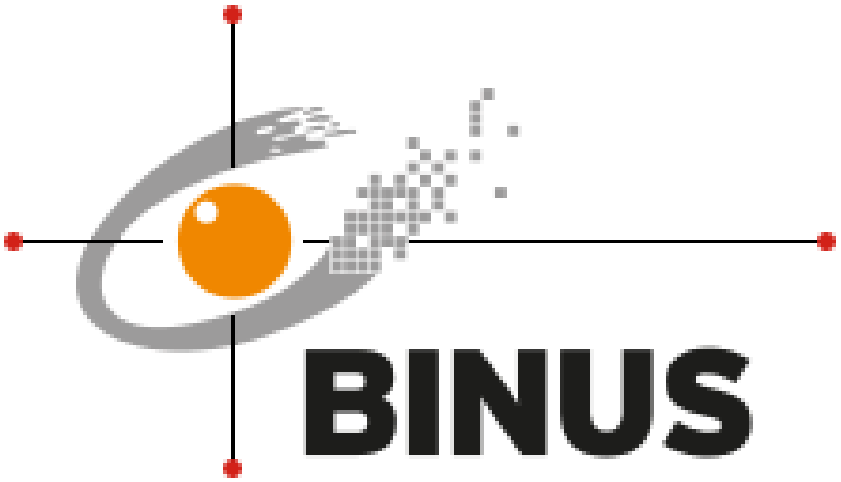
Algorithm and Programming Final Project: Sliding Puzzle Game



BINUS
UNIVERSITY
INTERNATIONAL

Lecturer:
Jude Joseph Lamug Martinez

Arranged by:
Edelyne Keisha - L1BC

Student ID:
2602169850

Character Building: Pancasila Education
Computer Science Faculty
Binus International University 2022/2026

Introduction

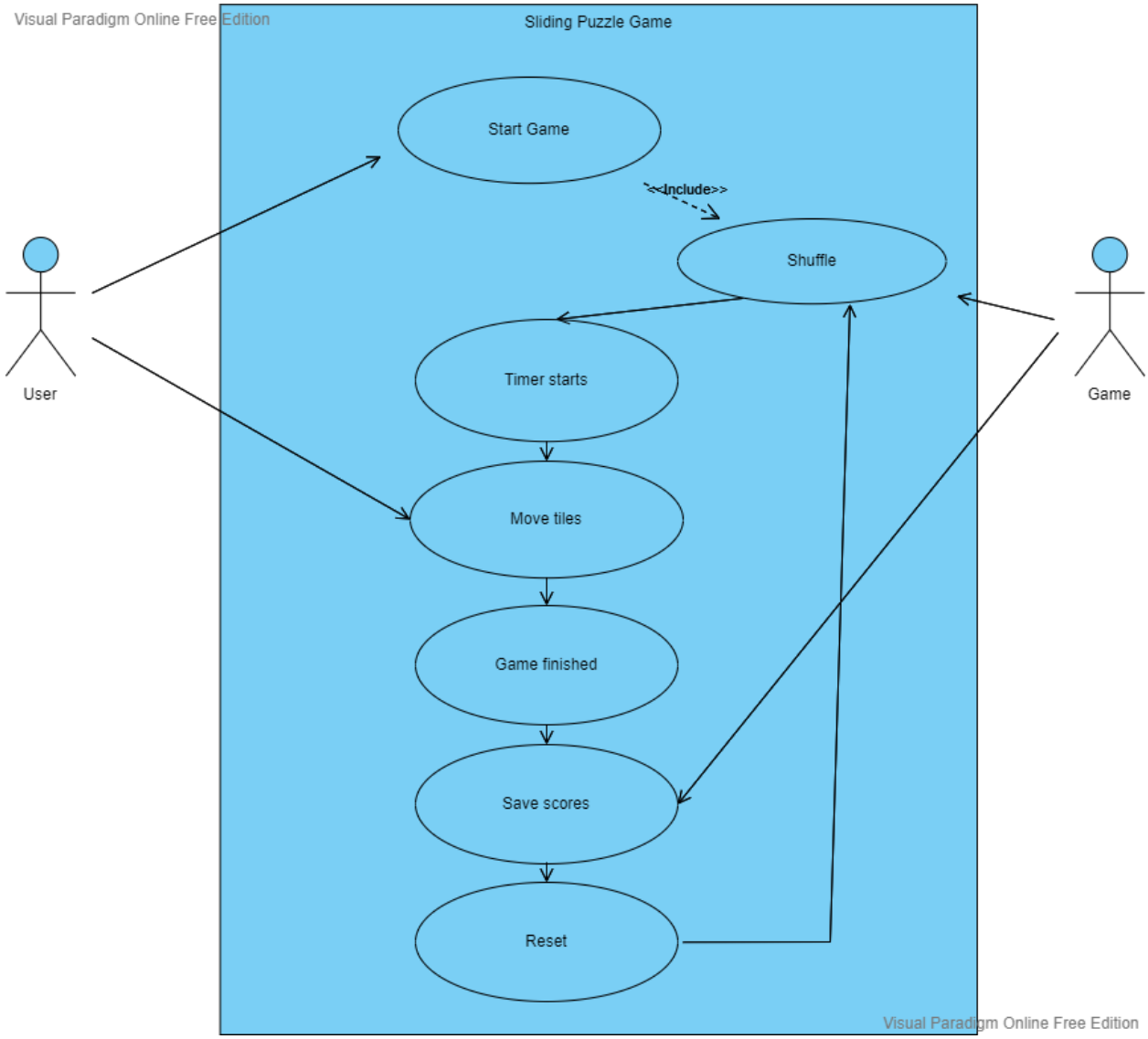
When it was announced that Algorithm and Programming will have a final project. I immediately went around looking for ideas where I can implement all the things that I've learned in this course. I had many ideas in the beginning, such as making a management system to even making a full fledged video game. There were many reasons why I never stayed with those ideas, one being that they were either too easy for a final project or too difficult for me at the current moment, but the biggest enemy is time constraints. Because of it, I eventually had to tone down my ideas into something a bit simpler. Which is what this game turned out to be.

Function of the Program

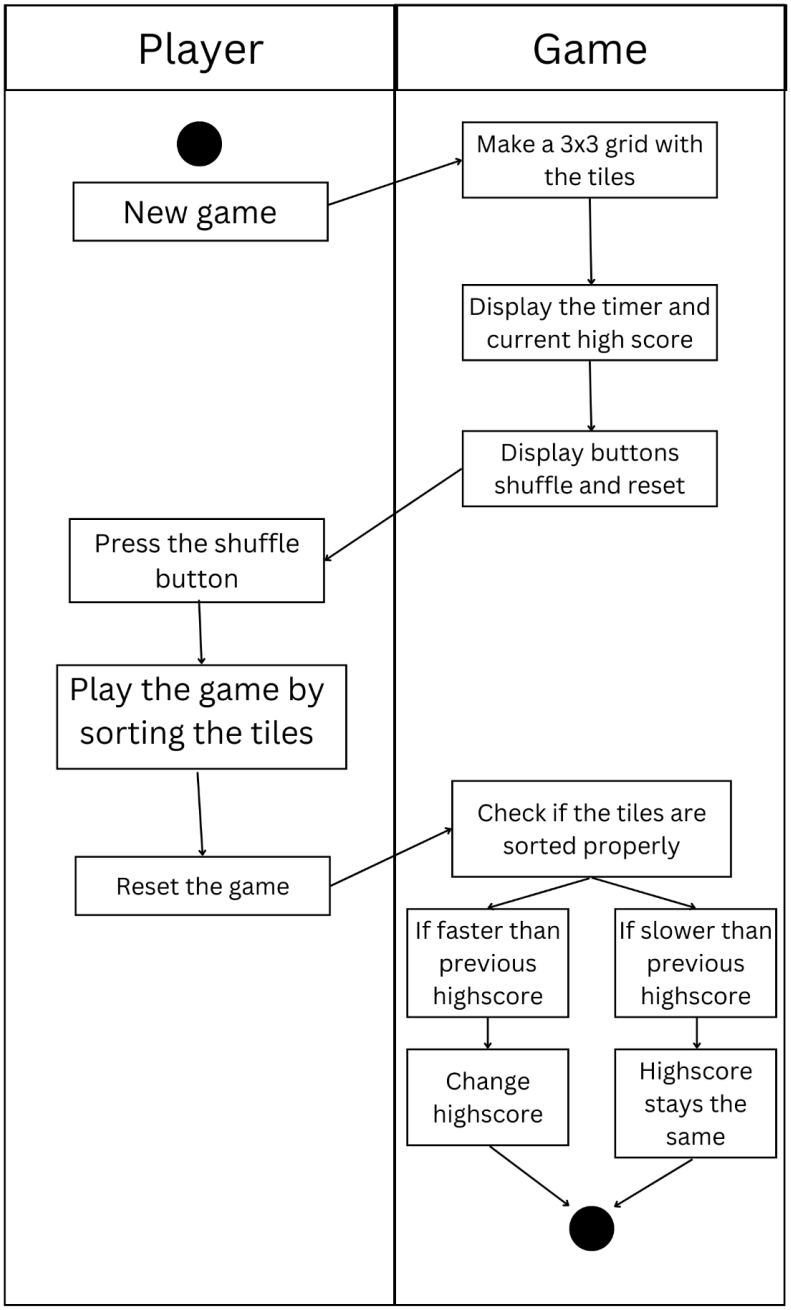
The function of my program is to play a sliding puzzle game, a simple, yet quite challenging way to pass time.

For those who don't know what it is, a sliding puzzle game is a type of puzzle where the player is presented with a grid of tiles with the size of 3x3, each with a number. The goal is to move the tiles around until the numbers are sorted in the correct order. The player can only move one tile at a time, and only if the space next to it is empty. The game is won only when all the numbers are sorted, and in my program, whoever wins is whoever obtains the highest score, as in, the player who solved the puzzle in the least amount of time.

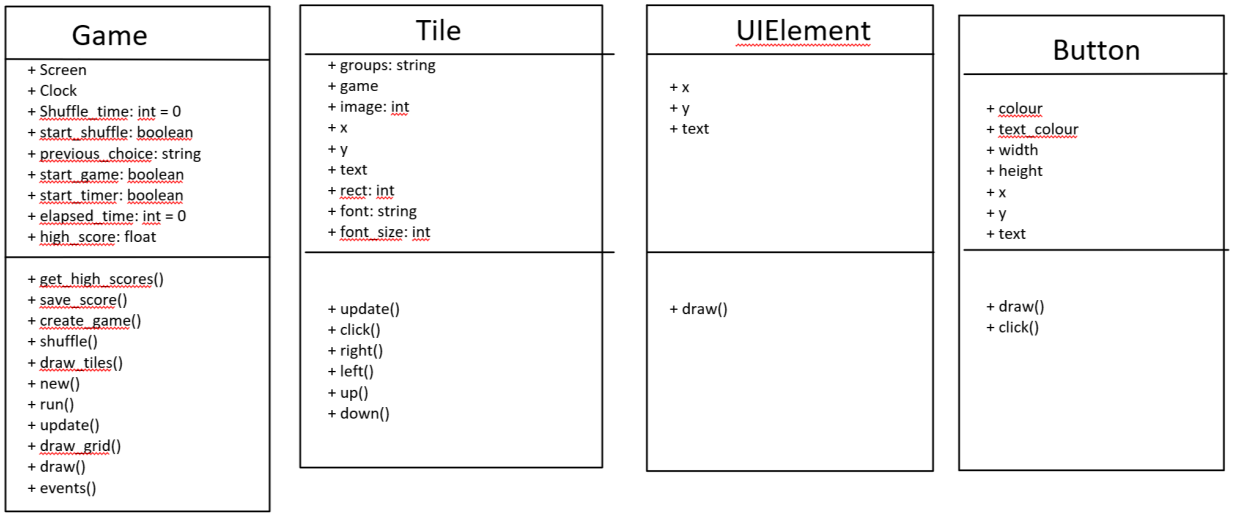
Use Case Diagram



Activity Diagram



Class Diagram



Modules

```
import pygame

import random

import time
```

Pygame: Used to make games in Python. This is what's able to make all the objects in the game, from the background color, to the tiles itself.

Random: Used to randomize things. This is used for the shuffling mechanic.

Time: Used for time. This is used for the timer in the game.

Essential Algorithms

Class Game:

- __init__(self):

  Set attributes screen to variables WIDTH and HEIGHT

  Set attributes self.clock to set the clock timer

  Set attributes self.shuffle_time to 0 as the shuffle hasn't begun yet

  Set attributes self.start_shuffle as boolean False because the shuffle button hasn't been clicked

  Set attributes self.previous_choice as an empty string

  Set attributes self.start_game as boolean False

  Set attributes self.start_timer as boolean False

  Set attributes self.elapsed_time as 0

  Set attributes self.high_score as a float

- get_high_scores(self):

  Reading the txt file to get the highscore and returning it

- save_score(self):

  Write in the txt file a new highscore and save it. High Score also written in 3 decimal places.

- create_game(self):

  Using for loops to make the row and column of the grid based on the GAME_SIZE, so there will be 3 rows and 3 columns.

  Last number is set to 0, that will be the empty tile.

- shuffle(self):

  Checking over all the possible ways the tiles can be shuffled. To make it more random, For example, the tile that had just been moved up cannot be moved back down, it only has options to move left, right, or up, this is done so that the shuffle won't have any chances of a tile being moved up and down repeatedly, making the shuffling more randomized

  Choose a tile, if it chooses to go right, swap the tile with the empty tile, or vice versa

- draw_tiles(self):

  Function to draw the tiles itself

  It uses a for loop that takes row and col as the indexes of the current element, and x and tile as the value of the current element

  If value not 0, it will be a tile with numbers from 1-8

  If value is 0, it will be the empty tile in the grid

- new(self):

  Function for every new game

Set attributes self.tiles_grid and self.tiles_grid_completed to self.create_game()

Set attributes self.elapsed_time to 0

Set attributes self.start_timer to False

Set attributes self.start_game to False

Set attributes self.buttons_list as an empty list for the buttons

Set attributes self.draw_tiles() to draw the tiles

- run(self)

  Function to run the game. When game is running, start clock, the events() function, the update() function, and the draw() function

- update(self)

  Function to update everything in the game, such as checking if the completed grid is the same as the first grid to end the game, comparing the previous high score to the current score, checking how much time passed from current time, modifying the shuffle function, and finally updating the entire game.

- draw_grid(self):

  Using for loops to make the grid based on the screen where the grid is drawn, line color for grid, starting point of line for current row, end point of line for current row. Pygame.draw.line is used by pygame so that it can draw the line based on those coordinate

- draw(self):

  Draw all the objects on the screen such as the background color, the sprites, the display for the timer and the highscore text

  Update the entire screen with pygame.display.flip()

- events(self):

  Function for the inputs, such as if the X button is clicked, or if a tile is clicked

  If statements to check if it's possible to swap the tiles

  Clicking shuffle will reset the timer

  Clicking reset will make a new game

Class Tile:

- __init__(self, game, x, y, text):

  Set attributes to make the tiles with a dimension of 128 x 128

  Any tiles that aren't empty will be drawn with a LIGHTBROWN color, has a text in the center will the numbers on it from 1-8

  Empty tiles will just be given background color

- update(self):

  Give coordinates on the grid but in pixels

- click(self, mouse_x, mouse_y):

  Check if the mouse is positioned inside the tile

- right(self)

  Check the right side of the tile

- left(self)

  Check the left side of the tile

- up(self)

  Check the above side of the tile

- down(self)

  Check the lower side of the tile

Class UIElement:

- __init__(self, x, y, text)

  Takes the x coordinate, y coordinate of the text

- draw(self, screen)

  Blits the text on the screen, such as the highscore text

Class Button:

- __init__(self, x, y, width, height, text, colour, text_colour):

  Set attributes self.colour, self.text_colour to variable colour, text_colour

  Set attributes self.width, self.height to variable width, height

  Set attributes self.x, self.y to variable x, y

  Set attributes self.text to variable text

- draw(self, screen):

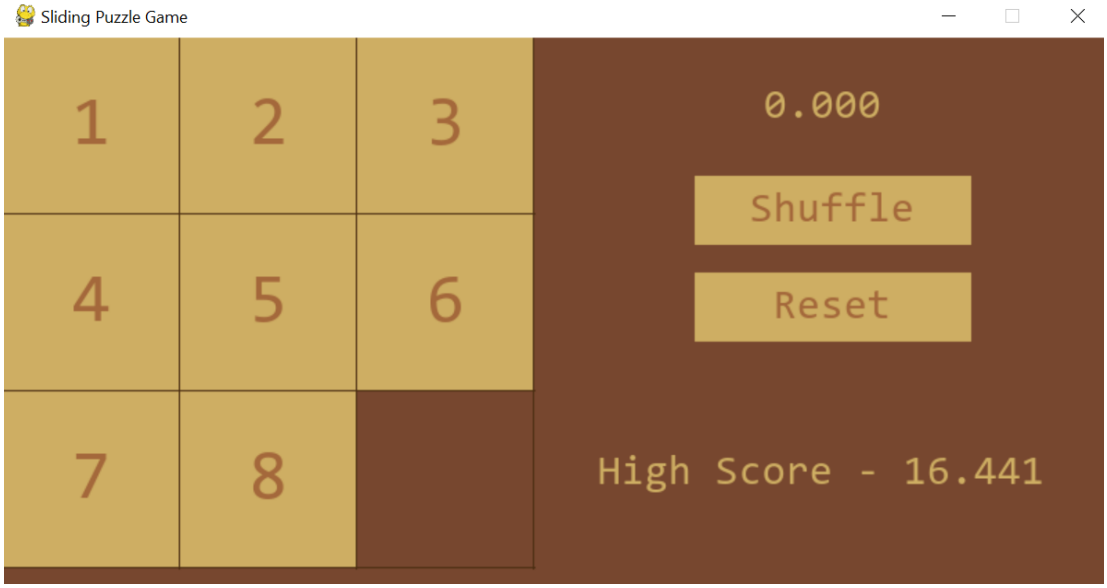
  Drawing the button itself using pygame.draw.rect, taking in the surface, color, and size of the button

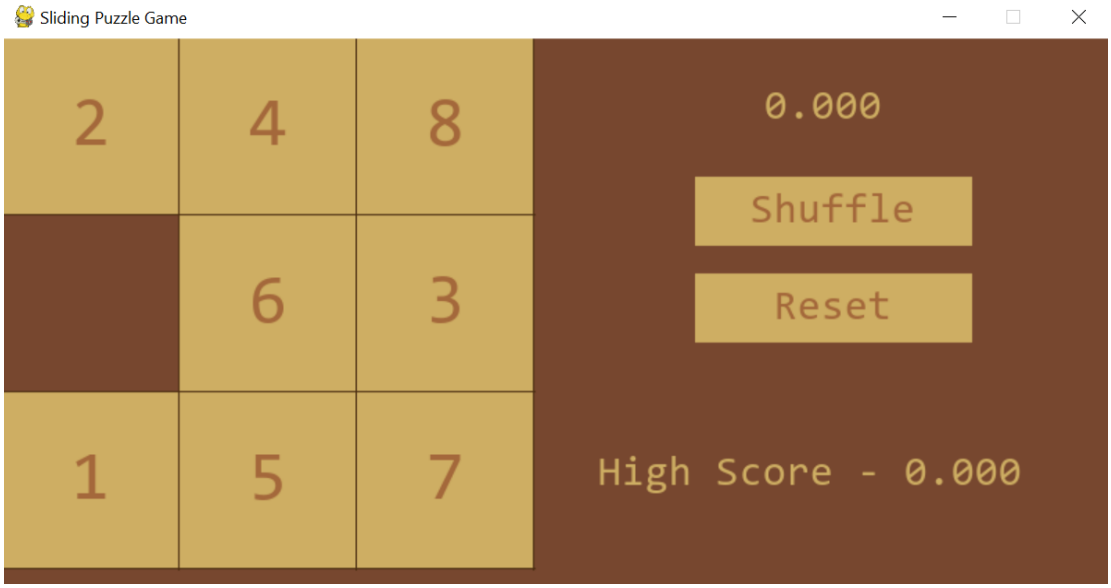
  Blit the text in the center of the button

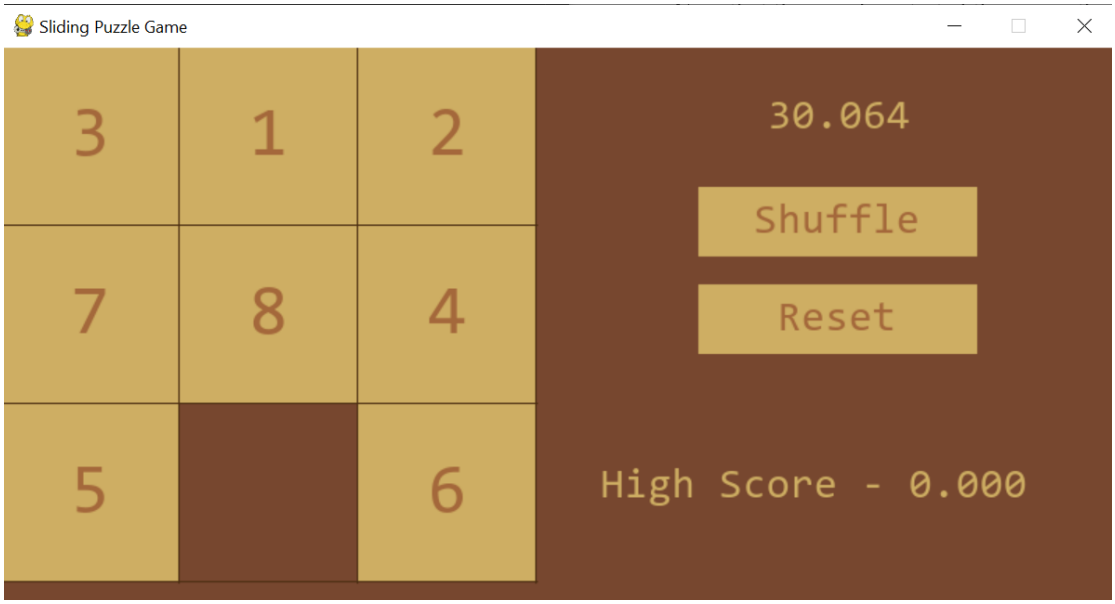
- click(self, mouse_x, mouse_y)

  Finding out if the mouse is positioned in the button

Screenshots







Lesson learned / Reflection

Making this game taught me a lot of things. It gave me experience in doing long term coding. It gave me insight on what it's going to be like as a programmer. In the beginning, it was frustrating, as I stumbled upon quite a few bugs that I originally didn't know how to fix, and had to debug myself. The process of debugging my code took a while, as it meant I had to familiarize myself with syntaxes and functions that I thought I had understood already, and learn many more new syntaxes such as the ones from pygame or from random. However, debugging my code, and eventually finishing the game, gave me a sense of accomplishment, as after all the struggle I finally managed to make a full working game, albeit a simple one for now.

Making this game also taught me more things about Python compared to what I thought I already knew. For example, it taught me about how to actually use Pygame. I learned how to actually draw things on the Window surface, making them move, place them in a specific position, and giving them actions based on the inputs from keyboard keys or a mouse.

All in all, although there were a lot more improvements to this game that I would like to add, I am for now satisfied with the results. As this is one of my first big projects, I'm glad this project can teach me many things that one can do in Python.