

รายงาน

โปรเจกต์การพัฒนาสร้างเกม Tetris ด้วย Pygame

โดย

นางสาวปาณิสดา กาญจนเพิ่มพูน รหัสนักศึกษา 630910176

นายโชคทวี ศรีศิลป์ รหัสนักศึกษา 630910315

นายโชคทวี ฟ้าคนอง รหัสนักศึกษา 630910316

นายตะวัน ไชยมาตร รหัสนักศึกษา 630910323

เสนอ

อาจารย์ศักดิ์ระพี ไพศาลนันท์

สาขาวิศวกรรมอิเล็กทรอนิกส์และระบบคอมพิวเตอร์

ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม

มหาวิทยาลัยศิลปากร

สารบัญ

บทที่ 1 บทนำ	1
1.วัตถุประสงค์	1
2.เกมที่เลือกใช้พัฒนาในโปรเจกต์	1
3.โปรแกรมที่ใช้งาน	1
4.เทคนิคที่ใช้การเขียนโปรแกรม	1
5.ลักษณะงานที่ได้	2
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง	4
1. Game Structure	4
2. Create the Grid	5
3. บล็อกในและตำแหน่งการหมุน	6
บทที่ 3 ขั้นตอนการดำเนินงาน	10
1. Install Pygame	10
2. Setup the Game Loop	10
3. Create the Grid	10
4. Create the Blocks	10
5. Move the Block.....	11
6. Rotate the Blocks	11
7. Checking for collisions	12
8. Check for completed rows.....	12
9. Game Over	12
10. Create User Interface.....	13
11. Add Score.....	13
12. Add Next Block	13
13. Add Sound	13
บทที่ 4 ขั้นตอนปฏิบัติงาน.....	14
โค้ดการเขียนโปรแกรม	14
1. Tetris.py	14
2. Position.py	16
3. Grid.py	16
4. Game.py	18

5. Colors.py	21
6. Block.py.....	22
7. Blocks.py.....	23
8. ผลการรัน.....	26

บทที่ 1 บทนำ

1.วัตถุประสงค์

เพื่อให้นักศึกษาได้ศึกษาพัฒนาการสร้างสรรค์เกมในรูปแบบต่างๆร่วมกันเป็นกลุ่ม โดยใช้เทคนิคหลักการเขียนโปรแกรมเชิงวัตถุ (OPP) ในจัดการระบบต่างๆของเกม

2.เกมที่เลือกใช้พัฒนาในโปรเจกต์

Tetris: เป็นเกม que ผู้เล่นต้องควบคุมการตกของบล็อกที่มีรูปทรงต่าง ๆ จากด้านบนลงมาในกรอบเพื่อจัดเรียงให้สมบูรณ์บนบางแถว โดยเมื่อมีแถวที่เต็มแล้ว แถวนั้นจะถูกลบออกไป และผู้เล่นจะได้คะแนน ความยากของเกมจะเพิ่มขึ้นเรื่อย ๆ โดยบล็อกจะตกลงลงมาเร็วขึ้นหรือมีรูปทรงที่ซับซ้อนมากขึ้น

3.โปรแกรมที่ใช้งาน

- Visual Studio Code
- Python
- Pygame

4.เทคนิคที่ใช้การเขียนโปรแกรม

ใช้เทคนิคหลักการเขียนโปรแกรมเชิงวัตถุ (OPP)

- 1) Encapsulation
- 2) Polymorphism
- 3) Inheritance

5.ลักษณะงานที่ได้

1) Classes

- Game จะเก็บสถานะและตัวแปรที่เกี่ยวข้องกับเกม Tetris
- Colors เก็บค่าสีที่ใช้ในเกม เช่นสีพื้นหลัง สีของข้อความ หรือสีของรูปทรงต่าง ๆ ที่ใช้ในการวาดหน้าจอของเกม Tetris
- Block สร้างบล็อกที่จะเพิ่มเข้ามาในเกม Tetris ในแต่ละรอบของการเล่นเกม
- Grid สร้างเส้นกริดที่ใช้ในการเล่นเกม Tetris
- Pygame ใช้งานฟังก์ชันและคลาสที่เกี่ยวข้องกับการวาดหน้าจอ
- Position เก็บตำแหน่งของเซลล์ในบล็อกแต่ละรูปแบบ
- Blocks สร้างบล็อกและเก็บบล็อกในรูปแบบต่างๆ

2) Functionality

Tetris.py

- เริ่มต้นใช้งาน pygame เพื่อให้โปรแกรมทำงานได้อย่างถูกต้อง
- สร้างอ็อบเจกต์ของฟอนต์ที่ใช้ในการแสดงข้อความ
- สร้างหน้าต่างเกมโดยกำหนดขนาดของหน้าต่างตามค่า width และ height
- ตั้งเวลาในการเรียกเมธอดที่กำหนดให้กับเหตุการณ์ GAME_UPDATE
- ดึงเหตุการณ์ทั้งหมดที่เกิดขึ้นในเกม
- เช็คว่ามีกริดคีย์บอร์ดหรือไม่
- จบการทำงานของ pygame และปิดหน้าต่างเกมลง

Position.py

- เรียกใช้เมธอดในการกำหนดค่าเริ่มต้นของอ็อบเจกต์เมื่อมีการสร้างอ็อบเจกต์

Grid.py

- เรียกใช้เมธอดเป็น constructor ใช้สำหรับกำหนดค่าเริ่มต้นของ Grid

- แสดงค่าใน Grid ออกทางคอนโซล
- ตรวจสอบว่าตำแหน่ง (row, column) นั้นอยู่ในขอบเขตของ Grid หรือไม่
- ตรวจสอบว่าตำแหน่ง (row, column) ใน Grid นั้นว่างหรือไม่
- ตรวจสอบว่าแถวนั้นมีเซลล์ทุกตัวเต็มหรือไม่
- ลบค่าทุกตัวในแถวนั้นให้เป็น 0
- เลื่อนแถวลงมาในกรณีที่มีแถวด้านบนเต็มแล้ว
- ตรวจสอบแถวที่เต็มแล้วและลบแถวนั้น โดยคืนค่าจำนวนแถวที่ถูกลบ
- ล้างค่าทั้ง Grid ให้เป็น 0 ทั้งหมด
- ใช้เมธอดในการวาด Grid และเซลล์ข้างในโดยใช้ pygame

Game.py

- เรียกใช้เมธอดเป็น constructor ใช้สำหรับกำหนดค่าเริ่มต้นของ Grid
- ใช้เมธอดสำหรับอัปเดตคะแนนเมื่อมีการลบแถว
- ใช้เมธอดสุ่มเลือกบล็อกจากบล็อกที่ยังไม่ถูกใช้แล้ว
- ใช้เมธอดใช้ในการเลื่อนบล็อกไปทางซ้าย, ขวา และลงล่างตามลำดับ
- เมื่อบล็อกพลัดลงไปจะทำการล๊อคบล็อกและเปลี่ยนบล็อกปัจจุบันเป็นบล็อกถัดไป
- รีเซ็ตเกมเพื่อเริ่มเกมใหม่
- ใช้เมธอดตรวจสอบว่าบล็อกตำแหน่งปัจจุบันสามารถอยู่ในกริดได้หรือไม่
- หมุนบล็อกตามเวลาที่กำหนดและเล่นเสียงหมุน
- ใช้เมธอดตรวจสอบว่าบล็อกตำแหน่งปัจจุบันอยู่ในกริดได้หรือไม่
- วาด Grid และบล็อกที่กำลังเล่นอยู่ รวมถึงบล็อกถัดไปบนหน้าจอ pygame

Colors.py

- ใช้เมธอดส่งคืนรายการของสีที่ใช้ในการแสดงบล็อกต่างๆ ในกริดของเกม

Block.py

- ใช้เมธอดในการกำหนดค่าเริ่มต้นของอ็อบเจกต์บล็อก
- ใช้เมธอดในการเลื่อนตำแหน่งของบล็อก
- ใช้เมธอดในการดึงตำแหน่งของเซลล์ภายในบล็อก
- ใช้เมธอดในการหมุนบล็อกและตรวจสอบสถานะ
- ใช้เมธอดในกรณีที่ต้องการย้อนกลับการหมุนบล็อกก่อนหน้านี้
- ใช้เมธอดในการวาดบล็อกลงบนหน้าจอ โดยใช้ขนาดของเซลล์

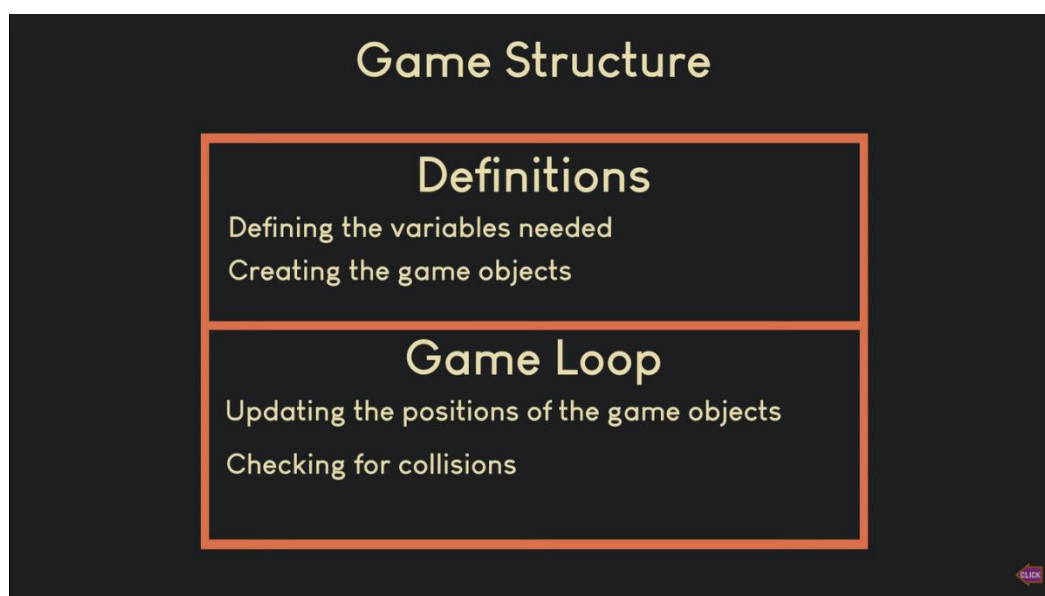
Blocks.py

- ใช้เมธอดในการสำหรับกำหนดค่าเริ่มต้นของบล็อกแต่ละแบบ

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

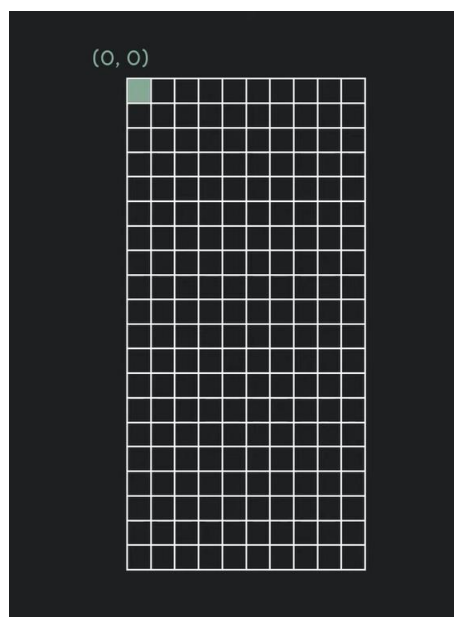
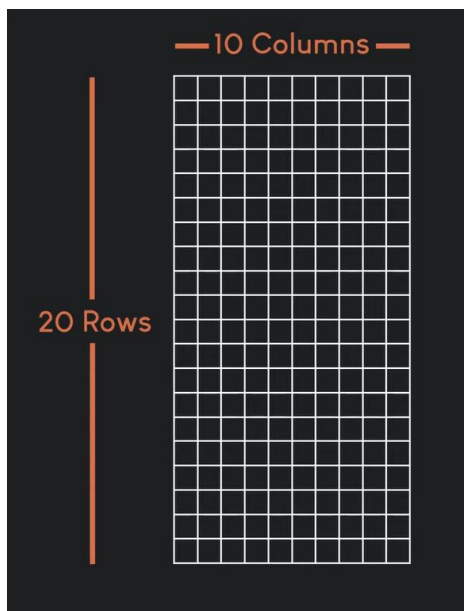
1. Game Structure ประกอบไปด้วย

- 1) Definitions = ต้องกำหนดตัวแปรที่จำเป็นและสร้างอ็อบเจกต์ของเกม
- 2) Game Loop = ต้องอัปเดตตำแหน่งของอ็อบเจกต์ในเกมและตรวจสอบการชนกันหรือทับซ้อน

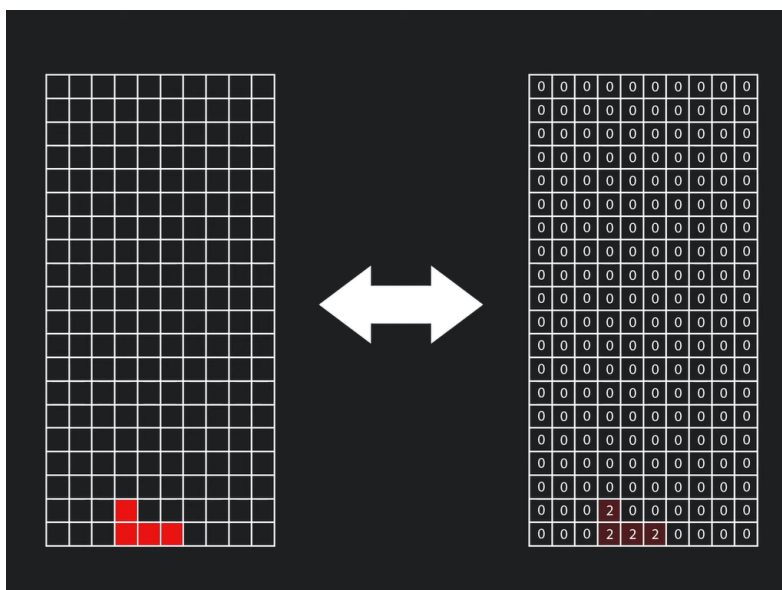


2. Create the Grid

ใน Tetris พื้นที่เล่นเกมจะเป็นตารางประกอบด้วย 20 แถว 10 คอลัมน์ ชิ้นส่วนของเกมจะตกลงมาจากตารางและให้ผู้เล่นเรียงชิ้นส่วนเหล่านั้นให้เรียงเป็นแถวโดยไม่มีช่องว่าง เราจะนับแถวจากบนลงล่างและคอลัมน์ซ้ายไปขวา โดยเซลล์บนซ้ายแถวที่ 0 และคอลัมน์ 0 เป็นจุดกำเนิด

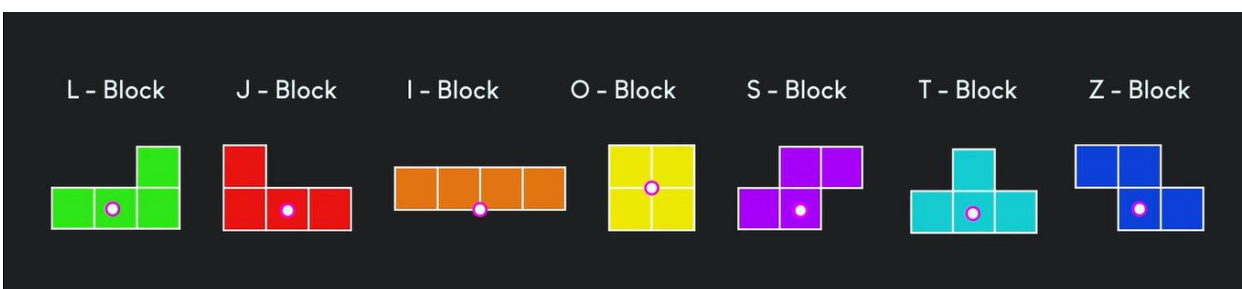


เพื่อให้เป็นตัวแทนของตาราง เราจะใช้อาเรย์สองมิติ ซึ่งสามารถนำไปใช้ในส่วนของ python ได้ในการทำงาน เซลล์ว่างจะแสดงด้วยค่า 0 เมื่อผู้เล่นวางบล็อกลงไปตารางแล้ว เซลล์ที่อยู่ในตำแหน่งจะถูกกำหนดค่าสีของมัน



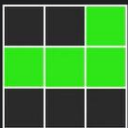
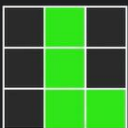

3. บล็อกในและตำแหน่งการหมุน

มีอยู่ 7 รูปแบบ



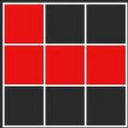
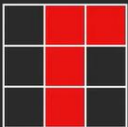

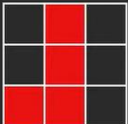
T - Block					
Rotation State 0		(0,1)	(1,0)	(1,1)	(1,2)
Rotation State 1		(0,1)	(1,1)	(1,2)	(2,1)
Rotation State 2		(1,0)	(1,1)	(1,2)	(2,1)
Rotation State 3		(0,1)	(1,0)	(1,1)	(2,1)

L - Block

Rotation State 0		(0,2)	(1,0)	(1,1)	(1,2)
Rotation State 1		(0,1)	(1,1)	(2,1)	(2,2)
Rotation State 2		(1, 0)	(1,1)	(1,2)	(2,0)
Rotation State 3		(0,0)	(0,1)	(1,1)	(2,1)





J - Block

Rotation State 0		(0,0)	(1,0)	(1,1)	(1,2)
Rotation State 1		(0,1)	(0,2)	(1,1)	(2,1)
Rotation State 2		(1, 0)	(1,1)	(1,2)	(2,2)
Rotation State 3		(0,1)	(1,1)	(2,0)	(2,1)



I - Block

Rotation State 0		(1,0)	(1,1)	(1,2)	(1,3)
Rotation State 1		(0,2)	(1,2)	(2,2)	(3,2)
Rotation State 2		(2,0)	(2,1)	(2,2)	(2,3)
Rotation State 3		(0,1)	(1,1)	(2,1)	(3,1)

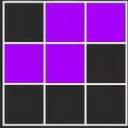
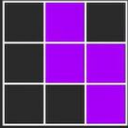
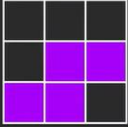
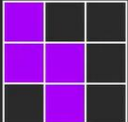


O - Block

Rotation State 0		(0,0)	(0,1)	(1,0)	(1,1)
Rotation State 1		(0,0)	(0,1)	(1,0)	(1,1)
Rotation State 2		(0,0)	(0,1)	(1,0)	(1,1)
Rotation State 3		(0,0)	(0,1)	(1,0)	(1,1)

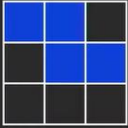
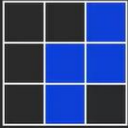
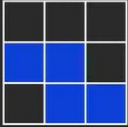
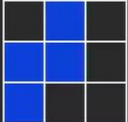


S - Block

Rotation State 0		(0,1) (0,2) (1,0) (1,1)
Rotation State 1		(0,1) (1,1) (1,2) (2,2)
Rotation State 2		(1,1) (1,2) (2,0) (2,1)
Rotation State 3		(0,0) (1,0) (1,1) (2,1)



Z - Block

Rotation State 0		(0,0) (0,1) (1,1) (1,2)
Rotation State 1		(0,2) (1,1) (1,2) (2,1)
Rotation State 2		(1,0) (1,1) (2,1) (2,2)
Rotation State 3		(0,1) (1,0) (1,1) (2,0)



บทที่ 3 ขั้นตอนการดำเนินงาน

ขั้นตอนการพัฒนาเกม Tetris มี 13 ขั้นตอน

1. Install Pygame

ติดตั้ง Pygame ใน Command Prompt ด้วยคำสั่ง `pip install pygame-ce`

2. Setup the Game Loop

- 1) สร้างไฟล์ main ตั้งชื่อว่า `Tetris.py` จากนั้น `import pygame`
- 2) สร้าง Game Window (Display Surface) ในโปรเจกต์ของเราจะใช้ขนาดกว้างxสูง 500x620 พิกเซล
- 3) สร้าง Game Loop

องค์ประกอบของสร้าง Game Loop ประกอบไปด้วย 3 ส่วนหลักๆ

 - Event Handling
 - Updating Positions
 - Drawing Objects

3. Create the Grid

- 1) สร้างไฟล์ใหม่ ตั้งชื่อว่า `Grid.py` จากนั้น ให้สร้าง class `Grid`
- 2) กำหนดค่าเริ่มต้นของอ็อบเจกต์ (Grid) โดยกำหนดขนาดและสีของGridแต่ละเซลล์
- 3) สร้างฟังก์ชันแสดง Grid ลงในจอร์รูปแบบตาราง
- 4) สร้างฟังก์ชันสำหรับวาด Grid ในเกมลงบนจอ Pygame ที่กำหนดไว้
- 5) เพิ่ม class colors ในไฟล์ `Grid.py`

4. Create the Blocks

- 1) สร้างไฟล์ใหม่ ตั้งชื่อว่า `Colors.py` จากนั้น ให้สร้าง class `Colors`
 - กำหนดค่าสีให้กับตัวแปรต่างๆ
 - เรียกใช้เมธอดของคลาสที่ถูกใช้ในการคืนค่าสีของเซลล์ต่างๆ
 - เพิ่ม class colors ในไฟล์ `Tetris.py`
- 2) สร้างไฟล์ใหม่ ตั้งชื่อว่า `Position.py` จากนั้นสร้าง class `Position` เพื่อเก็บข้อมูลในตารางหรือ Grid
- 3) สร้างไฟล์ใหม่ ตั้งชื่อว่า `Block.py` จากนั้น ให้สร้าง class `Block`
 - เรียกใช้คอนสตรักเตอร์ที่ใช้ในการสร้างอ็อบเจกต์ของคลาส `Block`
 - เรียกใช้เมธอดที่ใช้ในการวาดบล็อกลงบนหน้าจอ Pygame

- เพิ่ม class colors และ class position ในไฟล์ Block.py
- 4) สร้างไฟล์ใหม่ ตั้งชื่อว่า Blocks.py จากนั้น ให้สร้าง class Blocks เพื่อสร้างรูปแบบ Block มี 7 รูปแบบได้แก่
 - LBlock, JBlock, IBlock, OBlock, SBlock, TBlock, ZBlock
 - เพิ่ม class Block และ class Position ในไฟล์ Blocks.py

5. Move the Block

- 1) ส่วนของ Block.py
 - เรียกใช้เมธอดที่ใช้ในการเคลื่อนที่บล็อกในแนวแถวและคอลัมน์
 - เรียกใช้เมธอดที่ใช้ในการคำนวณตำแหน่งของเซลล์แต่ละตัวของบล็อก
- 2) ส่วนของ Blocks.py
 - เพิ่มคำสั่ง self.move ให้กับ class Blocks ทั้ง 7 รูปแบบ
- 3) สร้างไฟล์ใหม่ ตั้งชื่อว่า game.py จากนั้นสร้าง class Game
 - เพิ่ม class game ในไฟล์ Tetris.py
 - เรียกใช้เมธอดในการสร้างอ็อบเจกต์ของคลาส Game
 - เรียกใช้เมธอดเลือกบล็อกสุ่มจากลิสต์ที่ยังไม่ถูกใช้และสร้างอ็อบเจกต์ของบล็อกนั้นขึ้นมา
 - เรียกใช้เมธอดที่ใช้ในการเลื่อนบล็อกไปทางซ้าย
 - เรียกใช้เมธอดที่ใช้ในการเลื่อนบล็อกไปทางขวา
 - เรียกใช้เมธอดที่ใช้ในการเลื่อนบล็อกลง
 - เรียกใช้เมธอดที่ใช้ในการตรวจสอบว่าบล็อกอยู่ในกริดหรือตารางของเกม
- 4) ส่วนของ Tetris.py
 - เรียกใช้เมธอดตรวจสอบเป็นพินช์และสถานะของเกม
- 5) ส่วนของ Grid.py
 - สร้างฟังก์ชันตรวจสอบตำแหน่งที่กำหนดโดยพิกัดและคอลัมน์อยู่ขอบเขตของGridหรือไม่

6. Rotate the Blocks

- 1) ส่วนของ Block.py
 - เรียกใช้เมธอดเพิ่มและลดค่าสถานะการหมุนของ Block
- 2) ส่วนของ Tetris.py
 - เรียกใช้เมธอดตรวจสอบเป็นพินช์และสถานะของเกม

7. Checking for collisions

1) ส่วนของ Tetris.py

- กำหนดค่าให้ตัวแปรและเวลาให้กับ event
- เรียกใช้เมธอดตรวจสอบแป้นพิมพ์และสถานะของเกม

2) ส่วนของ Game.py

- เรียกใช้เมธอดที่ใช้ในการล๊อคบล็อกและเช็คสถานะเกมว่าเกมจบหรือยัง
- เรียกใช้เมธอดที่ใช้ในการตรวจสอบว่าบล็อกพอดีกับตำแหน่งของบล็อกหรือไม่

3) ส่วนของ Grid.py

- เรียกใช้เมธอดตรวจสอบตำแหน่งเซลล์ที่กำหนดโดยพิกัดและคอลัมน์อยู่ขอบเขตของGrid ที่ว่างอยู่หรือไม่

8. Check for completed rows

1) ส่วนของ Grid.py

- สร้างฟังก์ชันตรวจสอบแถวที่กำหนดใน Grid
- สร้างฟังก์ชันลบข้อมูลในแถวที่กำหนดในGrid
- สร้างฟังก์ชันเลื่อนข้อมูลในแถวที่กำหนดลงมาในแถวอื่นๆ ใน Grid
- สร้างฟังก์ชันตรวจสอบและลบแถวที่เต็มของ Grid ออก และเลื่อนแถวลงมาหากมีแถวที่ถูกลบออก โดยคืนค่าจำนวนแถวที่ถูกลบออกในการทำความสะอาดแถว

9. Game Over

1) ส่วนของ Tetris.py

- กำหนดสถานะของเกมเมื่อจบและรีเซ็ต

2) ส่วนของ Game.py

- เรียกใช้งานเมธอดรีเซ็ตเกม

10. Create User Interface

- 1) วิธีการสร้าง Text
 - สร้างฟอนต์
 - สร้าง Surface กับ text
 - สร้าง Surface ด้วยการเรียกใช้งานเมธอด blit()
- 2) จัดเรียงตำแหน่ง text ตามความต้องการ

11. Add Score

- 1) ส่วนของ Game.py
 - เรียกใช้เมธอดที่ใช้ในการอัปเดตคะแนนเมื่อมีการลบแถวบล็อกออก
- 2) ส่วนของ Tetris.py
 - สร้าง Surface นำไปแสดงประกอบเป็นคะแนนแล้วแสดงออกทางหน้าจอ

12. Add Next Block

- 1) ส่วนของ Game.py
 - เรียกใช้เมธอดที่ใช้ในการวาดสถานะปัจจุบันของเกมบนหน้าจอ

13. Add Sound

- 1) ส่วนของ Game.py
 - สร้างฟังก์ชันสำหรับโหลดเสียงเกม (เสียงพื้นหลัง, เสียงหมุน Block, เสียงล้างแถว)

บทที่ 4 ขั้นตอนปฏิบัติงาน

โค้ดการเขียนโปรแกรม

1. Tetris.py

```

1. import pygame # เพิ่มไลบรารี pygame
2. from Game import Game # เพิ่มคลาส Game จากไฟล์ Game.py
3. from Colors import Colors # เพิ่มคลาส Colors จากไฟล์ Colors.py
4.
5. pygame.init() # เรียกเมธอดเพื่อเริ่มต้นใช้งาน pygame
6.
7. title_font = pygame.font.Font(None, 40) # สร้างอ็อบเจกต์ของฟอนต์
8. score_surface = title_font.render("Score", True, Colors.white)
9. # แสดง Surface ของ "Score" ที่ตำแหน่ง (x, y) บนหน้าจอ
10.
11. next_surface = title_font.render("Next", True, Colors.white)
12. # แสดง Surface ของ "Next" ที่ตำแหน่ง (x, y) บนหน้าจอ
13.
14. game_over_surface = title_font.render("GAME OVER", True, Colors.drakred)
15. # แสดง Surface ของ "GAME OVER" ที่ตำแหน่ง (x, y) บนหน้าจอ
16.
17. # เรียกใช้เมธอดในการกำหนดตำแหน่งและตรวจสอบการชนกันสิ่งของในเกม
18. score_rect = pygame.Rect(320, 55, 170, 60)
19. next_rect = pygame.Rect(320, 215, 170, 180)
20.
21. screen_width=500 # กำหนดค่าให้ตัวแปร
22. screen_height=620 # กำหนดค่าให้ตัวแปร
23.
24. screen=pygame.display.set_mode((screen_width,screen_height))
25. # เรียกใช้เมธอดเพื่อสร้างหน้าต่างเกม
26. pygame.display.set_caption("Tetris 4027")
27. # เรียกใช้เมธอดเพื่อแสดงชื่อหัวข้อของหน้าต่างเกม
28.
29. clock = pygame.time.Clock() # กำหนดอ็อบเจกต์ของคลาส
30. FPS = 60 # กำหนดค่าตัวแปร
31.
32. game = Game()
33.
34. GAME_UPDATE = pygame.USEREVENT # กำหนดตัวแปรเพื่อเรียกใช้ event ต่างๆ
35. pygame.time.set_timer(GAME_UPDATE, 200)
36. # กำหนดเมธอดที่ใช้ตั้งค่าเวลาสำหรับ event ต่างๆ
37.
38. # LOOP
39. run=True # กำหนดค่าตัวแปร
40. while run:

```

```

41.
42.     #event hand
43.     for event in pygame.event.get(): # กำหนดลูปของเหตุการณ์ต่างๆ
44.         if event.type == pygame.QUIT: # กำหนดเงื่อนไขเป็นตรวจสอบเหตุการณ์การปิดหน้าต่าง
45.             run = False # ออกจากลูป(ปิดโปรแกรม)
46.         if event.type == pygame.KEYDOWN: # กำหนดเงื่อนไขตรวจสอบเป็นพิมพ์
47.             if game.game_over == True: # กำหนดสถานะของเกม (True คือจบไปแล้ว)
48.                 game.game_over = False # กำหนดสถานะของเกม (False คือการเริ่มรีเซ็ตเกม)
49.                 game.reset() # เรียกใช้เมธอดเพื่อรีเซ็ตเกม
50.             if event.key == pygame.K_LEFT and game.game_over == False:
51.                 # กำหนดเงื่อนไขตรวจสอบเป็นพิมพ์และสถานะของเกม
52.
53.                 game.move_left() # เรียกใช้เมธอดเพื่อเลื่อนไปด้านซ้าย 1 ช่อง
54.             if event.key == pygame.K_RIGHT and game.game_over == False:
55.                 # กำหนดเงื่อนไขตรวจสอบเป็นพิมพ์และสถานะของเกม
56.
57.                 game.move_right() # เรียกใช้เมธอดเพื่อเลื่อนไปด้านขวา 1 ช่อง
58.             if event.key == pygame.K_DOWN and game.game_over == False:
59.                 # กำหนดเงื่อนไขตรวจสอบเป็นพิมพ์และสถานะของเกม
60.
61.                 game.move_down() # เรียกใช้เมธอดเพื่อเลื่อนไปด้านล่าง 1 ช่อง
62.             if event.key == pygame.K_UP and game.game_over == False:
63.                 # กำหนดเงื่อนไขตรวจสอบเป็นพิมพ์และสถานะของเกม
64.
65.                 game.rotate() # เรียกใช้เมธอดเพื่อหมุนทิศทางบล็อก
66.             if event.type == pygame.UPDATE and game.game_over == False:
67.                 # กำหนดเงื่อนไขเป็นตรวจสอบเหตุการณ์และสถานะของเกม
68.
69.                 game.move_down() # เรียกใช้เมธอดเพื่อเลื่อนไปด้านล่าง 1 ช่อง
70.     #draw
71.     score_value_surface = title_font.render(str(game.score), True,
Colors.blue)
72.     # โหลดฟอนต์ออปเจกต์มา render เป็นค่า score --
73.
74.     screen.fill(Colors.blue) # เติมสีพื้นหลัง
75.     screen.blit(score_surface, (365, 20, 50, 50)) # แสดง surface ของค่า score
76.     screen.blit(next_surface, (375, 180, 50, 50)) # แสดง surface ของค่า next
77.
78.     #If game over
79.     if game.game_over == True: # เงื่อนไขตรวจสอบเกมจบหรือไม่
80.         screen.blit(game_over_surface, (320, 450, 50, 50))
81.         # ใช้แสดง Surface ของข้อความ "GAME OVER"
82.
83.     # ใช้เมธอดในการสร้างสี่เหลี่ยมผืนผ้าและแสดงออกทางจอ
84.     pygame.draw.rect(screen, Colors.lightgreen, score_rect, 0, 10)

```

```

85.     screen.blit(score_value_surface, score_value_surface.get_rect(centerx
    = score_rect.centerx,
86.         centery = score_rect.centery))
87.     pygame.draw.rect(screen, Colors.lightgrey, next_rect, 0, 10)
88.     game.draw(screen)
89.
90.     clock.tick(FPS)
91.     #update display
92.     pygame.display.update()
93. pygame.quit

```

2. Position.py

```

class Position: # สร้าง class Position
    def __init__(self, row, column): # รับพารามิเตอร์เพื่อเข้าถึงคุณสมบัติ
        self.row=row # สร้างแอตทริบิวต์ให้ row
        self.column=column # สร้างแอตทริบิวต์ให้ column

```

3. Grid.py

```

import pygame # เพิ่มไลบรารี pygame
from Colors import Colors # เพิ่มคลาส Colors จากไฟล์ Colors.py

class Grid: # สร้าง class Grid
    def __init__(self): # รับพารามิเตอร์เพื่อเข้าถึงคุณสมบัติ
        self.num_rows =20 # กำหนดจำนวนแถวในกริด
        self.num_cols =10 # กำหนดจำนวนคอลัมน์ในกริด
        self.cell_size =30 # กำหนดขนาดของเซลล์ในกริด
        self.grid = [[0 for j in range(self.num_cols)] for i in
range(self.num_rows)]
        # สร้างกริดเปล่าๆ โดยกำหนดค่าเริ่มต้นให้เป็น 0
        self.colors = Colors.get_cell_colors()
        # ดึงข้อมูลสีของเซลล์จาก Colors class โดยใช้เมทอด get_cell_colors()

    def print_grid(self):
        for row in range(self.num_rows):
            for column in range(self.num_cols):
                print(self.grid[row][column], end= " ")
                # แสดงค่าในเซลล์นั้นๆ และใช้ end=" " เพื่อให้เคอร์เซอร์เคลื่อนไปในบรรทัดถัดไป
            print() # ขึ้นบรรทัดใหม่หลังจากแสดงค่าในแต่ละคอลัมน์ในแถวนั้นเสร็จ

    def is_inside(self, row, column):
        if row >= 0 and row < self.num_rows and column >= 0 and column <
self.num_cols:

```

```

        return True # ถ้าตำแหน่งอยู่ในกริด ให้คืนค่า True
    return False # ถ้าตำแหน่งอยู่นอกกริด ให้คืนค่า False

def is_empty(self, row, column):
    if self.grid[row][column] == 0: # ถ้าค่าในเซลล์ตำแหน่งที่กำหนดเป็น 0 (หมายถึงว่าง)
        return True # คืนค่า True ว่าเซลล์นี้ว่าง
    return False # คืนค่า False ว่าเซลล์นี้ไม่ว่าง

def is_row_full(self, row):
    for column in range(self.num_cols):
        if self.grid[row][column] == 0: # ถ้ามีเซลล์ในแถวที่ยังว่าง
            return False # คืนค่า False ว่าแถวนี้ยังไม่เต็ม
    return True # คืนค่า True ว่าแถวนี้เต็มทั้งแถว

def clear_row(self, row):
    for column in range(self.num_cols):
        self.grid[row][column] = 0
        # กำหนดค่าของทุกเซลล์ในแถวที่กำหนดให้เป็น 0 (ค่าว่าง)

def move_row_down(self, row, num_rows):
    for column in range(self.num_cols):
        # เลื่อนค่าในเซลล์ในแถว row ลงไป num_rows แถวด้านล่าง
        self.grid[row+num_rows][column] = self.grid[row][column]
        # หลังจากเคลื่อนแล้วให้เซลล์ในแถว row เป็นค่าว่าง (0)
        self.grid[row][column] = 0

def clear_full_rows(self):
    complete = 0 # นับจำนวนแถวที่ถูกเติมและถูกลบ
    for row in range(self.num_rows-1, 0, -1):
        # วนลูปจากแถวล่างสุดไปข้างบน
        if self.is_row_full(row): # ถ้าแถวนี้เต็ม
            self.clear_row(row) # ลบแถวนี้
            complete += 1 # เพิ่มจำนวนแถวที่ถูกลบ
        elif complete > 0: # ถ้ามีแถวที่ถูกลบอยู่
            self.move_row_down(row, complete)
            # เลื่อนแถวลงเพื่อกระชับกับจำนวนแถวที่ถูกลบ
    return complete
# คืนค่าจำนวนแถวที่ถูกลบ

def reset(self):
    for row in range(self.num_rows):
        for column in range(self.num_cols):
            self.grid[row][column] = 0
            # กำหนดค่าในเซลล์ทุกเซลล์ในกริดให้เป็น 0 (ค่าว่าง)

```

```

def draw(self,screen):
    for row in range(self.num_rows):
        for column in range(self.num_cols):
            cell_value = self.grid[row][column] # ดึงค่าที่อยู่ในเซลล์นั้นๆ จากกริด
            # สร้างสี่เหลี่ยมสี่เหลี่ยมเล็กที่จะเป็นเซลล์ในกริด
            cell_rect = pygame.Rect(column*self.cell_size+11,
row*self.cell_size+11,
            self.cell_size-1, self.cell_size-1)
            # วาดสี่เหลี่ยมสี่เหลี่ยมเล็กที่จะแสดงเซลล์ในกริดด้วยสีที่เกี่ยวข้องกับค่าในเซลล์
            pygame.draw.rect(screen, self.colors[cell_value], cell_rect)

```

4. Game.py

```

1. from Grid import Grid # เพิ่ม class Grid จาก Grid.py
2. from Blocks import * # เพิ่ม class Blocks จาก Blocks.py
3. import random # เพิ่มไลบรารี random
4. import pygame # เพิ่มไลบรารี pygame
5.
6. class Game: # สร้าง class Game
7.     def __init__(self): # รับพารามิเตอร์เพื่อเข้าถึงคุณสมบัติ
8.         self.grid = Grid() # สร้างกริดข้อมูลเปล่าๆ
9.         self.blocks = [IBlock(), JBlock(), LBlock(), OBlock(), SBlock(),
TBLOCK(), ZBlock()]
10.        # สร้างบล็อกทั้งหมดที่มีในเกม
11.
12.        self.current_block = self.get_random_block() # กำหนดบล็อกปัจจุบันที่จะใช้ในการ
เล่น
13.        self.next_block = self.get_random_block() # กำหนดบล็อกต่อไปที่จะใช้ในการเล่น
14.        self.game_over = False # กำหนดสถานะเกมว่ายังไม่ Game Over
15.        self.score = 0 # กำหนดคะแนนเริ่มต้นให้เป็น 0
16.        self.rotate_sound = pygame.mixer.Sound("music/Cartoon Boing.mp3")
17.        # โหลดเสียงสำหรับการหมุนบล็อก
18.        self.clear_sound = pygame.mixer.Sound("music/Wood Plank
Flicks.mp3")
19.        # โหลดเสียงสำหรับเมื่อลบแถว
20.
21.        pygame.mixer.music.load("music/Girasol - Quincas Moreira.mp3")
22.        # โหลดเพลงเพื่อให้เป็นเสียงพื้นหลัง

```

```

23.     pygame.mixer.music.play()
24.     # เล่นเพลงเพื่อให้เป็นเสียงพื้นหลังของเกม
25.
26.     def update_score(self, lines_cleared, move_down_points):
27.         if lines_cleared == 1:
28.             self.score += 100 # เพิ่มคะแนน 100 เมื่อลบแถว 1 แถว
29.         elif lines_cleared == 2:
30.             self.score += 300 # เพิ่มคะแนน 300 เมื่อลบแถว 2 แถว
31.         elif lines_cleared == 3:
32.             self.score += 500 # เพิ่มคะแนน 500 เมื่อลบแถว 3 แถว
33.         self.score += move_down_points # เพิ่มคะแนนจากการเลื่อนลงของบล็อก
34.
35.     def get_random_block(self):
36.         if len(self.blocks) == 0: # ถ้าไม่มีบล็อกที่เหลือในลิสต์
37.             self.blocks = [IBlock(), JBlock(), LBlock(), OBlock(),
38.                             SBlock(), TBlock(), ZBlock()]
39.             # เพิ่มบล็อกทั้งหมดเข้าไปใหม่
40.         block = random.choice(self.blocks) # เลือกบล็อกจากลิสต์โดยสุ่ม
41.         self.blocks.remove(block) # ลบบล็อกที่ถูกสุ่มเลือกออกจากลิสต์
42.         return block # คืนค่าบล็อกที่ถูกสุ่มเลือกออกมา
43.
44.     def move_left(self):
45.         self.current_block.move(0,-1)
46.         # เคลื่อนที่บล็อกปัจจุบันไปทางซ้ายหนึ่งช่อง (เพิ่มตำแหน่ง column ลบออกหนึ่ง)
47.         if self.block_inside() == False or self.block_fits() == False:
48.             # ตรวจสอบว่าบล็อกอยู่ในกริดหรือไม่ และว่าบล็อกจะตั้งอยู่ในตำแหน่งนั้นหรือไม่
49.             self.current_block.move(0,1)
50.
51.     def move_right(self):
52.         self.current_block.move(0,1)
53.         # เคลื่อนที่บล็อกปัจจุบันไปทางขวาหนึ่งช่อง (เพิ่มตำแหน่ง column ขึ้นอยู่ 1)
54.         if self.block_inside() == False or self.block_fits() == False:
55.             # ตรวจสอบว่าบล็อกอยู่ในกริดหรือไม่ และว่าบล็อกจะตั้งอยู่ในตำแหน่งนั้นหรือไม่
56.             self.current_block.move(0,-1)
57.
58.     def move_down(self):
59.         self.current_block.move(1,0)
60.         # เคลื่อนที่บล็อกปัจจุบันลงมาหนึ่งช่อง (เพิ่มตำแหน่ง row ขึ้นอยู่ 1)
61.         if self.block_inside() == False or self.block_fits() == False:
62.             # ตรวจสอบว่าบล็อกอยู่ในกริดหรือไม่ และว่าบล็อกจะตั้งอยู่ในตำแหน่งนั้นหรือไม่
63.             self.current_block.move(-1,0)
64.             self.lock_block()
65.
66.     def lock_block(self):
67.         tiles = self.current_block.get_cell_position()

```

```

67.         # ค้างตำแหน่งของแต่ละเซลล์ในบล็อกที่ตั้งอยู่ปัจจุบัน
68.         for position in tiles:
69.             self.grid.grid[position.row][position.column]=self.current_block.id
70.             self.current_block = self.next_block # กำหนดให้บล็อกปัจจุบันเป็นบล็อกถัดไป
71.             self.next_block = self.get_random_block() # สุ่มบล็อกใหม่ที่จะเป็นบล็อกถัดไป
72.             rows_cleared = self.grid.clear_full_rows()
73.             # ตรวจสอบและลบแถวที่เต็มในกริดและนับจำนวนแถวที่ถูกลบ
74.
75.             # หากมีแถวที่ถูกลบ ให้เล่นเสียงที่ต้องการ (เช่น เสียงเมื่อลบแถว) และอัปเดตคะแนน
76.             if rows_cleared > 0:
77.                 self.clear_sound.play()
78.                 self.update_score(rows_cleared, 0)
79.             if self.block_fits() == False:
80.                 self.game_over = True
81.                 # ตรวจสอบว่าบล็อกถัดไปจะสามารถเข้าไปได้หรือไม่ หากไม่สามารถเข้าไปได้แสดงว่าเกมจบลง
82.
83.     def reset(self):
84.         self.grid.reset() # เรียกเมธอดในการรีเซ็ต
85.         self.blocks = [IBlock(), JBlock(), LBlock(), OBlock(), SBlock(),
86.             TBlock(), ZBlock()]
87.         # สร้างลิสต์ของบล็อกทั้งหมดใหม่
88.         self.current_block = self.get_random_block()
89.         # สุ่มบล็อกใหม่เพื่อเป็นบล็อกปัจจุบัน
90.         self.next_block = self.get_random_block()
91.         # สุ่มบล็อกใหม่เพื่อเป็นบล็อกถัดไป
92.
93.     def block_fits(self):
94.         tiles = self.current_block.get_cell_position()
95.         # ค้างตำแหน่งของเซลล์ในบล็อกปัจจุบัน
96.
97.         for tile in tiles:
98.             if self.grid.is_empty(tile.row, tile.column) == False:
99.                 # ตรวจสอบว่าตำแหน่งใดๆ ของบล็อกไม่ว่าง
100.                 return False # ถ้ามีตำแหน่งใดตำแหน่งหนึ่งไม่ว่าง คืนค่า False
101.                 return True # ถ้าทุกตำแหน่งของบล็อกว่าง คืนค่า True
102.
103.     def rotate(self):
104.         self.current_block.rotate() # เรียกใช้เมธอดในการหมุน
105.         if self.block_inside() == False:
106.             # ตรวจสอบว่าหลังจากหมุนบล็อกแล้ว บล็อกนั้นอยู่ในตำแหน่งที่ต้องการหรือไม่
107.             self.current_block.undo_rotation()
108.             # หากบล็อกหมุนแล้ว ไม่ได้อยู่ในตำแหน่งที่ต้องการ (ไม่อยู่ภายในกริด) จะทำการย้อนกลับการหมุนให้เป็นตำแหน่งเดิม
109.         else:

```

```

109.         self.rotate_sound.play()
110.         # กรณีที่บล็อกหมุนแล้วอยู่ในตำแหน่งที่ถูกต้อง ให้เล่นเสียงการหมุนบล็อกด้วย
111.
112.     def block_inside(self):
113.         tiles = self.current_block.get_cell_position()
114.         # เรียกใช้เมธอดดึงตำแหน่งของแต่ละเซลล์ในบล็อกปัจจุบัน
115.         for tile in tiles: # วนลูปผ่านเซลล์แต่ละตัวในบล็อกปัจจุบัน
116.             if self.grid.is_inside(tile.row, tile.column) == False:
117.                 # ตรวจสอบว่าตำแหน่งของเซลล์นั้นอยู่ภายในขอบเขตของกริดหรือไม่
118.                 return False
119.             # หากพบว่าเซลล์ใดๆ ในบล็อกปัจจุบันไม่ได้้อยู่ภายในขอบเขตของกริด จะคืนค่า False
120.         return True
121.         # หากทุกเซลล์ในบล็อกปัจจุบันอยู่ภายในขอบเขตของกริดทั้งหมด จะคืนค่า True
122.
123.     def draw(self, screen):
124.         self.grid.draw(screen)
125.         self.current_block.draw(screen, 10, 10)
126.         # เรียกใช้เมธอดของบล็อกปัจจุบัน
127.
128.         if self.next_block.id == 3: # ตรวจสอบบล็อกถัดไป
129.             self.next_block.draw(screen, 255, 290)
130.             # หาก self.next_block.id == 3 ให้วาดบล็อกถัดไปที่ตำแหน่ง x=255 และ y=290
131.         elif self.next_block.id == 4:
132.             # หาก self.next_block.id == 4 ให้วาดบล็อกถัดไปที่ตำแหน่ง x=255 และ y=280
133.             self.next_block.draw(screen, 255, 280)
134.         else:
135.             # หากไม่ตรงกับเงื่อนไขข้างต้น ให้วาดบล็อกถัดไปที่ตำแหน่ง x=270 และ y=270
136.             self.next_block.draw(screen, 270, 270)
137.

```

5. Colors.py

```

class Colors: # สร้าง class Colors
    lightgrey = (0, 0, 0) # สีเทาอ่อน
    lime = (102, 255, 102) # สีเขียวไทม์
    pink = (255, 102, 153) # สีชมพู
    lightorange = (255, 153, 102) # สีส้มอ่อน
    yellow = (255, 255, 153) # สีเหลือง
    purple = (153, 102, 255) # สีม่วง
    cyan = (204, 255, 255) # สีฉวอัน (ฟ้าอมเขียว)
    lightblue = (102, 153, 255) # สีฟ้าอ่อน
    white = (255, 255, 255) # สีขาว
    red = (255, 0, 0) # สีแดง
    green = (0, 255, 0) # สีเขียว
    blue = (0, 0, 255) # สีน้ำเงิน

```



```

lightgreen = (204, 255, 255) # สีเขียวอ่อน
drakred = (204, 0, 102) # สีแดงเข้ม
gold = (255, 204, 0) # สีทอง

@classmethod
def get_cell_colors(cls): # ที่รับค่าจากคลาสเพื่อคืนค่ารายการสีอื่นๆ
    return[cls.lightgrey, cls.lime, cls.pink, cls.lightorange, cls.yellow,
cls.purple, cls.cyan, cls.lightblue]

```

6. Block.py

```

from Colors import Colors # เพิ่ม class Colors ไฟล์ Colors.py
import pygame # เพิ่มไลบรารี pygame
from Position import Position # เพิ่ม class Position ไฟล์ Position.py

class Block: # สร้าง class Block
    def __init__(self, id):
        self.id = id
        self.cells = {} # สร้าง dictionary ที่ใช้เก็บเซลล์(cells)
        self.cell_size = 30 # ขนาดของเซลล์ 30 พิกเซล
        self.row_offset = 0 # กำหนดค่า offset สำหรับการเคลื่อนที่แถวของเซลล์ในกริด
        self.column_offset = 0 # กำหนดค่า offset สำหรับการเคลื่อนที่คอลัมน์ของเซลล์ในกริด
        self.rotation_state = 0 # กำหนดสถานะเริ่มต้นของการหมุนเซลล์ในกริด
        self.colors = Colors.get_cell_colors() # ดึงสีที่ใช้แสดงบนเซลล์มาจากคลาส Colors

    def move(self, rows, columns):
        self.row_offset += rows # เพิ่มหรือลดการเคลื่อนที่ของแถว
        self.column_offset += columns # เพิ่มหรือลดการเคลื่อนที่ของคอลัมน์

    def get_cell_position(self):
        tiles=self.cells[self.rotation_state]
        # เลือกตำแหน่งของเซลล์ที่มีการหมุนตาม self.rotation_state
        moved_tiles = [] # สร้างรายการเพื่อเก็บตำแหน่งของเซลล์หลังจากการเคลื่อนที่
        for position in tiles:
            # คำนวณตำแหน่งใหม่ของเซลล์หลังจากการเคลื่อนที่แล้วและเพิ่มเข้าไปในรายการ moved_tiles

```

```

        position = Position(position.row+ self.row_offset, position.column +
self.column_offset)
        moved_tiles.append(position)
        return moved_tiles # ส่งคืนรายการของตำแหน่งของเซลล์หลังจากการเคลื่อนที่แล้ว

def rotate(self):
    self.rotation_state += 1 # เพิ่มค่าของการหมุนเซลล์
    if self.rotation_state == len(self.cells):
        # ตรวจสอบว่าเมื่อหมุนถึงขอบเขตของตำแหน่งที่เป็นไปได้หมดแล้ว
        self.rotation_state = 0
        # กลับมาที่สถานะเริ่มต้น

def undo_rotation(self):
    self.rotation_state -= 1 # ลดค่าของการหมุนเซลล์
    if self.rotation_state == -1:
        # ตรวจสอบว่าถ้าหมุนไปถึงสถานะ -1 ให้กลับมาที่สถานะสุดท้ายของการหมุน
        self.rotation_state = len(self.cells) - 1
        # กลับไปยังสถานะสุดท้ายของการหมุน

def draw (self, screen, offset_x, offset_y):
    tiles = self.get_cell_position() # รับตำแหน่งของเซลล์ที่จะวาด
    for tile in tiles:
        # สร้าง rectangle สำหรับเซลล์แต่ละตัว
        tile_rect = pygame.Rect(offset_x + tile.column*self.cell_size+1,
                                offset_y + tile.row*self.cell_size+1, self.cell_size-1,
self.cell_size-1)
        # วาดเซลล์ด้วยสีที่กำหนดใน self.colors[self.id] บนหน้าจอ
        pygame.draw.rect(screen, self.colors[self.id], tile_rect)

```

7. Blocks.py

```

from Block import Block # เพิ่ม class Block 'ไฟล์Block.py
from Position import Position # เพิ่ม class Position 'ไฟล์Position.py

class LBlock(Block):
    def __init__(self):
        # เรียก constructor ของ superclass (Block) ด้วย super().__init__(id=1)
        super().__init__(id=1)

        # กำหนดตำแหน่งของเซลล์แต่ละตัวในแต่ละการหมุนของบล็อก
        self.cells = {
            0: [Position(0,2),Position(1,0),Position(1,1),Position(1,2)],
            1: [Position(0,1),Position(1,1),Position(2,1),Position(2,0)],

```

```

        2: [Position(1,0),Position(1,1),Position(1,2),Position(2,0)],
        3: [Position(0,0),Position(0,1),Position(1,1),Position(2,1)]
    }
    self.move(0, 3) # เลื่อนตำแหน่งเริ่มต้นของบล็อกไปที่แถว 0 และคอลัมน์ 3 ในกริด

class JBlock(Block):
    def __init__(self):
        # เรียก constructor ของ superclass (Block) ด้วย super().__init__(id=2)
        super().__init__(id = 2)

        # กำหนดตำแหน่งของเซลล์แต่ละตัวในแต่ละการหมุนของบล็อก
        self.cells = {
            0: [Position(0, 0), Position(1, 0), Position(1, 1), Position(1, 2)],
            1: [Position(0, 1), Position(0, 2), Position(1, 1), Position(2, 1)],
            2: [Position(1, 0), Position(1, 1), Position(1, 2), Position(2, 2)],
            3: [Position(0, 1), Position(1, 1), Position(2, 0), Position(2, 1)]
        }
        self.move(0, 3) # เลื่อนตำแหน่งเริ่มต้นของบล็อกไปที่แถว 0 และคอลัมน์ 3 ในกริด

class IBlock(Block):
    def __init__(self):
        # เรียก constructor ของ superclass (Block) ด้วย super().__init__(id=3)
        super().__init__(id = 3)

        # กำหนดตำแหน่งของเซลล์แต่ละตัวในแต่ละการหมุนของบล็อก
        self.cells = {
            0: [Position(1, 0), Position(1, 1), Position(1, 2), Position(1, 3)],
            1: [Position(0, 2), Position(1, 2), Position(2, 2), Position(3, 2)],
            2: [Position(2, 0), Position(2, 1), Position(2, 2), Position(2, 3)],
            3: [Position(0, 1), Position(1, 1), Position(2, 1), Position(3, 1)]
        }
        self.move(-1, 3) # เลื่อนตำแหน่งเริ่มต้นของบล็อกไปที่แถว -1 และคอลัมน์ 3 ในกริด

class OBlock(Block):
    def __init__(self):
        # เรียก constructor ของ superclass (Block) ด้วย super().__init__(id=4)
        super().__init__(id = 4)

        # กำหนดตำแหน่งของเซลล์แต่ละตัวในแต่ละการหมุนของบล็อก
        self.cells = {
            0: [Position(0, 0), Position(0, 1), Position(1, 0), Position(1, 1)]
        }
        self.move(0, 4) # เลื่อนตำแหน่งเริ่มต้นของบล็อกไปที่แถว 0 และคอลัมน์ 4 ในกริด

class SBlock(Block):

```

```

def __init__(self):
    # เรียก constructor ของ superclass (Block) ด้วย super().__init__(id=5)
    super().__init__(id = 5)

    # กำหนดตำแหน่งของเซลล์แต่ละตัวในแต่ละการหมุนของบล็อก
    self.cells = {
        0: [Position(0, 1), Position(0, 2), Position(1, 0), Position(1, 1)],
        1: [Position(0, 1), Position(1, 1), Position(1, 2), Position(2, 2)],
        2: [Position(1, 1), Position(1, 2), Position(2, 0), Position(2, 1)],
        3: [Position(0, 0), Position(1, 0), Position(1, 1), Position(2, 1)]
    }
    self.move(0, 3) # เลื่อนตำแหน่งเริ่มต้นของบล็อกไปที่แถว 0 และคอลัมน์ 3 ในกริด

class TBlock(Block):
    def __init__(self):
        # เรียก constructor ของ superclass (Block) ด้วย super().__init__(id=6)
        super().__init__(id = 6)

        # กำหนดตำแหน่งของเซลล์แต่ละตัวในแต่ละการหมุนของบล็อก
        self.cells = {
            0: [Position(0, 1), Position(1, 0), Position(1, 1), Position(1, 2)],
            1: [Position(0, 1), Position(1, 1), Position(1, 2), Position(2, 1)],
            2: [Position(1, 0), Position(1, 1), Position(1, 2), Position(2, 1)],
            3: [Position(0, 1), Position(1, 0), Position(1, 1), Position(2, 1)]
        }
        self.move(0, 3) # เลื่อนตำแหน่งเริ่มต้นของบล็อกไปที่แถว 0 และคอลัมน์ 3 ในกริด

class ZBlock(Block):
    def __init__(self):
        # เรียก constructor ของ superclass (Block) ด้วย super().__init__(id=7)
        super().__init__(id = 7)

        # กำหนดตำแหน่งของเซลล์แต่ละตัวในแต่ละการหมุนของบล็อก
        self.cells = {
            0: [Position(0, 0), Position(0, 1), Position(1, 1), Position(1, 2)],
            1: [Position(0, 2), Position(1, 1), Position(1, 2), Position(2, 1)],
            2: [Position(1, 0), Position(1, 1), Position(2, 1), Position(2, 2)],
            3: [Position(0, 1), Position(1, 0), Position(1, 1), Position(2, 0)]
        }
        self.move(0, 3) # เลื่อนตำแหน่งเริ่มต้นของบล็อกไปที่แถว 0 และคอลัมน์ 3 ในกริด

```

8. ผลการรัน

