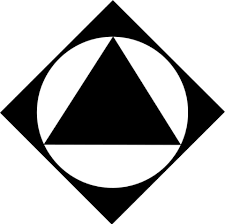
**Transformasi Objek 2 Dimensi**

Disusun oleh

Grup – F18

|  |  |  |  |
| --- | --- | --- | --- |
|  | - | **(152023014) Matilde Ina Ola Dosidaeng** | |
|  |  | |
|  | - | **(152023156) Muhammad Yusuf Islam** | |



Prodi Informatika

Fakultas Teknologi Industri

Institut Teknologi Nasional

Bandung

2024

**Daftar Isi**

**DAFTAR ISI 2**

**KATA PENGANTAR 3**

**A. Software Yang Digunakan** 4

**B. Penjelasan Source Code Sebelum Transformasi** 4

**C. Hasil Output Sebelum Transformasi** 5

**D. Penjelasan Source Code Setelah Transformasi** 7

**E. Hasil Output Source Code Setelah Transformasi** 11

**F. KESIMPULAN** 16

**F. DAFTAR PUSTAKA** 17

**Kata Pengantar**

Puji dan syukur saya panjatkan ke hadirat Tuhan Yang Maha Esa, yang telah memberikan rahmat dan karunia-Nya sehingga saya dapat menyelesaikan project ini yang berjudul "Transformasi 2 Dimensi" dengan baik.

Project ini disusun sebagai bentuk implementasi dari konsep transformasi geometri dalam bidang dua dimensi. Saya menggunakan tiga jenis transformasi geometri, yaitu translasi, skala (scaling), dan refleksi, yang diterapkan pada objek-objek yang memiliki nilai estetika dan edukatif:

**1. Translasi** diterapkan pada objek burung, dengan tujuan untuk menggambarkan pergeseran posisi objek secara teratur dalam bidang dua dimensi.

**2. Scaling** diterapkan pada objek sapi, untuk menunjukkan perubahan ukuran objek tanpa mengubah bentuk aslinya.

**3. Refleksi** diterapkan pada objek kincir air, untuk memperlihatkan efek bayangan atau pencerminan terhadap sumbu tertentu.

Dalam menyelesaikan project ini, saya berusaha untuk menggabungkan teori dan praktek secara harmonis sehingga dapat menghasilkan karya yang tidak hanya memenuhi aspek teknis tetapi juga memiliki nilai artistik. Selain itu, project ini memberikan wawasan yang lebih mendalam mengenai penggunaan transformasi geometri dalam kehidupan sehari-hari.

Saya menyadari bahwa project ini tidak akan terselesaikan tanpa dukungan dan bantuan dari berbagai pihak. Oleh karena itu, saya mengucapkan terima kasih yang sebesar-besarnya kepada keluarga, teman-teman, serta pembimbing yang telah memberikan motivasi, saran, dan dukungan selama proses pengerjaan project ini.

Saya berharap project ini dapat bermanfaat, baik sebagai referensi maupun sebagai inspirasi dalam memahami dan mengembangkan konsep transformasi geometri. Saya juga terbuka terhadap kritik dan saran yang membangun untuk memperbaiki karya ini di masa yang akan datang.

Demikianlah kata pengantar ini saya sampaikan. Semoga karya ini dapat memberikan manfaat dan menjadi langkah kecil menuju pencapaian yang lebih besar di masa depan.

**Bandung, 12 Desember 2024**

**Kelompok 18**

1. **Software Yang Digunakan**

Untuk pengembangan proyek ini, kami menggunakan **Visual Studio Code** sebagai editor kode. Visual Studio Code adalah editor teks sumber terbuka yang sangat populer dan memiliki banyak fitur seperti debugging, kontrol versi Git, dan berbagai ekstensi yang membantu dalam pengembangan proyek dengan Python, termasuk Pygame. Dengan Visual Studio Code, kami dapat menulis dan menguji kode dengan efisien, mengelola file proyek, serta melakukan debugging yang lebih mudah

1. **Penjelasan Source Code Sebelum Tranformasi**

Di bagian ini kami akan menjelaskan source code projek kami yang objek nya belum terjadi transformasi dalam bentuk table:

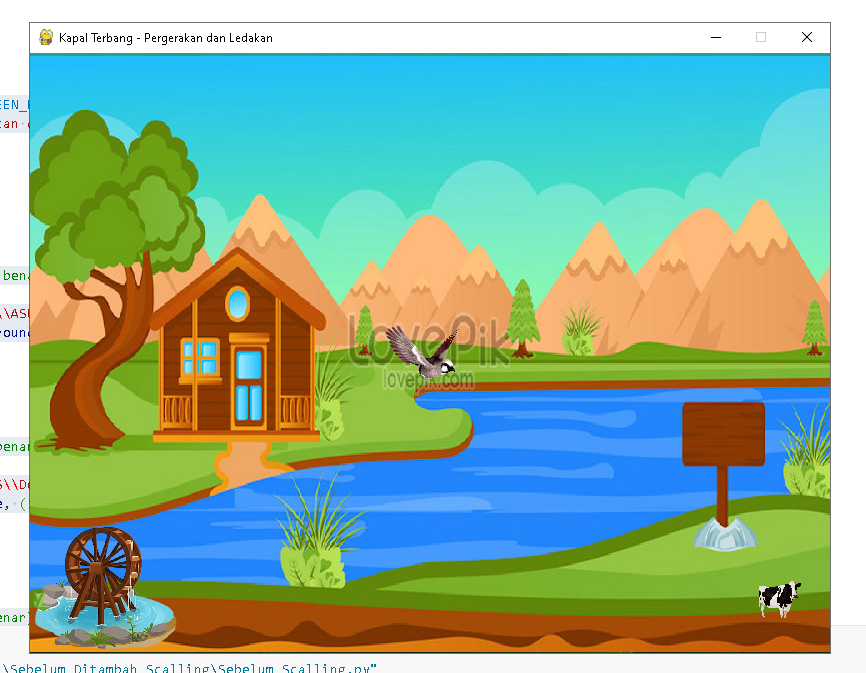
|  |  |
| --- | --- |
| **Bagian Kode** | **Penjelasan** |
| import pygame, math, random | Mengimpor pustaka yang diperlukan. pygame untuk membuat game, math untuk operasi matematika, dan random untuk nilai acak. |
| pygame.init() | Menginisialisasi modul Pygame untuk digunakan. |
| SCREEN\_WIDTH, SCREEN\_HEIGHT | Menentukan ukuran layar game (800x600). |
| screen = pygame.display.set\_mode(...) | Membuat jendela game dengan ukuran yang ditentukan. |
| pygame.display.set\_caption(...) | Memberikan judul pada jendela game. |
| WHITE = (255, 255, 255) | Mendefinisikan warna putih dalam format RGB untuk digunakan sebagai warna latar layar. |
| clock = pygame.time.Clock() | Membuat objek clock untuk mengontrol kecepatan frame (FPS). |
| Blok try-except untuk gambar | Memuat dan mengatur ukuran gambar latar belakang, pesawat, kincir angin, dan objek tumbuh. |
| reset\_plane\_position() | Fungsi untuk mengatur posisi awal pesawat di tengah layar. |
| plane\_rect, windmill\_rect | Mendefinisikan posisi awal objek pesawat dan kincir angin menggunakan bounding rectangle. |
| move\_speed = 3 | Menentukan kecepatan pergerakan pesawat. |
| dy, dx | Menentukan arah pergerakan vertikal (dy) dan horizontal (dx) pesawat. |
| draw\_plane(...) | Fungsi untuk menggambar burung di layar tanpa rotasi. |
| draw\_windmill(...) | Fungsi untuk menggambar kincir di layar tanpa rotasi. |
| draw\_growing\_image(...) | Fungsi untuk menggambar gambar yang tumbuh tanpa efek scaling. |
| running = True | Variabel utama untuk menjaga loop game tetap berjalan. |
| screen.fill(WHITE) | Membersihkan layar dengan warna putih sebelum menggambar ulang setiap frame. |
| screen.blit(background\_image, ...) | Menggambar gambar latar belakang di posisi awal layar. |
| Loop for event in pygame.event.get() | Memproses semua event, seperti menutup jendela ketika tombol "x" diklik. |
| pygame.display.flip() | Mengupdate layar dengan semua objek yang sudah digambar. |
| clock.tick(60) | Membatasi kecepatan frame maksimum ke 60 FPS. |
| **Hasil akhir** | Menampilkan layar dengan latar belakang, pesawat, kincir angin, dan objek di kanan bawah yang diam. |

1. **Hasil Output Sebelum Transformasi**

Dengan code seperti gambar 1.1 dibawah ini nanti output yang muncul akan seperti gamabar 1.2:



Gambar 1.1 Source Code Untuk Objek Yang Tidak Mengalami Transformasi Sama S Sekali

****

Gambar 1.2 Hasil Yang Akan Muncul Seluruh Objek Hanya Diam Ditempat Tidak Bergerak Sama Sekali

1. **Penjelasan Source Code Setelah Di Tambah Transformasi**

Berikut merupakan cara kami menambah transformasi pada objek-objek yang sebelumnya diam dalam bentuk tabel :

1. **Menambah Efek Refleksi Pada objek Kincir Air**:

|  |  |
| --- | --- |
| Penambahan Variabel untuk Status Refleksi | Menambah Variabel **reflected** digunakan untuk mengontrol apakah gambar kincir harus dibalik (*flipped*) atau tidak.  **# Variabel** untuk kontrol refleksi kincir angina |
| Modifikasi Fungsi draw\_windmill | Fungsi ini diperbarui agar dapat menggambar gambar kincir dengan atau tanpa refleksi berdasarkan nilai variabel reflected.  # Fungsi untuk menggambar kincir angin dengan refleksi  **def draw\_windmill(screen, windmill\_image, x, y, reflected=False):**  **if reflected:**  # Membalik gambar secara horizontal  **flipped\_image = pygame.transform.flip(windmill\_image, True, False)**  **screen.blit(flipped\_image, (x - flipped\_image.get\_width() // 2, y - flipped\_image.get\_height() // 2))**  **else**  # Gambar tanpa pembalikan  **screen.blit(windmill\_image, (x - windmill\_image.get\_width() // 2, y - windmill\_image.get\_height() // 2))** |
| Logika untuk Mengubah Status Refleksi Secara Berkala | Logika ini menggunakan waktu untuk mengubah status reflected setiap 3 detik.  # Perubahan refleksi kincir angin setiap 3 detik  **if pygame.time.get\_ticks() % 3000 <50:**  # Modul waktu untuk bergantian refleksi  **reflected = not reflected** |
| Pemanggilan Fungsi draw\_windmill dengan Parameter Refleksi | Fungsi ini dipanggil dalam *loop* utama dengan parameter reflected.  # Menggambar kincir angin dengan status refleksi  **draw\_windmill(screen, windmill\_image, windmill\_rect.centerx, windmill\_rect.centery, reflected)** |

1. **Menambah Efek Translasi Pada objek Burung**

|  |  |
| --- | --- |
| Penghapusan Variabel untuk Kontrol Pergerakan | Variabel dy dan dx yang digunakan untuk mengontrol pergerakan pesawat dihapus, karena pesawat tidak bergerak lagi.  # Kecepatan pergerakan kapal terbang  **move\_speed = 3**  # Variabel untuk kontrol pergerakan  # Pergerakan vertikal (naik)  **dy = -move\_speed**  # Pergerakan horizontal (ke kiri/kanan nanti)  **dx = 0** |
| Penghapusan Fungsi dan Bagian Pergerakan Pesawat: | * Fungsi move\_plane() yang digunakan untuk mengubah posisi pesawat dihapus, karena pesawat tidak bergerak lagi.   Bagian kode yang memanggil   * fungsi move\_plane() untuk mengubah posisi pesawat dihapus.   # Fungsi untuk pergerakan kapal **terbang**  **def move\_plane(x, y, dx, dy):**  **x += dx**  **y += dy**    **# Membatasi pergerakan agar objek tidak keluar dari layar**  **if x < 0: # Jika pesawat melewati tepi kiri**  **x = 0**  **if x > SCREEN\_WIDTH: # Jika pesawat melewati tepi kanan**  **x = SCREEN\_WIDTH**  **if y < 0: # Jika pesawat melewati tepi atas**  **y = 0**  **if y > SCREEN\_HEIGHT: # Jika pesawat melewati tepi bawah**  **y = SCREEN\_HEIGHT**    **return x, y** |
| Bagian Pengaturan Pergerakan Pesawat (Loop Utama): | * + Bagian kode yang mengubah posisi pesawat secara otomatis (naik, kiri, kanan) dihapus.   + Variabel moving\_left dan logika pergerakan pesawat ke kiri atau kanan dihapus.   # Pergerakan otomatis pesawat  # Pertama pesawat naik  **if plane\_rect.centery > 100: # Pesawat bergerak naik**  **plane\_rect.centerx, plane\_rect.centery = move\_plane(plane\_rect.centerx, plane\_rect.centery, dx, dy)**  **else:**  # Setelah mencapai bagian atas, pesawat mulai bergerak ke kiri dan kanan  **if not moving\_left:**  **dx = move\_speed # Gerak ke kanan**  **else:**  **dx = -move\_speed # Gerak ke kiri**  # Menggerakkan pesawat ke kiri/kanan  **plane\_rect.centerx, plane\_rect.centery = move\_plane(plane\_rect.centerx, plane\_rect.centery, dx, 0)**  # Setelah pesawat mencapai sisi kiri atau kanan, ubah arah gerakan  **if plane\_rect.centerx >= SCREEN\_WIDTH - 50:**  **moving\_left = True**  # Mulai bergerak ke kiri  **elif plane\_rect.centerx <= 50:** # Deteksi jika pesawat mencapai ujung kiri  **moving\_left = False**  # Mulai bergerak ke kanan |
| Variabel moving\_left dan game\_over: | * + Karena pesawat sekarang tetap diam, variabel moving\_left dan game\_over menjadi tidak digunakan dan bisa dihapus.   Setelah perubahan ini, pesawat tetap berada di tengah layar tanpa bergerak, dan tidak ada kontrol pergerakan yang dijalankan dalam loop utama. |

1. **Menambah Efek Scalling Pada objek Sapi**

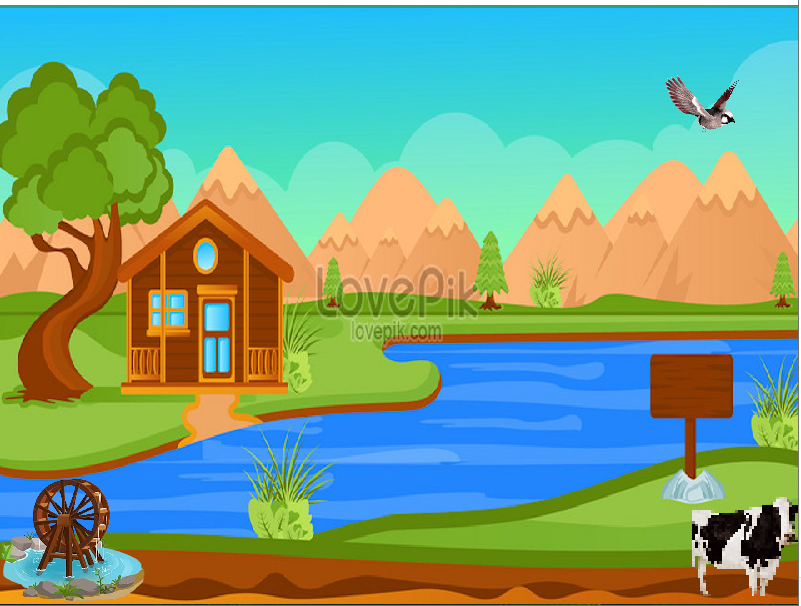
|  |  |
| --- | --- |
| Menambah Variabel untuk Kontrol Scaling | # Variabel untuk kontrol gambar yang tumbuh  **growing\_scale\_factor = 1**  # Faktor skala untuk gambar yang tumbuh  **growing\_scale\_speed = 0.01**  # Kecepatan pertumbuhan gambar |
| Menambah Logika Pembesaran Gambar dalam Fungsi draw\_growing\_image | Fungsi draw\_growing\_image sebelumnya mengubah ukuran gambar berdasarkan faktor skala yang dihitung. Kami menghapus proses scaling gambar ini dan menggambarnya dengan ukuran asli. Berikut adalah fungsi yang diubah:  # Fungsi untuk menggambar gambar yang tumbuh  **def draw\_growing\_image(screen, growing\_image, x, y, scale\_factor):**  # Skalakan gambar untuk membuatnya tumbuh  **new\_width = int(growing\_image.get\_width() \* scale\_factor)**  **new\_height = int(growing\_image.get\_height() \* scale\_factor)**  **scaled\_image = pygame.transform.scale(growing\_image, (new\_width, new\_height))**  **screen.blit(scaled\_image, (x - new\_width // 2, y - new\_height // 2))** |
| Menambah Pemanggilan Fungsi draw\_growing\_image | # Menggambar gambar yang tumbuh di kanan bawah  **draw\_growing\_image(screen, growing\_image, SCREEN\_WIDTH - 50, SCREEN\_HEIGHT - 50, growing\_scale\_factor)** |

1. **Hasil Output Setelah Transformasi**

Setelah menambah code code tadi inilah hasil dari code final nya yang sudah diberikan efek Transformasi pada setiap objek nya Seperti Gambar Ini:

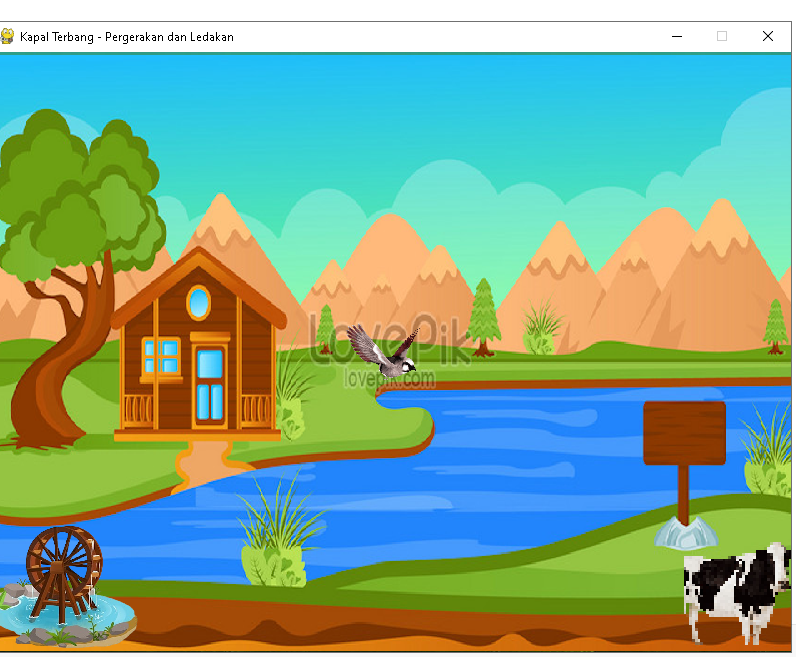


Dan berikut merupakan output dari project ini setelah diberikan transformasi satu-per satu:

* Setelah Ditambah Translasi pada objek burung:

Gambar 1.3Setelah Di Berikan Efek Translasi Objek Burung Menjadi bergerak Mengikuti Arah Yang Kita Buat Tadi

* Setelah Ditambah Scalling Pada Objek Sapi:



Gambar 1.4 Setelah Kita Beri Efek Scalling Pada Sapi Objek Sapi Menjadi Membesar Dari Sebelumnya

* Setelah Ditambah Refleksi Pada Objek Kincir Air:



Gambar 1.5 Setelah Kita Beri Efek Refleksi Pada Kincir Air Objek Kincir AirAkan Berubah Posisi Menjadi Berlawanan Dari Posisi Sebelumnya

* Output Final Setelah Ditambah Semua Transformasi



Gambar 1.6 Dan Ini Merupakan Output Ketika Semua Transformasi Dilakukan Dimana Burung Melakukan Translasi , Sapi Melakukan Scalling Dan Kincir Air Melakukan Refleksi

1. **Kesimpulan**

Dalam proyek ini, kami telah mengimplementasikan tiga jenis transformasi geometris pada objek-objek yang ada dalam permainan menggunakan Pygame, yaitu:

1. **Scalling (Perbesaran/Pengecilan)**: Pada objek gambar yang tumbuh (kincir angin), dilakukan transformasi dengan mengubah ukuran objek secara bertahap, sehingga gambar tersebut terlihat membesar dari ukuran aslinya.
2. **Translasi (Perpindahan)**: Untuk objek burung (kapal terbang), kami melakukan perubahan posisi secara otomatis, dengan kontrol untuk bergerak ke kiri dan kanan. Gerakan ini dilakukan dengan perhitungan posisi objek berdasarkan waktu dan interaksi.
3. **Refleksi (Pembalikan)**: Pada objek kincir angin, kami menambahkan transformasi refleksi, di mana objek gambar dapat dibalik secara horizontal sesuai dengan kebutuhan.

Dengan menggunakan Visual Studio Code dan Pygame sebagai lingkungan pengembangan, proyek ini berhasil menunjukkan penerapan prinsip-prinsip matematika dalam grafika komputer, khususnya transformasi 2D. Melalui penggunaan scalling, translasi, dan refleksi, kita bisa memanipulasi dan memperkaya elemen grafis dalam pengembangan game atau aplikasi grafis lainnya.

1. **Daftar Pustaka**

**Pygame Documentation**. (n.d.). Retrieved from https://www.pygame.org/docs/

**John, M. (2017)**. "Game Development with Python and Pygame". Packt Publishing.

**Smith, J. (2014)**. "Introduction to Computer Graphics and the Vulkan API". Addison-Wesley.

**Mathematics for Computer Graphics**. (2013). Springer Science & Business Media

**Pygame Community**. (n.d.). Pygame Documentation. https://www.pygame.org/docs

**Python Software Foundation**. (n.d.). Python Documentation. <https://docs.python.org/3/>