บทที่ 3

การออกแบบและพัฒนา

3.1 แนวคิดและกกฎพื้นฐาน

เกมนี้สร้างเพื่อให้ผู้เล่นได้ลองผิดลองถูก โดยการให้ผู้เล่นเติมตัวเลขลงช่องว่างในตารางขนาด 9x9 ตายกฎของ sudoku โดยกฎมีดังนี้

1.เติมเลข 1 ถึง 9 ให้ครบทุกตางราง

2.แต่ละแถว จะต้องไม่มีเลขซ้ำกัน

3. แต่ละคอลัมน์ จะต้องไม่มีตัวเลขซ้ำกัน

4.แต่ละบล็อกย่อยขนาด 3x3 จะต้องไม่มีตัวเลขซ้ำกัน

3.2 หลักการทำงานของโปรแกรม

1.หาช่องว่างแรกที่ยังไม่มีตัวเลข

2.ลองใส่ตัวเลขตั้งแต่ 1 ถึง 9 ในช่องนั้นแล้วเช็คว่าการใส่ตัวเลขนี้ยังเป็นไปตามกฎ Sudoku หรือไม่

3.ถ้าตัวเลขที่ใส่ทำให้ถูกต้อง (ตามกฎทุกข้อ) ให้เดินหน้าไปยังช่องถัดไป

4.ถ้าไม่มีตัวเลขไหนที่สามารถใส่ได้ในช่องนั้น ให้กลับมาที่ช่องก่อนหน้าและเปลี่ยนตัวเลขที่ใส่ไว้

5.ทำซ้ำไปเรื่อย ๆ จนกระทั่งทุกช่องถูกเติมด้วยตัวเลขที่ถูกต้อง

3.3 ขั้นตอนการทำงานของโค้ดโปรแกรม

3.3.1 Import binary

รูปที่ 1 Import binary

โดย Binary ที่ Import มา มีทั้งหมด 2 Binary ได้แก่ opencv กับ numpy

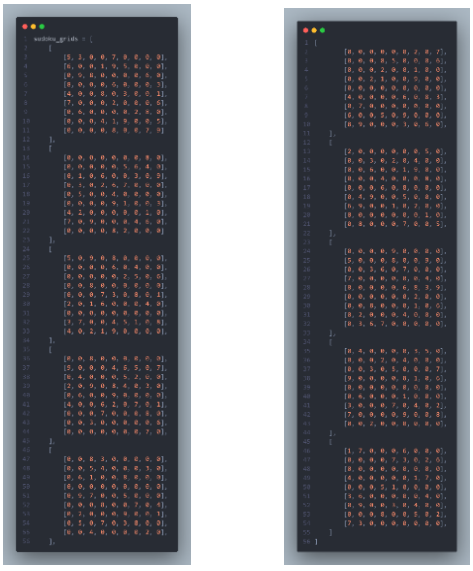
3.3.2 กำหนดตัวแปรของตารางและสี

```
grid_size = 9
cell_size = 50
board_size = grid_size * cell_size
line_color = (0, 0, 0)
text_color = (0, 0, 0)
input_color = (255, 0, 0)
invalid_color = (0, 0, 255)
FONT = cv2.FONT_HERSHEY_SIMPLEX
```

รูปที่ 2 กำหนดตัวแปรของตารางและสี

โดยกำหนดตัวแปร 1.grid_size ทนจำนวนช่องในแต่ละแถวหรือคอลัมน์ของตารางซูโดกุ ซึ่งโดยปกติแล้วตารางซูโดกุจะมีขนาด 9x9 2.cell_size แทนขนาดของแต่ละช่องในตาราง โดยวัดเป็นpixels แต่ละช่องจะมีขนาด 50x50 pixels 3.board_size คำนวณขนาดของตารางทั้งหมด เนื่องจากมี 9 ช่องในแต่ละแถว และแต่ละช่องมีขนาดกว้าง 50 pixels ส่วนตัวแปร line_color,text_color,input_color,invalid_color เป็นการกำหนดค่าสีเพื่อให้มองส่วนต่างๆได้ง่าย โดย line_color และ text_color เป็นสีดำให้เป็นค่าคงที่ input_color เป็นสีน้ำเงินเพื่อให้รู้ว่าเป็นค่าที่ใส่เข้าไป invalid_color เป็นสีแดงเพื่อให้รู้ว่าเป็นค่าที่ผิดเพื่อให้ง่ายต่อการเช็คว่าใส่ผิดไปกี่รอบ ส่วน FONT เป็นการกำหนดใช้ฟอนต์อะไรโดยใช้ Binary opencv มากำหนดตัวฟอนต์

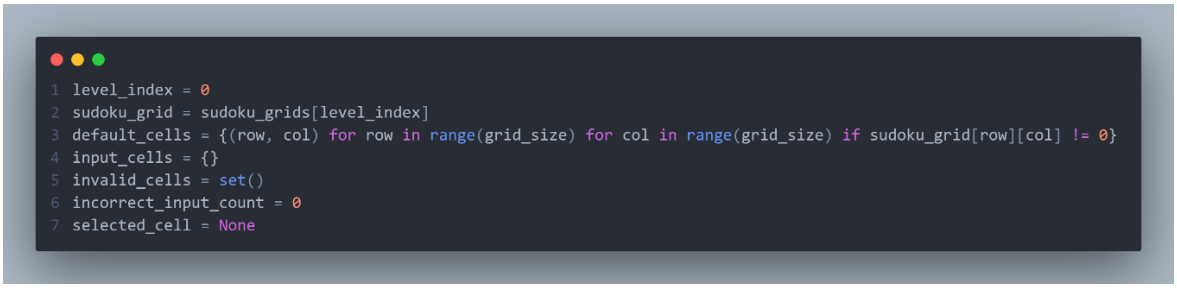
3.3.3 ด่านของเกม



รูปที่ 3 ด่านของเกม

กำหนดด่านของเกม sudoku โดยมีทั้งหมด 10 ด่าน จะเพิ่มระดับความยากขึ้นไปเรื่อยๆ ตามลำดับ

3.3.4 ตัวแปรข้อมูลของเกม

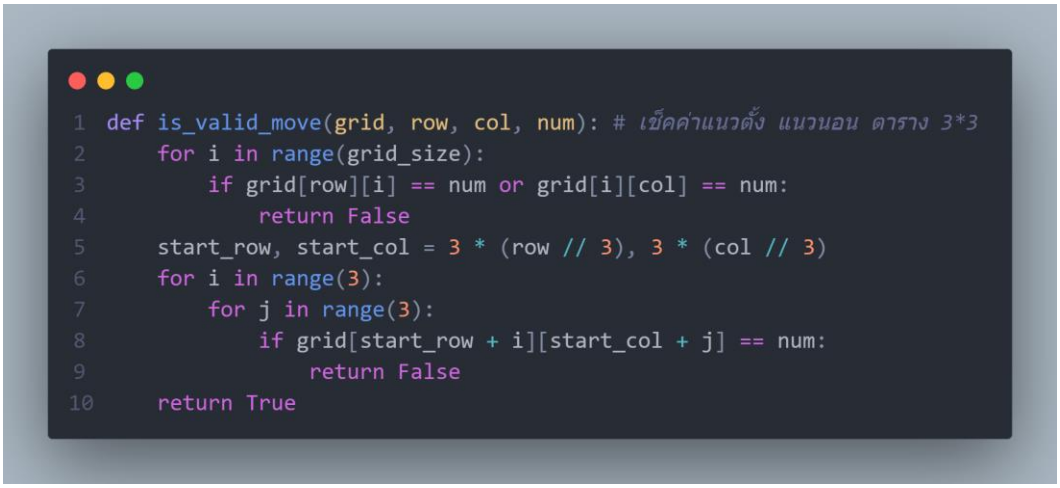


```
level_index = 0
sudoku_grid = sudoku_grids[level_index]
default_cells = {(row, col) for row in range(grid_size) for col in range(grid_size) if sudoku_grid[row][col] != 0}
input_cells = {}
invalid_cells = set()
incorrect_input_count = 0
selected_cell = None
```

รูปที่ 4 ตัวแปรข้อมูลของเกม

ตัวแปรที่กำหนดไว้จะเป็นตัวแปรที่ใช้เก็บข้อมูลของเกม โดยจะมี level_index และ sudoku_grid เป็นตัวแปรในการวนลูปด่าน default_cells สำหรับตัวเลขในตารางที่กำหนดไว้แล้ว input_cells สำหรับเก็บค่าที่ผู้เล่นป้อนเข้าไปในตาราง invalid_cells สำหรับเก็บตำแหน่งที่ผู้เล่นกรอกตัวเลขผิด incorrect_in สำหรับนับจำนวนครั้งที่ผู้เล่นกรอกตัวเลขผิด

3.3.5 ฟังก์ชันตรวจสอบการใส่ตัวเลข



```
def is_valid_move(grid, row, col, num): # เช็คค่าแนวตั้ง แนวนอน ตาราง 3*3
    for i in range(grid_size):
        if grid[row][i] == num or grid[i][col] == num:
            return False
    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for i in range(3):
        for j in range(3):
            if grid[start_row + i][start_col + j] == num:
                return False
    return True
```

รูปที่ 5 ฟังก์ชันตรวจสอบการใส่ตัวเลข

ฟังก์ชันนี้ตรวจสอบว่าการใส่ตัวเลขลงในช่องตารางนั้นถูกต้องตามกฎของเกมซูโดกุหรือไม่ โดยตัวเลขทุกตัวต้องไม่ซ้ำกันกับตัวเลขที่อยู่ในแถวเดียวกัน คอลัมน์เดียวกัน และตารางย่อย 3x3

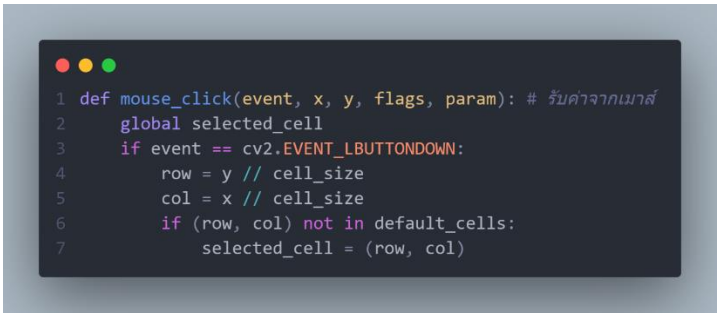
3.3.6 ฟังก์ชันวาดตารางซูโดกุ

```python
def draw_grid(img, grid, input_cells, invalid_cells, selected_cell): # สร้างตาราง
    for i in range(grid_size + 1):
        thickness = 2 if i % 3 == 0 else 1
        cv2.line(img, (0, i * cell_size), (board_size, i * cell_size), line_color, thickness)
        cv2.line(img, (i * cell_size, 0), (i * cell_size, board_size), line_color, thickness)

    if selected_cell is not None:
        sel_row, sel_col = selected_cell
        cv2.rectangle(img, (sel_col * cell_size, sel_row * cell_size),
                      ((sel_col + 1) * cell_size, (sel_row + 1) * cell_size),
                      (255, 26, 0), 2)

    for row in range(grid_size):
        for col in range(grid_size):
            if grid[row][col] != 0:
                text = str(grid[row][col])
                x = col * cell_size + cell_size // 4
                y = row * cell_size + int(cell_size * 0.75)
                color = text_color if (row, col) in default_cells else input_color
                cv2.putText(img, text, (x, y), FONT, 1, color, 2)

    for (row, col), value in input_cells.items():
        x = col * cell_size + cell_size // 4
        y = row * cell_size + int(cell_size * 0.75)
        color = invalid_color if (row, col) in invalid_cells else input_color
        cv2.putText(img, str(value), (x, y), FONT, 1, color, 2)

```

รูปที่ 6 ฟังก์ชันวาดตารางซูโดกุ

ฟังก์ชันนี้ใช้ในการวาดตารางซูโดกุและแสดงข้อมูลที่เกี่ยวข้อง รวมถึงการแสดงเซลล์ที่ถูกเลือก ตัวเลขที่ตั้งค่าเริ่มต้นเป็นสีดำ ตัวเลขที่ผู้เล่นป้อนเข้าไปเป็นสีน้ำเงิน และแสดงผลลัพธ์หากตัวเลขที่ป้อนผิดพลาดโดยแสดงเป็นสีแดง

3.3.7 ฟังก์ชันรับค่าจากเมาส์

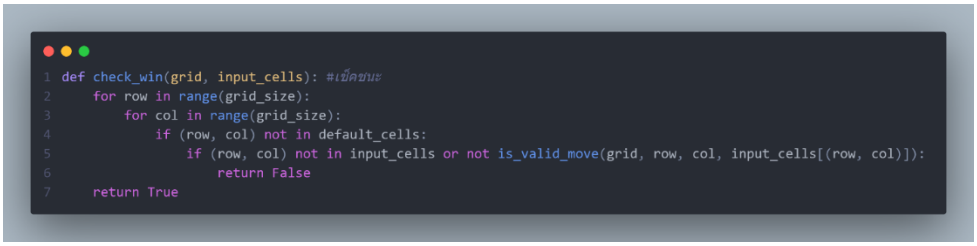


```python
def mouse_click(event, x, y, flags, param): # รับค่าจากเมาส์
    global selected_cell
    if event == cv2.EVENT_LBUTTONDOWN:
        row = y // cell_size
        col = x // cell_size
        if (row, col) not in default_cells:
            selected_cell = (row, col)
```

รูปที่ 7 ฟังก์ชันรับค่าจากเมาส์

ฟังก์ชันรับค่าเมาส์ ใช้เพื่อจัดการกับการคลิกเมาส์ซ้ายของผู้ใช้บนตาราง โดยผู้เล่นคลิกบนเซลล์ที่ว่างอยู่ โดยไม่ใช่เซลล์ที่มีตัวเลขเริ่มต้น ฟังก์ชันจะบันทึกตำแหน่งของเซลล์นั้นในตัวแปร selected_cell ซึ่งทำให้สามารถใช้ตำแหน่งนี้ในการ update ค่าที่ผู้เล่นใส่ไปในตารางซูโดกุ

3.3.8 ฟังก์ชันตรวจสอบว่าผู้เล่นชนะเกม

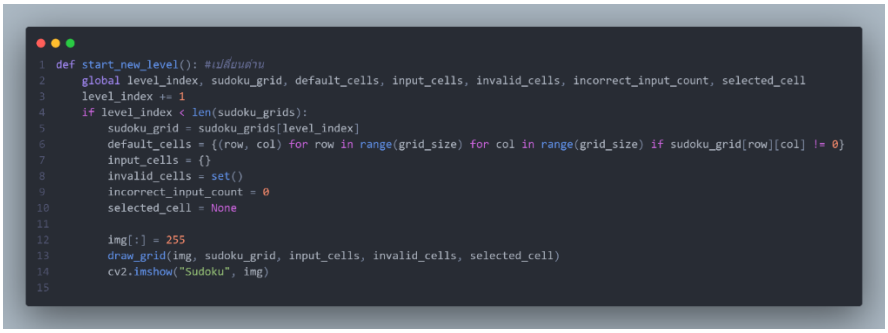


```
def check_win(grid, input_cells): #เช็คชนะ
    for row in range(grid_size):
        for col in range(grid_size):
            if (row, col) not in default_cells:
                if (row, col) not in input_cells or not is_valid_move(grid, row, col, input_cells[(row, col)]):
                    return False
    return True
```

รูปที่ 8 ฟังก์ชันตรวจสอบว่าผู้เล่นชนะเกม

ฟังก์ชันนี้ตรวจสอบว่าผู้เล่นแก้ตารางซูโดกุถูกต้องทั้งหมดหรือไม่ โดยเช็คว่าทุกเซลล์ที่ผู้เล่นป้อนค่าตัวเลขถูกต้องตามกฎของซูโดกุและไม่มีเซลล์ว่าง หากทุกอย่างถูกต้อง จะแสดงว่าผู้เล่นชนะเกมแล้ว

3.3.9 ฟังก์ชันสำหรับเปลี่ยนด่านของเกม

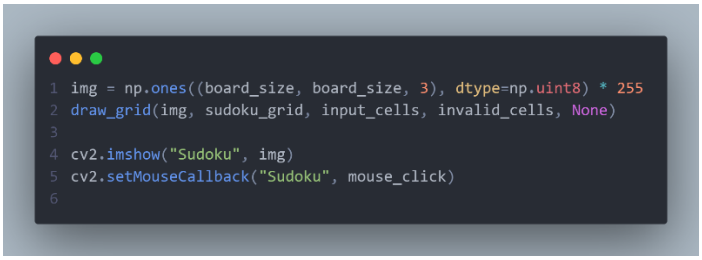


```
def start_new_level(): #เปลี่ยนด่าน
    global level_index, sudoku_grid, default_cells, input_cells, invalid_cells, incorrect_input_count, selected_cell
    level_index += 1
    if level_index < len(sudoku_grids):
        sudoku_grid = sudoku_grids[level_index]
        default_cells = {(row, col) for row in range(grid_size) for col in range(grid_size) if sudoku_grid[row][col] != 0}
        input_cells = {}
        invalid_cells = set()
        incorrect_input_count = 0
        selected_cell = None

        img[:] = 255
        draw_grid(img, sudoku_grid, input_cells, invalid_cells, selected_cell)
        cv2.imshow("Sudoku", img)

```

รูปที่ 9 ฟังก์ชันสำหรับเปลี่ยนด่านของเกม

ฟังก์ชันนี้จะเพิ่มระดับด่านของเกมซูโดกุไปยังด่านถัดไป ตั้งค่าตัวแปรใหม่สำหรับด่านนั้น รีเซ็ตค่าต่างๆ ที่เกี่ยวข้องกับการเล่น และแสดงตารางซูโดกุของด่านใหม่บนหน้าจอ เมื่อผู้เล่นผ่านด่านหนึ่งแล้ว ฟังก์ชันนี้จะเตรียมความพร้อมสำหรับการเริ่มเล่นในด่านต่อไปโดยอัตโนมัติ

3.3.10 การแสดงหน้าต่างของเกม



```
img = np.ones((board_size, board_size, 3), dtype=np.uint8) * 255
draw_grid(img, sudoku_grid, input_cells, invalid_cells, None)

cv2.imshow("Sudoku", img)
cv2.setMouseCallback("Sudoku", mouse_click)

```

รูปที่ 10 การแสดงหน้าต่างของเกม

โค้ดนี้จะสร้างภาพตารางsudoku แสดงภาพบนหน้าจอ และตั้งค่าให้ฟังก์ชัน mouse_click จัดการกับการคลิกเมาส์ที่เกิดขึ้นในหน้าต่างเกม เมื่อผู้เล่นคลิกเมาส์ในตาราง ฟังก์ชัน mouse_click จะช่วยให้เราทราบว่าผู้เล่นเลือกเซลล์ใดในตาราง

3.3.11 ฟังก์ชันการทำงานของเกม sudoku

```
while True: # เริ่มเกม
    cv2.imshow("Sudoku", img)
    key = cv2.waitKey(1)

    if key in [ord(str(i)) for i in range(1, 10)] and selected_cell:
        row, col = selected_cell
        num = int(chr(key))

        if is_valid_move(sudoku_grid, row, col, num):
            input_cells[(row, col)] = num
            invalid_cells.discard((row, col))
        else:
            if (row, col) not in invalid_cells:
                incorrect_input_count += 1
            invalid_cells.add((row, col))
            input_cells[(row, col)] = num

        img[:] = 255
        draw_grid(img, sudoku_grid, input_cells, invalid_cells, selected_cell)
        selected_cell = None

        if check_win(sudoku_grid, input_cells):
            cv2.rectangle(img, (0, board_size // 2 - 50),
                          (board_size, board_size // 2 + 100), (200, 200, 200), -1)

            win_text = "You win!!"
            incorrect_text = f"Incorrect inputs: {incorrect_input_count}"

            win_text_size = cv2.getTextSize(win_text, FONT, 2, 3)[0]
            incorrect_text_size = cv2.getTextSize(incorrect_text, FONT, 1, 2)[0]

            win_text_x = (board_size - win_text_size[0]) // 2
            win_text_y = board_size // 2

            incorrect_text_x = (board_size - incorrect_text_size[0]) // 2
            incorrect_text_y = win_text_y + 50

            cv2.putText(img, win_text, (win_text_x, win_text_y), FONT, 2, (0, 255, 0), 3)
            cv2.putText(img, incorrect_text, (incorrect_text_x, incorrect_text_y), FONT, 1, (0, 0, 255), 2)
            cv2.imshow("Sudoku", img)
            cv2.waitKey(10000)
            start_new_level()

    if key == 8 and selected_cell:
        row, col = selected_cell
        if (row, col) in input_cells:
            del input_cells[(row, col)]
        invalid_cells.discard((row, col))

        img[:] = 255
        draw_grid(img, sudoku_grid, input_cells, invalid_cells, selected_cell)
        selected_cell = None

    if key == 27:  # ESC key to exit
        break

    if cv2.getWindowProperty("Sudoku",cv2.WND_PROP_VISIBLE) < 1:
        break

cv2.destroyAllWindows()
```

รูปที่ 11 ฟังก์ชันการทำงานของเกม sudoku

ในฟังก์ชันส่วนนี้จะเป็นฟังก์ชันการทำงาน โดยมีขั้นตอนการทำงานดังนี้ 1.แสดงภาพตารางซูโดกุบนหน้าจอ 2.อ่านค่าคีย์ที่กด 3.การใส่ตัวเลขลงในตาราง 4.ตรวจสอบความถูกต้องของตัวเลขที่ใส่ 5.รีเฟรชหน้าจอและแสดงตารางใหม่ 6.ตรวจสอบว่าผู้เล่นชนะหรือไม่ 7.การลบตัวเลขที่ใส่ผิด 8.การออกจากเกม 9.ปิดหน้าต่างทั้งหมดเมื่อจบเกม โดยฟังก์ชันนี้คือการจัดการเกม sudoku แบบโต้ตอบ ผู้เล่นสามารถเลือกเซลล์ด้วยเมาส์ ใส่ตัวเลขผ่านคีย์บอร์ด ตรวจสอบการชนะ และเลื่อนไปยังด่านใหม่เมื่อชนะ หากผู้เล่นกด ESC หรือปิดหน้าต่าง เกมจะหยุดการทำงานและหน้าต่างจะถูกปิด