LAPORAN PRAKTIKUM
GRAFIKA KOMPUTER
(Dosen : *Rio Priantama S.T., M.T.I.*)

Modul 9

Nama : Muhammad Rizal Nurfirdaus

NIM : 20230810088

Kelas : TINFC-2023-04

TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS KUNINGAN

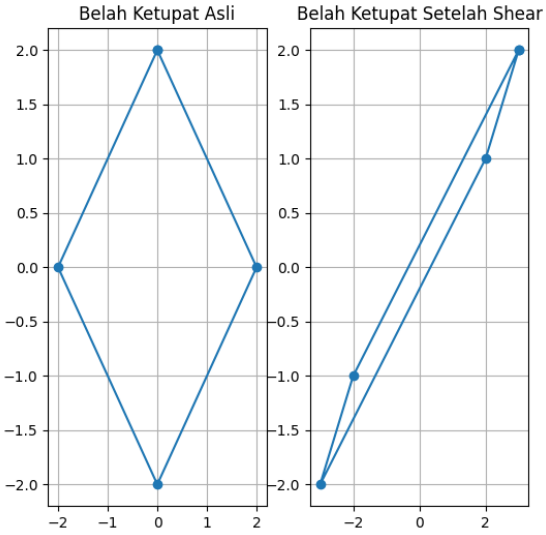
Praktikum

1. Praktikum 1 : Shear Objek Belah Ketupat
   Source Code:

```
# Praktikum 1
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar belah ketupat sebelum dan sesudah shear
def plot_belah_ketupat(original_points, sheared_points):
    # Gambar belah ketupat sebelum shear
    plt.figure(figsize=(6, 6))
    plt.subplot(1, 2, 1)
    plt.plot(*zip(*original_points, original_points[0]), marker='o')
    plt.title('Belah Ketupat Asli')
    plt.grid(True)

    # Gambar belah ketupat setelah shear
    plt.subplot(1, 2, 2)
    plt.plot(*zip(*sheared_points, sheared_points[0]), marker='o')
    plt.title('Belah Ketupat Setelah Shear')
    plt.grid(True)

    plt.show()

# Fungsi untuk melakukan shear pada objek belah ketupat
def shear_belah_ketupat(points, shear_x, shear_y):
    # Matriks transformasi shear
    shear_matrix = np.array([[1, shear_x],
                             [shear_y, 1]])

    # Mengalikan setiap titik dengan matriks shear
    sheared_points = []
    for point in points:
        new_point = np.dot(shear_matrix, point)
        sheared_points.append(new_point)

    return sheared_points

# Input dinamis dari user
print("Masukkan faktor shear horizontal (Shear X): ")
shear_x = float(input()) # Shear X (horizontal)
print("Masukkan faktor shear vertikal (Shear Y): ")
shear_y = float(input()) # Shear Y (vertikal)

# Koordinat asli belah ketupat
belah_ketupat_points = np.array([[0, 2], [2, 0], [0, -2], [-2, 0]])
```

```
# Lakukan transformasi shear
sheared_points = shear_belah_ketupat(belah_ketupat_points, shear_x,
shear_y)

# Plot hasilnya
plot_belah_ketupat(belah_ketupat_points, sheared_points)
```

Hasil Run:



Analisis: Praktikum ini berfokus pada penerapan transformasi **shear** pada objek poligon simetris berupa belah ketupat. Objek didefinisikan oleh empat titik koordinat, yaitu $(0, 2)$, $(2, 0)$, $(0, -2)$, dan $(-2, 0)$, yang secara geometris membentuk sebuah belah ketupat dengan pusat tepat di titik origin $(0, 0)$. Penempatan titik-titik ini pada keempat arah sumbu menunjukkan bahwa objek tersebar di seluruh kuadran bidang Kartesius, sehingga sangat sesuai untuk menganalisis efek transformasi affine secara menyeluruh.

Mekanisme transformasi dilakukan melalui sebuah fungsi bernama shear_belah_ketupat, yang menerapkan matriks transformasi berukuran 2×2. Setiap vektor posisi titik (x, y)dikalikan dengan matriks shear $\begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix}$, di mana sh_xdan sh_ymasing-masing merepresentasikan faktor shear horizontal dan vertikal. Proses perkalian matriks ini menghasilkan koordinat baru untuk setiap titik, yang kemudian membentuk objek hasil transformasi. Karena matriks shear termasuk dalam transformasi affine, hubungan kesejajaran antar sisi objek tetap dipertahankan.

Secara visual, penerapan shear pada belah ketupat ini menyebabkan objek tampak miring ke arah tertentu sesuai nilai faktor shear yang diberikan. Meskipun sudut-sudut internal belah ketupat berubah, sisi-sisi yang sejajar sebelum transformasi tetap sejajar setelah transformasi. Hal ini menegaskan sifat utama transformasi affine, yaitu mempertahankan kesejajaran garis dan rasio tertentu, sehingga luas area objek secara keseluruhan tidak mengalami perubahan. Efek ini memberikan pemahaman yang jelas bahwa shear memodifikasi bentuk tanpa melakukan rotasi maupun skala.

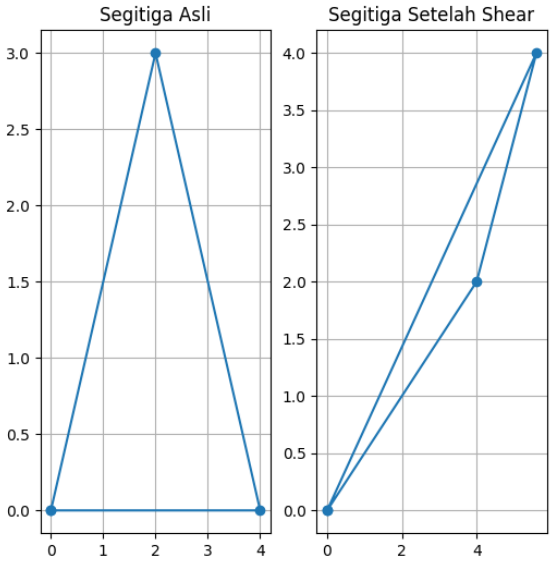
Selain itu, penggunaan fungsi zip(*points) dalam proses visualisasi memiliki peran penting. Fungsi ini digunakan untuk memisahkan pasangan koordinat (x, y)menjadi dua deretan terpisah, yaitu deretan nilai $x$dan deretan nilai y. Pemisahan ini diperlukan agar data koordinat dapat dibaca dan diplot dengan benar oleh pustaka **matplotlib**, sehingga bentuk belah ketupat sebelum dan sesudah transformasi dapat ditampilkan secara akurat.

2. Praktikum 2 : Shear Objek Segitiga
   Source Code:

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

# Fungsi untuk menggambar segitiga sebelum dan sesudah shear
def plot_segitiga(original_points, sheared_points):
    # Gambar segitiga sebelum shear
    plt.figure(figsize=(6, 6))
    plt.subplot(1, 2, 1)
    plt.plot(*zip(*original_points, original_points[0]), marker='o')
    plt.title('Segitiga Asli')
    plt.grid(True)

    # Gambar segitiga setelah shear
    plt.subplot(1, 2, 2)
    plt.plot(*zip(*sheared_points, sheared_points[0]), marker='o')
    plt.title('Segitiga Setelah Shear')
    plt.grid(True)

    plt.show()

# Fungsi untuk melakukan shear pada objek segitiga
def shear_segitiga(points, shear_x, shear_y):
    # Matriks transformasi shear
    shear_matrix = np.array([[1, shear_x],
                             [shear_y, 1]])

    # Mengalikan setiap titik dengan matriks shear
    sheared_points = []
    for point in points:
        new_point = np.dot(shear_matrix, point)
        sheared_points.append(new_point)

    return sheared_points

# Input dinamis dari user
print("Masukkan faktor shear horizontal (Shear X): ")
shear_x = float(input()) # Shear X (horizontal)
```

```
print("Masukkan faktor shear vertikal (Shear Y): ")
shear_y = float(input()) # Shear Y (vertikal)

# Koordinat asli segitiga
segitiga_points = np.array([[0, 0], [4, 0], [2, 3]])

# Lakukan transformasi shear
sheared_points = shear_segitiga(segitiga_points, shear_x, shear_y)

# Plot hasilnya
plot_segitiga(segitiga_points, sheared_points)
```

Hasil Run:



Analisis: Praktikum kedua memperlihatkan secara jelas bagaimana transformasi **shear** memengaruhi objek 2D yang memiliki jumlah titik ganjil serta posisi yang tidak simetris terhadap titik pusat. Objek yang digunakan adalah sebuah segitiga yang didefinisikan oleh tiga titik koordinat, yaitu $(0, 0)$, $(4, 0)$, dan $(2, 3)$. Pada struktur ini, alas segitiga berada tepat di sepanjang sumbu x, sedangkan titik puncak berada di atas alas, sehingga segitiga memiliki orientasi yang asimetris terhadap sumbu koordinat.

Pada penerapan **shear horizontal** dengan kondisi $sh_y = 0$dan $sh_x \neq 0$, efek transformasi sangat dipengaruhi oleh nilai koordinat $y$setiap titik. Titik $(0, 0)$dan $(4, 0)$tidak mengalami perpindahan karena nilai y-nya sama dengan nol, sehingga tidak terjadi pergeseran pada sumbu x. Sebaliknya, titik puncak $(2, 3)$mengalami pergeseran horizontal sebesar $3 \times sh_x$, karena pergeseran pada shear horizontal bergantung langsung pada nilai y. Akibatnya, bentuk segitiga tampak miring ke samping, namun alas segitiga tetap berada pada posisi semula.

Sementara itu, pada **shear vertikal** dengan kondisi $sh_x = 0$dan $sh_y \neq 0$, titik $(0, 0)$tetap tidak berpindah karena nilai x-nya nol. Namun, titik $(4, 0)$akan mengalami pergeseran ke arah atas atau bawah secara vertikal sesuai nilai sh_y, karena pergeseran pada shear vertikal bergantung pada nilai x. Perubahan ini menyebabkan alas segitiga

yang semula datar di sumbu $x$menjadi miring, sehingga orientasi segitiga secara keseluruhan berubah.

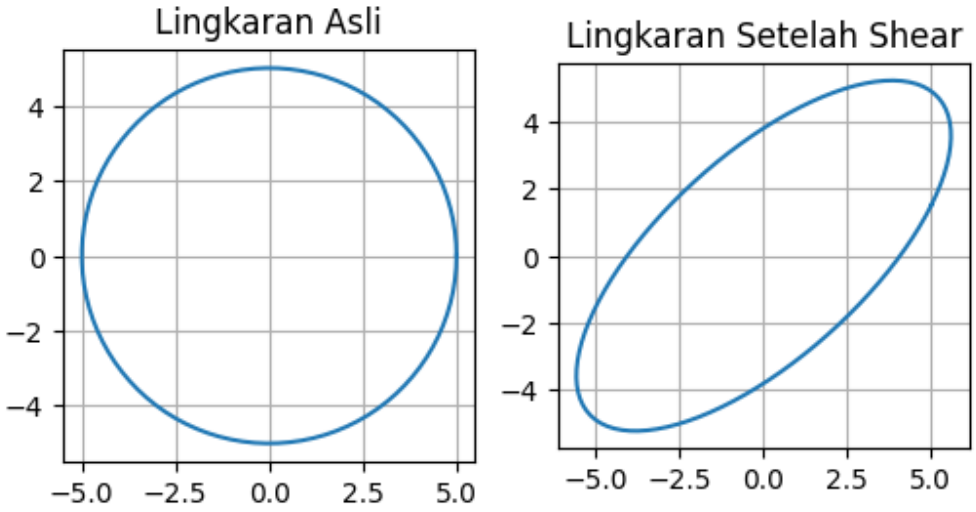
Secara teknis, praktikum ini membuktikan bahwa titik-titik yang berada tepat pada sumbu aksis dengan nilai koordinat nol berperan sebagai **jangkar** dalam transformasi shear yang sejajar dengan sumbu tersebut. Titik-titik ini tidak mengalami perpindahan, sehingga memberikan referensi tetap yang memperjelas arah dan efek deformasi objek akibat penerapan shear.

3. Praktikum 3 : Shear Objek Lingkaran
   Source Code:

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

# Fungsi untuk menggambar Lingkaran sebelum dan sesudah shear
def plot_lingkaran(original_points, sheared_points):
    # Gambar Lingkaran sebelum shear
    plt.figure(figsize=(6, 6))
    plt.subplot(1, 2, 1)
    plt.plot(original_points[0], original_points[1], label='Lingkaran
Asli')
    plt.title('Lingkaran Asli')
    plt.gca().set_aspect('equal', adjustable='box')
    plt.grid(True)

    # Gambar Lingkaran setelah shear
    plt.subplot(1, 2, 2)
    plt.plot(sheared_points[0], sheared_points[1], label='Lingkaran
Setelah Shear')
    plt.title('Lingkaran Setelah Shear')
    plt.gca().set_aspect('equal', adjustable='box')
    plt.grid(True)

    plt.show()

# Fungsi untuk melakukan shear pada Lingkaran
def shear_lingkaran(points, shear_x, shear_y):
    # Matriks transformasi shear
    shear_matrix = np.array([[1, shear_x],
                             [shear_y, 1]])

    # Mengalikan setiap titik Lingkaran dengan matriks shear
    sheared_points = np.dot(shear_matrix, points)

    return sheared_points
```

```
# Input dinamis dari user
print("Masukkan faktor shear horizontal (Shear X): ")
shear_x = float(input()) # Shear X (horizontal)
print("Masukkan faktor shear vertikal (Shear Y): ")
shear_y = float(input()) # Shear Y (vertikal)

# Menghasilkan koordinat Lingkaran asli
theta = np.linspace(0, 2 * np.pi, 100) # Sudut lingkaran
radius = 5 # Jari-jari lingkaran
x = radius * np.cos(theta) # Koordinat X lingkaran
y = radius * np.sin(theta) # Koordinat Y lingkaran

# Gabungkan koordinat X dan Y menjadi matriks
lingkaran_points = np.array([x, y])

# Lakukan transformasi shear
sheared_points = shear_lingkaran(lingkaran_points, shear_x, shear_y)

# Plot hasilnya
plot_lingkaran(lingkaran_points, sheared_points)
```

Hasil Run :



Analisis: Praktikum ketiga merupakan bagian yang paling kompleks karena melibatkan penerapan transformasi **shear** pada objek berbentuk kurva non-linear, yaitu lingkaran. Berbeda dengan poligon yang dibangun dari sejumlah titik sudut tertentu, lingkaran pada praktikum ini direpresentasikan menggunakan fungsi np.linspace(0, 2 * np.pi, 100) untuk menghasilkan 100 nilai sudut yang tersebar merata. Setiap nilai sudut tersebut kemudian dikonversi menjadi koordinat (x, y), sehingga terbentuk sekumpulan titik yang sangat rapat dan mampu merepresentasikan garis lengkung lingkaran secara halus dan kontinu.

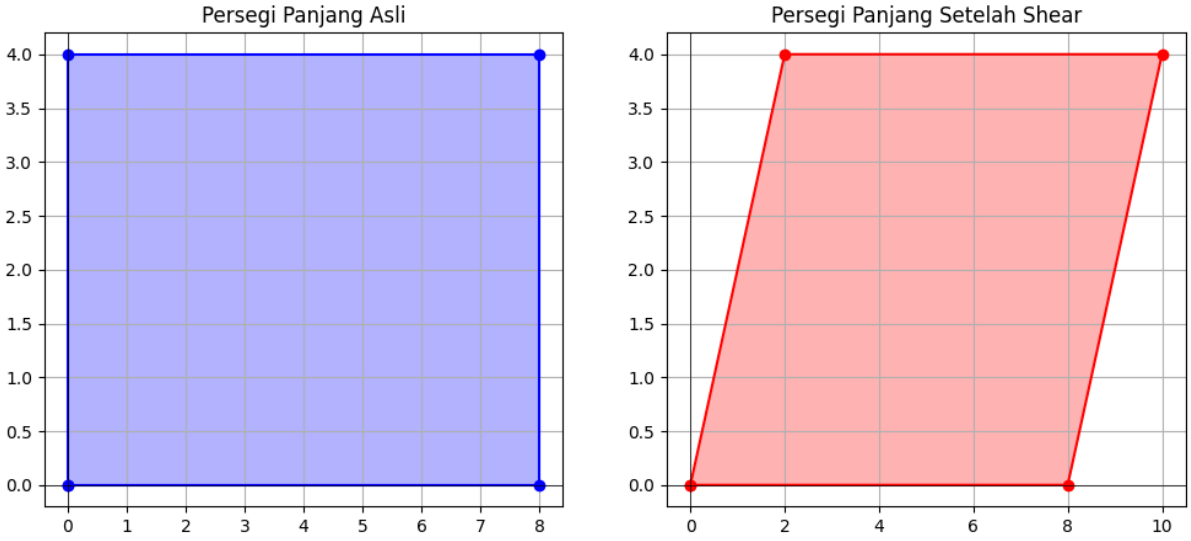
Ketika matriks shear diterapkan pada kumpulan titik lingkaran tersebut, terjadi perubahan geometris yang signifikan, yaitu lingkaran berubah menjadi **elips**. Perubahan ini terjadi karena transformasi shear menyebabkan jarak dari pusat lingkaran ke titik-titik pada tepi tidak lagi seragam di semua arah. Faktor sh_xatau sh_ymenggeser titik-titik berdasarkan koordinat pasangannya, sehingga simetri radial yang dimiliki lingkaran hilang dan bentuk hasil transformasi menjadi elips tanpa melakukan rotasi maupun penskalaan eksplisit.

Dari sisi implementasi, kode pada praktikum ini menunjukkan tingkat efisiensi yang lebih tinggi dibandingkan praktikum pertama dan kedua. Hal ini disebabkan oleh penggunaan operasi vektorisasi melalui np.dot(shear_matrix, points), yang memungkinkan seluruh matriks koordinat diproses sekaligus dalam satu operasi. Pendekatan ini jauh lebih optimal dibandingkan penggunaan perulangan for untuk menghitung koordinat baru satu per satu, baik dari segi kecepatan eksekusi maupun keterbacaan kode.

Selain itu, pengaturan rasio aspek menggunakan plt.gca().set_aspect('equal') memiliki peran yang sangat krusial dalam analisis visual. Pengaturan ini memastikan bahwa skala sumbu $x$dan $y$adalah sama, sehingga lingkaran awal benar-benar tampil proporsional dan tidak tampak lonjong akibat perbedaan skala tampilan. Dengan demikian, perubahan bentuk menjadi elips dapat dipastikan sepenuhnya disebabkan oleh transformasi shear, bukan oleh distorsi visual dari sistem koordinat layar.

LATIHAN/TUGAS

1. Buatlah kode program sederhana untuk menampilkan *shear* untuk objek persegi panjang berikut:

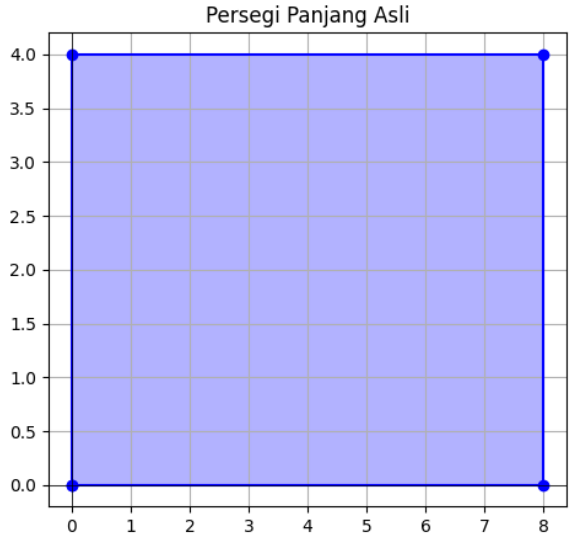


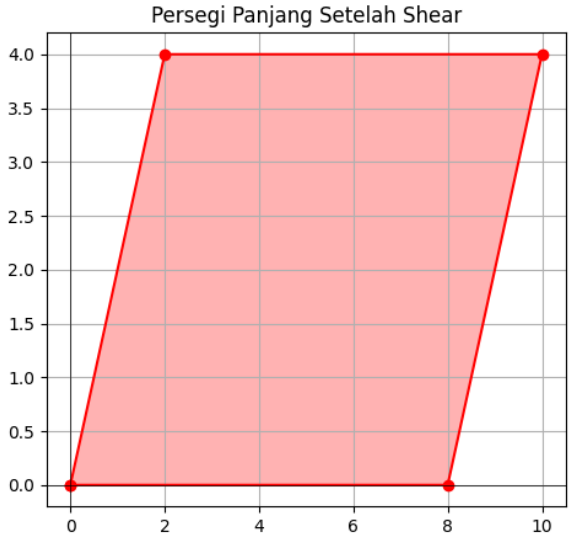
Source Code:

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

# Fungsi untuk menggambar persegi panjang sebelum dan sesudah shear
def plot_persegi_panjang(original_points, sheared_points):
    plt.figure(figsize=(12, 5))

    # Gambar Persegi Panjang Asli
    plt.subplot(1, 2, 1)
    # Menambahkan titik awal di akhir agar garis terhubung (menutup
kurva)
    points_to_plot = np.vstack([original_points, original_points[0]])
    plt.plot(points_to_plot[:, 0], points_to_plot[:, 1], 'b-o',
label='Asli')
    plt.fill(points_to_plot[:, 0], points_to_plot[:, 1], 'b',
alpha=0.3)
    plt.title('Persegi Panjang Asli')
    plt.grid(True)
    plt.axhline(0, color='black',linewidth=0.5)
    plt.axvline(0, color='black',linewidth=0.5)

    # Gambar Persegi Panjang Setelah Shear
    plt.subplot(1, 2, 2)
    sheared_to_plot = np.vstack([sheared_points, sheared_points[0]])
    plt.plot(sheared_to_plot[:, 0], sheared_to_plot[:, 1], 'r-o',
label='Shear')
    plt.fill(sheared_to_plot[:, 0], sheared_to_plot[:, 1], 'r',
alpha=0.3)
```

```
    plt.title('Persegi Panjang Setelah Shear')
    plt.grid(True)
    plt.axhline(0, color='black',linewidth=0.5)
    plt.axvline(0, color='black',linewidth=0.5)

    plt.show()

# Fungsi transformasi shear
def apply_shear(points, sh_x, sh_y):
    # Matriks transformasi shear 2D
    shear_matrix = np.array([[1, sh_x],
                             [sh_y, 1]])

    # Menghitung koordinat baru
    return np.dot(points, shear_matrix.T)

# 1. Tentukan koordinat titik persegi panjang (x, y)
# Sesuai gambar: (0,0), (8,0), (8,4), (0,4)
rect_points = np.array([
    [0, 0],
    [8, 0],
    [8, 4],
    [0, 4]
])

# 2. Input faktor shear
print("--- Transformasi Shear Persegi Panjang ---")
sh_x = float(input("Masukkan faktor shear horizontal (Shear X): "))
sh_y = float(input("Masukkan faktor shear vertikal (Shear Y): "))

# 3. Eksekusi transformasi
sheared_rect = apply_shear(rect_points, sh_x, sh_y)

# 4. Tampilkan hasil
plot_persegi_panjang(rect_points, sheared_rect)
```

Hasil Run :





Analisis: Berdasarkan visualisasi hasil transformasi pada gambar tugas, terlihat bahwa perubahan bentuk objek hanya terjadi pada arah horizontal. Titik-titik sudut bagian bawah, yaitu titik Adan B, tetap berada pada posisi semula, sedangkan titik-titik sudut bagian atas, yaitu titik Cdan D, mengalami pergeseran ke arah kanan. Fenomena ini menunjukkan bahwa transformasi yang diterapkan adalah **shear horizontal** tanpa adanya komponen shear vertikal. Secara matematis, pergeseran titik $C(8,4)$ke posisi baru $C'(10,4)$dapat dianalisis menggunakan persamaan transformasi shear horizontal $x' = x + (sh_x \cdot y)$. Dengan mensubstitusikan nilai yang diketahui, diperoleh $10 = 8 + (sh_x \cdot 4)$, sehingga nilai sh_xdapat dihitung sebesar 0,5. Dari perhitungan ini dapat disimpulkan bahwa faktor shear yang digunakan adalah $sh_x = 0{,}5$dan $sh_y = 0$.

Mekanisme kerja transformasi ini direalisasikan melalui perkalian matriks koordinat titik dengan matriks transformasi shear berukuran 2×2, yaitu $\begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$. Matriks ini menyebabkan setiap koordinat $x$baru diperoleh dari penjumlahan koordinat $x$lama dengan setengah dari nilai koordinat y-nya, sedangkan koordinat $y$tetap tidak berubah. Kondisi ini menjelaskan mengapa tinggi persegi panjang tetap sama setelah transformasi, namun bentuk keseluruhan objek menjadi miring ke arah horizontal.

Secara keseluruhan, hasil transformasi shear pada tugas ini mengubah persegi panjang menjadi bentuk **jajar genjang**. Analisis ini menegaskan bahwa shear horizontal sangat bergantung pada nilai ordinat (y) suatu titik. Semakin besar jarak titik tersebut dari sumbu x, semakin besar pula pergeseran horizontal yang dialaminya. Sebaliknya, titik-titik yang berada tepat pada sumbu $x$berperan sebagai titik jangkar yang menjaga kestabilan posisi objek pada bidang, sehingga transformasi yang terjadi tetap terkontrol dan mudah dianalisis secara geometris.