Materi M1 -> B1

Pengenalan Grafik Komputer

1. **Pengertian dan ruang lingkup grafik komputer**
   - Grafik adalah metode umum untuk mengilustrasikan hubungan dalam data secara visual.
   - Komputer adalah mesin yang dapat melakukan operasi matematika atau operasi logika dengan cepat dan otomatis.
   - Grafik komputer adalah ilmu yang mempelajari cara untuk membuat, memanipulasi, dan menyajikan objek yang terdiri atas model (representasi matematika dari bentuk tiga dimensi) dan gambar (representasi visual dari objek), lalu divisualisasi secara digital menggunakan komputer.
   - Ruang lingkup Grafika Komputer membahas berbagai hal terkait pengolahan grafis baik menciptakan maupun memanipulasi objek grafis dua dimensi melalui pemrograman visual.

   Beberapa aspek penting di gk: => 2D/3D R A UI V

   a. Pembuatan Gambar 2D dan 3D:
      - 2D: Melibatkan pembuatan gambar yang memiliki dua dimensi, yaitu panjang dan lebar. Contoh aplikasi termasuk pembuatan logo, ikon, dan ilustrasi digital.
      - 3D: Melibatkan pembuatan gambar yang memiliki tiga dimensi, yaitu panjang, lebar, dan tinggi. Ini termasuk pemodelan objek 3D, pembuatan karakter animasi, dan visualisasi arsitektural.

   b. Rendering:
   Proses mengubah model 3D menjadi gambar akhir. Rendering dapat mencakup berbagai teknik seperti pencahayaan, bayangan, dan tekstur untuk membuat gambar yang realistis.

   c. Animasi:
   Proses menciptakan gerakan dan perubahan bentuk pada gambar atau model. Animasi bisa dalam bentuk 2D atau 3D dan digunakan dalam berbagai aplikasi, seperti film animasi, video game, dan simulasi.

   d. Interaksi dan Antarmuka Pengguna (UI/UX):
   Grafik komputer juga mencakup desain antarmuka pengguna (user interface) yang intuitif dan interaktif, yang mempermudah pengguna dalam berinteraksi dengan perangkat lunak atau sistem komputer.

   e. Visualisasi Data:
   Menggunakan grafik komputer untuk menampilkan data dalam bentuk visual seperti grafik, diagram, dan peta untuk memudahkan pemahaman informasi kompleks.

2. **Sejarah grafik komputer**
   - Hardware revolution
   - Graphic subsystem
   Berpindahnya graphic processing dari CPU ⇒ dedicated chip (GPU) menyebabkan kapabilitas processing grafik komputer modern meningkat dengan pesat
   - Input devices
   Makin canggihnya input device (e.g kinect)
   - Mobile devices
   - Software improvements
     - Algorithms and data structure
     - Paralelization
     - Distributed and cloud computing

   ---

   1. CLI (Command Line Interface)
      - 1960s - now
      - Hanya menampilkan teks ASCII
      - Single task
   2. Vector
      - 1963 - 1980s
      - Menampilkan garis gambar dan goresan pada teks (2D dan 3D)
      - Single or multi tasks
   3. WIMP (Window, Icon, Menu, Pointer)
      - 2D bitmap raster display for PCs and workstation (’72 at Xerox PARC- now)
      - Menampilkan window, icon, dan teks
      - WYSIAYG (What you see is all you get)
      - GUI
   4. 3D Workstations
      - 1984 - now
      - Silicone Graphics (LGR Review)
      - Dapat membuat dan menampilkan gambar dan adegan realistis 3D secara real time
      - WYSIWYG
      - Multi-tasking, networked (client/server)
   5. High-End PC
      - NVIDIA GEFORCE dan ATI RADEON
      - Beberapa graphic card digabung dan dihubungkan dengan SLI bridge untuk mendapatkan GPU power yang maksimal

3. **Aplikasi grafik computer**
   a. Adobe Photoshop: Adobe Photoshop
   b. Adobe Photoshop: Software untuk editing dan manipulasi gambar serta pembuatan grafis.
   c. CorelDRAW: Aplikasi desain grafis vektor yang mencakup ilustrasi, layout, dan desain halaman.
   d. Adobe InDesign: Digunakan untuk desain layout cetak, seperti buku, majalah, brosur, dan dokumen lainnya.
   e. Adobe Illustrator: Aplikasi desain vektor untuk ilustrasi, logo, dan grafis lainnya.
   f. Inkscape: Aplikasi open-source untuk desain vektor yang sering digunakan untuk ilustrasi.
   g. Sketch: Aplikasi desain UI/UX yang populer di kalangan desainer untuk prototyping dan pembuatan interface.
   h. Affinity Designer: Aplikasi desain vektor dan raster yang alternatif dengan fitur komprehensif.
   i. Xara Designer Pro X: Software desain grafis yang mencakup vektor, foto, dan desain web.
   j. Blender: Software open-source untuk modeling, animasi, simulasi, rendering, compositing, dan motion tracking.
   k. Autodesk Maya: Aplikasi untuk modeling 3D, animasi, simulasi, dan rendering.

l. 3ds Max: Software untuk modeling, animasi, rendering, dan efek visual 3D.
m. Cinema 4D: Aplikasi untuk modeling, animasi, rendering, dan efek visual 3D yang populer di industri.
n. ZBrush: Software untuk modeling 3D dengan fokus pada sculpting digital.
o. GIMP: Program open-source untuk editing gambar dan grafis raster.
p. Procreate: Aplikasi digital art untuk sketsa, lukisan, dan ilustrasi di perangkat mobile.
q. Krita: Software open-source untuk lukisan digital, ilustrasi, dan animasi.
r. Fusion 360: Software CAD/CAM yang digunakan untuk desain produk dan manufaktur.
s. SketchUp: Aplikasi untuk modeling 3D, terutama untuk arsitektur dan desain interior.
t. Tinkercad: Aplikasi CAD untuk desain 3D yang sederhana, cocok untuk pemula dan pendidikan.
u. Substance Painter: Aplikasi untuk texturing 3D dengan fitur real-time painting dan rendering.

4. Komponen pendukung grafik computer

Komponen pendukung grafik komputer adalah perangkat keras dan perangkat lunak yang memungkinkan pembuatan, manipulasi, dan tampilan grafis yang kompleks dan berkualitas tinggi.

a. Perangkat Keras (Hardware) => GCMMSI

1. Graphics Processing Unit (GPU):
   - GPU adalah prosesor khusus yang dirancang untuk menangani tugas-tugas paralel, terutama perhitungan grafis.
   - Penting untuk rendering gambar dan video 3D yang kompleks dengan cepat dan efisien.
   - Contoh: NVIDIA GeForce, AMD Radeon.
2. Central Processing Unit (CPU):
   - CPU juga memainkan peran penting dalam grafik komputer, terutama dalam pemrosesan logika dan perhitungan umum yang tidak spesifik untuk grafis.
   - Menangani berbagai macam tugas secara serial.
   - Contoh: Intel Core, AMD Ryzen.
3. Monitor:
   - Perangkat output yang menampilkan informasi dalam bentuk visual
   - Monitor dengan resolusi tinggi dan kemampuan warna yang baik sangat penting untuk menampilkan grafis dengan akurasi tinggi.
   - Contoh: Monitor 4K, monitor dengan HDR.
4. Memory (RAM):
   - RAM yang cukup besar diperlukan untuk menangani data grafis yang besar dan kompleks selama proses rendering dan pengeditan.
   - Contoh: 16GB atau lebih untuk tugas grafis yang intensif.
5. Storage:
   - SSD (Solid State Drive) memberikan kecepatan baca/tulis yang cepat, penting untuk memuat file grafis besar dengan cepat.
   - Contoh: NVMe SSD.
6. Input Devices:
   - Perangkat input seperti tablet grafis, stylus, mouse khusus, dan keyboard ergonomis membantu dalam pembuatan dan manipulasi grafis.
   - Contoh: Wacom Tablet, Logitech MX Master Mouse.

b. Perangkat Lunak (Software) => OS 3G PE LF

1. Operating System (OS):
   - Sistem operasi yang mendukung grafik komputer dan perangkat lunak desain.
   - Perangkat lunak yang mengelola hardware dan menyediakan layanan umum untuk program komputer.
   - Contoh: Windows, macOS, Linux.
2. Graphics Drivers:
   - Driver grafis adalah perangkat lunak yang memungkinkan sistem operasi dan aplikasi berinteraksi dengan GPU.
   - Contoh: NVIDIA Drivers, AMD Radeon Software.
3. Graphics APIs (Application Programming Interfaces):
   - API adalah komponen software yang berfungsi sebagai perantara antara dua aplikasi yang tidak terhubung
   - Contoh: OpenGL, DirectX, Vulkan.
4. Graphics Software:
   - Perangkat lunak untuk pembuatan dan pengeditan grafis, animasi, dan model 3D.
   - Contoh: Adobe Photoshop, Blender, Autodesk Maya, CorelDRAW.
5. Plug-ins and Extensions:
   - Tambahan perangkat lunak yang menambahkan fitur atau kemampuan baru ke perangkat lunak grafis utama.
   - Contoh: V-Ray (plugin rendering untuk 3ds Max dan Maya), ZBrush plugins.
6. Libraries and Frameworks:
   - Perpustakaan dan kerangka kerja perangkat lunak yang menyediakan fungsi dan alat untuk pengembangan grafis komputer.
   - Contoh: Unity3D (game development framework), Unreal Engine (game engine).

c. Jaringan dan Infrastruktur => CS N

1. Cloud Services:
   - Layanan cloud yang menyediakan rendering dan penyimpanan berbasis awan untuk proyek grafis yang besar.

- Contoh: Google Cloud, Amazon Web Services (AWS).

2. Networking:
   - Infrastruktur jaringan yang mendukung kolaborasi dan transfer data grafis yang besar antara pengguna dan server.
   - Contoh: Jaringan berkecepatan tinggi (Gigabit Ethernet, Wi-Fi 6).

d. Perangkat Pendukung Tambahan => VR 3DPr

1. VR Headsets:
   - Headset VR untuk pengalaman grafis imersif dalam aplikasi realitas virtual.
   - Contoh: Oculus Rift, HTC Vive.
2. 3D Printers:
   - Printer 3D untuk mencetak model digital menjadi objek fisik.
   - Contoh: Ultimaker, Prusa i3.

5. Model dasar grafik Komputer => MRA

a. Pemodelan (Modeling)
   - Upaya untuk menggambarkan objek nyata ke dalam objek yang memiliki karakteristik geometris.
   - Pemodelan objek 3D dalam bentuk geometris ini dimaksudkan agar gambar dapat dimanipulasi tanpa kehilangan akurasi karena perhitungan dilakukan secara numeris berdasarkan kaidah matematis.

   Secara umum pemodelan geometris dapat diartikan sebagai:
   1. Memotret objek nyata dan lalu mengubahnya menjadi menjadi objek maya (virtual)
   2. Menjelaskan dunia nyata atau objek nyata menggunakan matematika
   3. Jika objek tidak ada, penggambaran dilakukan berdasarkan imajinasi artis

b. Rendering (Rendering)
   - Pemberian nuansa realistis kepada model-model geometris sehingga memiliki sifat/keadaan yang menyerupai sebenarnya.

   Langkah-langkah yang dilakukan pada proses rendering antara lain adalah:
   1. Penggambaran objek 3D dalam 2D
   2. Pemberian warna
   3. Pengaturan cahaya
   4. Pemberian gradasi warna
   5. Penambahan tekstur permukaan
   6. Pembuatan bayangan gambar
   7. Pantulan cahaya (reflection) maupun serapan cahaya (transparancy)
   8. Penghilangan objek-objek yang tersebunyi

c. Animasi (Animation)
   - Animasi adalah proses membuat objek atau model bergerak dan berubah bentuk seiring waktu. Animasi dapat diterapkan dalam 2D maupun 3D.

   Teknik: => KSMP
   1. Keyframe Animation: Mengatur posisi objek pada titik waktu tertentu, dan sistem akan mengisi gerakan di antaranya.
   2. Skeletal Animation: Menggerakkan objek kompleks dengan mengatur posisi tulang kerangka.
   3. Motion Capture: Merekam gerakan nyata dan menerapkannya pada model digital.
   4. Procedural Animation: Menggunakan algoritma untuk secara otomatis menghasilkan gerakan, seperti efek fisik atau gerakan alami.

Materi M2, M3 -> B2

OpenGL dan GLUT

1. Sejarah OpenGL

a. 1973: Graphical Kernel System (GKS)
b. 1982: Silicon Graphics Inc (SGI) mengimplementasikan konsep grafik pipeline 3D
c. 1992: OpenGL menjadi platform-independent API
d. 2002: OpenGL ES The Standard for Embedded Accelerated 3D Graphics

2. Pengenalan OpenGL

- OpenGL (Open Graphics Library) adalah sebuah standar API (Application Programming Interface) yang digunakan untuk membuat grafis komputer 2D dan 3D. Dikembangkan oleh Silicon Graphics Inc (SGI) pada tahun 1982, OpenGL memungkinkan program untuk bekerja dengan berbagai jenis perangkat keras grafis tanpa perlu ditulis ulang setiap kali ada pembaruan sistem.
- Untuk mempelajari pemrograman grafis dengan OpenGL, pemahaman tentang matematika, terutama operasi matriks, sangat diperlukan karena banyak melibatkan pembuatan shading, bentuk, dan transformasi (rotasi, translasi, skala).

Berikut adalah beberapa poin utama tentang OpenGL: => LP SIPM

a. Lintas-Platform dan Lintas-Bahasa: OpenGL bisa digunakan di berbagai sistem operasi (seperti Windows, macOS, dan Linux) dan mendukung berbagai bahasa pemrograman.
b. Spesifikasi Standar: OpenGL tidak berisi kode sumber, tetapi mendefinisikan standar spesifikasi yang diimplementasikan oleh produsen GPU (seperti NVIDIA, Intel, dan Samsung). Ini berarti perangkat keras dari berbagai produsen dapat bekerja dengan OpenGL.
c. Instruksi Grafis: OpenGL menyediakan lebih dari 250 fungsi yang dapat digunakan untuk menggambar dan mengelola objek 3D serta gambar-gambar kompleks dari bentuk-bentuk sederhana.
d. Penggunaan Luas: OpenGL digunakan dalam berbagai bidang seperti desain berbantuan komputer (CAD), realitas maya, visualisasi ilmiah, dan video game.
e. Manajemen oleh Khronos Group: OpenGL dikelola oleh Khronos Group, sebuah

konsorsium teknologi nirlaba yang memastikan pengembangan dan pemeliharaan standar ini.

3. **GLUT (GL Utility Toolkit)**
   GLUT (OpenGL Utility Toolkit) adalah sebuah toolkit yang menyediakan fungsi-fungsi dasar yang umum untuk semua sistem jendela (window system). Berikut adalah beberapa poin penting tentang GLUT:
   a. Membuka Jendela: GLUT dapat digunakan untuk membuka jendela di mana grafis OpenGL akan ditampilkan.
   b. Input dari Mouse dan Keyboard: GLUT menyediakan fungsi untuk mendapatkan input dari perangkat seperti mouse dan keyboard.
   c. Menu: GLUT mendukung pembuatan menu sederhana yang dapat digunakan dalam aplikasi grafis.
   d. Berbasis Peristiwa (Event-driven): GLUT menggunakan sistem berbasis peristiwa, yang berarti program merespons aksi dari pengguna seperti klik mouse atau penekanan tombol.
   e. Kode Portabel: Kode yang ditulis dengan GLUT dapat dijalankan di berbagai platform tanpa perubahan besar, membuatnya portabel.
   f. Keterbatasan: Meskipun portabel, GLUT tidak memiliki fitur lengkap seperti toolkit yang lebih spesifik untuk platform tertentu. Misalnya, GLUT tidak memiliki slide bars (bilah geser) atau widget canggih lainnya yang mungkin ditemukan dalam toolkit lain.
4. **Cara Kerja OpenGL**
   a. Display List: Mengumpulkan dan menyimpan perintah-perintah grafis untuk diproses nanti.
   b. Evaluator: Menghitung kurva dan geometri permukaan berdasarkan input polinomial.
   c. Per-Vertex Operations and Primitive Assembly: Memproses dan mentransformasikan sekumpulan vertex, serta memotong dan memasukkan primitif ke dalam viewport untuk rasterisasi.
   d. Rasterization: Mengubah deskripsi 2D (titik, garis, poligon) menjadi alamat frame-buffer dan nilai terkait.
   e. Per-Fragment Operations: Memperbarui frame-buffer berdasarkan nilai z untuk z-buffering dan mencampurkan warna pixel yang baru dengan yang lama serta melakukan operasi logika lainnya.
5. **Syntax perintah pada OpenGL**
   Sintaks perintah OpenGL mengikuti aturan penulisan dari library dimana fungsi tersebut berasal, berikut ini adalah format penulisan fungsi OpenGL. Semua perintah OpenGL menggunakan awalan gl diikuti dengan huruf capital pada setiap kata membentuk nama perintah, contohnya: **glClearColor**
   1. glClear(GL_COLOR_BUFFER_BIT): Menghapus buffer warna.
   2. glClearColor(r, g, b, a): Mengatur warna latar belakang dalam mode RGBA.
   3. glBegin(mode): Memulai penggambaran primitif (misalnya, GL_TRIANGLES untuk segitiga).
   4. glEnd(): Mengakhiri penggambaran.
   5. glVertex(x, y, z): Menentukan titik dalam koordinat (2D atau 3D).
   6. glColor3f(r, g, b): Menentukan warna dengan nilai red, green, dan blue.
   7. glTranslatef(x, y, z): Menggeser posisi objek ke koordinat yang ditentukan.
   8. glRotatef(angle, x, y, z): Memutar objek pada sudut tertentu di sekitar sumbu (x, y, z).
   9. glScalef(x, y, z): Mengubah ukuran objek berdasarkan faktor skala.
   10. glViewport(x, y, width, height): Mengatur ukuran dan posisi viewport.
   11. glEnable(option): Mengaktifkan fitur (misalnya, GL_DEPTH_TEST untuk depth testing).
   12. glDisable(option): Menonaktifkan fitur.
   13. glLoadIdentity(): Mengatur ulang transformasi ke keadaan awal (identitas).
   14. glMatrixMode(mode): Mengubah mode matriks (misalnya, GL_PROJECTION atau GL_MODELVIEW).
   15. glPushMatrix(): Menyimpan matriks saat ini.
   16. glPopMatrix(): Mengembalikan matriks yang disimpan dengan glPushMatrix().
   17. glClearDepth(value): Mengatur nilai pembersihan depth buffer.
   18. glDepthFunc(func): Mengatur fungsi perbandingan depth (misalnya, GL_LESS).
   19. glFinish(): Memastikan semua perintah OpenGL telah dieksekusi.
   20. glFlush(): Memastikan semua perintah penggambaran telah dieksekusi.
6. **Library yang berhubungan dengan OpenGL => CU IOS**
   a. OpenGL Core Library:
      - Windows: Library ini dikenal sebagai OpenGL32.
      - Unix/Linux: Dikenal sebagai GL (biasanya libGL.a).
   b. OpenGL Utility (GLU) Library:
      - GLU menyediakan fungsi tambahan yang melengkapi fungsi dasar OpenGL.
      - GLU membantu menghindari penulisan ulang kode untuk fungsi-fungsi yang umum digunakan.
   c. Integrasi dengan OS:
      - GLX: Digunakan untuk sistem jendela X Window di Unix/Linux.
      - WGL: Digunakan untuk Windows.
      - AGL: Digunakan untuk Macintosh.

Materi M4, M5 -> B3

Output primitif

Output primitif dalam grafika komputer menjelaskan bahwa elemen dasar pembentuk gambar adalah titik (point), garis (line), dan poligon (polygon).

1. **Titik dan Garis**

- Titik adalah satuan gambar terkecil. Dengan menggambar titik, kita bisa membuat objek apa pun, termasuk bentuk geometris. Operasi titik sering digunakan dalam pengolahan citra. Setiap titik di monitor memiliki koordinat dan warna.
- Garis adalah kumpulan titik-titik/pixel yang tersusun secara lurus dan linier dari titik awal sampai titik akhir.
- Poligon adalah elemen yang terdiri dari beberapa garis yang terhubung membentuk bentuk tertutup. Poligon umum seperti segitiga dan segi empat digunakan untuk membangun bentuk kompleks dalam grafika komputer. Algoritma seperti scan-line filling digunakan untuk mengisi warna dalam poligon.

2. Algoritma pembuatan Garis

a. Algoritma DDA (Digital Differential Analyzer)

- Algoritma DDA adalah algoritma yang digunakan untuk menggambar garis dalam komputer grafis.
- Algoritma ini menggunakan rumus dy=m.dx
- Semua koordinat titik yang membentuk garis diperoleh dari perhitungan kemudian dikonversikan menjadi nilai integer.
- Ide dasarnya adalah untuk menghitung inkremental dari x dan y untuk menentukan titik-titik yang membentuk garis dengan menggunakan perbedaan koordinat antara dua titik akhir.
- Algoritma ini lebih sederhana namun bisa kurang akurat dibandingkan dengan algoritma Bresenham.

Contoh 1:
Untuk menggambarkan algoritma DDA dalam pembentukan suatu garis yang menghubungkan titik (10,10) dan (17,16), pertama-tama ditentukan dx dan dy, kemudian dicari step untuk mendapatkan x_increment dan y_increment.
Δx = x1 — x0 = 17–10 = 7
Δy = y1 — y0 = 16 -10 = 6
selanjutnya hitung dan bandingkan nilai absolutnya.
|Δx| = 7
|Δy| = 6
karena |Δx| > |Δy|, maka step = |Δx| = 7,
maka diperoleh:
x_inc = 7/7= 1
y_inc = 6/7 = 0,86

Contoh 2:
Misalkan kita ingin menggambar garis dari titik (1, 2) ke titik (7, 5). Kita bisa menggunakan algoritma DDA untuk menghitung titik-titik yang membentuk garis tersebut dengan langkah sebagai berikut:

Hitung perbedaan antara koordinat akhir dan awal: Δx = 7 - 1 = 6 dan Δy = 5 - 2 = 3.
Hitung inkremental untuk x dan y:
Δx_inc = Δx / steps = 6 / 6 = 1
Δy_inc = Δy / steps = 3 / 6 = 0.5 (di mana steps adalah jumlah langkah yang dibutuhkan untuk menggambar garis).
Mulai dari titik awal (1, 2) dan tambahkan inkremental Δx dan Δy untuk setiap langkah hingga mencapai titik akhir (7, 5).
Hasil:
Titik-titik yang membentuk garis dari (1, 2) ke (7, 5) adalah sebagai berikut:
(1, 2), (2, 2.5), (3, 3), (4, 3.5), (5, 4), (6, 4.5), (7, 5)

b. Algoritma Bresenham

- Algoritma Bresenham digunakan untuk menggambar garis dalam komputer grafis secara efisien.
- Berbeda dengan DDA, algoritma ini menggunakan bilangan bulat saja, sehingga lebih efisien dalam komputasi.
- Rumus pembulatan: $p=2\Delta y-\Delta x$
- Algoritma Bresenham mencari titik-titik yang paling mendekati jalur sebenarnya dengan memperhitungkan nilai error untuk menentukan apakah titik berikutnya harus dipilih dari atas atau dari sisi garis.

Contoh 1:
Mari kita ambil contoh yang sama dengan sebelumnya, menggambar garis dari titik (1, 2) ke titik (7, 5) menggunakan algoritma Bresenham:
Hitung perbedaan antara koordinat akhir dan awal: Δx = 7 - 1 = 6 dan Δy = 5 - 2 = 3.
Hitung parameter pembulatan (p) = 2Δy - Δx = 2(3) - 6 = 0 (untuk nilai p awal).
Mulai dari titik awal (1, 2) dan tambahkan 1 untuk x dan/atau y berdasarkan nilai p untuk setiap langkah hingga mencapai titik akhir (7, 5).
Hasil:
Titik-titik yang membentuk garis dari (1, 2) ke (7, 5) adalah sebagai berikut: (1, 2), (2, 3), (3, 3), (4, 4), (5, 4), (6, 5), (7, 5)
Contoh 2:
Untuk membuat suatu garis yang menghubungkan titik (10,10) dan (17,16)
Hitung perbedaan antara koordinat akhir dan awal: Δx = 17 - 10 = 7 dan Δy = 16 - 10 = 6.
Hitung parameter pembulatan (p) = 2Δy - Δx = 2(6) - 7 = 5 (untuk nilai p awal).
Inisialisasi titik awal: (x0, y0) = (10, 10)
Iterasi untuk setiap langkah hingga mencapai titik akhir (17, 16):
Untuk setiap langkah, periksa nilai p:
Jika p ≥ 0, tambahkan 1 ke x dan y, dan perbarui p
$p=p+2(\Delta y-\Delta x)$
Jika p < 0, tambahkan 1 ke x dan perbarui p
$p=p+2\Delta y$
Hasil: Titik-titik yang membentuk garis dari (10, 10) ke (17, 16) adalah sebagai berikut:

(10, 10) => x=10,y=10,p=5
(11, 11) =>
p≥0→x=11,y=11,p=5+2(6−7)=5+2(−1)=3
(12, 12) =>
p≥0→x=12,y=12,p=3+2(6−7)=3+2(−1)=1
(13, 13) =>
p≥0→x=13,y=13,p=1+2(6−7)=1+2(−1)=−1
(14, 13) =>
p<0→x=14,y=13,p=−1+2(6)=−1+12=11
(15, 14) =>
p≥0→x=15,y=14,p=11+2(6−7)=11+2(−1)=9
(16, 15) =>
p≥0→x=16,y=15,p=9+2(6−7)=9+2(−1)=7
(17, 16) =>
p≥0→x=17,y=16,p=7+2(6−7)=7+2(−1)=5

3. **Algoritma pembuatan lingkaran**
   a. **Algoritma 8 titik simetris**
      - Algoritma titik simetris 8 digunakan untuk menggambar lingkaran dengan memanfaatkan simetri.
      - Memanfaatkan simetri lingkaran untuk menggambar hanya 1/8 lingkaran dan merefleksikan titik-titik tersebut ke 7 bagian lainnya.
      - Dalam 2D, ada delapan titik simetris yang penting:
        1. Simetri terhadap sumbu-x: (x, -y)
        2. Simetri terhadap sumbu-y: (-x, y)
        3. Simetri terhadap titik asal (0,0): (-x, -y)
        4. Simetri terhadap garis y = x: (y, x)
        5. Simetri terhadap garis y = -x: (-y, x)
        6. Simetri terhadap garis x = 0.5: (0.5 - y, 0.5 + x)
        7. Simetri terhadap garis x = -0.5: (-0.5 - y, 0.5 - x)
        8. Simetri terhadap garis y = 0.5: (0.5 + x, 0.5 - y)
        9. Simetri terhadap garis y = -0.5: (0.5 - x, -0.5 - y)
      - Memahami titik-titik simetris ini membantu dalam proses rendering dan transformasi objek dalam grafika komputer.

      Contoh:

      Misalkan kita memiliki titik (2, 3). Untuk menemukan titik-titik simetrisnya terhadap sumbu-x, sumbu-y, dan titik asal, kita dapat menggunakan hubungan simetris sederhana:

      - Simetri terhadap sumbu-x: (2, -3)
      - Simetri terhadap sumbu-y: (-2, 3)
      - Simetri terhadap titik asal: (-2, -3)

   b. **Algoritma titik tengah (Mid Point)**
      - Algoritma Midpoint digunakan untuk menggambar lingkaran secara efisien di komputer.
      - Algoritma ini mirip dengan algoritma Bresenham tetapi khusus untuk menggambar lingkaran.
      - Algoritma Midpoint menggunakan titik tengah (midpoint) dari garis untuk menentukan titik mana yang harus dipilih berikutnya untuk digambar.

      Contoh:

      Mari kita ambil contoh yang sama lagi, menggambar garis dari (1, 2) ke (7, 5) menggunakan algoritma Midpoint:

      1. Hitung parameter pendesain (p) awal berdasarkan titik tengah (midpoint) dari garis.
      2. Tentukan langkah-langkah berikutnya berdasarkan nilai p untuk memilih titik berikutnya yang mendekati jalur sebenarnya.
      3. Teruskan proses ini hingga mencapai titik akhir.

      Hasil:

      Titik-titik yang membentuk garis dari (1, 2) ke (7, 5) sesuai dengan algoritma Midpoint.

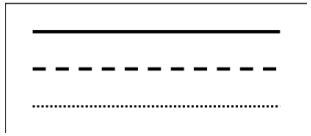
Atribut output primitif

4. **Pengertian atribut output primitive**
   Atribut Output Primitif adalah atribut yang berisi tentang pemberian warna dan ukuran besar kecilnya output primitive yang terbentuk.
5. **Atribut titik**
   Atribut dasar untuk titik adalah ukuran dan warna. Ukuran titik direpresentasikan sebagai beberapa piksel.
   a. Warna (Color): Menentukan warna titik menggunakan model warna seperti RGB (Red, Green, Blue).
   b. Ukuran (Size): Menentukan ukuran titik, biasanya dalam piksel. Misalnya, ukuran default titik sering kali adalah 1 piksel, tetapi bisa diubah sesuai kebutuhan.
6. **Atribut garis**
   Atribut dasar pada garis adalah tebal, warna, dan tipe.
   Atribut tebal garis:
   - Atribut ini menentukan ketebalan garis yang digambar. Ketebalan garis diukur dalam piksel.
   - Atribut tebal dapat ditentukan seperti panjang dan spasi antar titik. Algotitma raster menampilkan atribut ketebalan garis dengan memplot beberapa piksel sepanjang garis, baik horizontal (m<1) maupun vertical (m>1)
   - Di OpenGL, atribut ini dapat diatur menggunakan fungsi glLineWidth(GLfloat width), di mana width adalah ketebalan garis dalam piksel.

   Atribut warna garis:
   a. Atribut ini menentukan warna garis yang digambar. Warna biasanya diatur menggunakan model warna RGB (Red, Green, Blue).
   b. Di OpenGL, atribut ini dapat diatur menggunakan fungsi glColor3f(GLfloat red, GLfloat green, GLfloat blue), di mana red, green, dan blue adalah nilai intensitas warna yang berkisar antara 0.0 hingga 1.0.

   Atribut tipe garis:

a. Solid Line: Garis utuh tanpa putus.
b. Dashed Line: Garis putus-putus dengan segmen-segmen panjang.
c. Dotted Line: Garis titik-titik dengan segmen-segmen pendek.



Warna dan GrayScale

Opsi warna dalam sistem:

- Indeks warna dimasukkan ke dalam daftar nilai atribut sistem.
- Prosedur polyline menampilkan garis dengan warna yang diatur di frame buffer menggunakan prosedur setPixel.

Warna

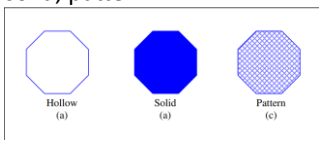
1. Pilihan nomor warna:
   - Bergantung pada jumlah bit per piksel di frame buffer.
2. Dua cara penyimpanan informasi warna:
   - Kode warna disimpan langsung di frame buffer.
   - Kode warna disimpan di tabel warna terpisah, dengan nilai piksel sebagai indeks.
3. Keuntungan tabel lookup warna:
   - Menyediakan banyak warna simultan tanpa memerlukan frame buffer besar.
   - Biasanya cukup dengan 256 atau 512 warna berbeda untuk gambar tunggal.
   - Tabel warna dengan ukuran 24 bit per entry, diakses dari frame buffer 8 bit per piksel.
   - Mengendalikan intensitas electron gun pada monitor RGB dengan tabel warna.

GrayScale

1. Fungsi warna pada monitor tanpa kapabilitas warna:
   - Menentukan tingkat keabuan untuk menampilkan primitif.
   - Nilai numerik antara 0 dan 1 dikonversi menjadi nilai biner untuk penyimpanan di sistem raster.
2. Spesifikasi intensitas untuk grayscale:
   - Sistem dengan 4 level grayscale, nilai intensitas 0.33 disimpan sebagai nilai biner 01.
   - Dengan 3 bit per piksel, bisa mengakomodasi 8 level keabuan.
   - Dengan 8 bit per piksel, bisa menampilkan 256 level keabuan.
3. Menggunakan tabel warna untuk berbagai monitor:
   - Tabel warna untuk monitor monokrom menggunakan nilai RGB.
   - Intensitas dihitung dengan rumus: Intensity = 0.5[min(r,g,b) + max(r,g,b)].

7. Fill area primitive

- Pengisian (filling) suatu area yang sudah didefinisikan dapat terbagi menjadi 3: hollow, solid, pattern.

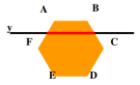


- Pilihan filling ini dapat diterapkan pada polygon atau area yang dibatasi oleh kurva tergantung pada kapabilitas dari software tersebut.
- Area dapat diwarnai dengan tipe brush, warna dan parameter transparansi

Algoritma Fill Area

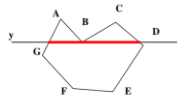
Pada system raster, ada tiga pendekatan dasar untuk mengisi area, yaitu:

a. Scan-line Polygon Fill Algorithm

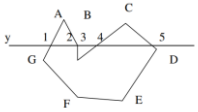


Gambar 4.x Ilustrasi prosedur scan line pada polygon cembung

- Scan line adalah garis horizontal yang digambar pada layar.
- Pendekatan ini bekerja dengan menentukan interval overlap untuk scan line yang melintasi area.
- Interval overlap adalah bagian dari scan line yang melewati area yang akan diisi.
- Piksel-piksel dalam interval overlap kemudian diisi dengan warna yang sesuai.
- Prosedur:
  a. Menentukan titik perpotongan scan line dengan batas polygon.
  b. Mengurutkan titik perpotongan dari kiri ke kanan.
  c. Mengisi frame buffer antara pasangan titik perpotongan dengan warna tertentu.
- Polygon cekung



Gambar 4.x Ilustrasi prosedur scan line pada polygon cekung berpotongan genap



Gambar 4.x Ilustrasi prosedur scan line pada polygon cekung berpotongan ganjil

   a. Scan line bisa berpotongan lebih dari sekali.
   b. Memodifikasi prosedur untuk menangani perpotongan ganjil/genap.

b. Boundary Fill Algorithm

- Mengisi area dengan warna tertentu, mulai dari titik dalam dan menyebar hingga mencapai batas yang ditentukan.
- Mulai dari titik interior dan mewarnai keluar hingga batas.

- Jika boundary ditentukan dengan satu warna, proses berlanjut per piksel sampai warna boundary ditemukan.
- Algoritma ini menerima input koordinat titik interior (x,y), warna isi dan warna boundary
- Prosedur akan menguji posisi tetangga untuk menentukan apakah posisi tersebut ada di boundary color.
- Jika tidak maka diwarnai dengan warna fill. Proses ini berlanjut sampai semua pixel yang ada pada boundary color diuji.

Contoh:

Titik awal: (5, 5).

Warna isi: merah.

Warna boundary: hitam.

Prosedur:

Isi (5, 5) dengan merah.

Periksa tetangga (5, 6), (5, 4), (6, 5), (4, 5).

Jika tetangga bukan hitam dan belum diisi, isi dengan merah dan lanjutkan proses.

c. Algoritma flood fill

- Mengisi area dengan warna tertentu, mulai dari titik dalam dan menyebar ke seluruh area hingga mencapai batas warna lain.
- Mulai dari titik (x, y).
- Jika warna titik tersebut adalah warna lama: Ganti warna titik tersebut dengan warna baru dan Panggil proses ini secara rekursif untuk tetangga (atas, bawah, kiri, kanan).
- Jika warna titik tersebut bukan warna lama, hentikan proses pada titik tersebut.

8. Karakter dan pembentukan karakter

Huruf, angka dan karakter lain dapat ditampilkan dalam berbagai ukuran (size) dan style. Jenis huruf atau typeface dikelompokkan menjadi beberapa kelompok utama antara lain:

a. Serif: Memiliki penambahan garis kecil atau 'serif' pada ujung hurufnya. Misalnya: Times New Roman, Georgia, dan Baskerville.
b. Sans serif: Tidak memiliki 'serif' atau garis tambahan pada ujung hurufnya, sehingga terlihat lebih bersih dan modern. Misalnya: Arial, Calibri, Helvetica, dan Tahoma.
c. Display: Jenis font yang dirancang untuk menarik perhatian dan digunakan pada judul atau teks besar. Jenis display yang umum digunakan antara lain Impact, Brush Script, dan Cooper Black.
d. Script: Meniru gaya tulisan tangan, sering kali dengan huruf-huruf yang saling terhubung. Contoh: Lucida Handwriting, Brush Script, dan Edwardian Script.
e. Dekoratif: Huruf dalam kategori dekoratif mempunyai bentuk indah. Misalnya: monotype corsiva
f. Monospace: Jenis font yang memiliki lebar karakter yang sama dan biasanya digunakan untuk coding atau menampilkan angka. Jenis monospace yang umum digunakan antara lain Courier New, Consolas, dan Lucida Console.
g. Handwriting: Handwriting adalah jenis font yang menyerupai tulisan tangan dan memberikan kesan personal. Jenis handwriting yang umum digunakan antara lain Comic Sans, Kristen ITC, dan Bradley Hand.

Note:

- Pilih font serif untuk desain formal dan klasik.
- Pilih font sans-serif untuk desain modern dan minimalis.
- Pilih font display untuk menarik perhatian pada judul atau teks besar.
- Pilih font script untuk memberikan kesan personal dan artistik.
- Pilih font display serif untuk desain klasik yang menarik perhatian.
- Pilih font monospace untuk coding atau menampilkan angka.
- Pilih font handwriting untuk memberikan kesan personal dan kreatif.

9. Antialiasing

Aliasing: Distorsi visual pada garis atau batas area yang terlihat seperti tangga (jaggies).
Antialiasing: Teknik untuk mengurangi efek jaggies.
Metode antialising:

a. Supersampling dan Postfiltering:

- Supersampling: Memecah piksel menjadi subpiksel, menggambar garis pada grid subpiksel, dan menentukan intensitas piksel berdasarkan jumlah subpiksel yang terkena garis.
- Metode:
  1. Garis infinitesimal: Intensitas berdasarkan jumlah subpiksel yang terkena.
  2. Garis tebal (1 piksel): Intensitas berdasarkan rasio subpiksel yang tertutupi.
- Pixelweighting Mask: Menggunakan mask seperti box atau Gaussian untuk menghitung intensitas.
- Continuous Filtering: Menggunakan fungsi permukaan kontinyu untuk smoothing, dengan konvolusi antara fungsi filter dan fungsi citra.

b. Area Sampling:

- Menganggap garis sebagai segi empat dengan lebar 1 piksel.
- Menghitung luas bagian piksel yang tertutup oleh garis secara geometris.
- Digunakan untuk anti-aliasing pada batas fill-area.
- Menggunakan modifikasi algoritma midpoint untuk menentukan persentase area yang tertutup.

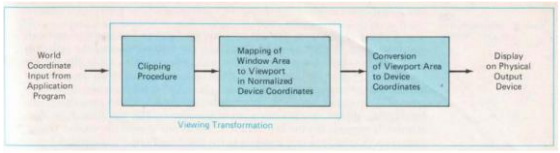
c. Pixel Phasing:

- Menggeser posisi elektron pada layar sebesar 1/4, 1/2, atau 3/4 diameter piksel.
- Biasanya built-in pada chipset grafis dan driver grafis.

Materi M6, M7, M8 -> B4

Viewing dan Clipping 2D

1. **Model konseptual grafik computer**
   - Model konseptual grafik komputer adalah kerangka kerja yang menjelaskan bagaimana pemandangan atau objek tiga dimensi di dunia nyata diubah menjadi gambar dua dimensi yang dapat ditampilkan di layar komputer.
   - Proses ini memerlukan teknik windowing untuk membatasi luas pandang objek sesuai dengan ukuran window. Window adalah area pada koordinat dunia yang dipilih untuk ditampilkan.
   - Viewport adalah area di layar yang menampilkan window tersebut.
   - Koordinat dan Mapping: Sistem koordinat dunia biasanya menggunakan derajat atau unit yang berbeda dari sistem koordinat layar yang menggunakan piksel. Oleh karena itu, diperlukan pemetaan atau mapping dari koordinat dunia ke koordinat layar untuk memastikan gambar ditampilkan dengan benar.
2. **Transformasi windows-viewport**



Gambar 4.4 Diagram Blok Transformasi *Windowing*

Transformasi windows ke viewport adalah proses dalam grafik komputer yang mengubah koordinat dari sistem koordinat dunia ke sistem koordinat layar sehingga gambar yang dihasilkan dapat ditampilkan dengan benar pada layar komputer. Proses ini mencakup beberapa langkah penting:

a. Window:
   - Definisi: Window adalah area yang dipilih dari pemandangan tiga dimensi yang ingin ditampilkan. Window berada dalam sistem koordinat dunia (world coordinates).
   - Fungsi: Window berfungsi sebagai jendela pandang yang membatasi bagian dari pemandangan yang akan ditampilkan di layar.

b. Viewport:
   - Definisi: Viewport adalah area pada layar komputer tempat window akan dipetakan dan ditampilkan. Viewport berada dalam sistem koordinat layar (screen coordinates).
   - Fungsi: Viewport menentukan di mana dan seberapa besar area window akan ditampilkan pada layar.

c. Proses Transformasi:
   - Langkah 1: Pemilihan Window: Tentukan koordinat window dalam sistem koordinat dunia. Misalnya, window bisa diatur dengan batas kiri, kanan, atas, dan bawah (xmin, xmax, ymin, ymax).
   - Langkah 2: Definisi Viewport: Tentukan koordinat viewport dalam sistem koordinat layar. Misalnya, viewport bisa diatur dengan batas kiri, kanan, atas, dan bawah (vxmin, vxmax, vymin, vymax).
   - Langkah 3: Skalasi: Hitung faktor skalasi untuk mengubah ukuran window ke ukuran viewport. Faktor skalasi untuk sumbu x dan y dihitung sebagai berikut:

$$S_x = \frac{vxmax - vxmin}{xmax - xmin}$$

$$S_y = \frac{vymax - vymin}{ymax - ymin}$$

   - Langkah 4: Translasi: Hitung translasi yang diperlukan untuk memetakan window ke viewport. Ini dilakukan dengan menyesuaikan koordinat awal window ke koordinat awal viewport:

$$T_x = vxmin - (xmin \times S_x)$$

$$T_y = vymin - (ymin \times S_y)$$

   - Langkah 5: Pemetaan Koordinat: Gunakan faktor skalasi dan translasi untuk mengonversi setiap titik dalam window ke titik yang sesuai dalam viewport. Formula pemetaan koordinat adalah:

$$x_v = x_w \times S_x + T_x$$

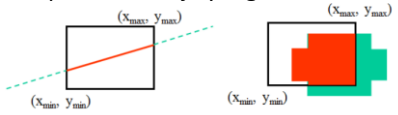
$$y_v = y_w \times S_y + T_y$$

Di mana x_w dan y_w adalah koordinat titik dalam window, dan x_v serta y_v adalah koordinat titik yang dipetakan dalam viewport.

d. Contoh: Misalkan window dalam koordinat dunia memiliki batas (xmin=0, xmax=100, ymin=0, ymax=100) dan viewport pada layar memiliki batas (vxmin=50, vxmax=450, vymin=50, vymax=450). Dengan menggunakan langkah-langkah di atas, kita dapat menghitung faktor skalasi dan translasi untuk memetakan koordinat window ke viewport.

e. Proses transformasi window ke viewport ini memastikan bahwa gambar yang dihasilkan dapat ditampilkan dengan benar dan proporsional pada layar komputer, memungkinkan pemandangan tiga dimensi dunia nyata untuk ditampilkan dalam dua dimensi pada layar.

3. **Clipping**
   - Clipping adalah proses pemotongan objek atau pengguntingan objek sehingga hanya objek yang berada pada area yang menjadi perhatian saja yang terlihat.



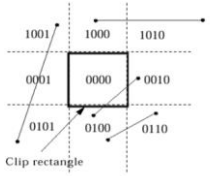
   a. **Penampakan garis**
   Garis-garis yang tampak pada area gambar atau viewport dapat dikelompokkan menjadi

tiga yaitu:
1. Garis yang terlihat seluruhnya (Fully visible).
2. Garis yang hanya terlihat sebagian (Partiality Visible).
3. Garis yang tidak terlihat sama sekali (Fully Invisible).

Proses clipping dilakukan terhadap garis-garis yang berada pada kondisi ke-2 yaitu garis-garis yang hanya terlihat sebagian saja

b. Algoritma Clipping Garis Cohen-Shuterland

- Pada algoritma Cohen-Sutherland, region viewport dibagi menjadi 9 dan masing-masing memiliki kode bit atau bit code yang terdiri dari 4 bit yang menyatakan kondisi dari garis yang melalui viewport atau region yang dimaksud.



Gambar 4.7 Bit Code dalam Cohen-Sutherland

- Kode empat bit menunjukkan posisi ujung garis pada region viewport. Sebagai contoh garis dengan ujung yang memiliki kode bit 1001 artinya ujungnya ada di kiri atas viewport.
- Ujung dengan kode bit 0010 berarti ada di kanan viewport. Dengan mengacu kepada kode bit maka proses clipping akan dilakukan secara lebih mudah dan efisien.

c. Algoritma Clipping Garis Liang-Barsky

- Algoritma Clipping Garis Liang-Barsky adalah metode dalam grafika komputer untuk memotong (clipping) garis sehingga hanya bagian dari garis yang berada di dalam area tertentu (disebut window atau clipping window) yang akan ditampilkan.
- Algoritma ini lebih efisien dibandingkan algoritma clipping garis lainnya, seperti algoritma Cohen-Sutherland, karena menggunakan representasi parametrik dari garis dan perhitungan linear untuk menentukan bagian garis yang berada di dalam window.
- Clipping yang lebih cepat dikembangkan berdasarkan persamaan parametrik dan hanya melakukan perhitungan linier tanpa perlu banyak pengecekan kondisi, membuatnya lebih cepat dan sederhana dalam implementasi untuk clipping garis dibandingkan metode lainnya.

Langkah-langkah Algoritma Liang-Barsky

1. **Definisikan Garis dan Window**:
   - Titik awal dan akhir garis: (x_1, y_1) dan (x_2, y_2).
   - Batas window: xmin, xmax, ymin, ymax.
2. **Hitung Parameter Delta**:
   - $\Delta x = x_2 - x_1$
   - $\Delta y = y_2 - y_1$
3. **Definisikan Parameter Parametrik p dan q**:
   - Untuk empat batas window:
     - $p_1 = -\Delta x, q_1 = x_1 - xmin$ (kiri)
     - $p_2 = \Delta x, q_2 = xmax - x_1$ (kanan)
     - $p_3 = -\Delta y, q_3 = y_1 - ymin$ (bawah)
     - $p_4 = \Delta y, q_4 = ymax - y_1$ (atas)
4. **Hitung Nilai t untuk Batas Window**:
   - Untuk setiap pasangan (p, q):
     - Jika $p_i = 0$ dan $q_i < 0$, garis berada di luar window dan tidak ada bagian yang akan ditampilkan.
     - Jika $p_i \neq 0$:
       - Hitung nilai t untuk enter dan leave:
         - $t_{\text{enter}} = \frac{q_i}{p_i}$ jika $p_i < 0$
         - $t_{\text{leave}} = \frac{q_i}{p_i}$ jika $p_i > 0$
5. **Tentukan Interval t yang Valid**:
   - Tentukan nilai t_{enter} terbesar dan t_{leave} terkecil yang memenuhi syarat $0 \leq t \leq 1$
   - Jika $t_{\text{enter}} > t_{\text{leave}}$, garis berada di luar window dan tidak ada bagian yang akan ditampilkan.
   - Jika $t_{\text{enter}} \leq t_{\text{leave}}$, hitung titik-titik clipping:
     - Titik awal clipping: $(x_1 + t_{\text{enter}}\Delta x, y_1 + t_{\text{enter}}\Delta y)$
     - Titik akhir clipping: $(x_2 + t_{\text{leave}}\Delta x, y_2 + t_{\text{leave}}\Delta y)$
6. **Gambar Garis yang Telah Diclip**:
   - Gambar garis dari titik awal clipping ke titik akhir clipping pada window.

Materi M9, M10 -> B5

Transformasi 2 Dimensi

1. Pengertian Transformasi

- Transformasi dalam grafika komputer adalah proses memodifikasi objek dua dimensi (2D) menggunakan operasi geometri untuk mengubah atau menyesuaikan komposisi pemandangan.
- Transformasi ini penting dalam pemodelan objek, yang merupakan sub bagian dari grafika komputer.

Contoh transformasi geometri meliputi:

a. Translasi (Translation): Memindahkan objek.
b. Penskalaan (Scaling): Mengubah ukuran objek, baik memperbesar atau memperkecil.
c. Rotasi (Rotation): Memutar objek.
d. Shearing (Shear): Menggeser bentuk objek secara paralel.
e. Refleksi (Reflection): Membalik objek terhadap sumbu tertentu.
f. Transformasi Gabungan (Composite Transformations): Kombinasi dari beberapa transformasi di atas.

Transformasi yang ada di atas dikenal sebagai transformasi affine, yang memindahkan objek tanpa merusak bentuknya.

Tujuan Transformasi:

a. Mengubah atau menyesuaikan komposisi pemandangan.
b. Mempermudah pembuatan objek yang simetris.
c. Melihat objek dari sudut pandang yang berbeda.
d. Memindahkan objek, sering digunakan dalam animasi komputer.

Proses transformasi dilakukan dengan mengalikan matriks objek dengan matriks transformasi, menghasilkan matriks baru yang berisi koordinat objek hasil transformasi.

Sebagai contoh, apabila sebuah garis yang melalui titik A(0,1) dan titik B(2,3) ditransformasikan dengan matriks

$$[T] = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$$

maka hasilnya adalah

$$[L][T]=\begin{bmatrix}0 & 1\\ 2 & 3\end{bmatrix}\begin{bmatrix}1 & 2\\ 3 & 1\end{bmatrix}=\begin{bmatrix}3 & 1\\ 11 & 7\end{bmatrix}=[L']$$

2. Translasi (Translation)

- Transformasi translasi merupakan suatu operasi yang menyebabkan perpindahan objek 2D dari satu tempat ke tempat yang lain.
- Perubahan ini berlaku dalam arah yang sejajar dengan sumbu X dan sumbu Y.
- Translasi dilakukan dengan penambahan translasi pada suatu titik koordinat dengan translation vector, yaitu (tx,ty), dimana tx adalah translasi menurut sumbu x dan ty adalah translasi menurut sumbu y.
- Koordinat baru titik yang ditranslasi dapat diperoleh dengan menggunakan rumus :

  $x' = x + tx \quad (x,y)$ = titik asal sebelum translasi

  $y' = y + ty \quad (x',y')$ = titik baru hasil translasi
- Translasi adalah transformasi dengan bentuk yang tetap, memindahkan objek apa adanya.
- Setiap titik dari objek akan ditranslasikan dengan besaran yang sama.
- Dalam operasi translasi, setiap titik pada suatu entitas yang ditranslasi bergerak dalam jarak yang sama.
- Pergerakan tersebut dapat berlaku dalam arah sumbu X saja, atau dalam arah sumbu Y saja atau keduanya.

  Contoh

  Untuk menggambarkan translasi suatu objek berupa segitiga dengan koordinat A(10,10) B(30,10) dan C(10,30) dengan tx,ty(10,20), tentukan koordinat yang barunya !

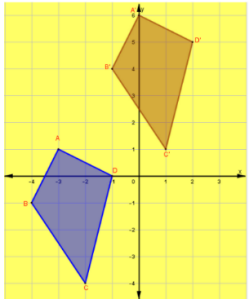
  Jawab

  A : x'=10+10=20
  y'=10+20=30
  A'=(20,30)

  B : x'=30+10=40
  y'=10+20=30
  B'=(40,30)

  C : x'=10+10=20
  y'=30+20=50
  C'=(20,50)

  Ilustrasi:



3. Penskalaan (Scaling)

- Penskalaan adalah suatu operasi yang membuat suatu objek berubah ukurannya baik menjadi mengecil ataupun membesar secara seragam atau tidak seragam tergantung pada faktor penskalaan (scalling factor) yaitu (sx,sy) yang diberikan.
- sx adalah faktor penskalaan menurut sumbu x dan sy faktor penskalaan menurut sumbu y.
- Koordinat baru diperoleh dengan:

  $x' = x + sx \quad (x,y)$ = titik asal sebelum diskala

  $y' = y + sy \quad (x',y')$ = titik setelah diskala
- Nilai lebih dari 1 menyebabkan objek diperbesar, sebaliknya bila nilai lebih kecil dari 1, maka objek akan diperkecil. Bila (sx,sy) mempunyai nilai yang sama, maka skala disebut dengan uniform scaling.

  Contoh

  Untuk menggambarkan skala suatu objek berupa segitiga dengan koordinat A(10,10)

  B(30,10) dan C(10,30) dengan (sx,sy) (3,2), tentukan koordinat yang barunya!

  Jawab:

  A : x' =10*3 =30

  y' =10*2 =20

  A' =(30,20)

  B : x' =30*3 =90

  y' =10*2 =20

  B' =(90,20)

  C : x' =10*3 =30

  y' =30*2 =60

  C' =(30,60)

  Ilustrasi:

4. Rotasi (Rotation)

- Rotasi adalah operasi yang menyebabkan objek berputar pada titik pusat atau sumbu putar tertentu berdasarkan sudut rotasi yang ditentukan. Untuk melakukan rotasi, diperlukan sudut rotasi dan titik pivot (xp, yp) sebagai pusat rotasi.
- Rotasi dapat dilakukan terhadap sumbu x, sumbu y, atau garis tertentu yang sejajar dengan sumbu-sumbu tersebut. Titik acuan rotasi bisa berada di pusat objek atau titik lainnya. Menurut aturan geometri:
  a. Rotasi searah jarum jam menggunakan sudut negatif.
  b. Rotasi berlawanan arah jarum jam menggunakan sudut positif.

Rotasi dapat dinyatakan dengan :

$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$

$y' = r\sin(\phi + \theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$

sedangkan diketahui

$x = r\cos\phi$, $y = r\sin\phi$

lakukan subtitusi, maka :

$x' = x\cos\phi - y\sin\theta$

$y' = x\sin\phi + y\cos\theta$

Matriks rotasi dinyatakan dengan :

$P' = R.P$

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

Rotasi suatu titik terhadap pivot point (xp,yp) :

$x' = xp + (x - xp)\cos\theta - (y - yp)\sin\theta$

$y' = yp + (x - xp)\sin\theta + (y - yp)\cos\theta$

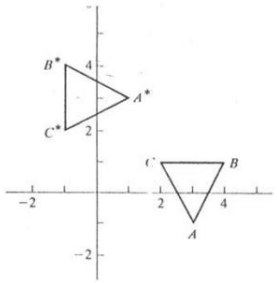
Contoh 1. Diketahui koordinat titik yang membentuk segitiga {(3, -1), (4, 1), (2, 1). Gambarkan objek tersebut kemudian gambarkan pula objek baru yang merupakan transformasi rotasi objek lama sebesar 90° CCW dengan pusat rotasi (0,0).

Jawab:
Matriks transformasi umum adalah

$[T_{90}] = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ $[T_{180}] = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

$[T_{270}] = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ $[T_{360}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Maka dengan mengalikan titik-titik segitiga tersebut dengan matriks transformasi yang sesuai, maka akan diperoleh hasil yang digambarkan sebagai berikut:



Gambar 4.2 Ilustrasi Rotasi

Contoh 2. Untuk menggambarkan rotasi suatu objek berupa segitiga dengan koordinat A(10,10), B(30,10) dan C(10,30) dengan sudut rotasi 300° terhadap titik pusat cartesian (10,10), dilakukan dengan menghitung koordinat hasil rotasi tiap titik satu demi satu.
Jawab:
Titik A
x'= xp+(x - xp) cos ɵ - (y - yp) sin ɵ
=10+(10-10)*0.9 – (10-10)*0.5 = 10
y'= yp+(x - xp) sin ɵ + (y - yp) cos ɵ
= 10+(10-10)*0.5 – (10-10)*0.9 = 10
Titik A'(10,10)

Titik B
x'= xp+(x - xp) cos ɵ - (y - yp) sin ɵ
=10+(30-10)*0.9 – (10-10)*0.5 = 28
y'= yp+(x - xp) sin ɵ + (y - yp) cos ɵ
= 10+(30-10)*0.5 – (10-10)*0.9 = 20
Titik B'(28,20)

Titik C
x' = xp+(x - xp) cos ɵ - (y - yp) sin ɵ
=10+(10-10)*0.9 – (30-10)*0.5 = 0
y' = yp+(x - xp) sin ɵ + (y - yp) cos ɵ
= 10+(10-10)*0.5 – (30-10)*0.9 = 28
Titik C'(0,28)

5. Refleksi (Reflection)

Proses refleksi dilakukan dengan mengalikan koordinat titik objek dengan matriks refleksi yang sesuai. Berikut adalah beberapa matriks refleksi untuk berbagai jenis refleksi:

1. Refleksi terhadap Sumbu X:

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

2. Refleksi terhadap Sumbu Y:

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

3. Refleksi terhadap Garis y = x:

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

4. Refleksi terhadap Garis y = -x:

$$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$

5. Refleksi terhadap Asal Koordinat (0, 0):

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

Contoh Refleksi
Misalkan kita memiliki titik A(3, 4) dan ingin merefleksikan titik ini terhadap sumbu X:

a. Koordinat awal A adalah (3, 4).
b. Menggunakan formula refleksi terhadap sumbu X:

$$x' = x = 3$$

$$y' = -y = -4$$

c. Titik hasil refleksi A' adalah (3, -4).

Dengan menggunakan refleksi, kita dapat menghasilkan bayangan cermin dari objek asli sesuai dengan sumbu atau bidang refleksi yang dipilih, yang berguna dalam berbagai aplikasi grafika komputer seperti desain grafis, animasi, dan simulasi.

- Refleksi adalah transformasi yang membuat mirror (pencerminan) dari image suatu objek. Image mirror untuk refleksi 2D dibuat relatif terhadap sumbu dari refleksi dengan memutar 180o terhadap refleksi. Sumbu refleksi dapat dipilih

pada bidang x,y. Refleksi terhadap garis y=0, yaitu sumbu x dinyatakan dengan matriks

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

- Transformasi membuat nilai x sama tetapi membalikan nilai y berlawanan dengan posisi koordinat. Langkah :
  a. Objek diangkat
  b. Putar 180o terhadap sumbu x dalam 3D
  c. Letakkan pada bidang x,y dengan posisi berlawanan
  d. Refleksi terhadap sumbu y membalikan koordinat dengan nilai y tetap.

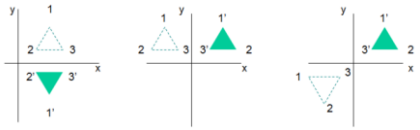
$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

- Refleksi terhadap sumbu x dan y sekaligus dilakukan dengan refleksi pada sumbu x terlebih dahulu, hasilnya kemudia direfleksi terhadap sumbu y. Transformasi ini dinyatakan dengan :

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

- Refleksi ini sama dengan rotasi 180° pada bidang xy dengan koordinat menggunakan titik pusat koordinat sebagai pivot point. Refleksi suatu objek terhadap garis y=x dinyatakan dengan bentuk matriks.

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$



- Matriks dapat diturunkan dengan menggabungkan suatu sekuen rotasi dari sumbu koordinat merefleksi matriks.
- Pertama-tama dilakukan rotasi searah jarum jam dengan sudut 45° yang memutar garis y=x terhadap sumbu x.
- Kemudian objek direfleksi terhadap sumbu y, setelah itu objek dan garis y=x dirotasi kembali ke arah posisi semula berlawanan arah dengan jarum jam dengan sudut rotasi 90°.
- Untuk mendapatkan refleksi terhadap garis y=-x dapat dilakukan dengan tahap :
  - Rotasi 45° searah jarum jam
  - Refleksi terhadap axis y
  - Rotasi 90° berlawanan arah dengan jarum jam
- Dinyatakan dengan bentuk matriks

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

- Refleksi terhadap garis y=mx+b pada bidang xy merupakan kombinasi transformasi translasi – rotasi – refleksi .
- Lakukan translasi mencapai titik perpotongan koordinat
  - Rotasi ke salah satu sumbu
  - Refleksi objek menurut sumbu tersebut

6. Shear (Shearing)

Shear adalah bentuk transformasi yang membuat distorsi dari bentuk suatu objek, seperti menggeser sisi tertentu. Terdapat dua macam shear yaitu:

a. Shear terhadap sumbu x

Shear terhadap sumbu x

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Dengan koordinat transformasi

$$x' = x + shx.y \qquad y' = y$$

Parameter shx dinyatakan dengan sembarang bilangan. Posisi kemudian digeser menurut arah horizontal.

b. Shear terhadap sumbu y.

Shear terhadap sumbu y

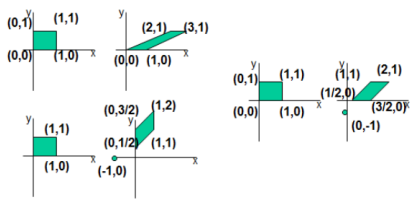
$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Dengan koordinat transformasi

$$x' = x \qquad y' = shy.x + y$$

Parameter shy dinyatakan dengan sembarang bilangan. Posisi koordinat kemudian menurut arah vertikal.

Gambar di bawah mengilustrasikan proses shearing.



Gambar 4.4 Ilustrasi Proses *Shearing*

7. Transformasi Homogen

- Transformasi homogen adalah jenis transformasi yang mencakup berbagai proses transformasi secara umum, berguna untuk menangani objek kompleks di dunia nyata yang memiliki koordinat masing-masing.
- Transformasi homogen sangat penting dalam grafika komputer untuk menyelesaikan masalah kompleks yang melibatkan pemodelan dan manipulasi objek.
- Poin-poin Penting dalam Transformasi Homogen:
  a. Origin Invarian: Titik asal (origin) bersifat invarian, artinya koordinatnya tidak akan berubah meskipun ditransformasikan, tetap berada di (0,0).
  b. Koordinat Homogen: Dalam kondisi nyata, origin tidak selalu berada di (0,0). Koordinat homogen digunakan untuk memetakan titik (0,0) ke posisi lain, memungkinkan fleksibilitas dalam transformasi.
  c. Elemen Tambahan: Matriks transformasi homogen memiliki elemen tambahan yang memungkinkan pemetaan titik ke posisi baru.

Untuk itu maka didefinisikan Matriks Transformasi Umum (MTU) sebagai berikut:

$$[T] = \begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

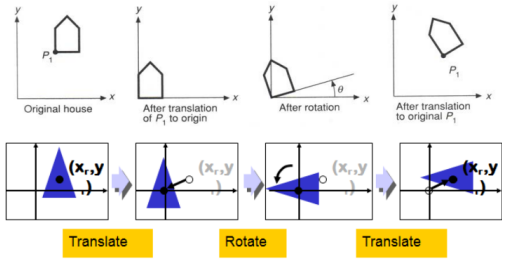
- Dimana a, b, c, d merupakan elemen untuk skala, rotasi,refleksi dan shearing; m, n merupakan elemen untuk translasi; s adalah elemen untuk overal scaling; dan p, q adalah elemen untuk proyeksi.

- Rotasi pada Sumbu Sembarang
  Jika sebuah objek dirotasikan sebesar θ° dengan pusat rotasi (m, n), maka langkah-langkah yang harus dilakukan adalah
  a. Translasikan pusat rotasi ke (0, 0); karena yang kita ketahui hanyalah rumus rotasi pada origin
  b. Lakukan rotasi sebesar yang diinginkan
  c. Re-translasi pusat rotasi ke posisi semula
  Dengan demikian matriks transformasinya menjadi

$$[T]=\begin{bmatrix}1&0&0\\0&1&0\\-m&-n&1\end{bmatrix}\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\m&n&1\end{bmatrix}$$

$$\begin{bmatrix}x^* & y^* & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}[T]$$

Proses ini diilustrasikan sebagai berikut:



Gambar 4.5 Ilustrasi Rotasi Pada Sumbu Sembarang

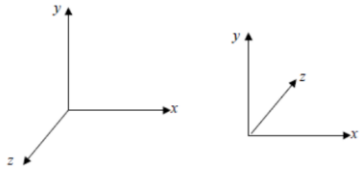
- Refleksi pada Garis Sembarang
  Jika sebuah objek direfleksikan pada sebuah garis maka langkah-langkah yang harus dilakukan adalah
  a. Translasikan cermin sedemikian rupa sehingga menyentuh titik origin
  b. Rotasikan cermin sehingga berimpit dengan salah satu sumbu utama
  c. Refleksikan objek
  d. Re-rotasi
  e. Re-translasi
  Jadi MTU (Matriks Transformasi Universal) terdiri dari 5 buat matriks transformasi sebagai berikut:

$$[T]=[T_r][Rot][\mathrm{R}][Rot^{-1}][T_r^{-1}]$$

Materi M12, M13 -> B6

Transformasi 3 Dimensi

1. **Pengertian transformasi 3D Operasi dasar Transformasi 3D**
   - Transformasi 3D adalah proses memanipulasi objek dalam ruang tiga dimensi (3D) menggunakan berbagai operasi matematika untuk mengubah posisi, orientasi, skala, atau bentuk objek tersebut. Operasi-operasi ini meliputi translasi, rotasi, refleksi, dan shear. Dalam ruang 3D, posisi suatu titik dinyatakan oleh tiga sumbu koordinat: x, y, dan z.
   - Dalam ruang dua dimensi, posisi suatu titik dinyatakan oleh dua sumbu, yaitu sumbu x dan sumbu y. Di ruang tiga dimensi, ada sumbu ketiga, yaitu sumbu z.
   - Ada dua konvensi untuk merepresentasikan titik dalam ruang 3D: kaidah tangan kanan dan kaidah tangan kiri.
     - Pada kaidah tangan kanan, jika sumbu x positif ke kanan dan sumbu y positif ke atas, maka sumbu z positif mendekati kita.
     - Pada kaidah tangan kiri, sumbu z positif menjauhi kita.



Gambar 6.1 Sumbu Koordinat 3 Dimensi

   - Transformasi geometris dasar di ruang 3D serupa dengan di ruang 2D, namun terdapat beberapa perbedaan penting:
     a. Pada rotasi, perlu membedakan rotasi terhadap masing-masing sumbu (x, y, z).
     b. Refleksi dilakukan terhadap bidang-bidang xy, yz, atau zx.
     c. Shear dilakukan terhadap dua sumbu, misalnya x dan z.
   - Selain itu, sistem koordinat homogen dapat digunakan untuk merepresentasikan titik (x, y, z) dalam ruang 3D sebagai matriks kolom [x y z h].

$$\begin{pmatrix}x'\\y'\\z'\\1\end{pmatrix}=\begin{pmatrix}m_{11}&m_{12}&m_{13}&m_{14}\\m_{21}&m_{22}&m_{23}&m_{24}\\m_{31}&m_{32}&m_{33}&m_{34}\\m_{41}&m_{42}&m_{43}&m_{44}\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}$$

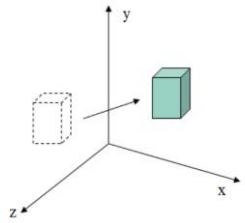
Operasi Dasar Transformasi 3D

- Matriks transformasi umum 3D dinyatakan sebagai matriks 4x4 sebagai berikut:

$$\begin{bmatrix}a&b&c&p\\d&e&f&q\\g&h&i&r\\l&m&n&s\end{bmatrix}$$

  Dimana a, b, c, d, e, f, g, h, i adalah elemen yang berpengaruh terhadap transformasi linier; p, q, r adalah elemen yang untuk proyeksi dan perspektif l, m, n adalah elemen untuk translasi pada sumbu x, y dan z s adalah elemen untuk overall scaling
- **Translasi**
  Transformasi translasi 3 dimensi dinyatakan sebagai
  $x'=x+t_x,\quad y'=y+t_y,\quad z'=z+t_z$
  Komposisi perkalian matriksnya adalah sebagai berikut:

$$\begin{bmatrix}x' & y' & z' & 1\end{bmatrix}=\begin{bmatrix}x & y & z & 1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\t_x&t_y&t_z&1\end{bmatrix}$$



Gambar 6.2 Translasi 3 Dimensi

- **Penskalaan**

  Penskalaan dalam ruang tiga dimensi adalah proses mengubah ukuran objek 3D dengan memperbesar atau memperkecil dimensinya.

  Dalam ruang tiga dimensi, terdapat dua jenis penskalaan:

  a. Local Scaling: Pensakalan dilakukan terhadap salah satu atau semua sumbu (x, y, dan z) secara individual.

  b. Overall Scaling (Global Scaling): Pensakalan dilakukan secara seragam untuk semua sumbu.

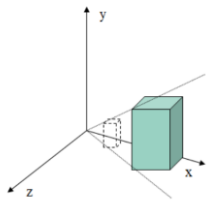
  Elemen matriks transformasi yang mempengaruhi local scaling adalah elemen diagonal, yaitu a, e, dan i. Sedangkan elemen yang mempengaruhi overall scaling adalah elemen s. Formulasi transformasi untuk global scaling dan local scaling berbeda sesuai dengan elemen matriks yang digunakan.

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{S_g} \end{bmatrix}$$

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Operasi transformasi scaling pada elemen matriksnya adalah operasi perkalian sebagia berikut.

$$x' = x \cdot s_x, \quad y' = y \cdot s_y, \quad z' = z \cdot s_z$$



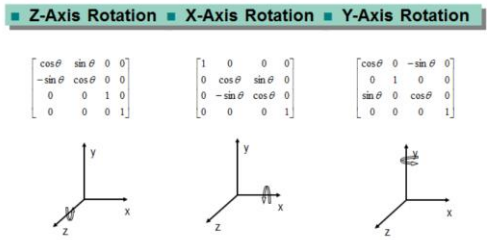
Gambar 6.3 Skala 3 Dimensi

- **Rotasi**

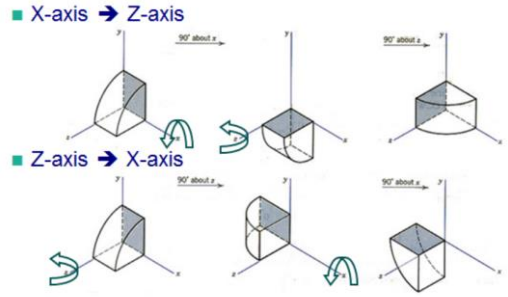
  Rotasi dalam ruang tiga dimensi adalah proses memutar objek di sekitar satu atau lebih sumbu koordinat (x, y, atau z). Tidak seperti rotasi dalam 2D, rotasi 3D membutuhkan perhatian khusus karena melibatkan tiga sumbu, sehingga ada beberapa jenis rotasi tergantung pada sumbu yang menjadi pusat rotasi.

  Ada tiga jenis rotasi utama dalam 3D, yaitu:

  1. Rotasi terhadap sumbu x (Rotasi x)
  2. Rotasi terhadap sumbu y (Rotasi y)
  3. Rotasi terhadap sumbu z (Rotasi z)



Dalam rotasi 3D perlu diperhatikan bahwa urutan proses rotasi tidaklah komutatif. Gambar berikut mengilustrasikan keadaan ini. Digambarkan bahwa hasil akhir antara urutan proses rotasi pada sumbu x dilanjutkan rotasi sumbu z TIDAK SAMA HASILNYA dengan rotasi pada sumbu z dilanjutkan dengan rotasi pada sumbu x.



Gambar 6.5 Rotasi 3 Dimensi Tidak Komutatif

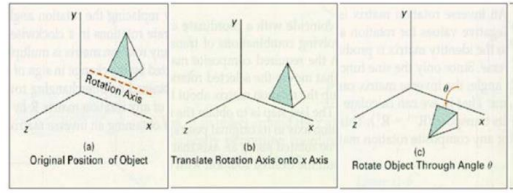
Rotasi pada Sumbu yang Paralel dengan Sumbu Utama

Langkah-langkah yang harus dilalui adalah

(i) translasikan objek sedemikian rupa sehingga berimpit dengan salah satu sumbu utama

(ii) lakukan rotasi

(iii) lakukan re-translasi. Ilustrasinya ditunjukkan pada gambar di bawah ini.

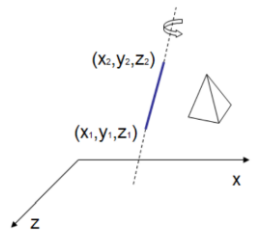


Gambar 6.6 Rotasi Pada Sumbu Sembarang yang Sejajar Sumb

Rotasi pada Sumbu Sembarang

Proses ini cukup kompleks dan dilakukan sebanyak 5 langkah. Formulasi perkalian matriksnya adalah sebagai berikut:

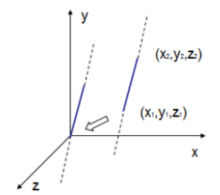
$$[T_R]_{ARB} = [T_{TR}]^{-1}[T_R]_x^{-\alpha}[T_R]_y^{-\phi}[T_R]_z^{\theta}[T_R]_y^{\phi}[T_R]_x^{\alpha}[T_{TR}]$$



Gambar 6.7 Rotasi 3 Dimensi Pada Sumbu Sembarang

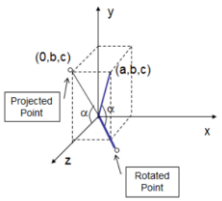
Langkah-langkah yang dilakukan adalah
1. Translasikan (x1, y1, z1) ke origin
2. Rotasikan (x'2, y'2, z'2) pada sumbu Z
3. Rotasikan objek pada sumbu Z
4. Re-rotasi sumbu ke orientasi semula
5. Re-translasi

Langkah 1: Translasi



$$T_{TR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix}$$

Langkah 2: Rotasi

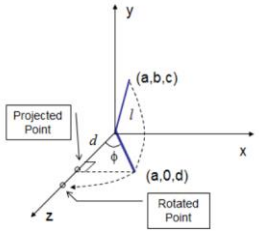


$$\sin\alpha = \frac{b}{\sqrt{b^2+c^2}} = \frac{b}{d}$$

$$\cos\alpha = \frac{c}{\sqrt{b^2+c^2}} = \frac{c}{d}$$

$$[T_R]_x^{\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Langkah 3: Rotasi



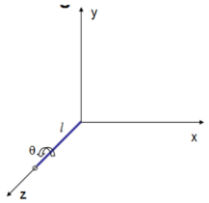
$$\sin\phi = \frac{a}{l}, \quad \cos\phi = \frac{d}{l}$$

$$l^2 = a^2 + b^2 + c^2 = a^2 + d^2$$

$$d = \sqrt{b^2+c^2}$$

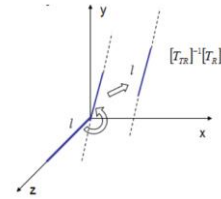
$$[T_R]_y^{\phi} = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d/l & & & \\ 0 & & & \\ -a/ & & & \\ 0 & & & \end{bmatrix}$$

Langkah 4: Re-Rotasi



$$[T_R]_z^{\theta} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Langkah 5: Re-translasi



$$[T_{TR}]^{-1}[T_R]_x^{\alpha}[T_R]_y^{\phi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_1 & y_1 & z_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Refleksi**

  Refleksi adalah salah satu jenis transformasi geometri yang memindahkan titik atau bangun dengan menggunakan sifat pembentukan bayangan oleh sebuah cermin. Dalam konteks 3D, refleksi terjadi pada objek di ruang tiga dimensi.

  Konsep Refleksi 3D:
  a. Refleksi 3D melibatkan pemindahan objek dalam tiga dimensi dengan menggambarkan bayangan objek di sekitarnya.
  b. Jika kita memiliki objek dalam ruang 3D, kita dapat memindahkannya dengan cara mencerminkannya terhadap bidang tertentu.

  Bidang Refleksi:
  a. Dalam refleksi 3D, kita menggunakan bidang sebagai cermin untuk mencerminkan objek.
  b. Bidang refleksi dapat berupa bidang XY, bidang XZ, atau bidang YZ.

  Rumus Refleksi 3D:
  a. Refleksi terhadap bidang XY: Jika kita memiliki titik (x, y, z), maka titik bayangannya adalah (x, y, -z).
  b. Refleksi terhadap bidang XZ: Jika kita memiliki titik (x, y, z), maka titik bayangannya adalah (x, -y, z).
  c. Refleksi terhadap bidang YZ: Jika kita memiliki titik (x, y, z), maka titik bayangannya adalah (-x, y, z).

  Contoh:
  Misalkan kita memiliki titik A(2, 3, 4). Jika kita mencerminkan titik A terhadap bidang XY, maka titik bayangannya adalah A'(2, 3, -4).
  Begitu pula jika kita mencerminkan titik A terhadap bidang XZ, maka titik bayangannya adalah A''(2, -3, 4).

2. **Sistem Koordinat berganda**
   - Sistem Koordinat Berganda digunakan untuk menggambarkan posisi dan orientasi objek dalam ruang tiga dimensi dengan menggunakan beberapa sistem koordinat yang berbeda tetapi saling terkait.
   - **Hierarchical Modeling**

     a. Sistem Koordinat Dunia (World Coordinate System): Ini adalah sistem koordinat global yang digunakan sebagai referensi utama. Semua sistem koordinat lainnya dinyatakan relatif terhadap sistem ini.
     b. Sistem Koordinat Traktor (Tractor Coordinate System): Ini adalah sistem koordinat yang bergerak bersama dengan traktor. Sistem ini didefinisikan relatif terhadap sistem koordinat dunia.
     c. Sistem Koordinat Roda Depan (Front-Wheel Coordinate System): Sistem koordinat ini bergerak bersama roda depan traktor dan didefinisikan relatif terhadap sistem koordinat traktor.
   - **Hierarki Sistem Koordinat**
     a. Saat traktor bergerak, sistem koordinat traktor dan sistem koordinat roda depan bergerak dalam sistem koordinat dunia.
     b. Roda depan berputar dalam sistem koordinat roda.
     c. Saat traktor berbelok, sistem koordinat roda depan berputar dalam sistem koordinat traktor.

- **Transformasi Deskripsi Objek dari Satu Sistem Koordinat ke Sistem Lain**
  Proses transformasi dari satu sistem koordinat ke sistem koordinat lainnya melibatkan beberapa langkah:
  a. Translasi: Memindahkan asal koordinat baru ke posisi asal koordinat yang lain.
  b. Rotasi: Menyelaraskan sumbu-sumbu koordinat yang bersesuaian.
  c. Skala: Transformasi skala, jika skala yang digunakan dalam kedua sistem koordinat berbeda.

Contoh Transformasi

Ilustrasi Sistem Koordinat:

- Terdapat dua sistem koordinat: sistem koordinat asal (x, y, z) dan sistem koordinat baru (x', y', z').
- Sistem koordinat baru diwakili oleh u'_x, u'_y, dan u'_z yang diletakkan pada posisi (x0, y0, z0) relatif terhadap sistem koordinat asal.

Matriks Transformasi:

- Matriks R.T menggabungkan rotasi dan translasi untuk mengubah koordinat dari sistem asal ke sistem baru.
- Matriks ini menunjukkan bagaimana vektor koordinat dalam sistem asal dikalikan untuk mendapatkan vektor koordinat dalam sistem baru.

Matriks transformasi biasanya berbentuk 4x4 untuk mengakomodasi transformasi linier (rotasi dan skala) serta translasi dalam ruang tiga dimensi. Elemen-elemen matriks tersebut secara khusus mendefinisikan transformasi yang dibutuhkan untuk memetakan satu sistem koordinat ke sistem yang lain.

Materi M14, M15 -> B7

Visualisasi 3 Dimensi

1. **Konsep visualisasi 3D**
   - **Apa itu Visualisasi 3D?**
     a. Visualisasi 3D adalah proses pembuatan dan penyajian konten digital 3D menggunakan perangkat lunak desain 3D dalam bentuk gambar 2D atau video animasi.
     b. Proses visualisasi 3D telah mengalami perkembangan signifikan selama beberapa dekade terakhir dan saat **ini digunakan untuk menghasilkan konten CGI digital yang luar biasa.**
   - **Perbedaan antara Visualisasi 3D dan Desain 3D:**
     a. Visualisasi 3D berbeda dari desain 3D karena fokus pada pembuatan konten grafis, termasuk gambar dan animasi.
     b. Visualisasi 3D tidak dapat langsung digunakan untuk menciptakan hasil nyata seperti file CAD (Computer-Aided Design) atau CAM (Computer-Aided Modeling).
   - **Elemen Visualisasi 3D:**
     a. Model 3D: Proses dimulai dengan pembuatan model 3D yang mendetail dengan spesifikasi lebar, panjang, dan kedalaman tertentu dalam tiga dimensi.
     b. Pencahayaan dan Tekstur: Setelah model 3D disetujui, pencahayaan dan tekstur diterapkan untuk menciptakan efek fotorealistik.
     c. Efek Tambahan: Desainer 3D dapat menambahkan efek seperti bayangan, tekstur, warna, dan bahkan efek blur untuk menciptakan persepsi objek bergerak.
   - **Manfaat Visualisasi 3D:**
     a. Komunikasi Optimal: Visualisasi 3D memungkinkan tim proyek untuk memahami bagaimana hasil akhir akan terlihat sejak awal. Konsep proyek dapat disajikan dengan jelas kepada para pemangku kepentingan.
     b. Promosi Produk: Dengan teknik ini, produk dapat ditampilkan dari berbagai sudut, memungkinkan bisnis untuk memperlihatkan tampilan produk yang menakjubkan.
   - **Penerapan teknologi visualisasi 3D:**
     a. Domain ilmiah
     b. Segmen manufaktur
     c. Industri konstruksi dan arsitektur
     d. Kampanye promosi dan pemasaran
     e. Sektor permainan
     f. Film
2. **Transformasi system pandang**
   - Transformasi sistem pandang dalam grafika komputer adalah proses mengubah atau menyesuaikan komposisi pemandangan untuk memudahkan pembuatan objek simetris, melihat objek dari sudut pandang yang berbeda, dan memindahkan satu atau beberapa objek dari satu tempat ke tempat lain
   - Transformasi sistem pandang adalah perubahan cara kita melihat atau memandang suatu hal.
   - Ini dapat mencakup perubahan sudut pandang, pergeseran fokus, atau penggunaan perspektif yang berbeda.
   - **Tujuan Transformasi Sistem Pandang:**
     a. Mengubah Komposisi Pemandangan: Transformasi sistem pandang memungkinkan kita mengubah atau menyesuaikan komposisi pemandangan.
     b. Memudahkan Simetri: Dengan melihat dari sudut pandang yang berbeda, kita dapat lebih mudah mengidentifikasi simetri pada objek.
     b. Melihat dari Berbagai Sudut: Transformasi ini memungkinkan kita melihat objek dari berbagai sudut, membantu kita memahami struktur dan detailnya.
     c. Memindahkan Objek: Kita dapat memindahkan satu atau beberapa objek dari satu tempat ke tempat lain dengan mengubah sistem pandang.
   - **Contoh Transformasi Sistem Pandang:**
     a. Rotasi: Mengubah sudut pandang objek dengan memutarinya sekitar sumbu tertentu.
     b. Translasi: Memindahkan objek secara sejajar dengan sumbu tertentu.

c. Perspektif: Menggunakan sudut pandang yang berbeda untuk menggambarkan objek dengan lebih akurat.

3. **Proyeksi**
   - Proyeksi dalam grafik komputer adalah proses mengonversi objek 3D menjadi representasi 2D. Ini mencakup dua jenis utama proyeksi: proyeksi paralel dan proyeksi perspektif, yang masing-masing memiliki subkategori dan aplikasi spesifik.
   - **Proyeksi Paralel**
     Dalam proyeksi paralel, garis-garis proyeksi tetap sejajar satu sama lain. Terdapat dua subtipe utama:
     a. Proyeksi Ortografis: Di mana garis proyeksi tegak lurus terhadap bidang pandang. Ini sering digunakan dalam aplikasi CAD dan teknik, karena menjaga ukuran objek yang sebenarnya.
     b. Proyeksi Oblique: Di mana garis proyeksi miring terhadap bidang pandang. Ini memberikan pandangan yang lebih lengkap dari suatu objek, sering digunakan dalam ilustrasi teknik.
   - **Proyeksi Perspektif**
     Proyeksi perspektif membuat objek terlihat lebih realistis dengan mensimulasikan cara mata manusia melihat dunia. Dalam proyeksi ini, garis-garis proyeksi bertemu di satu titik yang disebut titik hilang. Ada tiga subtipe berdasarkan jumlah titik hilang:
     a. Proyeksi Satu Titik: Menggunakan satu titik hilang, biasanya untuk menggambarkan objek yang menghadapi pengamat secara langsung.
     b. Proyeksi Dua Titik: Menggunakan dua titik hilang, sering digunakan untuk menggambarkan objek di sudut.
     c. Proyeksi Tiga Titik: Menggunakan tiga titik hilang, memberikan pandangan yang paling realistis, digunakan untuk menggambarkan objek yang dilihat dari sudut tinggi atau rendah.
   - **Perbandingan dan Aplikasi**
     a. Proyeksi Paralel: Lebih cocok untuk aplikasi teknik dan CAD karena mempertahankan dimensi dan bentuk objek yang sebenarnya.
     b. Proyeksi Perspektif: Lebih cocok untuk aplikasi visual dan animasi karena memberikan kesan kedalaman dan realisme.
   - Proyeksi digunakan dalam berbagai bidang, termasuk desain, arsitektur, dan medis. Dalam desain dan arsitektur, proyeksi membantu menggambarkan dan merencanakan struktur dengan tepat. Dalam bidang medis, proyeksi seperti sinar-X digunakan untuk diagnosis penyakit.

4. **View volume**
   - Konsep: View volume adalah ruang 3D yang terlihat oleh kamera atau pandangan dari titik pengamat.
   - Bentuk: Dalam proyeksi perspektif, view volume berbentuk piramida dengan titik puncak di posisi pengamat dan basis yang terletak di bidang pandang (view plane). Dalam proyeksi paralel, view volume berbentuk kubus.
   - Penggunaan: Digunakan untuk menentukan bagian mana dari dunia 3D yang akan ditampilkan pada layar, sehingga objek di luar view volume tidak perlu diproses lebih lanjut.

5. **Set up proyeksi perspektif**
   - Proyeksi Perspektif: Ini adalah metode proyeksi yang memetakan titik-titik dari ruang 3D ke bidang 2D dengan memperhitungkan jarak dari pengamat, sehingga menghasilkan efek realistis di mana objek yang lebih dekat dengan pengamat tampak lebih besar daripada objek yang lebih jauh.
   - Pusat Proyeksi (COP): Titik di mana semua garis proyeksi bertemu, seringkali berada di posisi mata atau kamera.
   - Transformasi: Titik-titik dalam ruang 3D ditransformasikan menggunakan matriks transformasi perspektif yang mengubah koordinat 3D menjadi koordinat 2D.
   - Efek Realistis: Proyeksi perspektif menghasilkan gambar yang tampak realistis karena mempertimbangkan perspektif manusia, di mana objek yang lebih jauh terlihat lebih kecil.

6. **Clipping 3D**
   - Konsep: Clipping adalah proses memotong bagian-bagian objek yang berada di luar view volume sehingga hanya bagian dalam view volume yang diproses dan ditampilkan.
   - Clipping Planes: View volume dibatasi oleh sejumlah bidang (planes), biasanya enam dalam sistem koordinat kamera (depan, belakang, kiri, kanan, atas, bawah).
   - Algoritma: Algoritma clipping, seperti algoritma Sutherland-Hodgman atau Cohen-Sutherland, digunakan untuk menentukan bagian mana dari objek yang harus dipotong.
   - Efisiensi: Clipping meningkatkan efisiensi rendering dengan mengabaikan objek atau bagian objek yang tidak terlihat oleh pengamat.