Penerapan Transformasi Matriks untuk Efek Animasi Dinamis pada Grafis 2D Sederhana

Mayla Yaffa Ludmilla 13523050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523050@std.stei.itb.ac.id, [2]maylayaffa@gmail.com

***Abstract*— Transformasi matriks memiliki peran penting dalam pengembangan animasi dan grafika komputer. Makalah ini mengeksplorasi implementasi transformasi matriks dalam menghasilkan animasi dinamis pada grafis 2D sederhana. Pembahasan dimulai dengan tinjauan historis perkembangan animasi dan grafika komputer, dilanjutkan dengan pembahasan konsep matematis yang mendasari transformasi matriks, termasuk vektor, matriks, dan koordinat homogen. Implementasi praktis dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan *library* OpenCV untuk memproses dan menampilkan animasi. Hasil implementasi menunjukkan bahwa transformasi matriks dapat digunakan secara efektif untuk menciptakan gerakan dalam animasi 2D. Meskipun implementasi masih terbatas pada transformasi-transformasi dasar, makalah ini menunjukan hasil penerapan transformasi matriks dalam animasi 2D..**

***Keywords*—animasi, transformasi, matriks**

I. PENDAHULUAN

Media visual adalah media yang tidak pernah lepas dari sejarah panjang kehadiran manusia di bumi ini. Manusia sudah menciptakan media visual sejak zaman prasejarah, terbukti dari peninggalan gambar gambar di dinding gua tempat tinggal masyarakat. Sejarah perkembangan animasi dan grafika komputer menunjukan evolusi teknologi yang mengubah cara manusia menciptakan dan mengonsumsi konten visual. Istilah animasi berakar dari bahasa Latin "*animare*" yang artinya menghidupkan. Perkembangannya dimulai dari alat sederhana yang bernama *thaumatrope* yang diciptakan pada tahun 1824. *Thaumatrope* adalah sebuah alat yang terdiri dari dua buah gambar yang ditempelkan secara bolak-balik dan dihubungkan dengan dua tali.di sisi kanan dan sisi kirinya. Ketika tali ini diputar, kedua gambar dapat diputar dengan cepat dan menghasilkan ilusi gambar bergerak.

Gambar 1.1 Thaumatrope
Sumber: [2]

Era awal animasi ditandai dengan penemuan alat alat animasi sederhana seperti *zoetrope* dan *praxinoscope* oleh Émile Reynaud pada sekitar tahun 1870-an yang membuka jalan bagi proyeksi gambar bergerak. Selanjutnya, pada 1908, Emile Cohl melahirkan karya berjudul "Fantasmagorie" yang dikenal sebagai film animasi pertama dengan teknik gambar manual. Industri animasi kemudian mengalami revolusi besar ketika Walt Disney memperkenalkan *cel animation* pada 1920-an, teknik yang mendominasi produksi animasi selama lebih dari enam dekade. Cel adalah kependekan untuk celluloid, bahan yang digunakan untuk menghasilkan lapisan lapisan semi-transparan atau tembus cahaya yang digunakan sebagai media gambar.

Grafika komputer adalah sebutan untuk penggunaan komputer untuk membuat dan memanipulasi gambar. Sejarah grafika komputer dimulai pada tahun 1960-an melalui inovasi Ivan Sutherland, Sketchpad, program pertama yang menggunakan antarmuka dengan grafika komputer yang lengkap. Penemuan ini menjadi pionir dalam sistem grafika komputer interaktif yang memperkenalkan konsep-konsep dasar seperti manipulasi objek langsung dan penggunaan transformasi geometris. Perkembangan selanjutnya adalah penemuan teknik-teknik dasar grafika komputer yang ditemukan oleh Ed Catmull dan Alvy Ray Smith, penemu studio animasi Pixar, pada tahun 1970-an.

Transformasi matriks menjadi komponen yang sangat penting dalam evolusi grafika komputer dan animasi digital. Konsep matematis ini memungkinkan manipulasi objek grafis melalui operasi dasar seperti translasi, rotasi, dan penskalaan yang diperlukan dalam menciptakan animasi. Dalam konteks grafis 2D, transformasi matriks berfungsi sebagai alat untuk menghasilkan gerakan pada objek animasi. Penerapannya memungkinkan kendali atas pergerakan objek dalam ruang dua dimensi

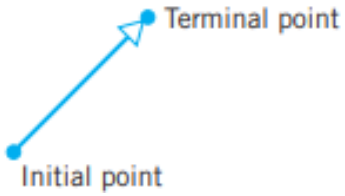
Kemajuan teknologi komputer dan perangkat lunak grafis telah menyederhanakan implementasi transformasi matriks dalam animasi. Ada banyak *platform* dan *library* dalam bahasa bahasa pemrograman yang dapat mengintegrasikan operasi transformasi matriks sebagai komponen dalam pengembangan media grafis dan animasi.

II. DASAR TEORI

A. Vektor

Dalam fisika, ada 2 jenis besaran yaitu besaran skalar dan besaran vektor. Besaran skalar adalah besaran yang hanya memiliki komponen nilai numerikal, sedangkan vektor adalah besaran yang selain memiliki komponen nilai numerikal juga memiliki arah.

Para insinyur dan fisikawan merepresentasikan vektor dalam dua dimensi (juga disebut 2-ruang) atau dalam 3 dimensi (juga disebut 3-ruang) menggunakan anak panah [1]. Arah dari vektor direpresentasikan oleh arah mata panah dan besar atau nilai dari vektor direpresentasikan menggunakan panjang anak panah. Ujung anak panah disebut titik awal atau *initial point* dari vektor tersebut, dan mata panah disebut titik akhir atau *terminal point.*



Gambar 2.1 Vektor
Sumber: [1]

Dalam ruang Euclidian, ruang berdimensi tempat vektor berada, vektor dapat ditulis dalam bentuk matriks yang disebut dengan bentuk vektor baris atau vektor kolom. Untuk vektor dalam n-ruang, vektor barisnya dapat ditulis

$$v = [v_1 \quad v_2 \quad \cdots \quad v_n]$$

dan vektor kolomnya dapat ditulis

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

Dalam animasi menggunakan transformasi matriks, konsep vektor digunakan untuk menyimpan posisi objek-objek dalam animasi.

B. Matriks

Matriks didefinisikan sebagai susunan bilangan yang diatur dalam baris dan kolom berbentuk persegi panjang dan dituliskan dalam kurung siku atau kurung biasa. Matriks berordo m × n adalah matriks yang terdiri dari m baris dan n kolom [1].

Matriks dapat dioperasikan melalui operasi operasi matematika, di antaranya:

1. Penjumlahan dan Pengurangan

   Dua matriks dapat dijumlahkan atau dikurangkan jika kedua matriks tersebut memiliki ukuran kolom dan baris yang sama. Operasi dilakukan dengan menjumlahkan atau mengurangkan elemen-elemen yang bersesuaian.

   Misalkan ada matriks A

   $$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

   dan matriks B

   $$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

   maka penjumlahan matriks A+B adalah

   $$A + B = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} \end{bmatrix}$$

   dan pengurangan matriks A-B adalah

   $$A - B = \begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} & a_{13}-b_{13} \\ a_{21}-b_{21} & a_{22}-b_{22} & a_{23}-b_{23} \\ a_{31}-b_{31} & a_{32}-b_{32} & a_{33}-b_{33} \end{bmatrix}$$

2. Perkalian

   Perkalian matriks A × B dapat dilakukan jika jumlah kolom matriks A sama dengan jumlah baris matriks B. Hasil perkalian matriks adalah matriks baru dengan jumlah baris sama dengan jumlah baris matriks A dan jumlah kolom sama dengan jumlah kolom matriks B.

   Misalkan ada matriks A

   $$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

   dan matriks B

   $$B = \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix}$$

   maka perkalian matriks AxB adalah

   $$A \times B = \begin{bmatrix} aj+bm+cp & ak+bn+cq & al+bo+cr \\ dj+em+fp & dk+en+fq & dl+eo+fr \\ gj+hm+ip & gk+hn+iq & gl+ho+ir \end{bmatrix}$$

3. Determinan

   Determinan adalah nilai yang dapat dihitung dari matriks persegi. Determinan matriks A dilambangkan dengan det(A) atau |A|.

4. Invers

   Invers dari matriks A (dilambangkan A^{-1}) adalah matriks yang jika dikalikan dengan A menghasilkan matriks identitas. Matriks identitas adalah matriks yang semua elemen di diagonal utamanya bernilai 1, dan elemen sisanya bernilai 0.

C. Transformasi Matriks

Transformasi matriks adalah bagian dari transformasi linier yang merupakan fungsi yang memetakan suatu vektor ke vektor lain dalam ruang vektor. Secara matematis, transformasi linear T dari R^n ke R^m dapat dinyatakan sebagai perkalian matriks A dengan vektor x, dimana A adalah matriks berukuran m × n [1].

Suatu transformasi matriks dapat dinyatakan dalam bentuk sistem persamaan linear:

$$w_1 = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n$$
$$w_2 = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n$$
$$\vdots \quad \vdots \quad \vdots \quad \vdots$$
$$w_m = a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n$$

yang dapat kita tulis dalam notasi matriks sebagai

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Sistem persamaan ini dapat ditulis dalam notasi matriks yang lebih ringkas sebagai:

$$w = Ax$$

di mana:

- w adalah vektor hasil transformasi dalam R^m
- A adalah matriks transformasi berukuran m × n
- x adalah vektor input dalam R^n

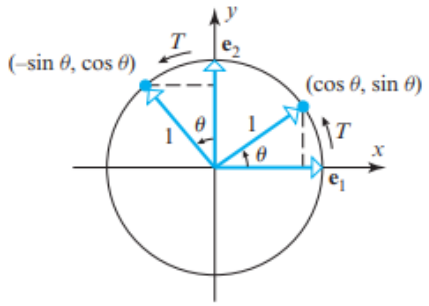
Meskipun persamaan w = Ax dapat dilihat sebagai bentuk ringkas dari sistem persamaan linear, pendekatan yang lebih mendalam adalah memahaminya sebagai suatu transformasi yang memetakan vektor x dalam ruang R^n ke vektor w dalam ruang R^m melalui operasi perkalian matriks A dari kiri. Secara formal, transformasi ini dinotasikan sebagai:

$$T_a: R^n \rightarrow R^m$$

Untuk sebuah matriks A, transformasi matriks $T_a: R^n \rightarrow R^m$ memiliki beberapa sifat untuk setiap vektor **u** dan **v** dan untuk setiap skalar *k:*

$$(a)\ T_A(0) = 0$$
$$(b)\ T_A(ku) = kT_A(u)$$
$$(c)\ T_A(u + v) = T_A(u) + T_A(v)$$
$$(d)\ T_A(u - v) = T_A(u) - T_A(v)$$

Operator matriks di R^2 dan R^3 yang memindahkan titik menyusuri busur dari lingkaran yang memiliki titik pusat di titik asal koordinat disebut dengan operator rotasi. Dalam konteks ini, kita akan fokus pada matriks standar untuk operator rotasi T: $R^2 \rightarrow R^2$ yang menggerakkan titik-titik berlawanan arah jarum jam terhadap titik asal dengan sudut θ positif [1].



Gambar 2.2 Operator rotasi vektor basis standar
Sumber : [1]

Berdasarkan gambar 2.5, untuk vektor basis standar, operator rotasinya adalah

$$T(e_1) = T(1,0) = (\cos\theta, \sin\theta)$$
$$T(e^2) = T(0,1) = (-\sin\theta, \cos\theta)$$

sehingga didapat matriks standarnya sebagai berikut

$$A = [T(e_1)|T(e_2)] = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

D. Koordinat Homogen

Koordinat homogen adalah sebuah representasi matematis yang dapat merepresentasikan transformasi linier dua dimensi atau tiga dimensi menggunakan matriks 3x3 atau matriks 4x4. Koordinat homogen menambahkan elemen tambahan ke vektor yang ada untuk memungkinkan operasi matematika yang sesuai dengan interpretasi geometris yang tepat.

Konversi koordinat kartesian dua dimensi menjadi koordinat homogen dapat ditulis sebagai

$$(x, y) \rightarrow (x, y, w)$$

dan konversi koordinat kartesian tiga dimensi menjadi koordinat homogen dapat ditulis sebagai

$$(x, y, z) \rightarrow (x, y, z, w)$$

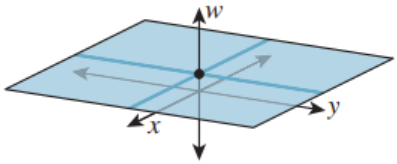
dengan nilai w bergantung pada jenis transformasi yang diterapkan.

Pada transformasi seperti translasi dan rotasi, nilai w biasanya diatur menjadi 1 untuk menjaga konsistensi dan kesederhanaan perhitungan. Namun, pada transformasi lain seperti proyeksi perspektif, nilai w dapat berubah untuk merepresentasikan skala atau kedalaman relatif terhadap bidang pandang. Setelah transformasi, jika w≠1, koordinat homogen perlu dinormalisasi dengan membagi setiap komponen koordinat (x,y,z) dengan nilai w untuk kembali ke bentuk koordinat kartesian.

Dalam grafik komputer, koordinat homogen digunakan dalam dua cara. Pertama, dengan menambahkan elemen tambahan (contohnya elemen ketiga dalam ruang dua dimensi atau elemen keempat dalam ruang tiga dimensi). Elemen tambahan ini bisa berupa nilai apapun yang bertindak sebagai pembagi komponen lainnya dan hanya digunakan pada kondisi tertentu. Kedua, bentuk terbatas dari koordinat homogen juga sangat berguna karena membantu menyelesaikan masalah dalam merepresentasikan dan menerapkan transformasi objek geometris. Sebagian besar grafik direpresentasikan dalam bentuk matriks, diterapkan pada vektor dalam bentuk kartesian, dengan mengalikan vektor kolom dengan matriks transformasi [4].

Sistem koordinat homogen memungkinkan setiap koordinat diekspresikan sebagai koordinat homogen untuk merepresentasikan semua persamaan transformasi geometris dalam bentuk perkalian matriks. Matriks transformasi yang dihasilkan dapat diekspresikan dalam bentuk matriks umum. Dalam penerapan kali ini, koordinat homogen digunakan untuk memungkinkan penggunaan matriks translasi dan rotasi.

Untuk matriks translasi, idenya adalah dengan mengambil bidang w = 1 di ruang xyw.



Gambar 2.3 Bidang w = 1 di ruang xyw

Sumber: [3]

Setelah itu, kita dapat menganggap transformasi dilakukan oleh sebuah matriks 3x3 di bidang xwy. Masalahnya, hasil perkalian matrks seperti itu mungkin tidak menghasilkan nilai 1 pada elemen terakhirnya. Kita dapat membatasi perkalian kita untuk memastikan menghasilkan angka 1:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ p & q & r \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Agar persamaan ini berlaku untuk semua x dan y, harus ada $px + qy + r = 1$ untuk semua x, y, sehingga $p = q = 0$ dan $r = 1$.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Jika kita mencermati kasus khusus ketika bagian kiri-atas adalah matriks identitas 2x2, maka kita dapatkan

$$\begin{bmatrix} 1 & 0 & c \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + c \\ y + f \\ 1 \end{bmatrix}$$

Selama kita hanya memperhatikan koordinat x dan y, ini serupa dengan transformasi translasi. Oleh karena itu, daapat disimpulkan bahwa matriks transformasi untuk w = 1 adalah

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$$

dengan dx adalah perubahan di sumbu x dan dy adalah perubahan di sumbu y [3].

Konsep yang sama diterapkan kepada transformasi rotasi dan menghasilkan matriks rotasi berupa

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

dengan θ adalah sudut rotasi.

III. ANIMASI

Animasi merupakan teknik menciptakan ilusi gerak melalui serangkaian gambar yang ditampilkan secara berurutan dengan kecepatan tertentu. Animasi bekerja dengan memanfaatkan fenomena *persistence of vision*, yaitu kemampuan mata manusia untuk mempertahankan bayangan dari suatu objek untuk sepersekian detik setelah objek tersebut hilang dari pandangan. Fenomena ini memungkinkan manusia untuk melihat potongan potongan gambar animasi sebagai suatu kesatuan yang kontinu.

Dalam konteks teknis, animasi dapat didefinisikan sebagai proses menampilkan rangkaian gambar statis *frame* secara berurutan dengan kecepatan tertentu untuk menciptakan ilusi gerakan yang mulus. Standar *frame rate* yang umum digunakan dalam industri animasi adalah 24-30 frame per detik (fps), meskipun beberapa produksi menggunakan 12-15 fps untuk animasi terbatas.

Dalam sejarah perkembangannya, animasi telah berkembang menjadi beberapa kategori yang berbeda dengan karakteristik uniknya masing-masing. Animasi tradisional merupakan teknik yang mengandalkan kemampuan menggambar manual setiap framenya, menghasilkan gaya visual yang sangat khas dari setiap animatornya.

Seiring perkembangan teknologi, muncul animasi digital yang memanfaatkan software khusus animasi dan berbagai tools digital. Animasi digital ini tidak hanya mempercepat proses produksi tetapi juga memudahkan proses revisi dan penyimpanan karya. Kategori lainnya adalah animasi stop motion, sebuah teknik yang menggunakan objek fisik yang difoto frame per frame. Teknik ini membutuhkan tingkat presisi yang sangat tinggi dalam pergerakan objek, dan menghasilkan tekstur visual yang tidak dapat ditiru oleh teknik animasi lainnya. Masing-masing kategori ini memiliki keunggulan dan tantangan tersendiri serta terus berkembang seiring dengan kemajuan teknologi dan kebutuhan industri kreatif.

IV. OPENCV

Dalam implementasi, penulis menggunakan *library* OpenCV. OpenCV (Open Source Computer Vision) adalah sebuah *library* yang terdiri dari kumpulan fungsi programming yang sebagian besar digunakan untuk *computer vision* secara *real-time*. OpenCV memiliki banyak bidang penerapan, seperti fitur 2D dan 3D, estimasi egomotion, sistem pengenalan wajah, pengenalan gestur, interaksi manusia-komputer, deteksi objek, robotika *mobile*, dan lain lain.

IV. IMPLEMENTASI

Implementasi dalam bentuk kode program untuk makalah ini dibuat dalam bahasa python. Animasi yang dibuat adalah sepotong adegan sederhana yang menggambarkan seekor burung yang sedang terbang di langit berawan.

```
import cv2
import numpy as np
```

Gambar 4.1
Sumber: Arsip Penulis

Pada awal kode, dilakukan import untuk *library*. *Library* yang digunakan dalam kode. *Library* pertama adalah yang sudah disebutkan sebelumnya, yaitu OpenCV. Library ini akan digunakan untuk memuat dan memproses gambar dari *file* lokal dan menampilkan frame animasi. *Library* kedua adalah NumPy yang digunakan untuk operasi *array* dan matriks yang diperlukan dalam pemrosesan gambar dan transformasi geometri.

```
cloud = cv2.imread("cloud.png", cv2.IMREAD_UNCHANGED)
cloud = cv2.resize(cloud, (100, 60))
bird = cv2.imread("bird.png", cv2.IMREAD_UNCHANGED)
bird = cv2.resize(bird, (50, 50))
```

Gambar 4.2
Sumber: Arsip Penulis

Pada bagian ini, dua gambar yaitu gambar awan dan burung dimuat dengan parameter cv2.IMREAD_UNCHANGED yang mengecek apabila ada bagian dari gambar yang transparan dengan

memungkinkan pembacaan *channel* alpha. Gambar-gambar tersebut kemudian diubah ukurannya menggunakan untuk menyesuaikan dengan kebutuhan animasi.

```
width, height = 800, 400
sky = np.full((height, width, 3), (235, 206, 135), dtype=np.uint8)

num_clouds = 15
cloud_positions = np.array([
    [np.random.randint(0, width + 200),
     np.random.randint(50, height),
     1]
    for _ in range(num_clouds)
])
cloud_speeds = [
    -2, -3, -4, -2.5, -3.5, -2, -3, -5,
    -0.5, -3.5, -2, -3, -4, -2.5, -3.5
    ]

bird_position = np.array([0, 175, 1])
bird_angle = 0
bird_speed = np.array([5, 5, 0])
rotation_direction = 1
```

Gambar 4.3
Sumber: Arsip Penulis

Latar belakang dibuat dengan ukuran 800x400 piksel dan 3 channel warna (BGR). Warna latar belakang diset dengan nilai BGR (235, 206, 135) yang merepresentasikan warna langit. Setelah itu, posisi awan-awan di awal diinisialisasi secara acak menggunakan fungsi bawaan numpy. Kecepatan awan dan burung bergerak setiap framenya juga diinisialisasi. Selain itu, diinisialisasi juga posisi awal burung. Koordinat homogen digunakan di posisi awan dan burung untuk memungkinkan operasi transformasi geometri. Nilai w yang digunakan adalah 1 dan tetap 1 karena transformasi yang digunakan adalah translasi dan rotasi sehingga hanya memengaruhi posisi x dan y dalam ruang, tanpa mengubah skala atau sifat homogen titik tersebut.

```
def translation_matrix(tx, ty):
    return np.array([
        [1, 0, tx],
        [0, 1, ty],
        [0, 0, 1]
    ])
```

Gambar 4.4
Sumber: Arsip Penulis

Fungsi ini mengembalikan matriks transformasi translasi dalam koordinat homogen. Matriks 3x3 ini memungkinkan pergeseran objek dalam ruang 2D. tx dan ty merepresentasikan besar pergeseran pada sumbu x dan y. Penggunaan koordinat homogen memungkinkan representasi transformasi linier sebagai perkalian matriks.

```
def rotation_matrix(angle):
    rad = np.radians(angle)
    return np.array([
        [np.cos(rad), -np.sin(rad), 0],
        [np.sin(rad),  np.cos(rad), 0],
        [0,           0,           1]
    ])
```

Gambar 4.5
Sumber: Arsip Penulis

Fungsi ini mengembalikan matriks transformasi rotasi dalam koordinat homogen. Matriks 3x3 ini memungkinkan rotasi objek terhadap titik pusatnya dalam ruang 2D. Parameter *angle* merepresentasikan besar rotasi dalam derajat yang lalu diubah menjadi radian untuk perhitungan trigonometri. Penggunaan koordinat homogen memungkinkan representasi transformasi linier sebagai perkalian matriks.

```
def overlay_image(canvas, image, position, angle=None):
    image_h, image_w = image.shape[:2]
    x_start, y_start = int(position[0]), int(position[1])

    if angle is not None:
        center = np.array([image_w // 2, image_h // 2, 1])
        rotation_m = rotation_matrix(angle)
        rotated_image = np.zeros_like(image, dtype=np.uint8)
        for y in range(image_h):
            for x in range(image_w):
                original_coords = np.array([x, y, 1]) - center
                rotated_coords = rotation_m @ original_coords + center
                x_rot, y_rot = int(rotated_coords[0]), int(rotated_coords[1])
                if 0 <= x_rot < image_w and 0 <= y_rot < image_h:
                    rotated_image[y_rot, x_rot] = image[y, x]
        image = rotated_image

    for y in range(image_h):
        for x in range(image_w):
            if x_start + x < 0 or y_start + y < 0:
                continue
            if y_start + y >= height or x_start + x >= width:
                break
            alpha = image[y, x, 3] / 255.0 if image.shape[2] == 4 else 1.0
            if alpha > 0:
                canvas[y_start + y, x_start + x] = (
                    alpha * image[y, x, :3] + (1 - alpha) * canvas[y_start + y, x_start + x]
                ).astype(np.uint8)
```

Gambar 4.6
Sumber: Arsip Penulis

Fungsi overlay_image terdiri dari dua bagian yaitu rotasi gambar dan penggabungan gambar menggunakan alpha blending. Pada bagian pertama, jika parameter angle diberikan, fungsi ini akan menangani rotasi gambar. Proses dimulai dengan menentukan titik pusat rotasi dan membuat matriks rotasi. Setelah itu, setiap piksel pada gambar asli diubah posisinya menggunakan matriks rotasi tersebut. Koordinat baru dihitung dengan mengurangkan titik pusat, menerapkan transformasi rotasi, lalu menambahkan kembali titik pusat. Piksel hasil rotasi disimpan dalam buffer sementara yang menjadi gambar baru setelah proses selesai.

Bagian kedua fungsi berfokus pada alpha blending atau teknik untuk menggabungkan gambar dengan transparansi. Fungsi ini melakukan iterasi pada setiap piksel gambar dan memeriksa apakah piksel berada dalam batas canvas untuk mencegah kesalahan akses indeks. Jika gambar memiliki alpha channel, nilai alpha digunakan untuk menentukan tingkat transparansi piksel tersebut. Formula alpha blending diterapkan untuk menghitung nilai warna akhir pada canvas. alpha adalah nilai alpha yang dinormalisasi antara 0 dan 1. Proses ini memastikan bahwa piksel dari gambar sumber dan tujuan digabungkan secara mulus, menghasilkan komposit dengan transparansi yang akurat.

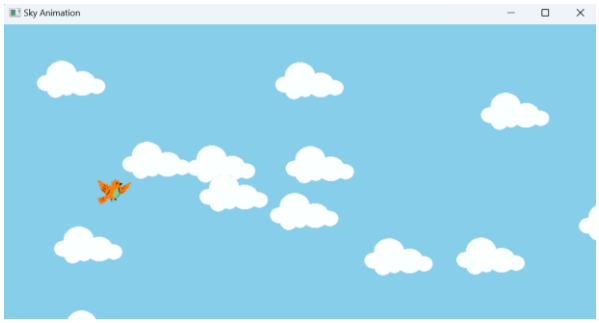
```
frames = 100
cv2.namedWindow("Sky Animation", cv2.WINDOW_AUTOSIZE)

for frame in range(frames):
    canvas = sky.copy()

    for i in range(num_clouds):
        cloud_positions[i] = translation_matrix(cloud_speeds[i], 0) @ cloud_positions[i]
        if cloud_positions[i, 0] < -100:
            cloud_positions[i, 0] = width + np.random.randint(50, height)
        overlay_image(canvas, cloud, cloud_positions[i])

    bird_position = translation_matrix(bird_speed[0], bird_speed[1]) @ bird_position
    if rotation_direction == 1 and bird_angle < 30:
        bird_angle += 5
    elif rotation_direction == -1 and bird_angle > -30:
        bird_angle -= 5
    else:
        rotation_direction *= -1

    if bird_position[1] > 175 :
        bird_speed = np.array([5, -5, 0])
    elif bird_position[1] < 150 :
        bird_speed = np.array([5, 5, 0])

    overlay_image(canvas, bird, bird_position, angle=bird_angle)

    cv2.imshow("Sky Animation", canvas)

cv2.destroyAllWindows()
```

Gambar 4.7
Sumber: Arsip Penulis

Inisialisasi awal dilakukan dengan menentukan jumlah frame animasi dan membuat *window* untuk menampilkan animasi menggunakan OpenCV. Pada setiap iterasi loop, program dimulai dengan membuat salinan baru dari latar belakang. Selanjutnya, awan diperbarui dengan menerapkan translasi horizontal sesuai kecepatannya menggunakan matriks translasi. Jika awan mencapai batas kiri layar, posisinya di-*reset* ke sisi kanan dengan jarak acak untuk menciptakan variasi. Setelah itu, awan digambar ke canvas menggunakan fungsi overlay_image.

Animasi burung dilakukan melalui kombinasi translasi dan rotasi. Posisi burung diperbarui berdasarkan kecepatannya menggunakan matriks translasi secara horizontal dan vertikal. Sudut rotasi burung diubah secara bertahap antara -30° hingga 30°. Ketika sudut mencapai batas atas atau bawah, arah rotasi dibalik untuk menciptakan gerakan berosilasi. Burung kemudian digambar ke canvas pada posisi dan sudut rotasi terbaru.

Di akhir setiap iterasi, frame yang telah direnderer ditampilkan dan program akan menampilkan setiap frame selama 1 milisekon. Setelah *loop* selesai, program membersihkan dan menutup *window* animasi.



Gambar 4.8
Sumber: Arsip Penulis

Hasil dari kode implementasi di makalah ini adalah sebuah adegan animasi sederhana yang menggambarkan burung yang sedang terbang di langit.

V. KESIMPULAN DAN SARAN

Makalah ini telah membahas perkembangan animasi dan grafika komputer, teori dasar seperti vektor, koordinat homogen, serta transformasi matriks, hingga implementasi sederhana animasi pendek menggunakan bahasa Python dan *library* OpenCV. Dari implementasi yang telah dilakukan, dapat disimpulkan bahwa konsep transformasi matriks dapat diterapkan dalam proses pembuatan animasi, khususnya di animasi dua dimensi.

Namun, implementasi yang dilakukan masih sangat sederhana yaitu hanya mengimplementasi animasi dasar yang melibatkan translasi dan rotasi objek 2D tanpa mengimplementasikan bentuk transfomasi yang lainnya seperti *scaling* dan *shearing*. Oleh karena itu, hasil makalah ini membuka peluang besar untuk implementasi lebih lanjut, seperti menambahkan bentuk transformasi yang lain, menambah dimensi menjadi animasi 3D, atau mengintegrasikan fitur lainnya. Pengembangan ini diharapkan dapat mendukung berbagai aplikasi kreatif, pendidikan, dan industri animasi di masa depan.

Selain itu, ada banyak *library* atau alat lain yang dapat digunakan untuk membuat program animasi selain OpenCV. Semua alat memiliki kelebihan dan kekurangannya sendiri, dan eksplorasi terhadap alat-alat yang sudah tersedia untuk membuat program animasi sesuai dengan kebutuhan sangat disarankan.

VI. UCAPAN TERIMA KASIH

Penulis menyampaikan ucapan terima kasih kepada:

1. Tuhan Yang Maha Esa, yang atas berkat-Nya penulis dapat menyelesaikan makalah ini,
2. Kedua orang tua penulis yang selalu memberikan dukungan kepada penulis,
3. Bapak Ir. Rinaldi Munir selaku dosen pengampu mata kuliah IF2123 Aljabar Linier dan Geometri kelas K2 yang telah membimbing penulis dan memberikan banyak ilmu yang bermanfaat selama perkuliahan.

Akhir kata, penulis mengharapkan makalah ini dapat bermanfaat bagi pihak yang membacanya..

REFERENCES

[1] Anton, H., & Rorres, C. (2014). Elementary Linear Algebra: Applications Version (11th ed.). Wiley.
[2] Parent, R. (2002). *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann.
[3] Foley, J. D., van Dam, A., McGuire, M., Sklar, D. F., Hughes, J. F, Feiner, S.K., & Akeley, K. (2013). *Computer Graphics: Principles and Practice*. Addison-Wesley Professional.
[4] www.geeksforgeeks.org/computer-graphics-homogeneous-coordinates/ diakses 28 Desember 2024
[5] https://nationalgeographic.grid.id/read/133345168/mengenang-bapak-animasi-yang-jenius-namun-berakhir-memilukan?page=all diakses 30 Desember 2023
[6] https://artlark.org/2022/08/17/fantasmagorie-the-first-ever-cartoon/ diakses 30 Desember 2024

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 30 Desember 2024

Mayla Yaffa Ludmilla 13523050