LAPORAN PROYEK

GRAFIKA KOMPUTER

Dosen Pengampu: Rio Priantama, S.T., M.T.I.

Disusun oleh :

1. Asep Haryana Saputra (20230810043)
2. Dimas Nurdiansyah (20230810087)
3. Muhammad Rizal Nurfirdaus (20230810088)
4. Rio Andika Andriansyah (20230810155)

Kelas : TINFC-2023-04

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

2026

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga laporan proyek berjudul **"Pengembangan Aplikasi Interaktif Grafika Komputer 2D"** ini dapat diselesaikan dengan baik. Laporan ini disusun untuk memenuhi tugas mata kuliah Grafika Komputer serta sebagai sarana untuk memahami konsep dasar grafika komputer dan transformasi geometri dua dimensi.

Proyek ini membahas pengembangan aplikasi grafika menggunakan bahasa pemrograman Python untuk menggambar objek geometri 2D seperti bujursangkar, segitiga, lingkaran, dan trapesium. Selain itu, aplikasi ini menerapkan transformasi geometri berupa penskalaan, pencerminan, dan rotasi, sehingga pengguna dapat melihat perubahan bentuk dan posisi objek secara langsung. Aplikasi ini diharapkan dapat membantu menghubungkan teori koordinat Kartesius dengan visualisasi objek secara interaktif.

Penulis menyadari bahwa proyek dan laporan ini masih memiliki keterbatasan, sehingga kritik dan saran yang membangun sangat diharapkan untuk pengembangan selanjutnya. Penulis mengucapkan terima kasih kepada Bapak **Rio Priantama, S.T., M.T.I.** selaku dosen pengampu mata kuliah Grafika Komputer serta semua pihak yang telah membantu dalam penyusunan laporan ini. Semoga laporan ini dapat memberikan manfaat dan menambah pemahaman pembaca mengenai grafika komputer 2D.

Kuningan, 30 Januari 2026

Kelompok

Daftar Isi

BAB I
PENDAHULUAN

1.1 Deskripsi Program

Program ini adalah aplikasi interaktif berbasis Python untuk menggambar bentuk geometris 2D dan menerapkan transformasi. Program menggunakan library Matplotlib untuk visualisasi dan NumPy untuk komputasi matematika.

1.2 Alur Kerja Program (Flowchart)

Berikut adalah alur kerja program secara keseluruhan:

- Program dimulai → Inisialisasi Matplotlib
- Tampilkan Menu → Pilih opsi (1-10)
- Jika pilih 1-5: Input parameter bentuk → Hitung vertices → Gambar bentuk
- Jika pilih 6-8: Input parameter transformasi → Hitung koordinat baru → Update gambar
- Jika pilih 9: Hapus semua bentuk
- Jika pilih 10: Keluar program
- Kembali ke langkah 2 (loop)

BAB II
CARA KERJA INISIALISASI

2.1 Inisialisasi Lingkungan Grafis

Langkah pertama program adalah menginisialisasi lingkungan grafis menggunakan Matplotlib. Berikut prosesnya:

Kode Inisialisasi:

```
#!/usr/bin/env python3
"""
Aplikasi Interaktif Grafika Komputer 2D
========================================

Aplikasi untuk menggambar bentuk geometris 2D dan menerapkan
transformasi menggunakan Matplotlib.

Fitur:
- Menggambar: Bujursangkar, Segitiga, Persegi Panjang, Lingkaran,
Trapesium
- Transformasi: Penskalaan, Pencerminan, Rotasi

Penggunaan:
    python main.py

Author: Rio Priantama
Created: 2026-01-13
"""

import matplotlib.pyplot as plt
import numpy as np
import sys
import os

# Tambahkan path untuk import modul lokal
sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)))

# Import modul shapes
from shapes.square import draw_square
from shapes.triangle import draw_triangle
from shapes.rectangle import draw_rectangle
from shapes.circle import draw_circle
from shapes.trapezoid import draw_trapezoid

# Import modul transformations
from transformations.scaling import apply_scaling, scale_vertices
```

```
from transformations.reflection import apply_reflection,
reflect_vertices
from transformations.rotation import apply_rotation, rotate_vertices

# Import utilities
from utils.plotting import init_cartesian_plot, plot_shape_on_ax

def show_menu():
    """Menampilkan menu utama aplikasi."""
    print("\n" + "="*50)
    print("   APLIKASI GRAFIKA KOMPUTER 2D")
    print("="*50)
    print("\n--- Menggambar Bentuk ---")
    print("1. Gambar Bujursangkar (Square)")
    print("2. Gambar Segitiga (Triangle)")
    print("3. Gambar Persegi Panjang (Rectangle)")
    print("4. Gambar Lingkaran (Circle)")
    print("5. Gambar Trapesium (Trapezoid)")
    print("\n--- Transformasi 2D ---")
    print("6. Terapkan Penskalaan (Scale)")
    print("7. Terapkan Pencerminan (Reflect)")
    print("8. Terapkan Rotasi (Rotate)")
    print("\n--- Lainnya ---")
    print("9. Hapus Semua Bentuk (Clear All)")
    print("10. Keluar (Exit)")
    print("-"*50)

def clear_all_shapes(ax):
    """Menghapus semua bentuk dari axes."""
    for p in ax.patches:
        p.remove()
    for txt in ax.texts:
        txt.remove()
    ax.set_xlim(-10, 10)
    ax.set_ylim(-10, 10)
    ax.set_aspect('equal')
    ax.grid(True)
    plt.title('Interactive Drawing and Transformation Application')
    print("Semua bentuk telah dihapus.")

def main():
    """Fungsi utama aplikasi."""
    print("\n" + "="*50)
    print("   SELAMAT DATANG DI APLIKASI GRAFIKA KOMPUTER")
    print("="*50)
    print("\nMenginisialisasi lingkungan grafis...")

    # Inisialisasi plot
    fig, ax = init_cartesian_plot(
```

```
        figsize=(8, 8),
        xlim=(-10, 10),
        ylim=(-10, 10),
        title='Interactive Drawing and Transformation Application'
    )

    # Enable interactive mode
    plt.ion()
    plt.show()

    # Variabel untuk menyimpan data bentuk terakhir
    current_shape_data = None

    while True:
        show_menu()
        choice = input("\nPilih opsi (1-10): ").strip()

        shape_data = None

        if choice == '1':
            shape_data = draw_square(ax)

        elif choice == '2':
            shape_data = draw_triangle(ax)

        elif choice == '3':
            shape_data = draw_rectangle(ax)

        elif choice == '4':
            shape_data = draw_circle(ax)

        elif choice == '5':
            shape_data = draw_trapezoid(ax)

        elif choice == '6':
            if current_shape_data is not None:
                new_data = apply_scaling(ax, current_shape_data)
                if new_data is not None:
                    current_shape_data = new_data
            else:
                print("Tidak ada bentuk yang tersedia. Gambar bentuk
terlebih dahulu.")

        elif choice == '7':
            if current_shape_data is not None:
                new_data = apply_reflection(ax, current_shape_data)
                if new_data is not None:
                    current_shape_data = new_data
            else:
```

```
                print("Tidak ada bentuk yang tersedia. Gambar bentuk
terlebih dahulu.")

        elif choice == '8':
            if current_shape_data is not None:
                new_data = apply_rotation(ax, current_shape_data)
                if new_data is not None:
                    current_shape_data = new_data
            else:
                print("Tidak ada bentuk yang tersedia. Gambar bentuk
terlebih dahulu.")

        elif choice == '9':
            clear_all_shapes(ax)
            current_shape_data = None

        elif choice == '10':
            print("\nKeluar dari aplikasi. Sampai jumpa!")
            break

        else:
            print("Pilihan tidak valid. Harap masukkan angka antara 1
dan 10.")

        # Update current_shape_data jika menggambar bentuk baru
        if choice in ['1', '2', '3', '4', '5'] and shape_data is not
None:
            current_shape_data = shape_data
            plot_shape_on_ax(ax, current_shape_data,
clear_previous=True)

        # Redraw canvas
        plt.draw()
        plt.pause(0.1)

    # Cleanup
    plt.ioff()
    plt.close(fig)
    print("Aplikasi gambar telah ditutup.")

if __name__ == "__main__":
    main()
```

Penjelasan:

Program Python yang ditampilkan merupakan sebuah aplikasi interaktif grafika komputer 2D yang dirancang untuk menggambar berbagai bentuk geometri dasar serta

menerapkan transformasi geometri menggunakan pustaka Matplotlib. Aplikasi ini dijalankan melalui terminal dan menampilkan hasil visual pada jendela grafik, sehingga pengguna dapat berinteraksi secara langsung dengan objek yang digambar. Bentuk geometri yang didukung meliputi bujursangkar, segitiga, persegi panjang, lingkaran, dan trapesium, sedangkan transformasi yang tersedia mencakup penskalaan, pencerminan, dan rotasi. Dengan pendekatan ini, aplikasi tidak hanya berfungsi sebagai alat gambar, tetapi juga sebagai media pembelajaran konsep dasar grafika komputer dua dimensi.

Pada bagian awal program, beberapa library utama diimpor, yaitu Matplotlib untuk visualisasi grafik, NumPy untuk perhitungan numerik dan pengolahan titik-titik koordinat, serta modul sys dan os untuk mengatur jalur direktori agar modul lokal dapat diakses. Pengaturan path menggunakan sys.path.insert bertujuan agar Python dapat mengenali folder proyek sebagai sumber modul, sehingga file-file seperti shapes, transformations, dan utils dapat diimpor tanpa menimbulkan kesalahan. Struktur ini menunjukkan bahwa program disusun secara modular, sehingga setiap bagian memiliki tanggung jawab yang jelas dan mudah dikembangkan.

Modul shapes berisi fungsi-fungsi untuk menggambar bentuk geometri 2D. Setiap fungsi, seperti draw_square, draw_triangle, dan lainnya, bertugas menggambar bentuk tertentu pada axes Matplotlib dan mengembalikan data bentuk tersebut. Data ini biasanya mencakup informasi titik-titik koordinat (vertices), jenis bentuk, serta referensi objek grafis (patch) yang digunakan Matplotlib untuk menampilkan bentuk. Data inilah yang nantinya dipakai kembali ketika transformasi diterapkan, sehingga perubahan bentuk dapat dilakukan secara konsisten tanpa harus menggambar ulang dari awal.

Selain menggambar bentuk, program juga menyediakan modul transformations yang berisi fungsi-fungsi transformasi geometri. Transformasi penskalaan digunakan untuk memperbesar atau memperkecil ukuran objek, pencerminan untuk membalik objek terhadap sumbu tertentu, dan rotasi untuk memutar objek dengan sudut tertentu. Secara konsep, transformasi ini memanfaatkan operasi matriks terhadap koordinat titik-titik bentuk. Setelah transformasi diterapkan, fungsi akan mengembalikan data bentuk yang telah diperbarui sehingga dapat ditampilkan kembali pada grafik.

Pengelolaan tampilan grafik dilakukan melalui modul utils.plotting. Fungsi init_cartesian_plot bertugas membuat sistem koordinat kartesius dengan skala yang seimbang, grid aktif, serta judul grafik. Sementara itu, fungsi plot_shape_on_ax digunakan untuk menampilkan atau memperbarui bentuk pada axes, termasuk opsi untuk menghapus bentuk sebelumnya. Dengan pemisahan ini, logika penggambaran dan pengaturan tampilan menjadi lebih rapi dan mudah dipahami.

Fungsi show_menu menampilkan menu utama aplikasi di terminal. Menu ini berisi pilihan untuk menggambar bentuk, menerapkan transformasi, menghapus semua bentuk, dan keluar dari aplikasi. Seluruh interaksi pengguna dilakukan melalui menu ini, sehingga alur penggunaan program menjadi jelas dan terstruktur. Setiap pilihan menu akan diproses di dalam loop utama program yang berjalan terus-menerus hingga pengguna memilih opsi keluar.

Fungsi main merupakan inti dari keseluruhan aplikasi. Pada bagian awal, program menginisialisasi lingkungan grafis dengan membuat figure dan axes Matplotlib serta mengaktifkan mode interaktif menggunakan plt.ion(). Mode interaktif ini memungkinkan grafik diperbarui secara langsung tanpa menutup jendela setiap kali terjadi perubahan. Di dalam fungsi ini juga terdapat variabel current_shape_data yang berfungsi untuk menyimpan data bentuk terakhir yang digambar atau dimodifikasi. Variabel ini sangat penting karena semua transformasi hanya diterapkan pada bentuk yang sedang aktif.

Selama program berjalan, sebuah loop while digunakan untuk terus menampilkan menu dan membaca input pengguna. Jika pengguna memilih untuk menggambar bentuk, maka fungsi gambar yang sesuai akan dipanggil dan data bentuk tersebut disimpan sebagai bentuk aktif. Jika pengguna memilih transformasi, program akan terlebih dahulu memeriksa apakah sudah ada bentuk yang tersedia. Apabila tidak ada, maka akan ditampilkan pesan peringatan. Namun jika ada, transformasi akan diterapkan dan hasilnya langsung diperbarui pada tampilan grafik. Setiap perubahan divisualisasikan menggunakan perintah plt.draw() dan plt.pause() agar grafik tetap responsif.

Ketika pengguna memilih opsi untuk menghapus semua bentuk, fungsi clear_all_shapes akan dipanggil. Fungsi ini menghapus seluruh objek grafis dan teks dari

axes, kemudian mengatur ulang tampilan grafik ke kondisi awal. Sementara itu, saat pengguna memilih keluar dari aplikasi, loop utama akan dihentikan, mode interaktif Matplotlib dimatikan, dan jendela grafik ditutup dengan rapi. Dengan mekanisme ini, program dapat berhenti tanpa meninggalkan proses atau penggunaan memori yang tidak perlu.

Secara keseluruhan, aplikasi ini merupakan contoh implementasi grafika komputer 2D yang baik dan terstruktur. Program ini menerapkan konsep dasar penggambaran objek, pengelolaan koordinat, serta transformasi geometri secara interaktif. Dengan desain modular dan alur kerja yang jelas, aplikasi ini sangat cocok digunakan sebagai media pembelajaran grafika komputer, khususnya untuk memahami hubungan antara representasi matematis objek dan visualisasinya dalam bentuk grafik dua dimensi.

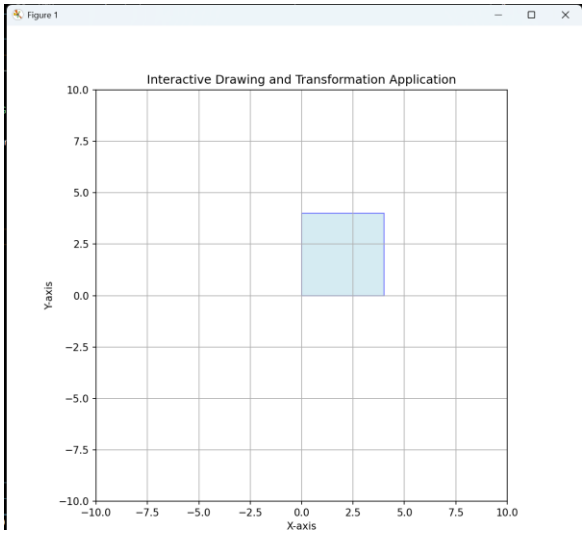
BAB III
CARA KERJA MENGGAMBAR BENTUK

3.1 Cara Kerja Menggambar Bujur Sangkar

Program meminta input panjang sisi dan koordinat pojok kiri bawah. Kemudian menghitung 4 vertices:

```
vertices = [(x, y), (x+sisi, y), (x+sisi, y+sisi), (x, y+sisi)]
```

Contoh: Kotak 4x4 dengan pojok kiri bawah di titik (0,0)



Objek yang ditampilkan adalah sebuah bujur sangkar (persegi) dua dimensi dengan panjang sisi 4 unit yang diposisikan secara presisi pada Kuadran I sistem koordinat Kartesius. Secara matematis, objek ini didefinisikan oleh empat titik sudut (vertices) utama, dengan titik acuan kiri bawah berada tepat di pusat koordinat atau origin (0,0). Sisi-sisinya membentang secara horizontal dan vertikal hingga mencapai koordinat (4,0), (4,4), dan (0,4), sehingga membentuk bidang dengan luas 16 unit persegi. Penempatan yang sejajar dengan sumbu X dan sumbu Y ini menunjukkan bahwa objek berada dalam kondisi awal (default state) yang stabil, tanpa adanya pengaruh transformasi rotasi ataupun kemiringan (shear).

Dari sisi teknis pemrograman, bujursangkar ini dirender menggunakan kombinasi garis tepi (stroke) berwarna biru solid untuk mempertegas batas bidang, serta isian (fill) biru muda transparan agar garis kisi-kisi (grid) pada latar belakang tetap terlihat.

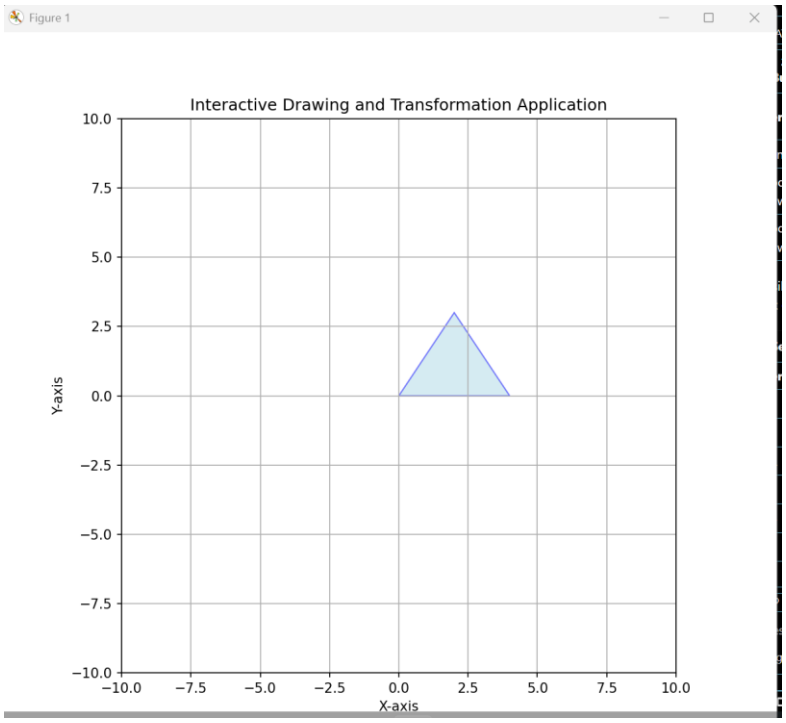
Penggunaan transparansi ini sangat efektif untuk membantu pengguna memverifikasi akurasi posisi setiap titik sudut terhadap bidang koordinat secara visual. Di dalam aplikasi, objek ini bukan sekadar gambar statis, melainkan representasi dari struktur data koordinat yang dinamis; artinya, setiap titik sudut tersebut telah tersimpan dalam memori program dan siap digunakan sebagai parameter input untuk proses transformasi geometri lebih lanjut, seperti diperbesar skalanya, dicerminkan, maupun diputar sesuai kebutuhan pengguna

3.2 Cara Kerja Menggambar Segitiga

Program meminta input 3 pasang koordinat (x1,y1), (x2,y2), (x3,y3). Ketiga titik ini langsung digunakan sebagai vertices segitiga.

```
vertices = [(x1, y1), (x2, y2), (x3, y3)]
```

Contoh: Segitiga dengan 3 titik di (0,0), (4,0), dan (2,3)



Gambar tersebut menampilkan sebuah antarmuka grafis dari program komputer yang dinamakan "Interactive Drawing and Transformation Application". Di dalamnya terdapat sebuah sistem koordinat Kartesius dua dimensi dengan sumbu-X dan sumbu-Y

yang memiliki rentang nilai dari -10.0 hingga 10.0. Pada area kerja tersebut, terdapat objek utama berupa sebuah segitiga sama kaki berwarna biru muda transparan dengan garis tepi berwarna biru yang lebih tegas.

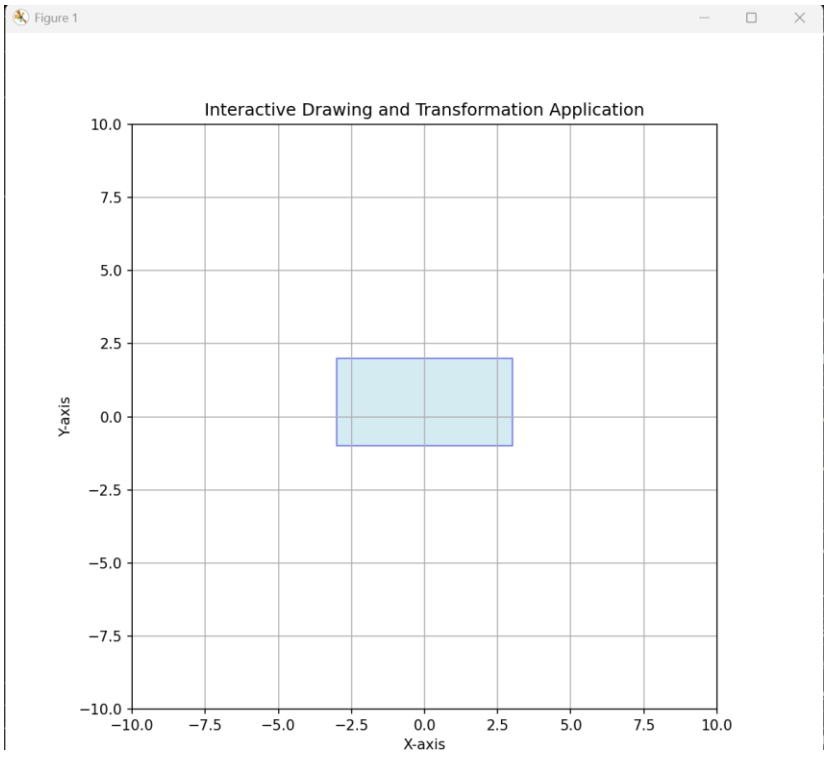
Berdasarkan posisi pada kisi-kisi (grid), segitiga tersebut memiliki tiga titik sudut utama: titik pertama berada di pangkal koordinat , titik kedua berada di pada sumbu horizontal, dan titik puncaknya berada tepat di tengah keduanya secara horizontal, yaitu pada koordinat . Struktur visual ini menunjukkan bahwa aplikasi tersebut sedang digunakan untuk merepresentasikan objek geometri sederhana di atas bidang koordinat, yang kemungkinan besar dapat dimanipulasi atau ditransformasikan (seperti digeser, diputar, atau diubah ukurannya) sesuai dengan nama aplikasinya.

3.3 Cara Kerja Menggambar Persegi Panjang

Program meminta input lebar, tinggi, dan koordinat pojok. Vertices dihitung:

```
vertices = [(x, y), (x+lebar, y), (x+lebar, y+tinggi), (x, y+tinggi)]
```

Contoh: Persegi panjang 6x3 dengan pojok kiri bawah di (-3,-1)



Gambar diatas menunjukkan antarmuka dari perangkat lunak yang sama, yaitu **"Interactive Drawing and Transformation Application"**, namun dengan objek geometri yang berbeda dari sebelumnya. Di atas bidang koordinat Kartesius yang memiliki rentang dari -10.0 hingga 10.0 pada kedua sumbunya, kini terdapat sebuah **persegi panjang** yang digambarkan dengan warna biru muda transparan dan garis tepi biru padat.

Berdasarkan posisi pada grid, persegi panjang tersebut terletak secara simetris terhadap sumbu-Y. Titik-titik sudutnya berada pada koordinat (-3, 2) di kiri atas, (3, 2) di kanan atas, (3, -1) di kanan bawah, dan (-3, -1) di kiri bawah. Hal ini menunjukkan objek memiliki lebar sebesar 6 satuan dan tinggi sebesar 3 satuan. Penempatan objek yang memotong sumbu-X dan sumbu-Y ini memberikan gambaran visual yang jelas mengenai posisi koordinat negatif dan positif dalam ruang kerja aplikasi tersebut.

3.4 Cara Kerja Menggambar Lingkaran

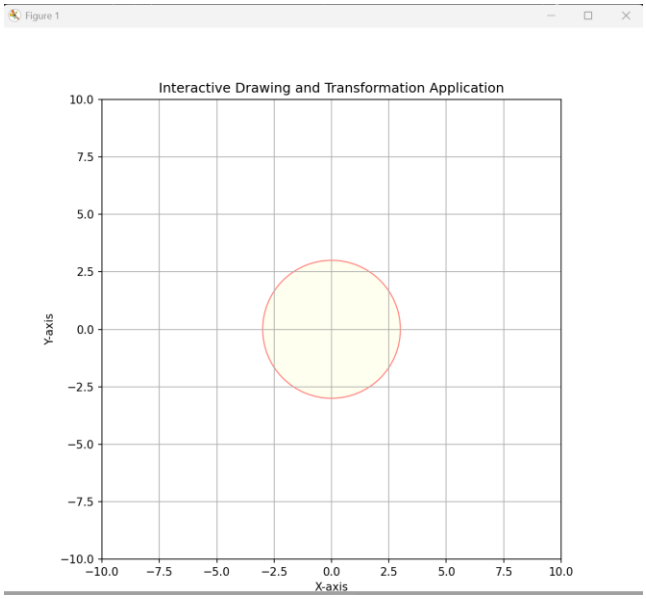
Program meminta input koordinat pusat (cx, cy) dan radius. Lingkaran digambar menggunakan matplotlib.patches.Circle:

```
circle = Circle((cx, cy), radius)
```

Untuk transformasi, lingkaran dikonversi ke 100 titik menggunakan persamaan parametrik:

$$x = cx + r * \cos(\theta),\ y = cy + r * \sin(\theta),\ \theta \in [0, 2\pi]$$

Contoh: Lingkaran dengan pusat (0,0) dan jari-jari 3



Gambar kedua ini menampilkan objek geometri yang berbeda pada aplikasi yang sama, yang menunjukkan fleksibilitas alat tersebut dalam menangani berbagai bentuk. Pada gambar pertama, kita melihat **persegi panjang** yang sama dengan sebelumnya, terletak di antara koordinat x = -3 hingga 3 dan y = -1 hingga 2, memberikan kesan stabilitas di tengah grid. Sementara itu, pada gambar kedua, aplikasi menampilkan sebuah **lingkaran** berwarna krem dengan garis tepi merah yang halus. Lingkaran ini berpusat tepat di titik asal (0,0) dengan jari-jari sebesar 3 satuan, sehingga tepi-tepinya menyentuh angka 3 dan -3 pada kedua sumbu. Perubahan dari bentuk persegi panjang berwarna biru ke lingkaran berwarna merah ini mendemonstrasikan kemampuan aplikasi dalam menggambar objek kurva secara presisi di atas sistem koordinat.

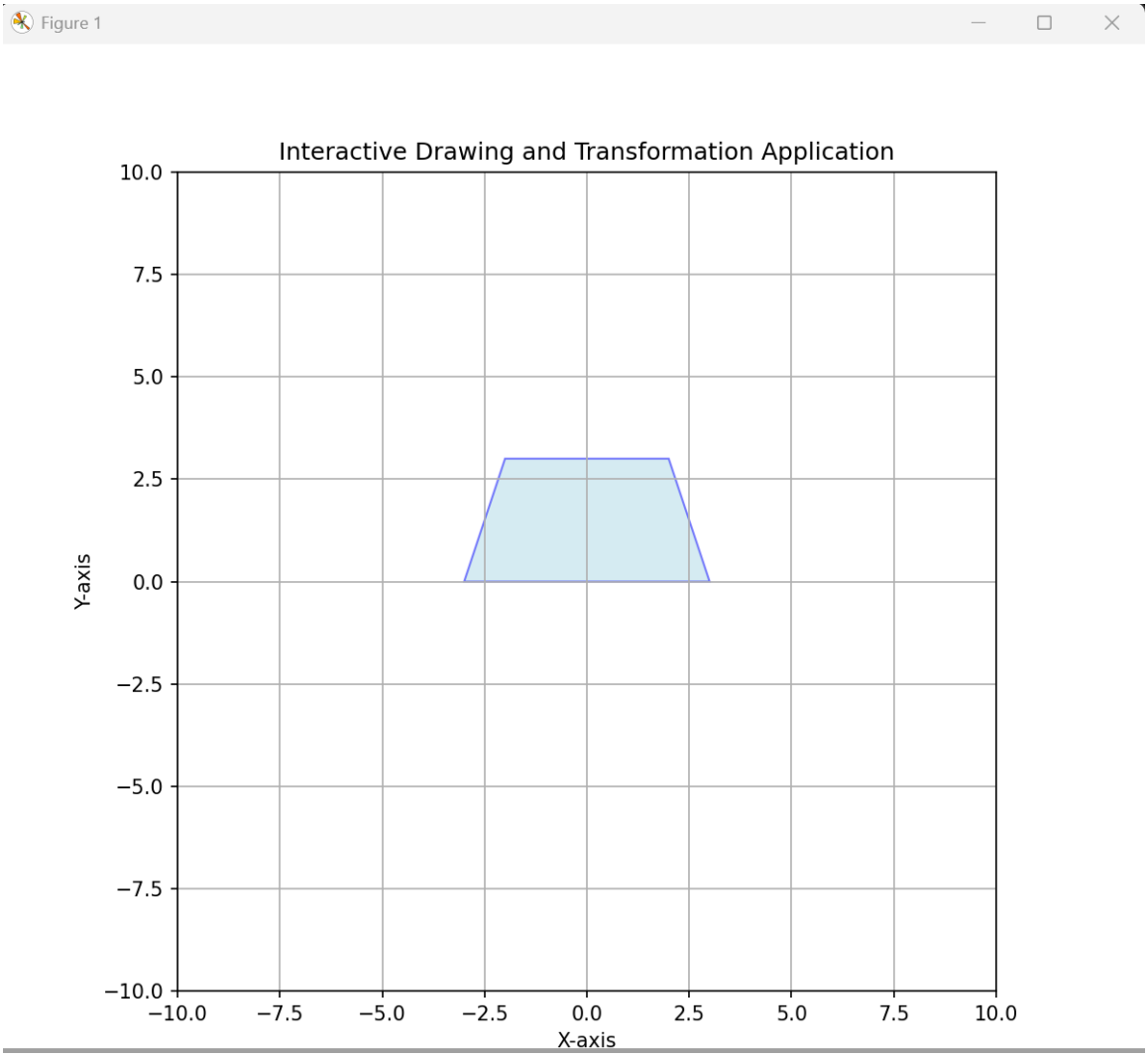
3.5 Cara Kerja Menggambar Trapesium

Program meminta input lebar bawah, lebar atas, tinggi, dan koordinat pojok. Offset dihitung agar sisi atas berada di tengah:

```
offset = (lebar_bawah - lebar_atas) / 2
```

```
vertices = [(x, y), (x+lb, y), (x+lb-offset, y+t), (x+offset, y+t)]
```

Contoh: Trapesium simetris dengan sisi bawah 6, sisi atas 4, tinggi 3



Gambar diatas tersebut memperlihatkan perkembangan objek geometri pada aplikasi **"Interactive Drawing and Transformation Application"**, yang menggunakan sistem koordinat Kartesius dengan rentang hingga . Pada gambar pertama, objek yang ditampilkan adalah sebuah **persegi panjang** biru transparan yang diposisikan secara simetris terhadap sumbu-Y, dengan lebar satuan (dari ke) dan tinggi satuan (dari ke). Gambar kedua menunjukkan perubahan drastis pada bentuk objek menjadi sebuah **lingkaran** berwarna krem dengan garis tepi merah; lingkaran ini berpusat tepat di titik dengan jari-jari satuan, sehingga secara visual mengisi area tengah grid dengan sangat seimbang.

Terakhir, pada gambar ketiga, aplikasi menampilkan sebuah bangun datar **trapesium sama kaki** berwarna biru transparan. Trapesium ini memiliki alas bagian

bawah yang terletak tepat di atas sumbu-X sepanjang satuan (dari ke), sedangkan sisi atasnya lebih pendek dengan panjang satuan (dari ke) yang terletak pada ketinggian . Perubahan berurutan dari persegi panjang, ke lingkaran, hingga ke trapesium ini menunjukkan kemampuan perangkat lunak dalam memvisualisasikan berbagai poligon dan kurva secara matematis berdasarkan koordinat titik sudut maupun parameter geometri tertentu.

3.6 Proses Menambahkan Bentuk ke Canvas

Setelah proses perhitungan vertices selesai dilakukan, tahap selanjutnya adalah merender bentuk geometris tersebut ke dalam canvas grafis. Proses ini merupakan langkah krusial dalam visualisasi grafika komputer karena mengubah data matematis (koordinat vertices) menjadi representasi visual yang dapat dilihat oleh pengguna.

3.6.1 Pembuatan Objek Polygon

Langkah pertama dalam proses rendering adalah membuat objek Polygon menggunakan library Matplotlib. Objek Polygon merupakan representasi dari bentuk geometris tertutup yang didefinisikan oleh sekumpulan titik vertices. Sintaks pembuatannya adalah sebagai berikut:

```
polygon = plt.Polygon(vertices, closed=True, edgecolor='blue',
facecolor='lightblue', alpha=0.5)
```

Pada kode di atas, terdapat beberapa parameter penting yang digunakan:

Parameter	**Fungsi**	**Keterangan**
vertices	Menentukan titik-titik sudut polygon	Berupa list of tuples koordinat (x, y)
closed=True	Menutup polygon secara otomatis	Menghubungkan titik terakhir dengan titik pertama

edgecolor	Warna garis tepi (outline)	Dapat berupa nama warna atau kode hex
facecolor	Warna isi (fill) polygon	Dapat berupa nama warna atau kode hex
alpha	Tingkat transparansi	Nilai 0-1 (0=transparan, 1=solid)

3.6.2 Penambahan Patch ke Axes

Setelah objek Polygon berhasil dibuat, langkah berikutnya adalah menambahkan objek tersebut ke dalam sistem koordinat Axes menggunakan method add_patch():

Method add_patch() berfungsi untuk mendaftarkan objek grafis (patch) ke dalam container Axes. Proses ini melibatkan beberapa tahapan internal:

1. **Registrasi Objek**: Matplotlib mendaftarkan polygon ke dalam koleksi patches yang dimiliki oleh Axes
2. **Transformasi Koordinat**: Sistem melakukan transformasi dari koordinat data (user space) ke koordinat display (pixel space)
3. **Penentuan Bounding Box**: Matplotlib menghitung area yang ditempati oleh polygon untuk keperluan rendering dan interaksi
4. **Penjadwalan Rendering**: Polygon ditandai untuk direnderpada siklus draw berikutnya

3.6.3 Proses Rendering Pipeline

Ketika plt.show() atau plt.draw() dipanggil, Matplotlib menjalankan rendering pipeline yang terdiri dari:

1. **Backend Processing**: Matplotlib berkomunikasi dengan backend grafis (seperti TkAgg, Qt5Agg) untuk mempersiapkan konteks rendering
2. **Artist Rendering**: Setiap artist (termasuk Polygon) direnderberdasarkan urutan z-order

3. **Rasterization**: Untuk backend berbasis raster, polygon dikonversi menjadi piksel-piksel pada framebuffer
4. **Display**: Hasil render ditampilkan pada window atau disimpan ke file

BAB IV
CARA KERJA TRANSFORMASI 2D

4.1 Cara Kerja Penskalaan (Scaling)

Penskalaan mengubah ukuran bentuk dengan faktor skala. Proses kerjanya:

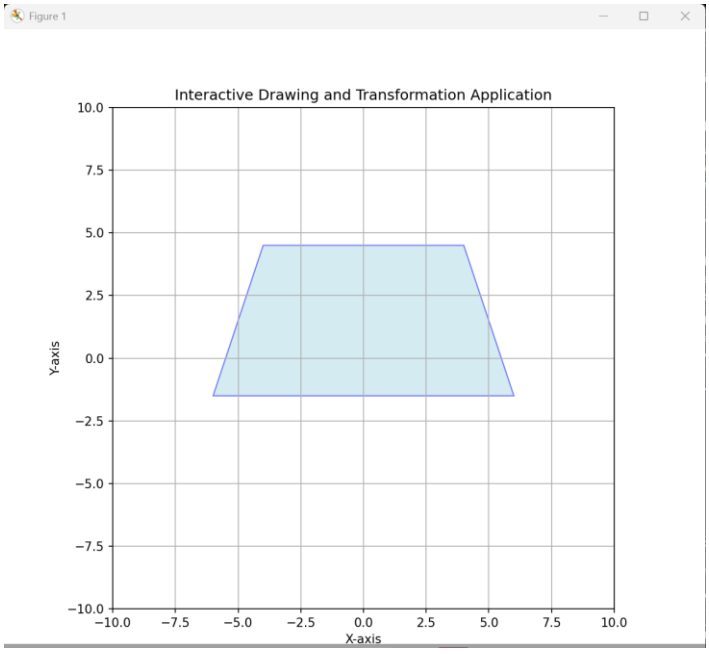
- Input faktor skala sx (sumbu X) dan sy (sumbu Y)
- Hitung centroid bentuk sebagai pusat transformasi
- Untuk setiap vertex: translasi ke origin, kalikan dengan faktor skala, translasi kembali
- Gambar ulang bentuk dengan vertices baru

Rumus:

$x' = (x - cx) \times sx + cx$

$y' = (y - cy) \times sy + cy$

Contoh: Trapesium simetris dengan sisi bawah 6, sisi atas 4, tinggi 3



Rangkaian gambar tersebut memvisualisasikan proses **penskalaan (scaling)** pada berbagai objek geometri di dalam aplikasi "Interactive Drawing and Transformation

Application". Berdasarkan rumus dan contoh yang Anda berikan, narasi dari transformasi tersebut dapat disusun sebagai berikut:

Aplikasi ini mendemonstrasikan bagaimana sebuah objek geometri, seperti **trapesium simetris**, dapat dibentuk dan dimodifikasi menggunakan parameter matematika yang presisi. Berdasarkan rumus vertices yang digunakan, trapesium dibangun dengan menentukan titik awal (x, y), lebar alas bawah (lb), serta lebar atas yang disesuaikan melalui nilai offset. Sebagai contoh, pada salah satu gambar, terlihat trapesium dengan sisi bawah 6 satuan dan sisi atas 4 satuan dengan tinggi 3. Ketika fungsi **penskalaan** diterapkan mirip dengan contoh segitiga yang dikalikan dua kali lipat—seluruh dimensi objek akan membesar secara proporsional.

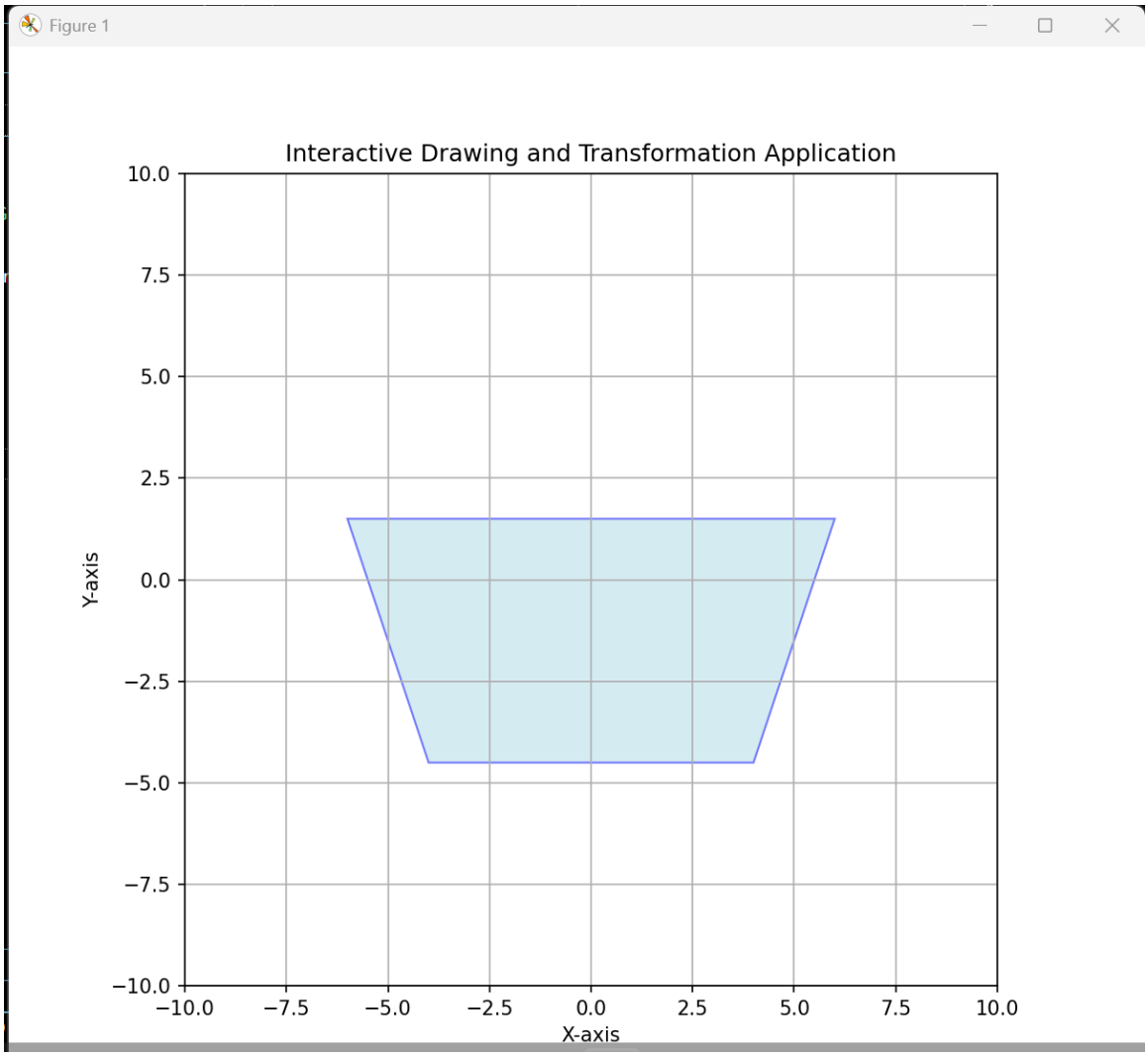
Penskalaan ini tidak hanya mengubah ukuran panjang sisi, tetapi juga menggeser posisi koordinat titik sudutnya agar tetap sinkron dengan pusat massa atau titik referensi yang ditentukan. Hal ini terlihat pada perbedaan antara gambar trapesium kecil (lebar bawah 6) dan trapesium yang lebih besar (lebar bawah mencapai 12 satuan), di mana objek tersebut tampak "tumbuh" keluar dari pusatnya. Proses ini membuktikan bahwa transformasi geometri dalam aplikasi tersebut bekerja dengan menghitung ulang setiap posisi *vertex* berdasarkan faktor skala yang diinput, sehingga menghasilkan visualisasi perubahan ukuran yang akurat di atas bidang koordinat.

4.2 Cara Kerja Pencerminan (Reflection)

Pencerminan menghasilkan bayangan cermin bentuk. Program menyediakan 4 opsi:

Jenis Cermin	Rumus	Contoh
Sumbu X	x' = x, y' = -y	(3,2) → (3,-2)
Sumbu Y	x' = -x, y' = y	(3,2) → (-3,2)
Origin	x' = -x, y' = -y	(3,2) → (-3,-2)
Garis y=x	x' = y, y' = x	(3,2) → (2,3)

Sumbu X



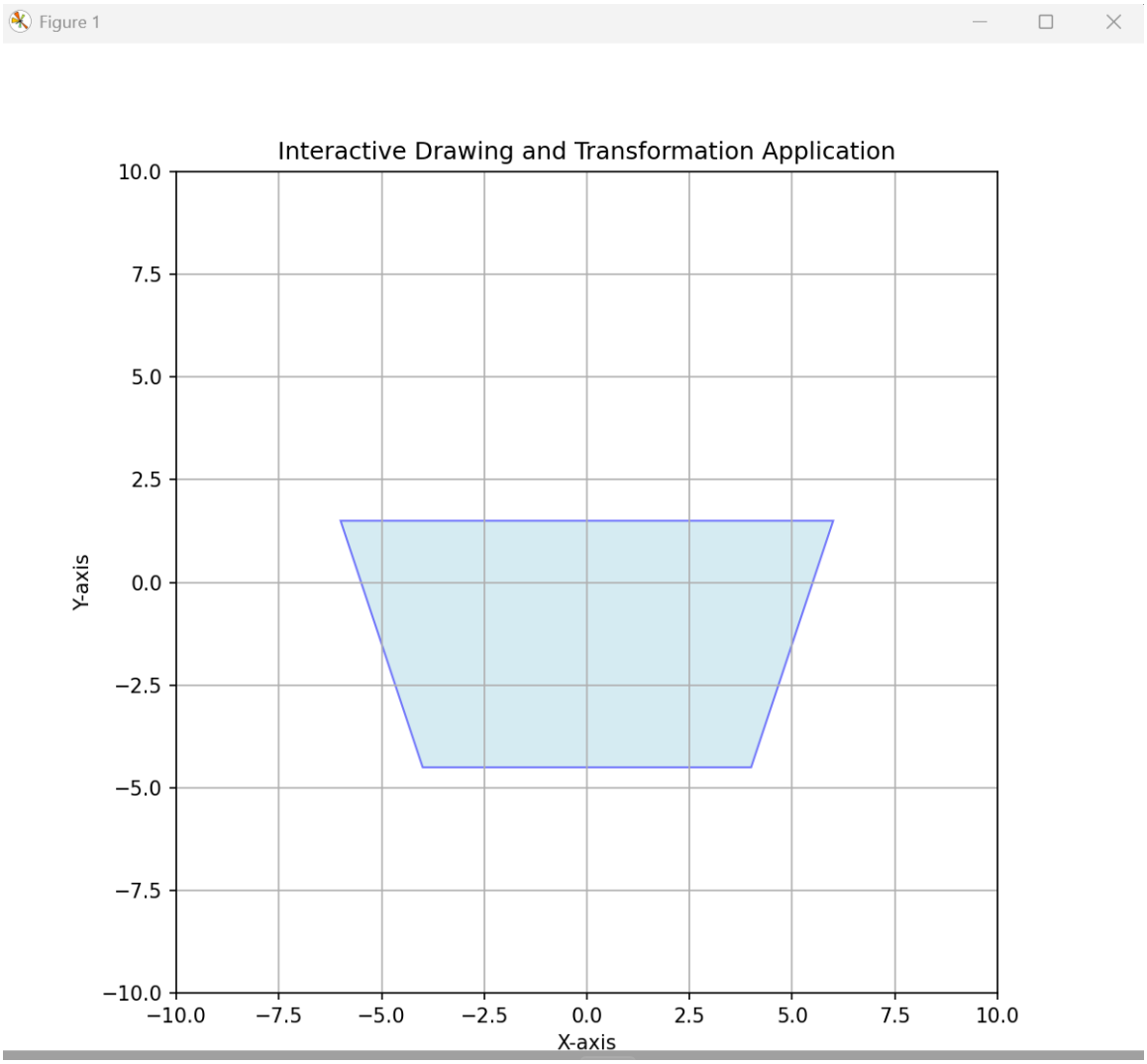
Rangkaian gambar diatas mengilustrasikan penerapan berbagai **transformasi geometri** pada aplikasi interaktif, mulai dari perubahan bentuk dasar hingga proses pencerminan (refleksi) dan penskalaan. Berdasarkan tabel rumus yang Anda sertakan, narasi untuk proses transformasi, khususnya pada **sumbu-X**, dapat dijelaskan sebagai berikut:

Dalam aplikasi ini, setiap objek didefinisikan oleh sekumpulan titik sudut atau *vertices*. Ketika kita menerapkan pencerminan terhadap **sumbu-X**, aplikasi menjalankan rumus x' = x dan y' = -y. Artinya, nilai horizontal (x) tetap sama, namun posisi vertikal (y) akan dibalikkan nilainya. Sebagai contoh nyata dalam visualisasi tersebut, jika kita

memiliki titik puncak trapesium di (3, 2), setelah dicerminkan terhadap sumbu-X, titik tersebut akan berpindah secara akurat ke posisi (3, -2).

Proses ini terlihat jelas pada perbandingan antara gambar trapesium yang berada di area atas (koordinat y positif) dengan gambar trapesium yang terbalik di area bawah (koordinat y negatif). Objek tersebut seolah-olah melihat bayangannya sendiri di permukaan sumbu-X yang bertindak sebagai cermin. Selain pencerminan, penggunaan parameter lebar bawah, lebar atas, dan tinggi memungkinkan trapesium tersebut diubah skalanya secara dinamis. Gabungan antara rumus penskalaan dan rumus refleksi sumbu-X ini memastikan bahwa bentuk geometri tetap presisi secara matematis, meskipun ukuran dan orientasinya berubah drastis di dalam ruang koordinat tersebut.

Sumbu Y



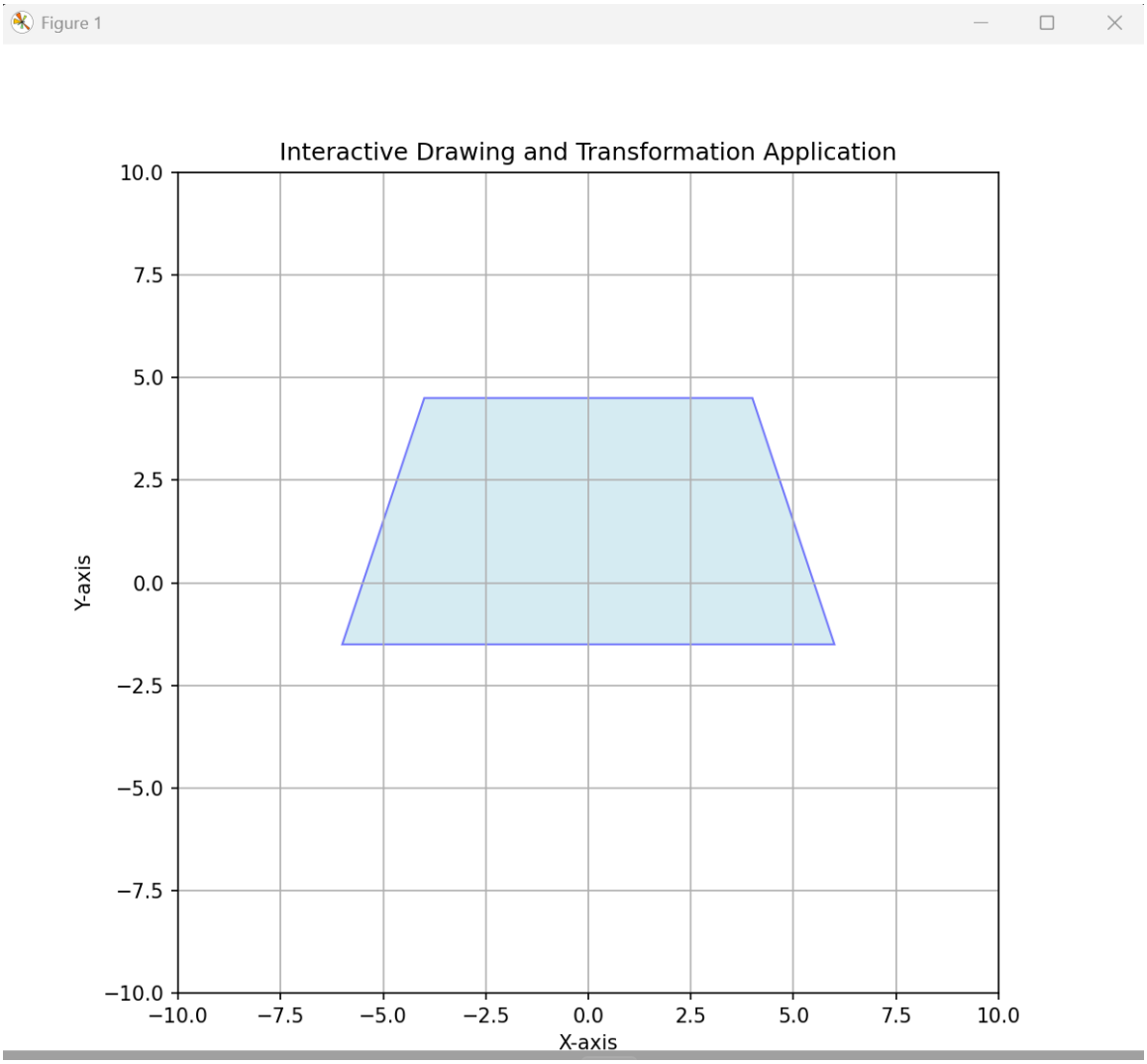
Rangkaian gambar diatas mendemonstrasikan proses transformasi geometri berupa **pencerminan (refleksi) terhadap sumbu-Y** di dalam aplikasi interaktif. Berdasarkan rumus yang Anda berikan, narasi teknis dari proses ini dapat disusun sebagai berikut:

Dalam sistem koordinat ini, pencerminan terhadap sumbu-Y dilakukan dengan menerapkan rumus x' = -x dan y' = y. Secara visual, proses ini membalikkan posisi horizontal objek tanpa mengubah ketinggiannya, sehingga sumbu-Y bertindak sebagai garis lipat atau cermin tegak lurus. Sebagai contoh, jika kita memiliki titik sudut trapesium di koordinat (3, 2), setelah proses refleksi, titik tersebut akan berpindah secara otomatis ke (-3, 2). Hal ini terlihat pada transisi antar gambar di mana objek trapesium yang semula

dominan berada di sisi kanan grid (nilai x positif) berpindah secara simetris ke sisi kiri grid (nilai x negatif).

Aplikasi ini secara akurat menjaga proporsi objek melalui perhitungan *vertices* yang dinamis. Ketika trapesium tersebut memiliki lebar bawah 6 satuan dan lebar atas 4 satuan, pencerminan terhadap sumbu-Y memastikan bahwa jarak setiap titik ke garis tengah tetap sama namun dalam arah yang berlawanan. Narasi visual ini menunjukkan bahwa transformasi sumbu-Y adalah cara yang efektif untuk menciptakan simetri sempurna pada objek geometri, memungkinkan pengguna untuk memanipulasi tata letak bangun datar dengan kepastian matematis yang tinggi.

Origin

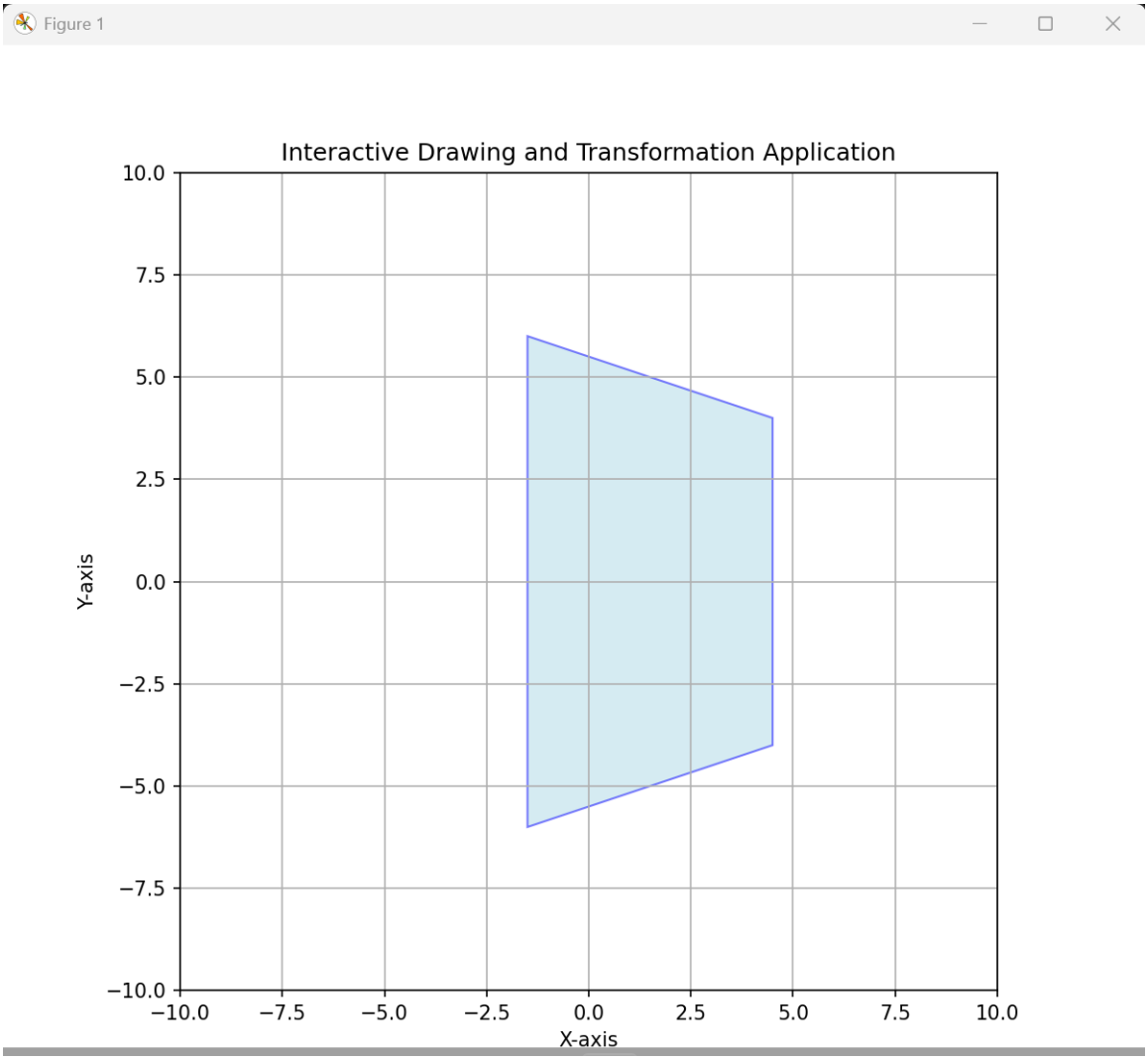


Berdasarkan rangkaian gambar diatas dan parameter matematika yang telah kita bahas, narasi untuk penerapan transformasi **Origin (Titik Pusat)** pada objek geometri dalam aplikasi tersebut adalah sebagai berikut:

Transformasi melalui **Origin** merupakan bentuk refleksi ganda yang melibatkan pembalikan posisi objek baik secara horizontal maupun vertikal secara bersamaan. Mengacu pada aturan matematika yang Anda berikan, aplikasi ini menerapkan rumus dan untuk setiap *vertex* objek. Hal ini berarti, jika sebuah titik pada trapesium berada di koordinat , setelah transformasi Origin, titik tersebut akan dipetakan tepat di seberangnya pada koordinat . Secara visual, efek ini menciptakan rotasi di mana objek tidak hanya berpindah sisi dari kanan ke kiri, tetapi juga terjungkir dari atas ke bawah.

Dalam konteks gambar trapesium yang ditampilkan, kita dapat melihat bagaimana struktur bangun datar tersebut mengalami perubahan orientasi yang total. Trapesium yang semula memiliki alas lebar di bagian bawah dan terletak di area koordinat positif (atas-kanan), akan berubah menjadi trapesium terbalik dengan alas lebar berada di sisi atas dan berpindah ke area koordinat negatif (bawah-kiri). Ketelitian aplikasi dalam menghitung *offset* dan dimensi vertices memastikan bahwa meskipun objek mengalami pembalikan nilai dan yang drastis, bentuk geometrisnya tetap konsisten dan simetris terhadap titik pusat , memberikan ilustrasi yang sempurna mengenai keseimbangan matematis dalam ruang dua dimensi.

Garis y=x



Berdasarkan rangkaian gambar dan tabel transformasi yang Anda berikan, narasi untuk penerapan refleksi terhadap **Garis** pada objek geometri dalam aplikasi tersebut adalah sebagai berikut:

Refleksi terhadap garis diagonal merupakan salah satu transformasi yang paling menarik secara visual karena melibatkan pertukaran peran antara sumbu horizontal dan vertikal. Mengacu pada rumus transformasi yang Anda sertakan, yaitu dan , setiap titik pada objek akan "bertukar tempat" koordinatnya. Jika kita merujuk pada objek trapesium yang telah mengalami penskalaan, titik yang semula berada pada koordinat akan dipetakan secara akurat ke posisi setelah dicerminkan terhadap garis miring tersebut.

Secara visual, efek ini terlihat jelas pada gambar terakhir di mana orientasi objek berubah secara diagonal. Trapesium yang tadinya berdiri tegak sejajar dengan sumbu-X kini tampak "miring" atau merebah mengikuti arah sumbu-Y. Hal ini terjadi karena dimensi lebar objek (pada sumbu-X) kini menjadi dimensi tinggi (pada sumbu-Y), dan sebaliknya. Penggunaan parameter *vertices* yang dinamis dalam aplikasi memastikan bahwa meskipun posisi dan kemiringannya berubah drastis, karakteristik bentuk trapezium seperti perbandingan sisi atas dan bawahnya tetap terjaga dengan presisi matematis, membuktikan bahwa refleksi adalah cara efektif untuk memutar dan membalik orientasi objek secara simetris terhadap garis diagonal pusat.

4.3 Cara Kerja Rotasi (Rotation)

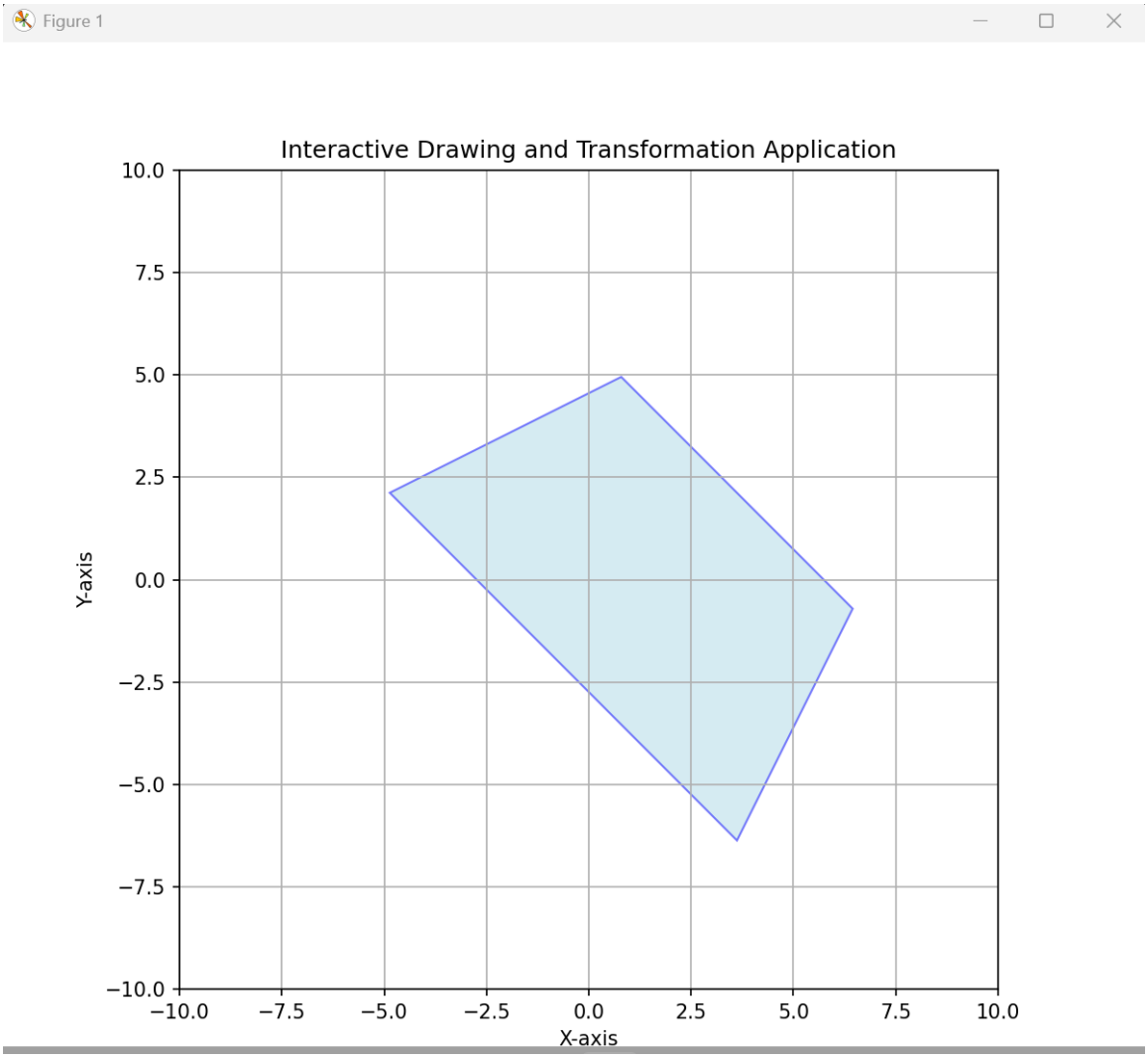
Rotasi memutar bentuk terhadap pusat (centroid) dengan sudut tertentu. Proses kerjanya:

- Input sudut rotasi dalam derajat (positif = counter-clockwise)
- Konversi sudut ke radian: $\theta\_rad = \theta \times \pi / 180$
- Hitung centroid sebagai pusat rotasi
- Untuk setiap vertex: translasi ke origin, terapkan matriks rotasi, translasi kembali
- Gambar ulang bentuk dengan vertices baru

Rumus Rotasi:

$x' = (x-cx)\times\cos(\theta) - (y-cy)\times\sin(\theta) + cx$

$y' = (x-cx)\times\sin(\theta) + (y-cy)\times\cos(\theta) + cy$



Berdasarkan rangkaian gambar diatas dan konsep geometri yang telah kita bahas, narasi untuk penerapan **Rotasi** pada objek dalam aplikasi tersebut adalah sebagai berikut:

Transformasi rotasi dalam aplikasi ini memungkinkan objek untuk berputar di sekitar titik pusat (Origin) dengan sudut tertentu, yang mengubah orientasi objek tanpa mengubah bentuk atau ukurannya. Secara matematis, rotasi sejauh sudut Θ dihitung menggunakan koordinat baru:

$x' = x \cos(\Theta) - y \sin(\Theta)$

$y' = x \sin(\Theta) + y \cos(\Theta)$

Pada visualisasi terakhir, kita dapat melihat bagaimana trapesium yang semula berdiri tegak mengalami perputaran posisi. Jika pada refleksi objek hanya membalik secara kaku, pada rotasi objek tampak "berayun" mengelilingi titik (0,0). Sebagai contoh, rotasi 90° berlawanan arah jarum jam akan memindahkan titik yang semula berada di sumbu-X positif ke sumbu-Y positif, sehingga trapesium yang awalnya melebar secara horizontal kini menjadi memanjang secara vertikal.

Proses ini menunjukkan kemampuan aplikasi dalam menangani perhitungan trigonometri yang kompleks untuk setiap *vertex*. Objek tidak hanya berpindah kuadran, tetapi seluruh kemiringan garis pembentuknya disesuaikan secara presisi sesuai derajat rotasi yang dipilih. Narasi visual ini memberikan pemahaman bahwa rotasi adalah cara dinamis untuk mengubah sudut pandang objek di atas bidang Kartesius, memastikan bahwa meskipun posisi titik-titiknya berubah, integritas geometri trapesium tersebut tetap terjaga sempurna.

BAB V
PENUTUP

5.1 Kesimpulan

Program ini merupakan aplikasi interaktif berbasis **Python** yang mengintegrasikan pustaka **Matplotlib** dan **NumPy** untuk memvisualisasikan konsep grafika komputer 2D secara dinamis. Proyek yang dirancang untuk mendemonstrasikan bagaimana parameter matematis diterjemahkan menjadi objek visual di atas koordinat Kartesius. Alur kerja program dimulai dengan inisialisasi lingkungan grafis yang menetapkan batas layar dari -10 hingga 10 pada sumbu-X dan sumbu-Y, menciptakan ruang kerja yang presisi untuk menggambar bentuk dasar seperti bujursangkar, segitiga, lingkaran, hingga trapesium.

Setiap bentuk yang digambar tidak hanya sekadar visualisasi statis, melainkan kumpulan data koordinat (**vertices**) yang disimpan dalam memori. Sebagai contoh, sebuah trapesium simetris dibangun menggunakan parameter lebar bawah, lebar atas, dan tinggi, yang kemudian diproses melalui fungsi draw_trapezoid untuk menghasilkan titik-titik sudut yang akurat. Proses rendering melibatkan pembuatan objek Polygon dengan pengaturan estetika seperti garis tepi biru solid dan isian biru muda transparan, yang memungkinkan pengguna tetap melihat kisi-kisi grid di latar belakang untuk keperluan verifikasi posisi koordinat secara langsung.

Kekuatan utama aplikasi ini terletak pada modul **Transformasi 2D** yang memungkinkan manipulasi objek melalui perhitungan matriks. Program mendukung tiga jenis transformasi utama:

1. **Penskalaan (Scaling):** Mengubah ukuran objek berdasarkan faktor skala sx dan sy relatif terhadap titik pusat (*centroid*), sehingga objek dapat "tumbuh" atau mengecil secara proporsional.

2. **Pencerminan (Reflection):** Menggunakan rumus pembalikan koordinat untuk menciptakan simetri pada **Sumbu-X** (x' = x, y' = -y), **Sumbu-Y** (x' = -x, y' = y), titik **Origin** (x' = -x, y' = -y), maupun diagonal **Garis y=x** (x' = y, y' = x).

3. **Rotasi (Rotation):** Memutar objek di sekitar titik pusat menggunakan perhitungan trigonometri (cos Θ, sin Θ), yang mengubah orientasi objek tanpa merusak integritas bentuknya.

Secara keseluruhan, proyek ini berhasil membuktikan bahwa hubungan antara teori matematika dan pemrograman dapat menghasilkan alat bantu visual yang kuat. Melalui antarmuka menu yang terstruktur, pengguna dapat melihat bagaimana perubahan satu nilai koordinat atau satu parameter transformasi dapat langsung memperbarui tampilan pada kanvas grafis. Hal ini menjadikan aplikasi ini sebagai media pembelajaran yang efektif bagi mahasiswa informatika dalam memahami mekanisme di balik perangkat lunak desain grafis modern.