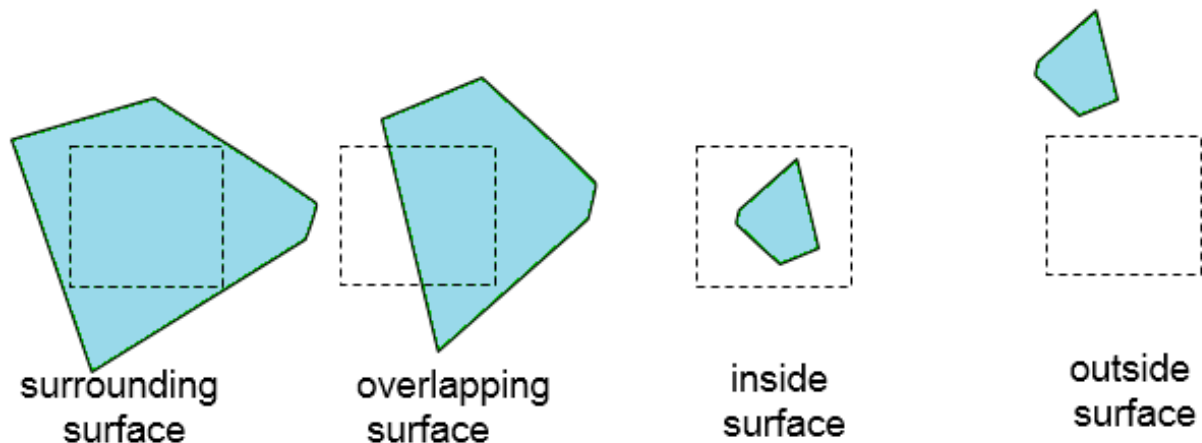
Metode Pembagian Area

Metode pembagian area mengambil keuntungan dengan menempatkan area tampilan yang mewakili bagian dari satu permukaan. Membagi area tampilan total menjadi persegi panjang yang lebih kecil dan lebih kecil sampai setiap area kecil adalah proyeksi bagian dari permukaan yang terlihat atau tidak ada permukaan sama sekali.

Lanjutkan proses ini sampai subdivisi mudah dianalisis sebagai milik permukaan tunggal atau sampai mereka dikurangi menjadi ukuran satu piksel. Cara mudah untuk melakukan ini adalah

untuk membagi area menjadi empat bagian yang sama pada setiap langkah. Ada empat kemungkinan hubungan yang dapat dimiliki suatu permukaan dengan batas area tertentu.

- **Permukaan sekitar:** Satu yang benar-benar menutupi area.
- **Permukaan yang tumpang tindih:** Satu yang sebagian di dalam dan sebagian di luar daerah.
- **Permukaan dalam:** Satu yang benar-benar di dalam area.
- **Permukaan luar:** Yang benar-benar di luar area.



Pengujian untuk menentukan visibilitas permukaan dalam suatu area dapat dinyatakan dalam empat klasifikasi ini. Tidak diperlukan subdivisi lebih lanjut dari area tertentu jika salah satu dari kondisi berikut ini benar:

- Semua permukaan adalah permukaan luar sehubungan dengan area tersebut.
- Hanya satu di dalam, tumpang tindih atau permukaan yang ada di area tersebut.
- Permukaan sekitarnya mengaburkan semua permukaan lain di dalam batas area.

Deteksi Wajah-Belakang

Metode objek-ruang cepat dan sederhana untuk mengidentifikasi wajah belakang polyhedron didasarkan pada tes "dalam-luar". Suatu titik (x, y, z) adalah "di dalam" permukaan poligon dengan parameter bidang A, B, C, dan D jika Ketika titik dalam berada di sepanjang garis pandang ke permukaan, poligon tersebut harus berupa permukaan belakang (kita berada di dalam wajah itu dan tidak dapat melihat bagian depannya dari posisi melihat kita).

Kita dapat menyederhanakan tes ini dengan mempertimbangkan vektor normal N ke permukaan poligon, yang memiliki komponen Cartesian (A, B, C).

Secara umum, jika V adalah vektor dalam arah pandang dari posisi mata (atau "kamera"), maka poligon ini adalah wajah belakang jika

$$VN > 0$$

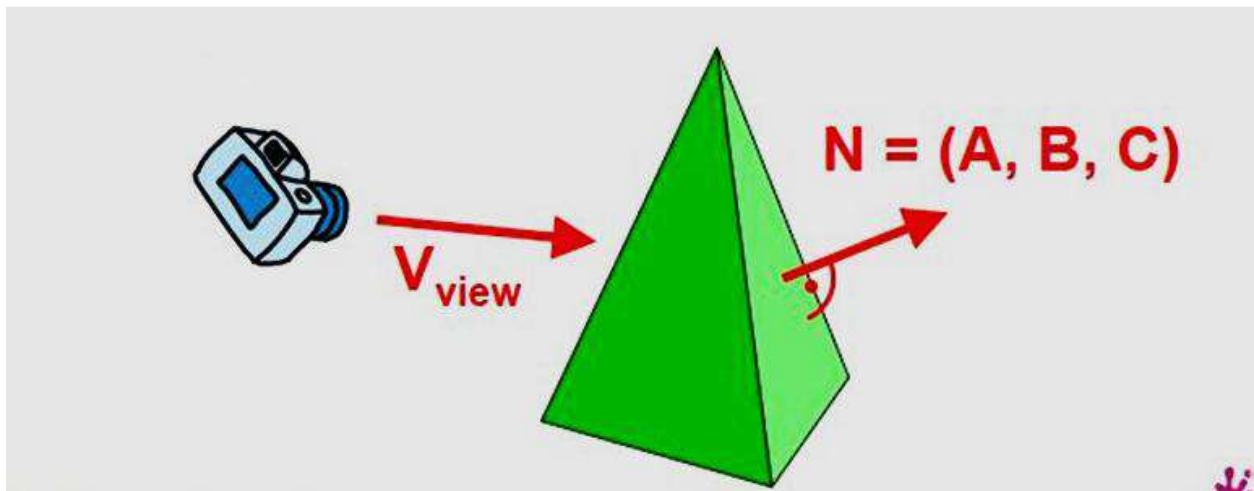
Selanjutnya, jika deskripsi objek dikonversi ke koordinat proyeksi dan arah tampilan Anda sejajar dengan sumbu z tampilan, maka:

$$V = (0, 0, V_z) \quad \text{dan} \quad VN = V_z C$$

Sehingga kita hanya perlu mempertimbangkan tanda C komponen vektor normal N.

Dalam sistem penglihatan tangan kanan dengan arah penglihatan sepanjang Z negatif sumbu, poligon adalah wajah belakang jika $C < 0$. Juga, kita tidak dapat melihat wajah yang normalnya memiliki komponen z $C = 0$, karena arah pandangan Anda mengarah ke poligon itu. Dengan demikian, secara umum, kita dapat memberi label poligon sebagai wajah belakang jika vektor normalnya memiliki nilai komponen az:

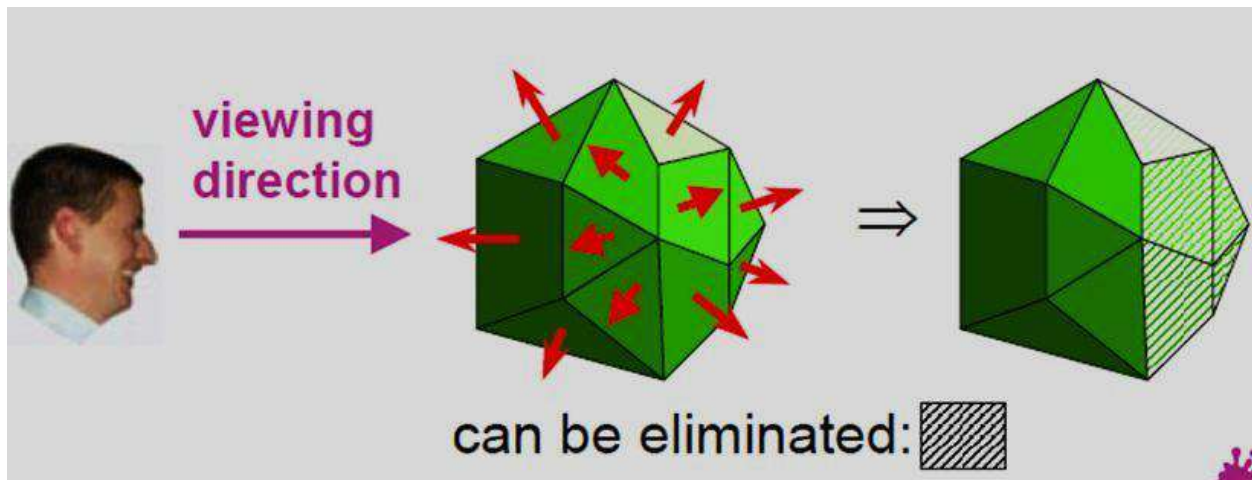
$$C \leq 0$$



Metode serupa dapat digunakan dalam paket yang menggunakan sistem tampilan kidal. Dalam paket ini, parameter bidang A, B, C dan D dapat dihitung dari koordinat vertigon poligon yang ditentukan dalam arah searah jarum jam (tidak seperti arah berlawanan arah jarum jam yang digunakan dalam sistem tangan kanan).

Juga, wajah belakang memiliki vektor normal yang mengarah menjauh dari posisi penonton dan

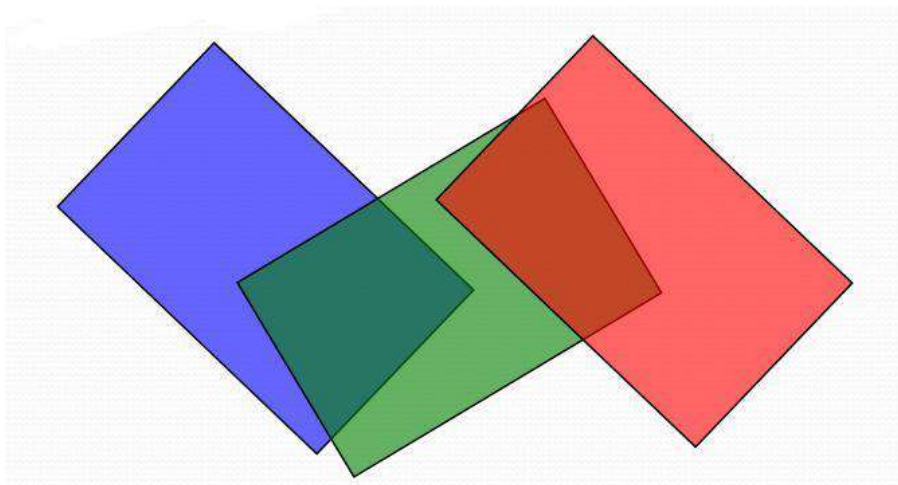
diidentifikasi oleh $C > 0$ ketika arah melihat sepanjang positif Z sumbu. Dengan memeriksa parameter C untuk bidang-bidang berbeda yang mendefinisikan objek, kita dapat segera mengidentifikasi semua permukaan belakang.



Metode Penyangga A

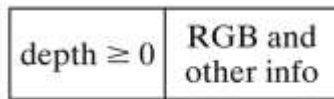
Metode A-buffer adalah perpanjangan dari metode kedalaman-buffer. Metode A-buffer adalah metode deteksi visibilitas yang dikembangkan di Lucas film Studios untuk sistem rendering Renders Everything You Ever Saw (REYES).

A-buffer memperluas metode buffer kedalaman untuk memungkinkan transparansi. Struktur data utama dalam buffer-A adalah buffer akumulasi.



Setiap posisi dalam buffer A memiliki dua bidang:

- 1) **Bidang kedalaman:** Ini menyimpan bilangan real positif atau negatif
- 2) **Bidang intensitas:** Ini menyimpan informasi intensitas permukaan atau nilai pointer



(a)



(b)

Jika kedalaman ≥ 0 , angka yang disimpan pada posisi itu adalah kedalaman permukaan tunggal yang tumpang tindih dengan area piksel yang sesuai. Bidang intensitas kemudian menyimpan komponen RGB warna permukaan pada titik itu dan persentase cakupan piksel.

Jika kedalaman < 0 , ini menunjukkan kontribusi multi-permukaan terhadap intensitas piksel. Bidang intensitas kemudian menyimpan pointer ke daftar data permukaan yang ditautkan. Buffer permukaan dalam buffer-A meliputi:

- Komponen intensitas RGB
- Parameter Keburaman
- Kedalaman
- Persen cakupan wilayah
- Pengidentifikasi permukaan

Algoritma hasil seperti algoritma buffer kedalaman. Nilai kedalaman dan opacity digunakan untuk menentukan warna akhir suatu piksel.

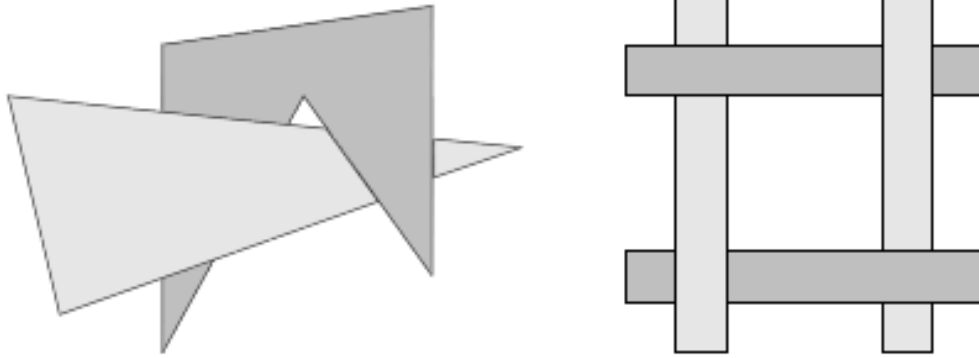
Metode Penyortiran Kedalaman

Metode pengurutan kedalaman menggunakan ruang gambar dan operasi ruang-objek. Metode kedalaman melakukan dua fungsi dasar:

- Pertama, permukaan diurutkan dalam urutan kedalaman menurun.
- Kedua, permukaan diubah-pindai secara berurutan, dimulai dengan permukaan yang paling dalam.

Konversi pemindaian permukaan poligon dilakukan di ruang gambar. Metode ini untuk memecahkan masalah permukaan tersembunyi sering disebut sebagai **algoritma pelukis**.

Gambar berikut menunjukkan efek penyortiran kedalaman:



Gambar: Penyortiran Kedalaman

Algoritma dimulai dengan mengurutkan berdasarkan kedalaman. Misalnya, perkiraan "kedalaman" awal poligon dapat dianggap sebagai nilai z terdekat dari titik mana pun dari poligon.

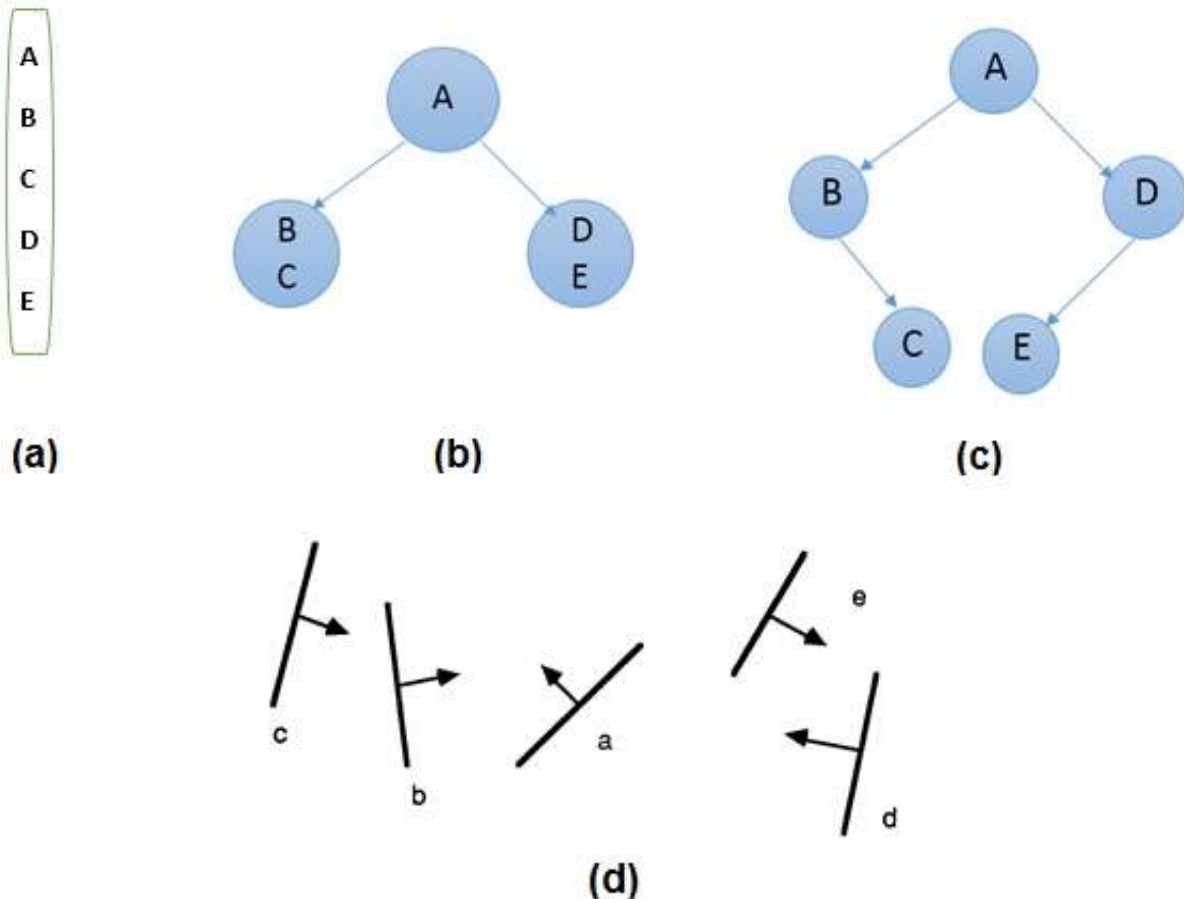
Mari kita ambil poligon P di akhir daftar. Pertimbangkan semua poligon Q yang z -luasannya tumpang tindih dengan P . Sebelum menggambar P , kami membuat tes berikut. Jika salah satu dari tes berikut ini positif, maka kita dapat mengasumsikan P dapat diambil sebelum Q .

1. Apakah x -extents tidak tumpang tindih?
2. Apakah ekstensi- y tidak tumpang tindih?
3. Apakah P sepenuhnya berada di sisi berlawanan dari bidang Q dari sudut pandang?
4. Apakah Q sepenuhnya di sisi yang sama dari bidang P sebagai sudut pandang?
5. Apakah proyeksi poligon tidak tumpang tindih?

Jika semua tes gagal, maka kami membagi P atau Q menggunakan bidang yang lain. Poligon yang dipotong baru dimasukkan ke dalam urutan kedalaman dan proses berlanjut. Secara teoritis, partisi ini dapat menghasilkan $O(n^2)$ setiap poligon, tetapi dalam praktiknya, jumlah poligon jauh lebih kecil.

Pohon Partisi Ruang Biner (BSP)

Partisi ruang biner digunakan untuk menghitung visibilitas. Untuk membangun pohon BSP, seseorang harus mulai dengan poligon dan memberi label semua sisi. Berurusan hanya dengan satu sisi pada satu waktu, panjangkan setiap sisi sehingga membagi bidang menjadi dua. Tempatkan tepi pertama di pohon sebagai root. Tambahkan tepi berikutnya berdasarkan apakah mereka di dalam atau di luar. Tepi yang merentangkan ekstensi tepi yang sudah ada di pohon dibagi menjadi dua dan keduanya ditambahkan ke pohon.



Gambar: Pohon BSP

- Dari gambar di atas, take pertama **SEBUAH** sebagai root.
- Buat daftar semua node dalam gambar (a).
- Letakkan semua node yang ada di depan root **SEBUAH** ke sisi kiri simpul **SEBUAH** dan letakkan semua simpul yang berada di belakang root **SEBUAH** ke sisi kanan seperti yang ditunjukkan pada gambar (b).
- Proses semua node depan terlebih dahulu dan kemudian node di belakang.
- Seperti yang ditunjukkan pada gambar (c), pertama-tama kita akan memproses node **B**. Karena tidak ada apa-apa di depan node **B**, kami telah menempatkan NIL. Namun, kami memiliki simpul **C** di belakang node **B**, jadi simpul **C** akan menuju ke sisi kanan simpul **B**.
- Ulangi proses yang sama untuk node **D**.

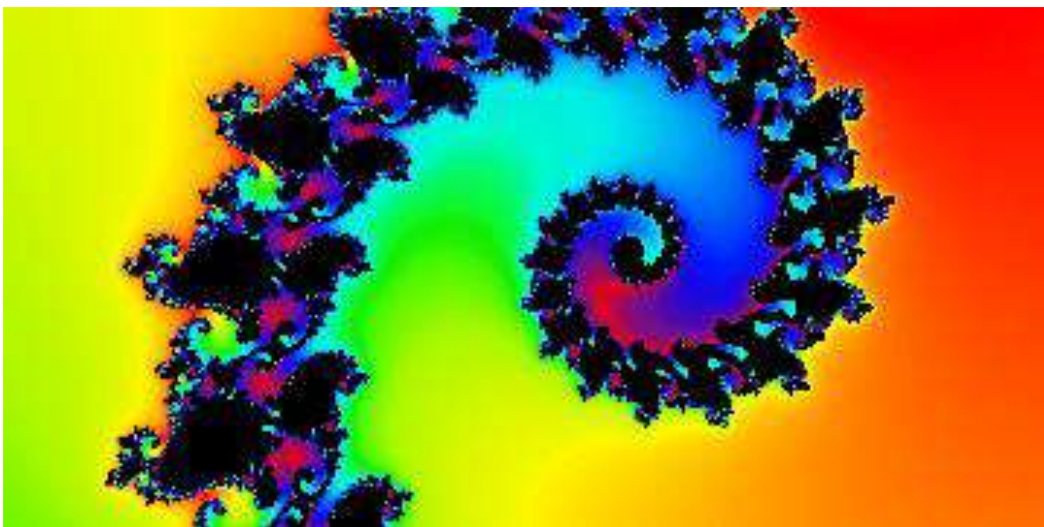
Seorang ahli matematika Prancis / Amerika Dr Benoit Mandelbrot menemukan Fraktal. Kata fraktal berasal dari kata Latin *frusus* yang berarti rusak.

Apa itu Fraktal?

Fraktal adalah gambar yang sangat kompleks yang dihasilkan oleh komputer dari satu formula. Mereka dibuat menggunakan iterasi. Ini berarti satu rumus diulangi dengan nilai yang sedikit berbeda berulang-ulang, dengan mempertimbangkan hasil dari iterasi sebelumnya.

Fraktal digunakan di banyak bidang seperti:

- **Astronomi:** Untuk menganalisis galaksi, cincin Saturnus, dll.
- **Biologi / Kimia:** Untuk menggambarkan kultur bakteri, reaksi kimia, anatomi manusia, molekul, tanaman,
- **Lainnya:** Untuk menggambarkan awan, garis pantai dan garis batas, kompresi data, difusi, ekonomi, seni fraktal, musik fraktal, lanskap, efek khusus, dll.



Gambar: Fraktal

Generasi Fraktal

Fraktal dapat dihasilkan dengan mengulangi bentuk yang sama berulang-ulang seperti yang ditunjukkan pada gambar berikut. Pada gambar (a) menunjukkan segitiga sama sisi. Pada gambar (b), kita dapat melihat bahwa segitiga diulang untuk membuat bentuk seperti bintang. Pada gambar (c), kita dapat melihat bahwa bentuk bintang pada gambar (b) berulang-ulang untuk membuat bentuk baru.

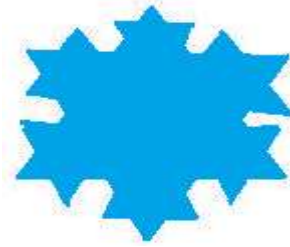
Kita dapat melakukan iterasi dalam jumlah tak terbatas untuk membuat bentuk yang diinginkan. Dalam istilah pemrograman, rekursi digunakan untuk membuat bentuk seperti itu.



(a) Generasi Zeroth



(B) Generasi Pertama

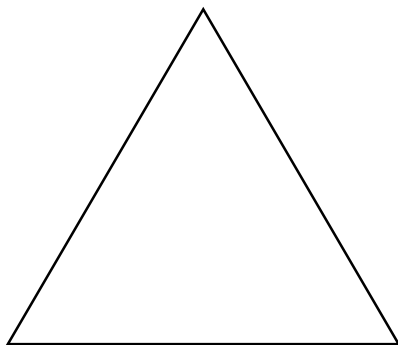


(c) Generasi Kedua

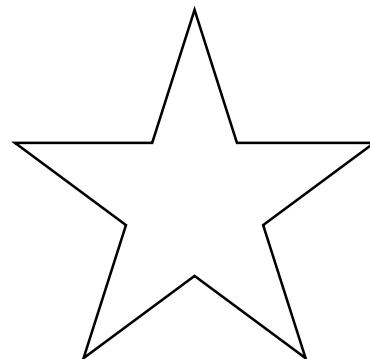
Gambar: Generasi Fraktal

Fraktal Geometris

Fraktal geometris berhubungan dengan bentuk-bentuk yang ditemukan di alam yang memiliki dimensi non-integer atau fraktal. Untuk secara geometris membangun fraktal yang mirip diri deterministik (nonrandom), kita mulai dengan bentuk geometris tertentu, yang disebut **pemrakarsa**. Bagian dari penggagas kemudian diganti dengan pola, yang disebut **generator**.



Pemrakarsa






Generator

Gambar: Inisiator dan Generator untuk fraktal

Sebagai contoh, jika kita menggunakan inisiator dan generator yang ditunjukkan pada gambar di atas, kita dapat membangun pola yang baik dengan mengulangnya. Setiap segmen garis lurus pada inisiator diganti dengan empat segmen garis dengan panjang yang sama pada setiap langkah. Faktor penskalaan adalah $1/3$, sehingga dimensi fraktal adalah $D = \ln 4 / \ln 3 \approx 1.2619$.

Juga, panjang setiap segmen garis dalam inisiator meningkat dengan faktor $4/3$ pada setiap langkah, sehingga panjang kurva fraktal cenderung tak terhingga karena lebih banyak detail ditambahkan ke kurva seperti yang ditunjukkan pada gambar berikut:

Panjang Segmen = 1	<u>Panjang Segmen = $1/3$ Panjang Segmen = $1/9$</u>	
		
Panjang = 1	Panjang = $4/3$	Panjang = $16/9$

Animasi berarti memberi kehidupan pada objek apa pun dalam grafik komputer. Ia memiliki kekuatan menyuntikkan energi dan emosi ke benda-benda yang tampaknya mati. Animasi berbasis komputer dan animasi yang dihasilkan komputer adalah dua kategori animasi komputer. Dapat disajikan melalui film atau video.

Ide dasar di balik animasi adalah untuk memutar ulang gambar yang direkam dengan kecepatan yang cukup cepat untuk menipu mata manusia agar menafsirkannya sebagai gerakan terus menerus. Animasi dapat membuat serangkaian gambar mati menjadi hidup. Animasi dapat digunakan di banyak bidang seperti hiburan, desain berbantuan komputer, visualisasi ilmiah, pelatihan, pendidikan, e-commerce, dan seni komputer.

Teknik Animasi

Animator telah menemukan dan menggunakan berbagai teknik animasi yang berbeda. Pada dasarnya ada enam teknik animasi yang akan kita bahas satu per satu di bagian ini.

Animasi Tradisional (bingkai demi bingkai)

Secara tradisional sebagian besar animasi dilakukan dengan tangan. Semua bingkai dalam animasi harus digambar dengan tangan. Karena setiap detik animasi membutuhkan 24 frame (film), jumlah upaya yang diperlukan untuk membuat bahkan film terpendek dapat luar biasa.

Keyframing

Dalam teknik ini, storyboard diletakkan dan kemudian para seniman menggambar bingkai utama dari animasi. Bingkai utama adalah yang di mana perubahan yang menonjol terjadi. Mereka adalah poin utama dari animasi. Keyframing mensyaratkan bahwa animator menentukan posisi kritis atau kunci untuk objek. Komputer kemudian secara otomatis mengisi frame yang hilang dengan menyisipkan antar posisi tersebut dengan lancar.

Prosedural

Dalam animasi prosedural, objek dianimasikan dengan prosedur - seperangkat aturan bukan dengan keyframing. Animator menetapkan aturan dan kondisi awal serta menjalankan simulasi. Aturan sering didasarkan pada aturan fisik dari dunia nyata yang diungkapkan oleh persamaan matematika.

Perilaku

Dalam animasi perilaku, karakter otonom menentukan tindakannya sendiri, setidaknya sampai batas tertentu. Ini memberi karakter kemampuan untuk berimprovisasi, dan membebaskan animator dari kebutuhan untuk menentukan setiap detail gerakan setiap karakter.

Berbasis Kinerja (Tangkapan Gerakan)

Teknik lain adalah Motion Capture, di mana sensor magnetik atau berbasis penglihatan merekam tindakan objek manusia atau hewan dalam tiga dimensi. Komputer kemudian menggunakan data ini untuk menghidupkan objek.

Teknologi ini telah memungkinkan sejumlah atlet terkenal untuk memasok aksi untuk karakter dalam permainan video olahraga. Motion capture cukup populer di kalangan animator terutama karena beberapa tindakan manusia biasa dapat ditangkap dengan relatif mudah. Namun, mungkin ada perbedaan serius antara bentuk atau dimensi subjek dan karakter grafis dan ini dapat menyebabkan masalah eksekusi yang tepat.

Berbasis Fisik (Dinamika)

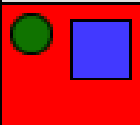
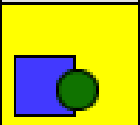
Tidak seperti pembingkai kunci dan gambar bergerak, simulasi menggunakan hukum fisika untuk menghasilkan gerak gambar dan objek lainnya. Simulasi dapat dengan mudah digunakan untuk menghasilkan urutan yang sedikit berbeda dengan tetap mempertahankan realisme fisik. Kedua, simulasi real-time memungkinkan tingkat interaktivitas yang lebih tinggi di mana orang yang sebenarnya dapat bermanuver tindakan karakter yang disimulasikan.

Sebaliknya aplikasi yang didasarkan pada pembingkai tombol dan gerak memilih dan memodifikasi gerakan membentuk pustaka gerak yang dihitung sebelumnya. Salah satu kelemahan yang diderita simulasi adalah keahlian dan waktu yang dibutuhkan untuk membuat sistem kontrol yang sesuai.

Framing Kunci

Bingkai kunci adalah bingkai tempat kami mendefinisikan perubahan dalam animasi. Setiap bingkai adalah bingkai kunci ketika kita membuat animasi bingkai demi bingkai. Ketika seseorang membuat animasi 3D di komputer, mereka biasanya tidak menentukan posisi pasti dari objek yang diberikan pada setiap frame. Mereka membuat keyframe.

Bingkai kunci adalah bingkai penting di mana objek mengubah ukuran, arah, bentuk, atau properti lainnya. Komputer kemudian mencari tahu semua frame di antara dan menghemat jumlah waktu yang ekstrim untuk animator. Ilustrasi berikut menggambarkan frame yang dibuat oleh pengguna dan frame yang dihasilkan oleh komputer.

1	2	3	4	5	6	7	8
							

Gambar: Bingkai yang digambar oleh pengguna

1	2	3	4	5	6	7	8
							

Gambar: Di antara frame yang dihasilkan oleh komputer

Perubahan

Transformasi bentuk-bentuk objek dari satu bentuk ke bentuk lain disebut morphing. Ini adalah salah satu transformasi paling rumit.



Gambar: Grafik asli



Gambar: Versi asli yang melengkung

Sebuah morf tampak seolah-olah dua gambar saling melebur dengan gerakan yang sangat lancar. Dalam istilah teknis, dua gambar terdistorsi dan terjadi pemudaran di antara keduanya.