# LAPORAN PRAKTIKUM

**GRAFIKA KOMPUTER**

**(Dosen : *Rio Priantama S.T., M.T.I.*)**

**Modul 5**


**Nama : Muhammad Rizal Nurfirdaus**

**NIM : 20230810088**

**Kelas : TINFC-2023-04**

**TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS KUNINGAN**

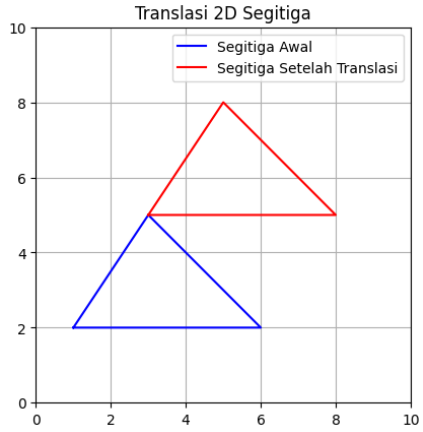
**Praktikum**

1. Praktikum 1 : Translasi Objek Segitiga
   Source Code:

```
# Praktikum 1
# Import library yang diperlukan
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar segitiga
def gambar_segitiga(titik, warna='b', label=None):
    segitiga = np.array(titik)
    # Tambahkan titik pertama ke akhir untuk menutup segitiga
    segitiga = np.vstack([segitiga, segitiga[0]])
    plt.plot(segitiga[:, 0], segitiga[:, 1], color=warna, label=label)

# Fungsi untuk melakukan translasi
def translasi(titik, vektor):
    # Menambahkan vektor translasi ke semua titik segitiga
    titik_baru = [(X + vektor[0], Y + vektor[1]) for X, Y in titik]
    return titik_baru

# Titik-titik awal segitiga
segitiga_awal = [[1, 2], [3, 5], [6, 2]]

# Vektor translasi
vektor_translasi = [2, 3]

# Translasi segitiga
segitiga_tertranslasi = translasi(segitiga_awal, vektor_translasi)

# Plot segitiga sebelum dan sesudah translasi
plt.figure()

# Gambar segitiga awal
gambar_segitiga(segitiga_awal, warna='b', label='Segitiga Awal')

# Gambar segitiga setelah translasi
gambar_segitiga(segitiga_tertranslasi, warna='r', label='Segitiga
Setelah Translasi')

# Menampilkan grid dan setting sumbu
plt.grid(True)
plt.xlim(0, 10)
plt.ylim(0, 10)
plt.gca().set_aspect('equal', adjustable='box')
plt.legend()
plt.title('Translasi 2D Segitiga')
```

```
# Tampilkan plot
plt.show()
```

Hasil Run:




Analisis: Pada program ini, sebuah segitiga didefinisikan oleh tiga titik awal yaitu (1,2), (3,5), dan (6,2). Program kemudian menerapkan translasi menggunakan vektor (2,3), yang berarti seluruh titik segitiga digeser 2 satuan ke kanan dan 3 satuan ke atas. Proses translasi dilakukan dengan menambahkan komponen vektor translasi ke setiap titik segitiga, sehingga menghasilkan segitiga baru dengan posisi yang lebih tinggi dan lebih ke kanan dari posisi awalnya.

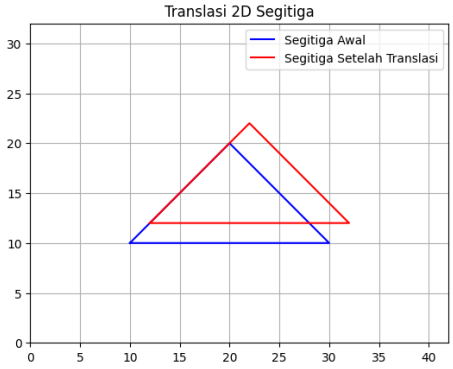
Fungsi gambar_segitiga() digunakan untuk menggambar bentuk segitiga dengan menutup poligon secara otomatis, sedangkan fungsi translasi() bertugas menghitung koordinat baru setelah translasi. Grafik yang ditampilkan memperlihatkan dua segitiga: segitiga awal berwarna biru dan segitiga hasil translasi berwarna merah. Perbandingan ini menunjukkan bahwa translasi hanya menggeser posisi objek tanpa mengubah bentuk, ukuran, atau orientasinya. Penggunaan grid, batas sumbu, serta aspect ratio yang sama memungkinkan visualisasi translasi terlihat jelas dan proporsional.

2. Praktikum 2 : Translasi Objek Segitiga Dengan Input Dinamis
   Source Code:

```
# Praktikum 2
# Import library yang diperlukan
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar segitiga
def gambar_segitiga(titik, warna='b', label=None):
    segitiga = np.array(titik)
    # Tambahkan titik pertama ke akhir untuk menutup segitiga
    segitiga = np.vstack([segitiga, segitiga[0]])
    plt.plot(segitiga[:, 0], segitiga[:, 1], color=warna, label=label)

# Fungsi untuk melakukan translasi
def translasi(titik, vektor):
```

```
    # Menambahkan vektor translasi ke semua titik segitiga
    titik_baru = [(X + vektor[0], Y + vektor[1]) for X, Y in titik]
    return titik_baru

# Fungsi utama untuk input dinamis dan translasi
def main():
    # Input titik segitiga dari pengguna
    print("Masukkan koordinat segitiga (dalam format x, y):")
    x1, y1 = map(float, input("Titik 1 (x1, y1): ").split(','))
    x2, y2 = map(float, input("Titik 2 (x2, y2): ").split(','))
    x3, y3 = map(float, input("Titik 3 (x3, y3): ").split(','))

    # Titik segitiga awal
    segitiga_awal = [[x1, y1], [x2, y2], [x3, y3]]

    # Input vektor translasi dari pengguna
    print("Masukkan vektor translasi (dalam format dx, dy):")
    dx, dy = map(float, input("Vektor Translasi (dx, dy):
").split(','))

    # Translasi segitiga
    segitiga_tertranslasi = translasi(segitiga_awal, (dx, dy))

    # Plot segitiga sebelum dan sesudah translasi
    plt.figure()

    # Gambar segitiga awal
    gambar_segitiga(segitiga_awal, warna='b', label='Segitiga Awal')

    # Gambar segitiga setelah translasi
    gambar_segitiga(segitiga_tertranslasi, warna='r', label='Segitiga
Setelah Translasi')

    # Menampilkan grid dan setting sumbu
    plt.grid(True)
    plt.xlim(min(x1, x2, x3, x1+dx, x2+dx, x3+dx) - 10, max(x1, x2, x3,
x1+dx, x2+dx, x3+dx) + 10)
    plt.ylim(min(y1, y2, y3, y1+dy, y2+dy, y3+dy) - 10, max(y1, y2, y3,
y1+dy, y2+dy, y3+dy) + 10)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.legend()
    plt.title('Translasi 2D Segitiga')

    # Tampilkan plot
    plt.show()

# Panggil fungsi utama
main()
```

Hasil Run:




Analisis: Program ini memungkinkan pengguna untuk memasukkan tiga titik koordinat segitiga dan vektor translasi secara dinamis melalui input. Setiap titik segitiga awal disimpan, kemudian fungsi translasi menambahkan nilai dx dan dy ke seluruh titik sehingga diperoleh posisi baru segitiga setelah dipindahkan. Program kemudian menggambar segitiga awal dan segitiga hasil translasi menggunakan fungsi gambar_segitiga(), yang secara otomatis menutup bentuk segitiga dengan menghubungkan titik terakhir kembali ke titik pertama.

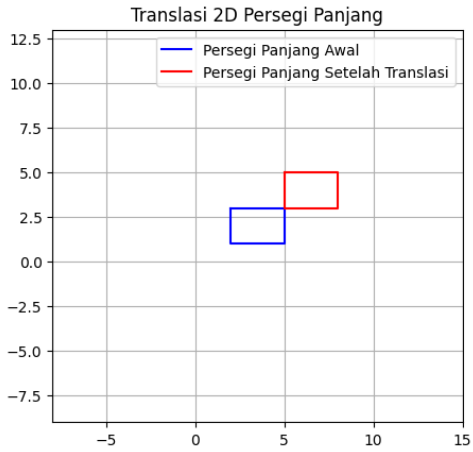
Pengaturan sumbu menggunakan rentang dinamis berdasarkan titik awal dan titik hasil translasi, memastikan kedua segitiga terlihat jelas dalam satu tampilan grafik. Grid, aspect ratio yang disamakan, serta legenda membantu memperjelas perbedaan antara posisi sebelum dan sesudah translasi. Secara keseluruhan, program sudah efektif dan interaktif karena menerima input pengguna serta menampilkan visualisasi perubahan posisi objek akibat translasi dalam koordinat 2D.

3. Praktikum 3 : Translasi Objek Persegi Panjang Dengan Input Dinamis
   Source Code:

```
# Praktikum 3
# Import library yang diperlukan
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar persegi panjang
def gambar_persegi_panjang(titik, warna='b', label=None):
    persegi_panjang = np.array(titik)
    # Tambahkan titik pertama ke akhir untuk menutup persegi panjang
    persegi_panjang = np.vstack([persegi_panjang, persegi_panjang[0]])
    plt.plot(persegi_panjang[:, 0], persegi_panjang[:, 1], color=warna,
label=label)

# Fungsi untuk melakukan translasi
def translasi(titik, vektor):
    # Menambahkan vektor translasi ke semua titik persegi panjang
    titik_baru = [(X + vektor[0], Y + vektor[1]) for X, Y in titik]
    return titik_baru

# Fungsi utama untuk input dinamis dan translasi
```

```
def main():
    # Input titik persegi panjang dari pengguna
    print("Masukkan koordinat persegi panjang (dalam format x, y):")
    x1, y1 = map(float, input("Titik 1 (x1, y1): ").split(','))
    x2, y2 = map(float, input("Titik 2 (x2, y2): ").split(','))
    x3, y3 = map(float, input("Titik 3 (x3, y3): ").split(','))
    x4, y4 = map(float, input("Titik 4 (x4, y4): ").split(','))

    # Titik persegi panjang awal
    persegi_panjang_awal = [[x1, y1], [x2, y2], [x3, y3], [x4, y4]]

    # Input vektor translasi dari pengguna
    print("Masukkan vektor translasi (dalam format dx, dy):")
    dx, dy = map(float, input("Vektor Translasi (dx, dy):
").split(','))

    # Translasi persegi panjang
    persegi_panjang_tertranslasi = translasi(persegi_panjang_awal, (dx,
dy))

    # Plot persegi panjang sebelum dan sesudah translasi
    plt.figure()

    # Gambar persegi panjang awal
    gambar_persegi_panjang(persegi_panjang_awal, warna='b',
label='Persegi Panjang Awal')

    # Gambar persegi panjang setelah translasi
    gambar_persegi_panjang(persegi_panjang_tertranslasi, warna='r',
label='Persegi Panjang Setelah Translasi')

    # Menampilkan grid dan setting sumbu
    plt.grid(True)
    plt.xlim(min(x1, x2, x3, x4) - 10, max(x1, x2, x3, x4) + 10)
    plt.ylim(min(y1, y2, y3, y4) - 10, max(y1, y2, y3, y4) + 10)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.legend()
    plt.title('Translasi 2D Persegi Panjang')

    # Tampilkan plot
    plt.show()

# Panggil fungsi utama
main()
```

Hasil Run :




Analisis: Program ini bertujuan menggambarkan translasi 2D pada sebuah persegi panjang berdasarkan input titik yang diberikan pengguna. Pengguna memasukkan empat titik sudut persegi panjang, kemudian program menyusunnya menjadi sebuah bentuk poligon tertutup. Setelah itu, vektor translasi berupa nilai dx dan dy dimasukkan untuk menentukan pergeseran posisi persegi panjang. Fungsi translasi() bekerja dengan menambahkan nilai dx dan dy ke tiap titik, menghasilkan persegi panjang baru yang bergeser dari posisi awalnya.

Visualisasi dilakukan dengan menggambar persegi panjang awal berwarna biru dan persegi panjang hasil translasi berwarna merah. Penggunaan plt.vstack() memastikan bentuk tertutup sehingga persegi panjang tergambar utuh. Rentang sumbu (xlim dan ylim) diatur secara dinamis berdasarkan titik awal untuk memastikan grafik selalu menampilkan seluruh bentuk secara proporsional. Grid dan pengaturan aspect ratio membantu menjaga visualisasi agar akurat dan tidak terdistorsi. Secara keseluruhan, program ini memberikan representasi jelas tentang bagaimana translasi memindahkan sebuah objek tanpa mengubah bentuk, ukuran, atau orientasinya.

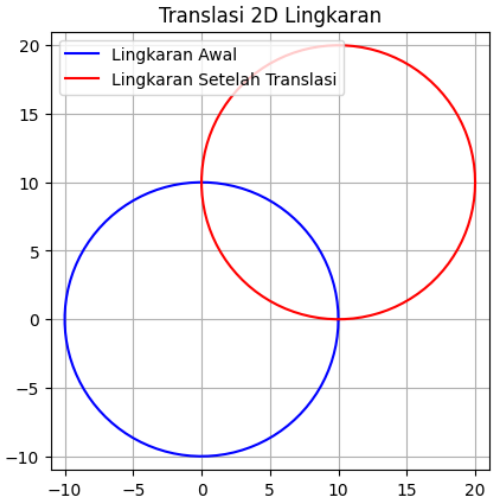
4. Praktikum 4 : Translasi Objek Lingkaran

```
# Praktikum 4
# Import library yang diperlukan
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar lingkaran
def gambar_lingkaran(tengah, radius, warna='b', label=None):
    # Buat data untuk lingkaran
    theta = np.linspace(0, 2 * np.pi, 100)
    X = tengah[0] + radius * np.cos(theta)
    Y = tengah[1] + radius * np.sin(theta)
    plt.plot(X, Y, color=warna, label=label)

# Fungsi untuk melakukan translasi
def translasi(tengah, vektor):
```

```
    # Menambahkan vektor translasi ke pusat lingkaran
    tengah_baru = [tengah[0] + vektor[0], tengah[1] + vektor[1]]
    return tengah_baru

# Fungsi utama untuk input dinamis dan translasi
def main():
    # Input pusat lingkaran dan radius dari pengguna
    print("Masukkan Pusat Lingkaran (dalam format x, y):")
    x, y = map(float, input("Pusat Lingkaran (x, y): ").split(','))

    radius = float(input("Masukkan Radius Lingkaran: "))

    # Pusat lingkaran awal
    pusat_lingkaran_awal = (x, y)

    # Input vektor translasi dari pengguna
    print("Masukkan vektor translasi (dalam format dx, dy):")
    dx, dy = map(float, input("Vektor Translasi (dx, dy):
").split(','))

    # Translasi pusat lingkaran
    pusat_lingkaran_tertranslasi = translasi(pusat_lingkaran_awal, (dx,
dy))

    # Plot lingkaran sebelum dan sesudah translasi
    plt.figure()

    # Gambar lingkaran awal
    gambar_lingkaran(pusat_lingkaran_awal, radius, warna='b',
label='Lingkaran Awal')

    # Gambar lingkaran setelah translasi
    gambar_lingkaran(pusat_lingkaran_tertranslasi, radius, warna='r',
label='Lingkaran Setelah Translasi')

    # Menampilkan grid dan setting sumbu
    # Menentukan batas plot agar kedua lingkaran terlihat
    min_x = min(x, x + dx)
    max_x = max(x, x + dx)
    min_y = min(y, y + dy)
    max_y = max(y, y + dy)

    # Menambahkan margin yang cukup (radius + 1)
    plt.xlim(min_x - radius - 1, max_x + radius + 1)
    plt.ylim(min_y - radius - 1, max_y + radius + 1)
    plt.grid(True)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.legend()
```

```
    plt.title('Translasi 2D Lingkaran')

    # Tampilkan plot
    plt.show()

# Panggil fungsi utama
main()
```

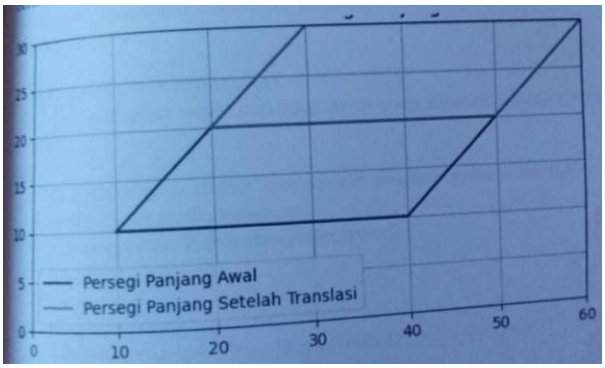
Hasil Run :




Analisis : Program tersebut berfungsi untuk menggambar sebuah lingkaran awal kemudian memindahkan posisinya menggunakan operasi translasi 2D berdasarkan input pengguna. Proses dimulai dengan meminta pengguna memasukkan koordinat pusat lingkaran dan nilai radius. Selanjutnya, pengguna diminta memberikan vektor translasi dalam bentuk (dx, dy), yang digunakan untuk menghitung posisi pusat lingkaran baru melalui penambahan nilai translasi pada koordinat awal. Program kemudian menggambar dua lingkaran: lingkaran awal berwarna biru dan lingkaran setelah translasi berwarna merah. Untuk memastikan kedua lingkaran tampil jelas, batas sumbu ditentukan secara dinamis berdasarkan posisi awal dan posisi hasil translasi, ditambah margin sebesar radius lingkaran. Selain itu, plot diberi grid, rasio sumbu dibuat seimbang agar lingkaran tidak terdistorsi, dan legend ditampilkan untuk memperjelas perbedaan lingkaran sebelum dan sesudah translasi. Hasil akhirnya adalah visualisasi yang menunjukkan bagaimana translasi memindahkan lingkaran tanpa mengubah bentuk atau ukurannya.

## LATIHAN/TUGAS

1. Buatlah kode program sederhana untuk menggambar jajargenjang seperti gambar berikut:




Source Code:

```
# Tugas
import matplotlib.pyplot as plt
import numpy as np

# --- 1. Definisikan Koordinat Jajargenjang ---

# Koordinat Jajargenjang Awal (Persegi Panjang Awal)
# Titik-titik yang terbaca dari grafik:
# A (10, 10), B (40, 12), C (40+10=50, 12+20=32), D (10+10=20,
10+20=30)
# (Menggunakan asumsi bahwa jajargenjang awal yang dimaksud adalah yang
berwarna gelap)
x_awal = [10, 40, 50, 20, 10]  # Tambahkan (10, 10) di akhir untuk
menutup bentuk
y_awal = [10, 12, 32, 30, 10]

# Koordinat Jajargenjang Setelah Translasi
# Titik-titik yang terbaca dari grafik:
# A' (10, 22), B' (40, 24), C' (50, 44), D' (20, 42)
# (Translasi yang terjadi adalah vektor T = (0, 12) atau T = (0, 10).
Mari kita gunakan yang terlihat jelas: A'=(10, 22))
x_translasi = [10, 40, 50, 20, 10]
y_translasi = [22, 24, 44, 42, 22] # Koordinat y ini melebihi batas 30
di gambar,
                                   # namun berdasarkan garis
horizontal y=22, kita asumsikan pergeseran y
                                   # yang lebih kecil untuk
Jajargenjang Setelah Translasi (garis tipis).

# Berdasarkan gambar (garis tipis): Jajargenjang setelah translasi
memiliki batas y sekitar 22.
# Asumsi baru berdasarkan garis horizontal y=22 (garis tipis pada
gambar):
# A'' (10, 22), B'' (40, 22), C'' (50, 32), D'' (20, 32)
```

```
x_translasi_final = [10, 40, 50, 20, 10]
y_translasi_final = [22, 22, 32, 32, 22]


# --- 2. Plotting Grafik ---

plt.figure(figsize=(8, 6))

# Plot Jajargenjang Awal (Garis tebal)
plt.plot(x_awal, y_awal, color='navy', linewidth=2, label='Persegi
Panjang Awal')

# Plot Jajargenjang Setelah Translasi (Garis tipis yang lebih
lurus/horizontal)
plt.plot(x_translasi_final, y_translasi_final, color='darkgreen',
linewidth=1, label='Persegi Panjang Setelah Translasi')

# --- 3. Pengaturan Tampilan ---

# Mengatur batas sumbu x dan y sesuai dengan gambar
plt.xlim(0, 60)
plt.ylim(0, 35)
plt.xticks(np.arange(0, 61, 10))
plt.yticks(np.arange(0, 31, 5))

# Menampilkan grid
plt.grid(True)

# Menampilkan legenda
plt.legend(loc='lower left')

# Memberi judul (opsional)
plt.title('Translasi Jajargenjang 2D')

# Tampilkan plot
plt.show()
```

Hasil Run :

Analisis: Program ini bertujuan menampilkan dua jajargenjang: posisi awal (garis tebal/gelap) dan posisi setelah translasi (garis tipis/hijau). Anda membaca koordinat dari gambar dan menuliskannya sebagai daftar x/y untuk masing-masing poligon, lalu memplot kedua poligon tersebut. Pengaturan sumbu, ticks, grid, judul, dan legenda sudah dipasang sehingga visual menjadi informatif.

Ada satu titik penting yang harus diperhatikan: **ketepatan pengambilan koordinat dan konsistensi vektor translasi**. Pada kode Anda muncul dua langkah penarikan koordinat berbeda: pertama Anda menuliskan A'=(10,22), B'=(40,24), C'=(50,44), D'=(20,42) (ini konsisten dengan translasi vektor $T = (0,12)$jika A dari 10→22), tetapi kemudian Anda memilih x_translasi_final/y_translasi_final di mana sisi atas dipaksa menjadi y = [22,22,32,32,22] ini **tidak konsisten** dengan vektor translasi yang disimpulkan sebelumnya (karena B seharusnya menjadi 24, bukan 22; C seharusnya 44, bukan 32). Akibatnya plot final mencerminkan asumsi translasi yang berbeda dari perhitungan awal.

Untuk membuatnya konsisten: tentukan satu pasangan titik korespondensi (mis. A→A') dan hitung vektor translasi $T = (t_x, t_y)$dari sana, lalu terapkan ke semua titik awal. Dari data awal Anda:

- A awal = (10, 10) → A' yang Anda baca = (10, 22) → sehingga $T = (0,12)$.

Jika kita gunakan $T = (0,12)$secara konsisten, koordinat setelah translasi menjadi:

- A' = (10, 10) + (0,12) = **(10, 22)**
- B' = (40, 12) + (0,12) = **(40, 24)**
- C' = (50, 32) + (0,12) = **(50, 44)**
- D' = (20, 30) + (0,12) = **(20, 42)**

Perhatikan bahwa beberapa titik (mis. y = 44) berpotensi berada di luar batas sumbu ylim(0,35) yang Anda tetapkan itulah mengapa Anda melihat perbedaan visual antara asumsi pertama dan plot akhir. Jadi jika ingin memvisualisasikan $T = (0,12)$dengan benar, perlu memperbesar ylim agar semua titik terlihat.

Rekomendasi perbaikan praktis:

1. **Ambil satu vektor translasi** dari pasangan titik yang jelas (mis. A→A') dan gunakan ke seluruh titik supaya konsisten.
2. **Jangan memaksa** nilai y hasil translasi ke angka yang "lebih pas" pada plot; sesuaikan batas sumbu (xlim, ylim) agar benar-benar menampilkan semua titik.
3. Tambahkan plt.gca().set_aspect('equal', adjustable='box') (jika belum) agar bentuk tidak terdistorsi sudah Anda pakai di contoh lain, baik untuk dipertahankan.
4. Untuk pemeriksaan cepat, tambahkan anotasi koordinat titik di plot (mis. plt.text(x,y,f'({x},{y})')) sehingga pembacaan manual menjadi lebih mudah dan sedikit kesalahan dalam ekstraksi dapat dikurangi.

Ringkasnya: kode plotting sudah benar secara struktur; masalah utama adalah **ketidakkonsistenan asumsi koordinat/vektor translasi** saat Anda menyalin dari

gambar. Pilih satu vektor translasi yang konsisten (mis. $T = (0,12)$), terapkan ke semua vertex, dan sesuaikan batas plot agar hasil visual sesuai perhitungan.