

LAPORAN PROJECT UAS
SLIDE PUZZLE KELOMPOK 6 2023C



Dosen Pengampu :

I Gde Agung Sri Sidhimantra S.Kom., M.Kom.

Binti Kholifah, S.Kom., M.Tr.Kom.

Moch Deny Pratama, S.Tr.Kom., M.Kom.

Dimas Novian Aditia Syahputra, S.Tr.T., M.Tr.T.

Disusun Oleh :

1. Dickrullah Brilian Akbar (23091397087/2023C)
2. Atika Haniifatun Nisa' (23091397098/2023C)
3. Roberto Christian (23091397101/2023C)

Program Studi D4 Manajemen Informatika

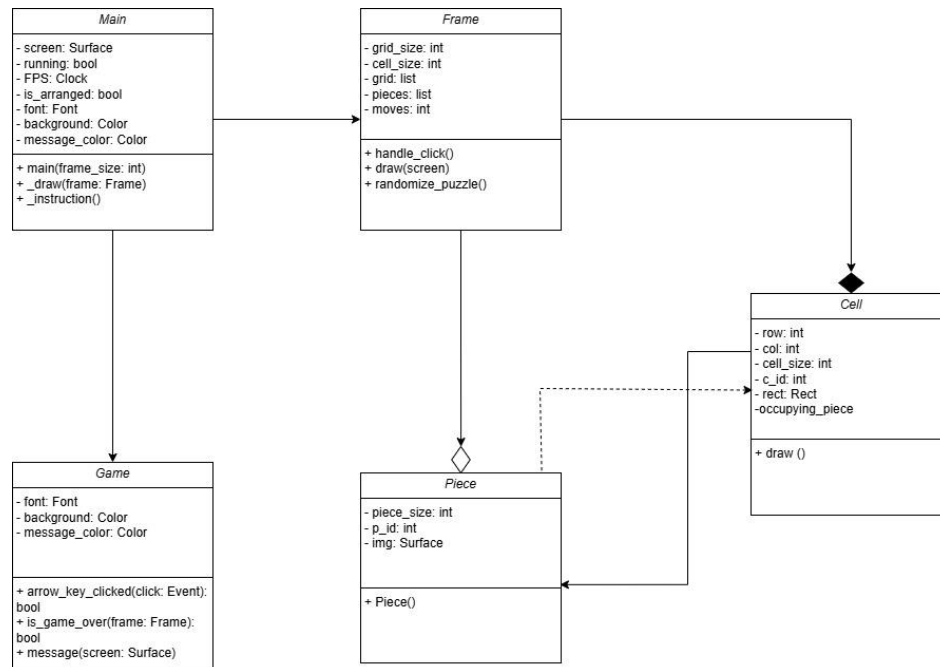
Fakultas Vokasi

Universitas Negeri Surabaya 2023/2024

Pengertian

Slide Puzzle adalah aplikasi permainan sederhana yang mengimplementasikan logika permainan susun puzzle menggunakan bahasa pemrograman Python dan library Pygame. Permainan ini bertujuan untuk menyusun potongan-potongan puzzle agar sesuai dengan urutan yang benar. Proyek ini dirancang untuk mengasah kemampuan logika dan visualisasi pemain dalam menyelesaikan masalah.

Class Diagram



Fitur Game

A. Tampilan Sederhana

- Antarmuka permainan dirancang dengan tampilan sederhana menggunakan warna yang mudah dibaca.
- Instruksi permainan ditampilkan dengan jelas di layar.

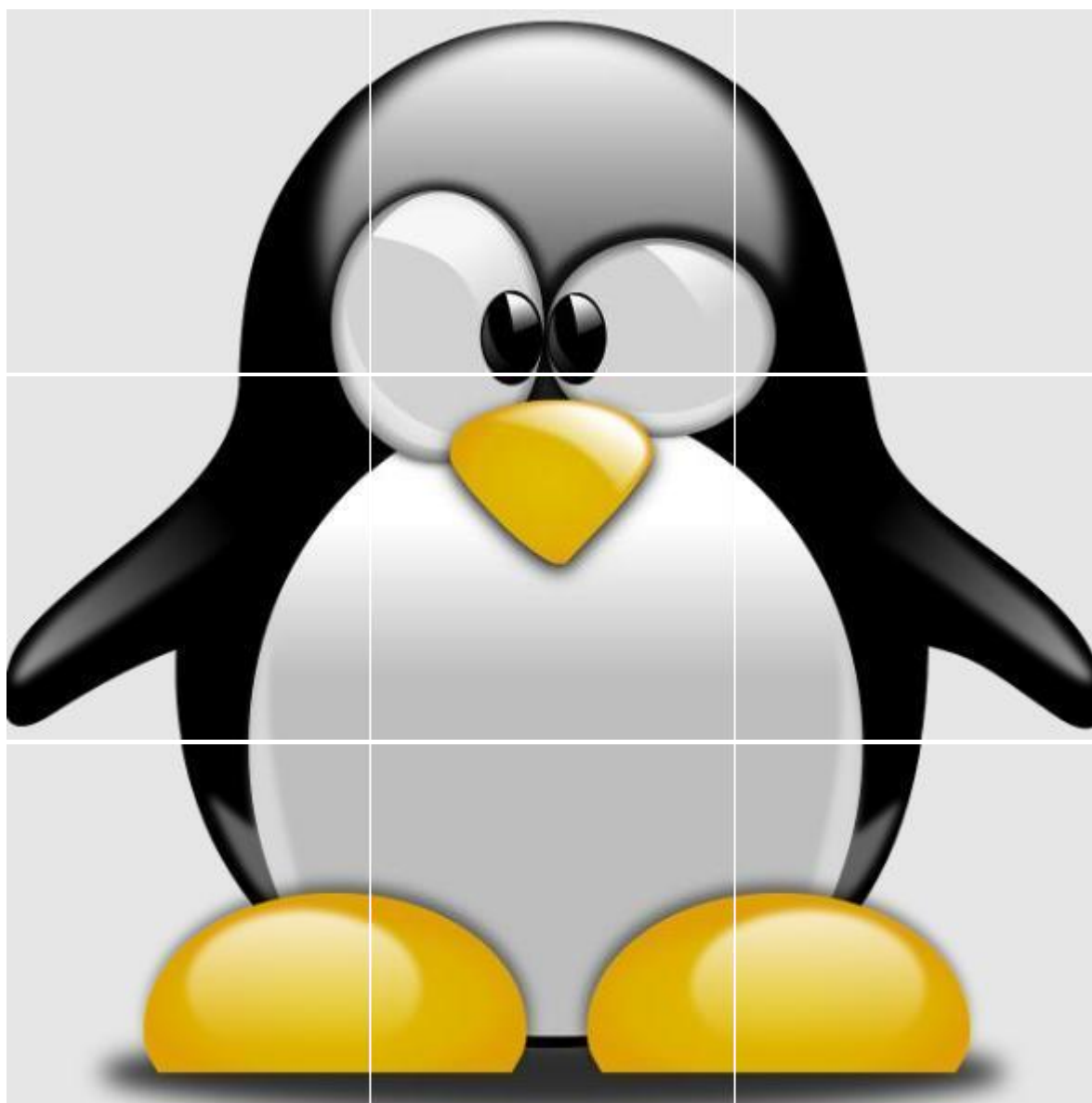
B. Interaksi Yang Dinamis

- Pemain menggunakan tombol panah pada keyboard untuk memindahkan elemen puzzle.
- Program secara otomatis memeriksa apakah puzzle sudah tersusun dengan benar.

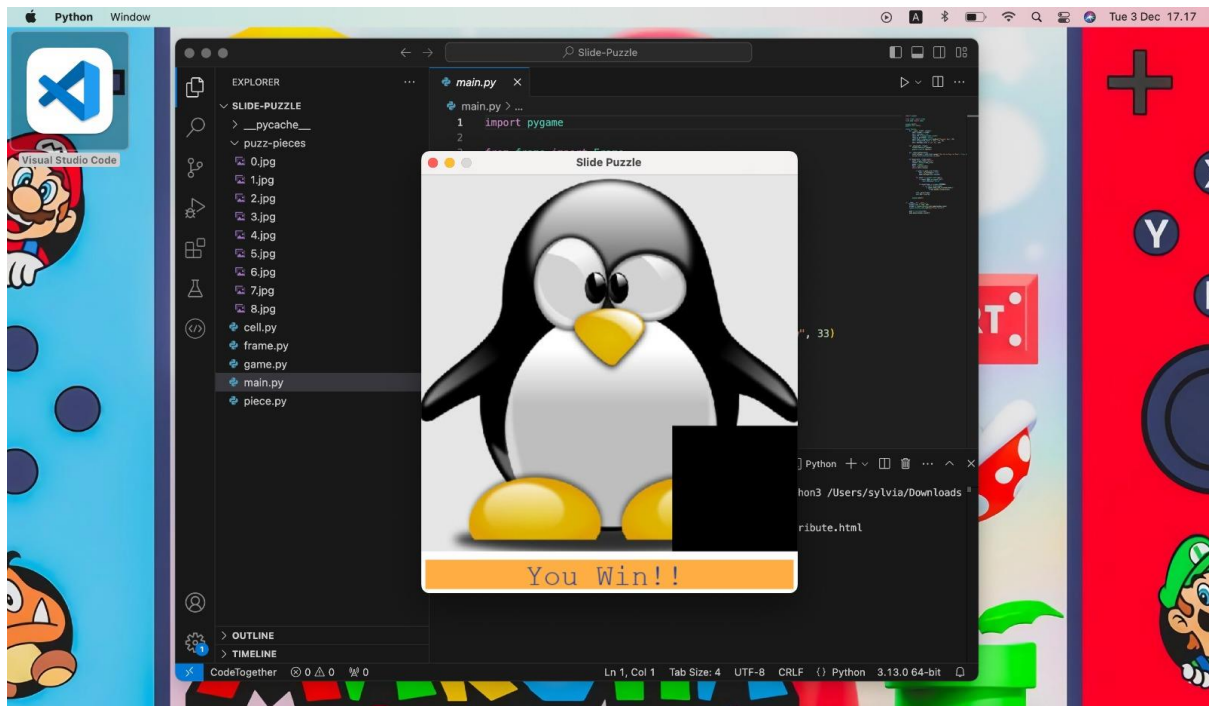
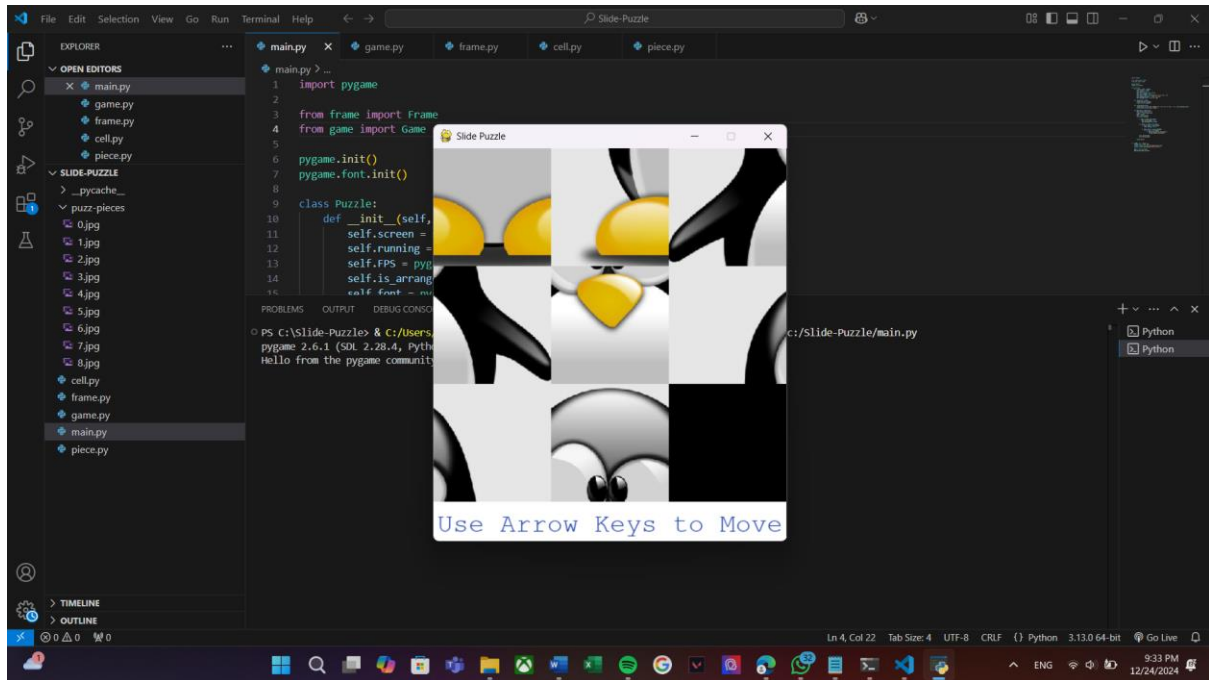
C. Sistem Winner

- Program memberikan pesan kepada pemain jika puzzle berhasil terselesaikan.

PUZZ-PIECES



Hasil Game



LINK GITHUB

<https://github.com/Dickrullah/UAS-OOP>

Code dan Penjelasan

CELL.PY

```
1  # /* cell.py
2  import pygame
3
4  class Cell:
5      def __init__(self, row, col, cell_size, c_id):
6          self.row = row
7          self.col = col
8          self.cell_size = cell_size
9          self.width = self.cell_size[0]
10         self.height = self.cell_size[1]
11         self.abs_x = row * self.width
12         self.abs_y = col * self.height
13         self.c_id = c_id
14         self.rect = pygame.Rect(
15             self.abs_x,
16             self.abs_y,
17             self.width,
18             self.height
19         )
20         self.occupying_piece = None
21
22     def draw(self, display):
23         pygame.draw.rect(display, (0,0,0), self.rect)
24         if self.occupying_piece != None and self.occupying_piece.p_id != 8:
25             centering_rect = self.occupying_piece.img.get_rect()
26             centering_rect.center = self.rect.center
27             display.blit(self.occupying_piece.img, centering_rect.topleft)
```

1. Import Pygame

`import pygame` digunakan untuk mengimpor library Pygame yang menyediakan berbagai fungsi dan modul untuk membuat game dan aplikasi grafis 2D. Pygame akan digunakan untuk menggambar objek pada layar dan menangani berbagai elemen grafis, seperti sel dan potongan yang menempati sel tersebut.

2. Metode `__init__` (Inisialisasi Kelas Cell)

Metode `__init__` adalah konstruktor yang digunakan untuk menginisialisasi objek `Cell`. Pada metode ini, terdapat beberapa parameter yang diperlukan: `row` (baris), `col` (kolom), `cell_size` (ukuran sel), dan `c_id` (ID sel).

- **Atribut** `self.row`, `self.col`, `self.cell_size`: Menyimpan informasi tentang baris dan kolom sel, serta ukuran sel.
- **Atribut** `self.width`, `self.height`: Mengambil dimensi lebar dan tinggi dari `cell_size` yang diberikan, digunakan untuk menentukan ukuran sel.
- **Atribut** `self.abs_x`, `self.abs_y`: Menghitung posisi absolut dari sel di layar berdasarkan posisi baris dan kolom serta ukuran sel. Ini digunakan untuk memposisikan sel dengan tepat di layar.
- **Atribut** `self.rect`: Membuat objek `pygame.Rect` yang digunakan untuk menggambar persegi panjang pada layar yang mewakili sel. `pygame.Rect` menerima parameter posisi `x`, `y`, lebar, dan tinggi untuk menentukan posisi dan ukuran sel.
- **Atribut** `self.occupying_piece`: Menyimpan referensi ke objek yang menempati sel tersebut (misalnya, potongan catur). Awalnya diset ke `None`, yang berarti sel tersebut kosong.

3. Metode draw

Metode `draw` bertugas untuk menggambar sel di layar.

- **Menggambar Sel**: `pygame.draw.rect(display, (0,0,0), self.rect)` menggambar sebuah persegi panjang berwarna hitam dengan ukuran dan posisi sesuai dengan objek `rect` yang telah didefinisikan sebelumnya. Ini menggambarkan sel sebagai sebuah kotak pada layar.
- **Menampilkan Potongan yang Menempati Sel**: Jika ada objek yang menempati sel (`self.occupying_piece` tidak `None`), dan objek tersebut memiliki ID yang tidak sama dengan 8 (`self.occupying_piece.p_id != 8`), maka objek tersebut akan digambar di dalam sel.
 - Gambar objek yang menempati sel (`self.occupying_piece.img`) diposisikan agar terpusat di tengah sel dengan cara mengatur posisi gambar menggunakan `centering_rect.center = self.rect.center`.
 - `display.blit(self.occupying_piece.img, centering_rect.topleft)` digunakan untuk menggambar gambar objek pada posisi yang telah dihitung sebelumnya. `blit` adalah metode Pygame untuk menggambar gambar di layar pada posisi yang diberikan.

Metode `draw` secara keseluruhan bertanggung jawab untuk menggambar sel yang dapat berisi objek di dalamnya, serta memposisikan objek tersebut agar terlihat dengan benar di tengah sel sesuai dengan grid yang ada.

FRAME.PY

```
1 import pygame
2 import random
3
4 from cell import Cell
5 from piece import Piece
6
7 class Frame:
8     def __init__(self, frame_size):
9         self.grid_size = 3
10        self.cell_width = frame_size // self.grid_size
11        self.cell_height = frame_size // self.grid_size
12        self.cell_size = (self.cell_width, self.cell_height)
13
14        self.grid = self._generate_cell()
15        self.pieces = self._generate_piece()
16
17        self._setup()
18        self.randomize_puzzle()
19
20    def _generate_cell(self):
21        cells = []
22        c_id = 0
23        for col in range(self.grid_size):
24            new_row = []
25            for row in range(self.grid_size):
26                new_row.append(Cell(row, col, self.cell_size, c_id))
27                c_id += 1
28            cells.append(new_row)
29        return cells
30
31    def _generate_piece(self):
32        puzzle_pieces = []
33        p_id = 0
34        for col in range(self.grid_size):
35            for row in range(self.grid_size):
36                puzzle_pieces.append(Piece(self.cell_size, p_id))
37                p_id += 1
38        return puzzle_pieces
39
40    def _setup(self):
41        for row in self.grid:
42            for cell in row:
43                tile_piece = self.pieces[-1]
44                cell.occupying_piece = tile_piece
45                self.pieces.remove(tile_piece)
46
```

```

1  def randomize_puzzle(self):
2      moves = [(0, 1), (0, -1), (1, 0), (-1, 0)]
3      for i in range(30):
4          shuffle_move = random.choice(moves)
5          for row in self.grid:
6              for cell in row:
7                  tile_x = self.grid.index(row) + shuffle_move[0]
8                  tile_y = row.index(cell) + shuffle_move[1]
9                  if tile_x >= 0 and tile_x <= 2 and tile_y >= 0 and tile_y <= 2:
10                     new_cell = self.grid[tile_x][tile_y]
11                     if new_cell.occupying_piece.img == None:
12                         c = (cell, new_cell)
13                         try:
14                             c[0].occupying_piece, c[1].occupying_piece = c[1].occupying_piece, c[0].occupying_piece
15                         except:
16                             return False
17                     else:
18                         continue
19
20 def _is_move_valid(self, click):
21     moves = {
22         79: (0, 1),
23         80: (0, -1),
24         81: (1, 0),
25         82: (-1, 0)
26     }
27     for row in self.grid:
28         for cell in row:
29             move = moves[click.scancode]
30             tile_x = self.grid.index(row) + move[0]
31             tile_y = row.index(cell) + move[1]
32             if tile_x >= 0 and tile_x <= 2 and tile_y >= 0 and tile_y <= 2:
33                 new_cell = self.grid[tile_x][tile_y]
34                 if new_cell.occupying_piece.img == None:
35                     return (cell, new_cell)
36             else:
37                 continue
38
39 def handle_click(self, click):
40     c = self._is_move_valid(click)
41     try:
42         c[0].occupying_piece, c[1].occupying_piece = c[1].occupying_piece, c[0].occupying_piece
43     except:
44         return False
45
46 def draw(self, display):
47     for row in self.grid:
48         for cell in row:
49             cell.draw(display)

```

1. Import Pygame dan Random

import pygame mengimpor library Pygame yang digunakan untuk penggambaran grafis dan elemen game, sementara import random digunakan untuk menghasilkan gerakan acak pada puzzle (misalnya, untuk mengacak posisi potongan puzzle).

2. Kelas Frame

Kelas Frame bertanggung jawab untuk mengelola tampilan dan logika dari papan puzzle, serta pengaturan dan pengacakan potongan-potongan puzzle. Kelas ini menginisialisasi grid (papan puzzle) dan potongan-potongan yang ada di dalamnya.

3. Metode `__init__` (Inisialisasi Kelas Frame)

Metode ini adalah konstruktor yang digunakan untuk menginisialisasi objek Frame dan pengaturan awal grid serta potongan-potongan puzzle.

- **Atribut:**
 - `self.grid_size`: Menentukan ukuran grid puzzle (3x3).
 - `self.cell_width` dan `self.cell_height`: Menentukan ukuran sel berdasarkan ukuran frame yang diberikan.
 - `self.cell_size`: Kombinasi lebar dan tinggi sel.
 - `self.grid`: Memanggil metode `_generate_cell()` untuk membuat grid yang berisi objek Cell.
 - `self.pieces`: Memanggil metode `_generate_piece()` untuk membuat daftar potongan puzzle (Piece).
- Setelah inisialisasi, dua metode dipanggil: `_setup()` untuk menempatkan potongan di sel-sel, dan `randomize_puzzle()` untuk mengacak posisi potongan.

4. Metode `_generate_cell`

Metode ini digunakan untuk menghasilkan grid 3x3 dari objek Cell.

- **Proses:**
 - Membuat grid kosong dan mengisi setiap sel dengan objek Cell.
 - Setiap objek Cell diberi ID unik (`c_id`), dan ID ini akan meningkat setiap kali sel baru dibuat.
 - Grid ini diorganisir dalam bentuk list of lists (matriks) yang berisi objek Cell.

5. Metode `_generate_piece`

Metode ini digunakan untuk menghasilkan potongan-potongan puzzle (Piece).

- **Proses:**
 - Membuat list kosong untuk menampung potongan-potongan puzzle.
 - Untuk setiap sel dalam grid, objek Piece baru dibuat dan ditambahkan ke dalam list `puzzle_pieces`.
 - Setiap objek Piece diberi ID unik (`p_id`), yang juga meningkat untuk setiap potongan baru.

6. Metode `_setup`

Metode ini digunakan untuk mengatur setiap sel pada grid dengan potongan-potongan puzzle yang telah dibuat.

- **Proses:**
 - Iterasi melalui setiap baris dan kolom dalam grid.

- Potongan puzzle terakhir dari daftar pieces dipindahkan ke dalam sel yang bersangkutan.
- Setelah setiap potongan ditempatkan di sel, potongan tersebut dihapus dari daftar pieces.

7. Metode `randomize_puzzle`

Metode ini digunakan untuk mengacak posisi potongan-potongan puzzle secara acak.

- **Proses:**

- Daftar moves berisi kemungkinan gerakan: atas, bawah, kiri, dan kanan.
- Selama 30 iterasi, sebuah gerakan acak dipilih dari daftar moves.
- Untuk setiap sel dalam grid, perhitungan posisi sel yang bisa dipindahkan ke posisi kosong dilakukan, dan potongan yang ada di sana akan dipindahkan ke posisi tersebut.

8. Metode `_is_move_valid`

Metode ini digunakan untuk memeriksa apakah gerakan yang diminta oleh pemain valid, yakni apakah potongan puzzle dapat dipindahkan ke posisi kosong yang ada di grid.

- **Proses:**

- Berdasarkan klik pengguna (yang diterjemahkan ke dalam kode pindah menggunakan `click.scancode`), metode ini mencari sel yang bisa dipindahkan dan memiliki potongan kosong.
- Jika ada posisi yang valid untuk dipindahkan, metode ini akan mengembalikan sepasang sel yang bisa dipertukarkan.

9. Metode `handle_click`

Metode ini digunakan untuk menangani klik pengguna dan memindahkan potongan puzzle sesuai dengan gerakan yang valid.

- **Proses:**

- Metode ini memanggil `_is_move_valid` untuk memeriksa apakah gerakan yang diminta valid.
- Jika valid, potongan pada dua sel (sel yang diklik dan sel kosong) akan dipertukarkan.

10. Metode `draw`

Metode ini digunakan untuk menggambar grid dan potongan-potongan puzzle di layar.

- **Proses:**

- Iterasi melalui setiap sel dalam grid dan memanggil metode draw dari setiap objek Cell untuk menggambar sel beserta potongan yang menempati sel tersebut (jika ada).

Secara keseluruhan, kelas Frame berfungsi untuk mengelola seluruh permainan puzzle, mulai dari membuat grid dan potongan-potongan, mengacak puzzle, memvalidasi gerakan, hingga menggambar dan menangani interaksi pengguna.

GAME.PY

```

1 import pygame
2
3 pygame.font.init()
4
5 class Game:
6     def __init__(self):
7         self.font = pygame.font.SysFont("Courier New", 35)
8         self.background_color = (255, 174, 66)
9         self.message_color = (17, 53, 165)
10
11     def arrow_key_clicked(self, click):
12         try:
13             if click.key == pygame.K_LEFT or click.key == pygame.K_RIGHT or click.key == pygame.K_UP or click.key == pygame.K_DOWN:
14                 return(True)
15         except:
16             return(False)
17
18     def is_game_over(self, frame):
19         for row in frame.grid:
20             for cell in row:
21                 piece_id = cell.occupying_piece.p_id
22                 if cell.c_id == piece_id:
23                     is_arranged = True
24                 else:
25                     is_arranged = False
26                 break
27         return is_arranged
28
29     def message(self, screen):
30         screen.fill(self.background_color, (5, 460, 440, 35))
31         instructions = self.font.render('You Win!!', True, self.message_color)
32         screen.blit(instructions, (125, 460))

```

1. Import dan Inisialisasi Pygame

pygame adalah pustaka Python yang digunakan untuk membuat game dan aplikasi multimedia. Fungsi `pygame.font.init()` digunakan untuk menginisialisasi modul font, yang memungkinkan untuk menampilkan teks dalam permainan.

2. Kelas Game

Game adalah sebuah kelas yang akan digunakan untuk mengelola logika permainan.

Dalam metode `__init__`:

- `self.font = pygame.font.SysFont("Courier New", 35)` membuat objek font dengan jenis huruf Courier New dan ukuran font 35.

- `self.background_color = (255, 174, 66)` adalah warna latar belakang (background color) dalam format RGB.
- `self.message_color = (17, 53, 165)` adalah warna untuk teks yang akan ditampilkan pada layar (teks pesan kemenangan) dalam format RGB.

3. Metode `arrow_key_clicked`

Metode ini digunakan untuk memeriksa apakah sebuah tombol arah pada keyboard (kiri, kanan, atas, atau bawah) telah ditekan.

`click` adalah parameter yang diharapkan berisi objek event yang diterima oleh pygame.

`pygame.K_LEFT`, `pygame.K_RIGHT`, `pygame.K_UP`, dan `pygame.K_DOWN` adalah kode-kode tombol arah di pygame.

Jika tombol arah ditekan, fungsi ini mengembalikan `True`. Jika tidak, atau jika terjadi kesalahan (misalnya, objek `click` tidak sesuai dengan yang diharapkan), ia mengembalikan `False`.

4. Metode `is_game_over`

Metode ini bertugas untuk memeriksa apakah permainan telah selesai atau belum, tergantung pada kondisi di dalam objek `frame`.

`frame.grid` di sini diasumsikan sebagai grid atau papan permainan, yang terdiri dari beberapa `cell`.

Untuk setiap `cell` dalam grid, kode ini memeriksa apakah `piece_id` dari potongan (`piece`) yang ada di dalam sel cocok dengan `c_id` dari sel tersebut.

- `cell.occupying_piece.p_id` adalah ID potongan yang ada di dalam sel.
- `cell.c_id` adalah ID sel.

Jika ada potongan yang ID-nya tidak cocok dengan ID sel, maka permainan dianggap belum selesai (`is_arranged = False`).

Jika seluruh grid cocok, maka permainan dianggap selesai dan `is_arranged` akan tetap `True`.

5. Metode `message`

Fungsi ini bertugas menampilkan pesan "You Win!!" di layar permainan ketika pemain menang.

`screen.fill(self.background_color, (5, 460, 440, 35))` menggambar latar belakang dengan warna yang sudah ditentukan (`self.background_color`) pada area yang ditentukan oleh koordinat dan ukuran (5, 460, 440, 35).

`self.font.render('You Win!!', True, self.message_color)` membuat objek gambar (surface) dari teks "You Win!!" dengan warna yang sudah ditentukan (blue).

`screen.blit(instructions, (125, 460))` menempatkan teks yang sudah dirender ke layar pada posisi (125, 460).

MAIN.PY

```

1 import pygame
2
3 from frame import Frame
4 from game import Game
5
6 pygame.init()
7 pygame.font.init()
8
9 class Puzzle:
10     def __init__(self, screen):
11         self.screen = screen
12         self.running = True
13         self.FPS = pygame.time.Clock()
14         self.is_arranged = False
15         self.font = pygame.font.SysFont("Courier New", 33)
16         self.background_color = (255, 174, 66)
17         self.message_color = (17, 53, 165)
18
19     def _draw(self, frame):
20         frame.draw(self.screen)
21         pygame.display.update()
22
23     def _instruction(self):
24         instructions = self.font.render('Use Arrow Keys to Move', True, self.message_color)
25         screen.blit(instructions, (5, 460))
26
27     def main(self, frame_size):
28         self.screen.fill("white")
29         frame = Frame(frame_size)
30         game = Game()
31         self._instruction()
32         while self.running:
33
34             if game.is_game_over(frame):
35                 self.is_arranged = True
36                 game.message(self.screen)
37
38             for event in pygame.event.get():
39                 if event.type == pygame.QUIT:
40                     self.running = False
41
42                 if event.type == pygame.KEYDOWN:
43                     if not self.is_arranged:
44                         if game.arrow_key_clicked(event):
45                             frame.handle_click(event)
46
47             self._draw(frame)
48             self.FPS.tick(30)
49
50         pygame.quit()
51
52 if __name__ == "__main__":
53     window_size = (450, 500)
54     screen = pygame.display.set_mode(window_size)
55     pygame.display.set_caption("Slide Puzzle")
56
57     game = Puzzle(screen)
58     game.main(window_size[0])

```

1. Impor Pustaka dan Modul

pygame adalah pustaka untuk membuat game dan aplikasi multimedia.

from frame import Frame dan from game import Game mengimpor kelas Frame dan Game dari file lain. Ini menunjukkan bahwa kode ini bergantung pada kelas lain yang didefinisikan dalam file terpisah (frame.py dan game.py), yang mungkin mengelola logika game dan representasi grid.

pygame.init() digunakan untuk menginisialisasi semua modul pygame, seperti grafik, suara, dan input.

pygame.font.init() menginisialisasi modul font untuk memungkinkan penggunaan teks dalam game.

2. Kelas Puzzle

__init__ adalah konstruktor kelas Puzzle. Parameter screen adalah objek layar untuk menampilkan game.

- self.screen = screen menyimpan objek layar untuk digunakan di seluruh metode dalam kelas ini.
- self.running = True menandakan bahwa game masih berjalan. Variabel ini akan diubah menjadi False untuk menghentikan game.
- self.FPS = pygame.time.Clock() digunakan untuk mengontrol kecepatan frame (jumlah frame per detik) dalam game.
- self.is_arranged = False adalah status apakah puzzle sudah tersusun dengan benar atau belum.
- self.font = pygame.font.SysFont("Courier New", 33) mendefinisikan font yang digunakan untuk menampilkan teks (dengan ukuran 33).
- self.background_color = (255, 174, 66) adalah warna latar belakang layar (RGB).
- self.message_color = (17, 53, 165) adalah warna teks yang akan ditampilkan (RGB).

3. Metode _draw

Metode _draw bertugas untuk menggambar frame (grid puzzle) pada layar.

- frame.draw(self.screen) akan memanggil metode draw dari objek frame untuk menggambar grid ke layar.
- pygame.display.update() memperbarui tampilan layar setelah menggambar.

4. Metode _instruction

_instruction digunakan untuk menampilkan instruksi pada layar (dalam hal ini, cara bermain dengan menggunakan tombol arah).

- `self.font.render('Use Arrow Keys to Move', True, self.message_color)` merender teks instruksi menjadi objek gambar (surface).
- `screen.blit(instructions,(5,460))` menggambar teks instruksi pada posisi (5, 460) pada layar.

5. Metode main

main adalah metode utama yang menjalankan permainan.

- `self.screen.fill("white")` mengisi latar belakang layar dengan warna putih.
- `frame = Frame(frame_size)` membuat objek frame yang mungkin berisi grid puzzle (dari kelas Frame).
- `game = Game()` membuat objek game yang mengelola logika permainan (dari kelas Game).
- `self._instruction()` memanggil metode yang menampilkan instruksi cara bermain.
- `while self.running:` adalah loop utama permainan yang terus berjalan selama `self.running` bernilai True.

6. Loop Utama Permainan

Di dalam loop, kode ini memeriksa apakah permainan telah selesai (misalnya, puzzle sudah disusun dengan benar). Jika ya, maka `self.is_arranged` diubah menjadi True dan pesan kemenangan ditampilkan menggunakan `game.message(self.screen)`.

`for event in pygame.event.get():` menangani semua event yang terjadi selama permainan (seperti menekan tombol atau menutup jendela).

- `if event.type == pygame.QUIT:` jika pemain menutup jendela permainan, maka loop berhenti dengan mengubah `self.running` menjadi False.
- `if event.type == pygame.KEYDOWN:` jika ada tombol yang ditekan:
 - `if not self.is_arranged:` hanya menerima input jika puzzle belum tersusun dengan benar.
 - `if game.arrow_key_clicked(event):` memeriksa apakah tombol yang ditekan adalah tombol arah (panah).
 - `frame.handle_click(event)` menangani input pemain untuk memindahkan potongan puzzle sesuai dengan tombol arah yang ditekan.

`self._draw(frame)` menggambar frame (grid puzzle) pada layar.

`self.FPS.tick(30)` mengontrol kecepatan permainan dengan membatasi frame rate menjadi 30 frame per detik.

7. Menutup Pygame

Setelah loop selesai, `pygame.quit()` akan menutup pustaka pygame dan membersihkan semua sumber daya yang digunakan.

8. Bagian Eksekusi

`if __name__ == "__main__":` memastikan bahwa kode ini hanya dijalankan jika file ini dieksekusi langsung, bukan diimpor sebagai modul.

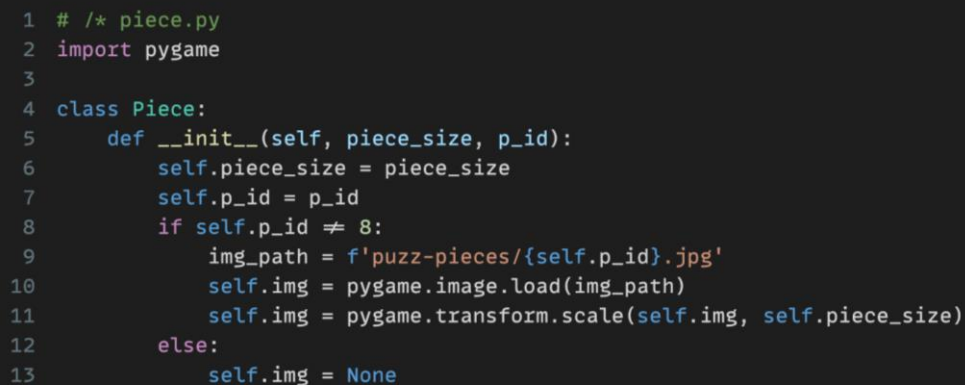
`window_size = (450, 500)` mendefinisikan ukuran jendela permainan.

`screen = pygame.display.set_mode(window_size)` membuat objek layar dengan ukuran yang ditentukan.

`pygame.display.set_caption("Slide Puzzle")` menetapkan judul jendela menjadi "Slide Puzzle".

`game = Puzzle(screen)` membuat objek Puzzle dan menjalankan permainan dengan `game.main(window_size[0])`.

PIECE.PY



```
1  # /* piece.py
2  import pygame
3
4  class Piece:
5      def __init__(self, piece_size, p_id):
6          self.piece_size = piece_size
7          self.p_id = p_id
8          if self.p_id != 8:
9              img_path = f'puzz-pieces/{self.p_id}.jpg'
10             self.img = pygame.image.load(img_path)
11             self.img = pygame.transform.scale(self.img, self.piece_size)
12         else:
13             self.img = None
```

- **Impor pygame**

Pustaka pygame digunakan untuk menangani elemen grafis dalam permainan, seperti memuat gambar dan merendernya.

- **Kelas Piece**

Piece adalah kelas yang merepresentasikan sebuah potongan puzzle.

piece_size adalah ukuran potongan gambar puzzle, biasanya berupa tuple (lebar, tinggi).

p_id adalah ID unik yang mengidentifikasi potongan puzzle tersebut. Misalnya, ID ini bisa digunakan untuk menentukan gambar yang sesuai untuk potongan puzzle tersebut.

- **Menangani Gambar untuk Potongan Puzzle**

if self.p_id != 8:: Potongan dengan p_id == 8 dianggap sebagai potongan kosong atau tidak terlihat (biasanya dalam slide puzzle, potongan kosong tidak memiliki gambar). Untuk potongan lainnya:

- `img_path = f'puzz-pieces/{self.p_id}.jpg'`: Membuat path file gambar untuk potongan puzzle berdasarkan ID-nya. Misalnya, jika `p_id == 1`, path gambar akan menjadi `puzz-pieces/1.jpg`.
- `self.img = pygame.image.load(img_path)`: Memuat gambar dari file dengan menggunakan `pygame.image.load()`.
- `self.img = pygame.transform.scale(self.img, self.piece_size)`: Mengubah ukuran gambar agar sesuai dengan ukuran potongan puzzle yang telah ditentukan (`piece_size`).

else::

- Jika `p_id == 8` (potongan kosong), maka `self.img = None` menetapkan gambar potongan tersebut menjadi None, karena tidak ada gambar yang ditampilkan untuk potongan kosong.

