



เรื่อง Amath

จัดทำโดย

นาย ภูมิพัฒน์ ภิรมย์ราช 67332310144-2 ECP3R

นาย จรัสชัย ตันจำปา 67332310068-8 ECP3R

เสนอ

อาจารย์ ประภาส ผ่องสนาม

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา

การออกแบบระบบอินเทอร์เน็ตของสรรพลิ่ง

ภาคเรียนที่ 2 ปีการศึกษา 2568

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

เชียงใหม่ จังหวัดเชียงใหม่

ที่มาและความสำคัญของโครงการเกม Amath

1. สถานการณ์และปัญหา (Problem Statement)

คณิตศาสตร์เป็นวิชาพื้นฐานที่มีความสำคัญต่อการพัฒนาทักษะการคิดวิเคราะห์และการแก้ปัญหาอย่างไร้กัมมานะ ผู้เรียนจำนวนมากมองว่า คณิตศาสตร์เป็นวิชาที่ซับซ้อนและน่าเบื่อ เกมกระดานต่อสมการคณิตศาสตร์ (A-Math) จึงถูกประดิษฐ์ขึ้นเพื่อช่วยให้การฝึกฝนทักษะการคำนวณ มีความสนุกสนานและท้าทายมากขึ้น แต่ในปัจจุบัน การเล่นเกมกระดาน A-Math แบบเดิม (Physical Board Game) ยังมีข้อจำกัดหลายประการ เช่น ปัญหาเบี้ยสูงหาย การใช้เวลาในการคำนวณคะแนนด้วยตนเองซึ่งอาจเกิดข้อผิดพลาดได้ง่าย (Human Error) รวมถึงข้อจำกัดเรื่องสถานที่และผู้เล่น ที่จำเป็นต้องมีเพื่อนเล่นด้วยจึงสามารถเริ่มเกมได้ ทำให้ขาดความตื่นเต้นในการฝึกฝนทักษะ

2. แรงบันดาลใจในการสร้างสรรค์

จากข้อจำกัดดังกล่าว คณะผู้จัดทำจึงเกิดแรงบันดาลใจที่จะนำเทคโนโลยีคอมพิวเตอร์และทักษะการเขียนโปรแกรมมาประยุกต์ใช้ เพื่อยกระดับเกมกระดาน A-Math ให้อยู่ในรูปแบบดิจิทัล (Simulator) โดยมีความมุ่งมั่นที่จะสร้างเกมที่ผู้เล่นสามารถเข้าถึงได้ง่าย เล่นได้ทุกที่ทุกเวลา รวมถึงต้องการท้าทายความสามารถของตนเองในการเขียนโปรแกรมด้วยภาษา Python และไลบรารี Pygame โดยเฉพาะการพัฒนาระบบปัญญาประดิษฐ์ (AI Bot) ที่สามารถคิดคำนวณและตัดสินใจวางแผนการได้เหมือนผู้เล่นจริง เพื่อให้ผู้เล่นสามารถฝึกฝนทักษะได้แม้ในเวลาที่ไม่มีคู่เล่น

3. วัตถุประสงค์หลัก

- เพื่อออกแบบและพัฒนาเกมกระดานต่อสมการคณิตศาสตร์ (A-Math Simulator) ในรูปแบบเกมคอมพิวเตอร์ 2 มิติ โดยใช้ภาษา Python และ Pygame
- เพื่อพัฒนาระบบผู้เล่นอัตโนมัติ (AI Bot) ที่สามารถค้นหาสมการและวางแผนเบี้ยนกระดานได้อย่างถูกต้องตามกฎติกา สำหรับรองรับการเล่นในโหมดผู้เล่นคนเดียว (Single-Player)
- เพื่อสร้างระบบตรวจสอบความถูกต้องของสมการ (Equation Validation) และระบบคำนวณคะแนนโดยนัยศึกษาอัตโนมัติที่มีความแม่นยำ ป้องกันการทุจริตและข้อผิดพลาดจากการคำนวณของมนุษย์
- เพื่อศึกษาและประยุกต์ใช้อัลกอริทึมในการค้นหาและจัดการโครงสร้างข้อมูล (Data Structure) ภายในเกม

4. ประโยชน์ที่คาดว่าจะได้รับ

ด้านผู้เล่น: ได้รับสื่ออิเล็กทรอนิกส์ที่ช่วยฝึกฝนทักษะการคิดเลขเร็วและการสร้างสมการคณิตศาสตร์ในรูปแบบที่สนุกสนาน เข้าถึงง่าย และสามารถใช้เวลาว่างให้เกิดประโยชน์โดยมี AI เป็นคู่ช่วย

ด้านการใช้งาน: ขัดปัญหาความยุ่งยากในการจัดเตรียมอุปกรณ์เกมกระดานแบบเดิม หมดปัญหาเบี้ยหาย และมีความโปร่งใสในการตัดสินใจแบบ 100%

ด้านผู้พัฒนา: ได้รับประสบการณ์ตรงในการพัฒนาซอฟต์แวร์เกมอย่างเป็นระบบ ตั้งแต่การออกแบบ UI/UX, การเขียนระบบ Logic เชิงคณิตศาสตร์, การจัดการ State ของเกม ไปจนถึงการเพียงบอทอัจฉริยะ และการแก้ไขปัญหา (Debugging) ซึ่งสามารถนำไปต่อยอดในการพัฒนาซอฟต์แวร์ที่ซับซ้อนขึ้นในอนาคตได้

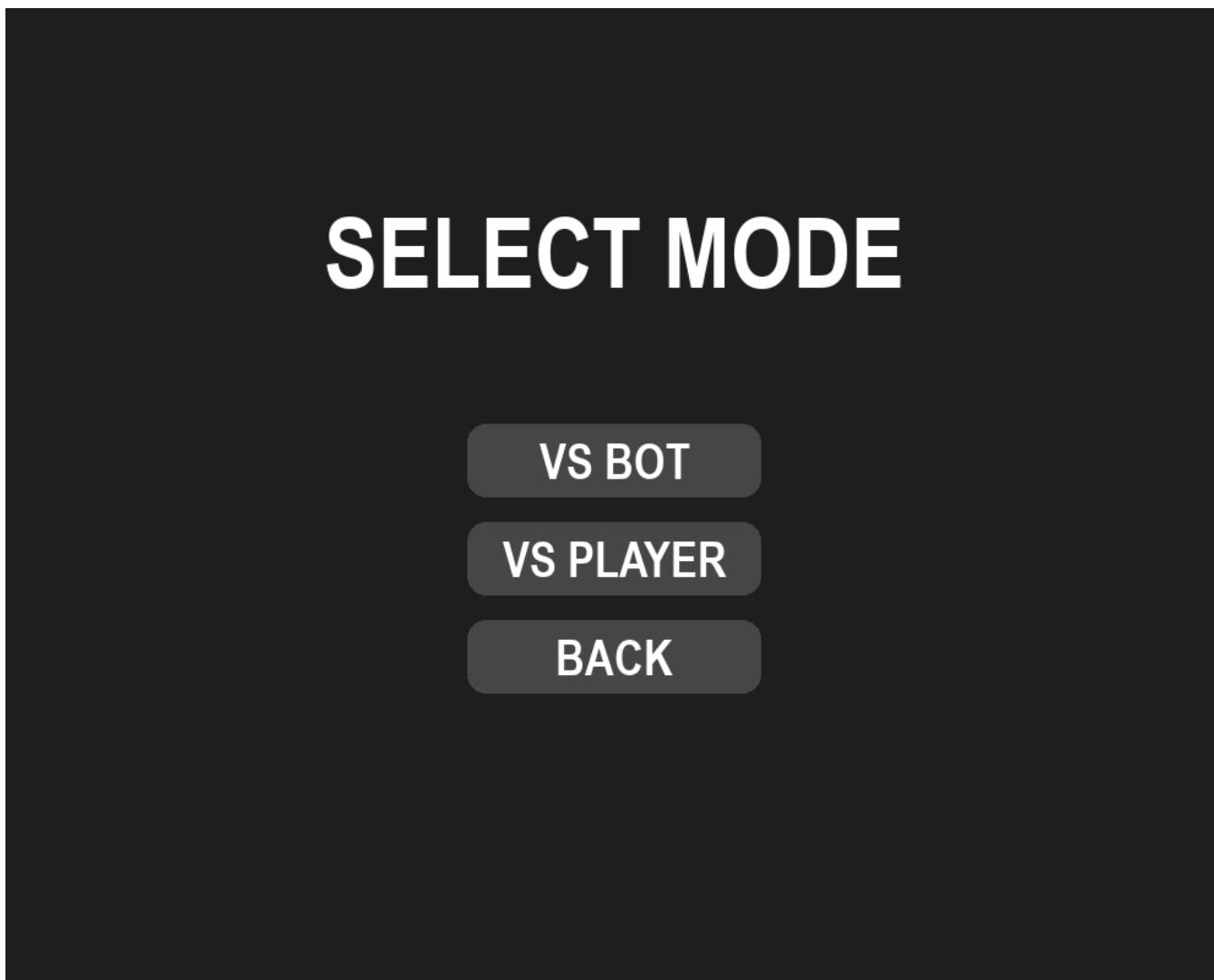
คู่มือการเล่นเกม “Amath”

คู่มือการเล่น A-Math Simulator

A-Math (เอแมธ) คือเกมต่อสมการคณิตศาสตร์ ผู้เล่นจะต้องน้าเบี้ยตัวเลขและเครื่องหมายมาวางแผนเรียงบนกระดานเพื่อสร้างสมการที่เป็นจริง โดยใช้ทักษะการคำนวณและการวางแผนเพื่อทำคะแนนให้ได้มากที่สุด

🎮 1. โหมดการเล่น (Game Modes)

- **Single Player:** เล่นแข่งกับ Bot อัจฉริยะ (สามารถปรับเปลี่ยนเบี้ยและคำนวณหาจุดวางแผนที่ดีที่สุดได้)
- **Multiplayer:** เล่นแข่งกับเพื่อนแบบ 2 คน (ผลัดกันเล่นบนเครื่องเดียวกัน)



✳️ 2. จำนวนเบี้ยในเกม

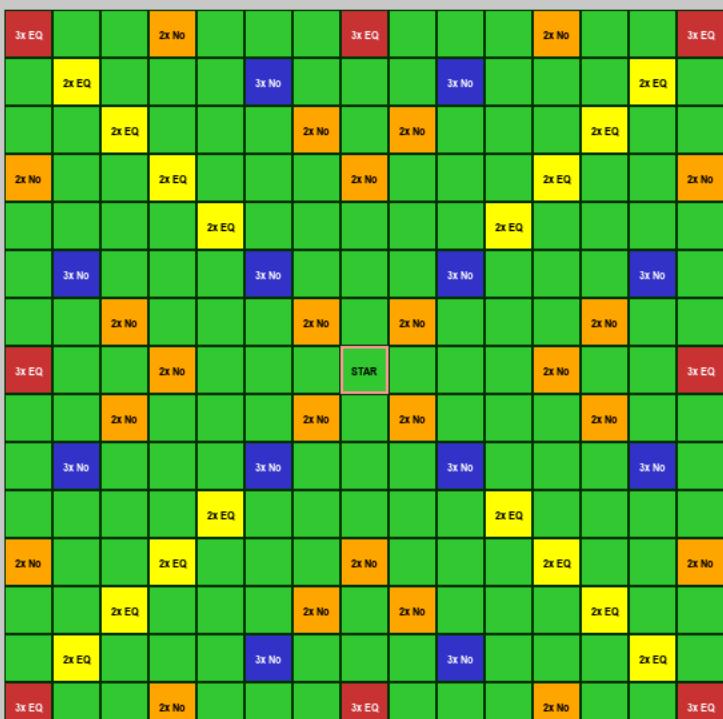
เกมนี้มีเบี้ยในถุงทั้งหมด 80 ตัว (เริ่มเกมจะแจกให้ผู้เล่นคนละ 8 ตัว เหลือในถุง 64 ตัว)

- **ตัวเลข (0-15):** 40 ตัว
- ***เครื่องหมาย (+, -, /):** อย่างละ 5 ตัว (รวม 20 ตัว)
- **เครื่องหมายเท่ากัน (=):** 20 ตัว

💡 **ระบบ Smart Draw (แจกเบี้ยอัจฉริยะ):** ทุกครั้งที่คุณจับเบี้ย ระบบจะพยายามรักษาสมดุลบนมือคุณให้มี ตัวเลขอย่างน้อย 5 ตัว, เครื่องหมายอย่างน้อย 2 ตัว และ "=" อย่างน้อย 1 ตัวเสมอ ทำให้คุณพร้อมสร้างสมการได้ทุกเทิร์น!

Game Restarted!

PLAYER 1
0
Bag: 64



PLAYER 2
0

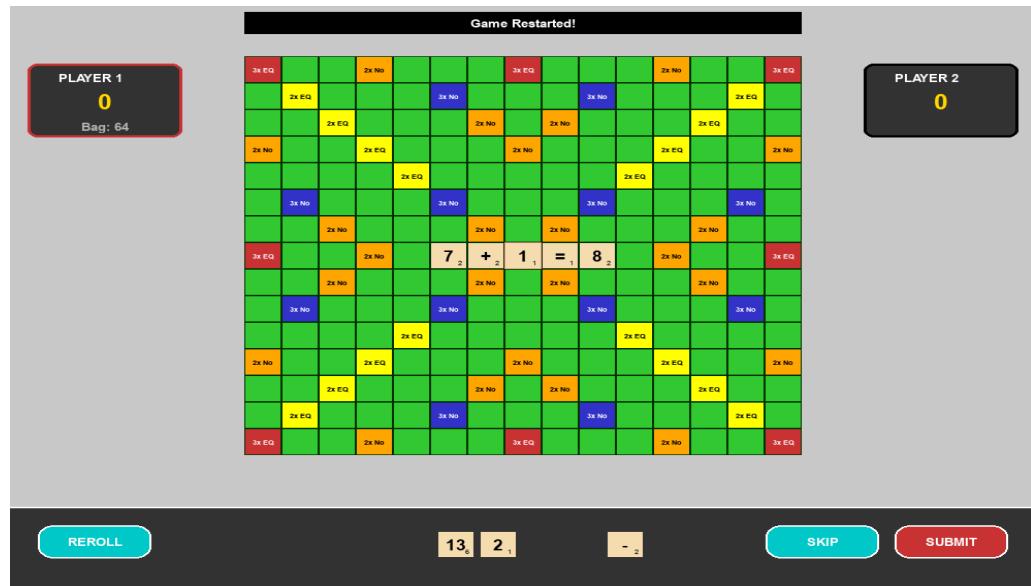
REROLL

8₂ **2₁** **2₁** **13₅** **3₁** **+**₂ **/₂** **=₁**

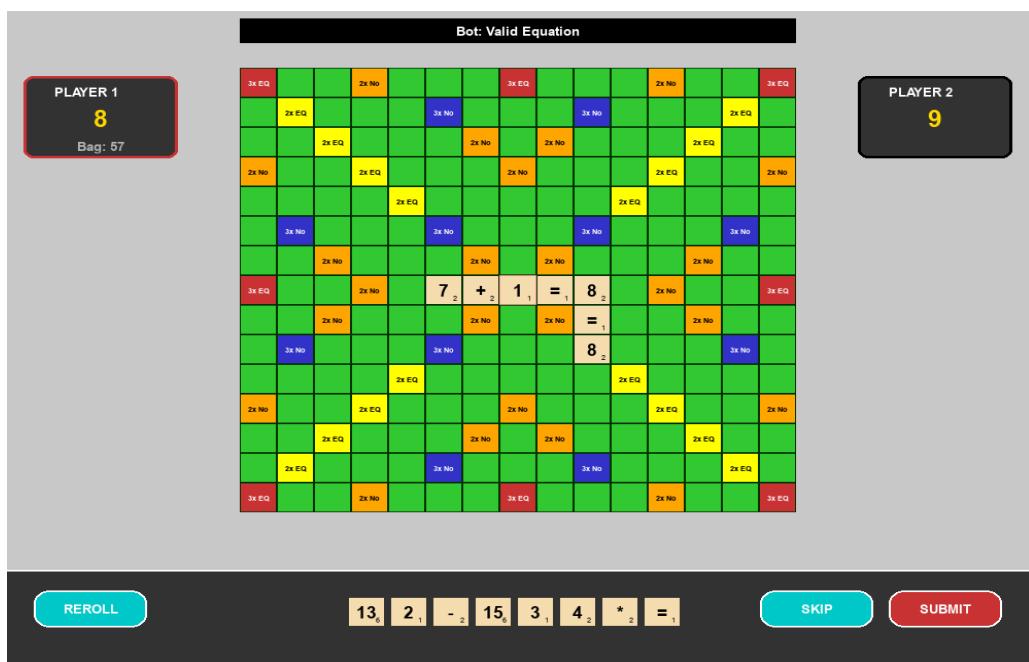
SUBMIT

3. วิธีการเล่นพื้นฐาน

- **ตัวแกร่งของเกม:** ผู้เล่นคนแรกต้องนำเบี้ยมาสร้างสมการ โดยต้องมีเบี้ยอย่างน้อย 1 ตัววางทับบนช่อง **STAR (ดาว)** ตรงกับกล่องกระดาan



- **ตาคัดไป:** ผู้เล่นจะต้องนำเบี้ยในมือ ไปวางต่อเติมจากสมการเดิมที่มีอยู่แล้วบนกระดาan (ต้องมีตัวเชื่อมอย่างน้อย 1 ตัว)



- **ทิศทางสมการ:** การวางแผนการจะต้องอ่านจาก **ซ้ายไปขวา** (**แนวอน**) หรือ **บนลงล่าง** (**แนวตั้ง**)

Bot: Valid Equation

PLAYER 1
18
Bag: 39

PLAYER 2
49

REROLL SKIP SUBMIT

ตั้ง) เท่านั้น

- **ความถูกต้อง:** สมการทุกແຄວທີ່ເກີດຂຶ້ນໃໝ່ມັນກະຕານ ຕັ້ງເປັນຈິງເສມອ (ເຊັ່ນ $5+2=7$ ອີຣ້ $10/2=5$) ທ້າມວາງຕົວເລີຂໂດດາ ໂດຍໄນ້ມີເຄື່ອງໝາຍ =

4. ปุ่มคำสั่งในเกม (Controls)

เมื่อถึงตากองคณ คณสามารถเลือกทำได้ 1 อย่างจาก 3 ตัวเลือกนี้:

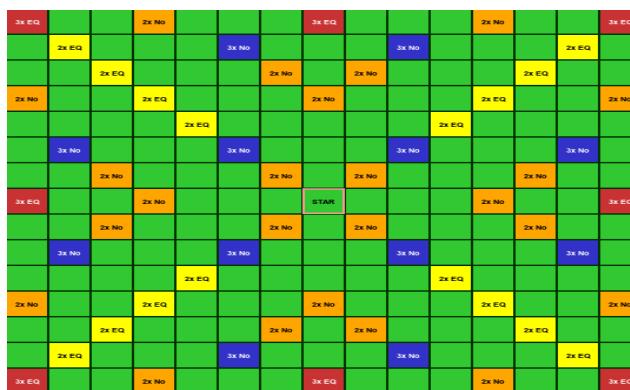
- **SUBMIT** (ยืนยันการวาง): เมื่อลากเบี้ยไปจัดเรียงบนกระดานเสร็จแล้ว ให้กดปุ่มนี้เพื่อจบระบบจะตรวจสอบว่าสมการถูกต้องหรือไม่ หากถูกต้องจะได้คะแนน
 - **REROLL** (เปลี่ยนเบี้ย): หากทางลงไม่ได้ คุณสามารถเอาเบี้ยบันมือทั้งหมดโยนกลับลงถัง แล้วสุ่มขึ้นมาใหม่ 8 ตัวได้ (หมายเหตุ: การกด Reroll จะถือว่าเสียเทิร์นนั้นไปฟรีๆ และนับเป็นการ Skip 1 ครั้ง)
 - **Skip** (ข้ามตา): หากไม่อยากลงเบี้ยและไม่อยากเปลี่ยนเบี้ย สามารถกดผ่านตาให้ผู้ทรงชัยเล่นต่อได้เลย

| | | | | | | | | | | |
|---------------|----------------------|-----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|-------------|---------------|
| REROLL | 9₂ | 12₄ | 1₁ | -₂ | 0₁ | 3₁ | /₂ | =₁ | SKIP | SUBMIT |
|---------------|----------------------|-----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|-------------|---------------|

★ 5. ช่องคะแนนพิเศษบนกระดาน

เมื่อวางแผนเบี้ยทับช่องสีต่างๆ จะได้รับคะแนนโบนัส:

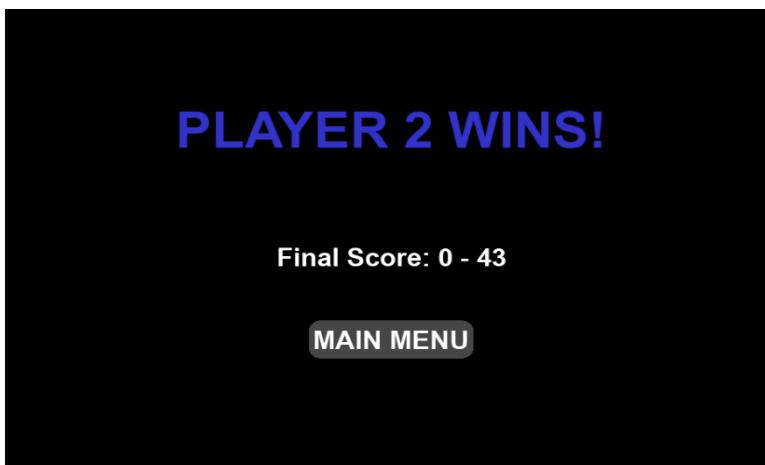
- 3x EQ (สีแดง): นำคะแนนของสมการที่สร้างใหม่ คุณ 3
- 2x EQ (สีเหลือง): นำคะแนนของสมการที่สร้างใหม่ คุณ 2
- 3x No (สีน้ำเงิน): นำคะแนนของเบี้ยตัวที่วางทับ คุณ 3
- 2x No (สีส้ม): นำคะแนนของเบี้ยตัวที่วางทับ คุณ 2



▣ 6. เงื่อนไขการจบเกม (Game Over)

เกมจะจบลงทันทีและสรุปคะแนนผู้ชนะ เมื่อเกิดเหตุการณ์ใดเหตุการณ์หนึ่งต่อไปนี้:

- เบี้ยหมด: เบี้ยในถุงหมด (0 ตัว) และมีผู้เล่นคนใดคนหนึ่งใช้เบี้ยในมือจนหมด
- กระดานเต็ม: ไม่มีช่องว่างเหลือให้วางเบี้ยบนกระดานได้อีก
- ผ่านตาต่อเนื่อง (Deadlock): มีการกดข้ามตา (SKIP) หรือ เปลี่ยนเบี้ย (REROLL) ติดต่อกัน รวมกันครบ 6 ครั้ง (แปลว่าทั้งสองฝ่ายทางลงไม่ได้แล้ว)



7. กฎการสร้างตัวเลขและสมการ (Equation Rules)

เพื่อให้การต่อสมการถูกต้องตามหลักคณิตศาสตร์ ผู้เล่นต้องระวังกฎเหล่านี้:

- การสร้างเลขหลายหลัก (Multi-digit): คุณสามารถนำเบี้ยตัวเลขมาวางติดกันเพื่อสร้างเลขหลักสิบหรือหลักร้อยได้ (เช่น นำเบี้ย 1 มาวางติดกับ 5 จะมีค่าเท่ากับ 15)
- ห้ามศูนย์นำหน้า (No Leading Zeros): ไม่อนุญาตให้ใช้เลข 0 นำหน้าตัวเลขใดๆ (เช่น 05 + 2 = 7 ถือว่า ผิดกติกา ต้องใช้ 5 + 2 = 7 เท่านั้น) แต่ถ้ายังไร์ก็ตาม สามารถใช้เลข 0 โดยเดียวได้ (เช่น 5 + 0 = 5)
- ลำดับการคำนวณ (PEMDAS): เกมนี้อิงตามหลักคณิตศาสตร์สากล โดยระบบจะคำนวณ คูณ (*) และ หาร (/) ก่อน บวก (+) และ ลบ (-) เช่น
 - ตัวอย่างที่ถูก: $2 + 3 * 2 = 8$ (เพราะระบบจะเอา $3 * 2$ ก่อน และค่อยบวก 2)
 - ตัวอย่างที่ผิด: $2 + 3 * 2 = 10$ (วางแผนนี้ระบบจะดึงตัวคูณทันที)

🚫 8. กฎข้อห้ามของเครื่องหมาย (Operator Restrictions)

- ห้ามเครื่องหมายติดกัน: ไม่อนุญาตให้วางเบี้ยเครื่องหมาย 2 ตัวชนกันเด็ดขาด (เช่น + -, * =, / /)
- ห้ามหารด้วยศูนย์ (Divide by Zero): สมการใดๆ ที่มีตัวหารเป็นเลข 0 (เช่น $5 / 0 = 0$) ถือว่า ผิดหลักคณิตศาสตร์และไม่สามารถลงได้
- สมการต้องสมบูรณ์: ทุกແກ້ວທີ່ເກີດຂຶ້ນໃໝ່ມັນกระດານ ໄມ່ວ່າຈະເປັນແນວອນຫຼືແນວຕັ້ງ ຕ້ອງມີ ເຄື່ອງໝາຍ = ເສັມອ (ຈະວາງແຕ່ $5 + 2$ ລອຍໆ ໂດຍໄມ່ມີຜລລັບຮູ່ໄມ່ໄດ້)



9. โบนัสพิเศษ "บิงโก" (BINGO Bonus)

- กฎการดัดแปลง: หากในตาใดก็ตาม ผู้เล่นสามารถใช้ความคิดสร้างสรรค์ นำเบี้ยในมือทั้ง 8 ตัว ลงไปวางบนกระดานได้รวดเดียวหมดในตาเดียว ผู้เล่นคนนั้นจะได้รับ คะแนนโบนัสพิเศษ +40 คะแนน! ทันที (บวกเพิ่มจากคะแนนสมการที่ทำได้ในตานั้น)



10. กฎการเปลี่ยนเบี้ย (Reroll / Swap)

- หากผู้เล่นมองว่าเบี้ยในมือไม่สามารถทำคะแนนได้ หรือไม่มีตัวเลข/เครื่องหมายที่ต้องการ สามารถกดปุ่ม REROLL เพื่อนำเบี้ยในมือทั้งหมดกลับลงกอง และสุ่มหยิบขึ้นมาใหม่ได้
- ข้อควรระวัง: การ Reroll จะถือเป็นการ "ลละลิทีการทำคะแนนในตาหนึ่น" (ได้ 0 คะแนน) และเปลี่ยนให้เป็นตาของอีกฝ่ายทันที (ซึ่งจะถูกนับเป็น 1 ใน 6 ของเงื่อนไขการจบเกมแบบ Deadlock ด้วย)

Data Structure & Algorithm

1. Data Structures ที่ใช้ในโปรเจค

♦ 1. List (Dynamic Array)

ใช้เป็นโครงสร้างหลักของเกม

- player1_hand, player2_hand → เก็บเบี้ยของผู้เล่น
- board_tiles → เก็บเบี้ยที่วางบนกระดาน
- tile_bag → ถุงเบี้ยสำหรับสุ่ม
- resolutions → เก็บค่าความละเอียดหน้าจอ

ลักษณะการใช้งาน:

- append()
- pop()
- remove()
- clear()
- shuffle()

♦ 2. 2D List (Matrix)

```
grid_logic = [[None for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
```

ใช้แทนกระดานเกม (ตาราง 15×15)

การเข้าถึงข้อมูล:

```
grid_logic[row][column]
```

◆ 3. Object (Class)

- คลาส Tile
- เก็บข้อมูล:
 - ค่าเมีย (value)
 - ตำแหน่ง
 - grid position
 - สถานะ (dragging, on_board)

เป็นการใช้แนวคิด Object-Oriented Programming (OOP)

◆ 4. Tuple

ใช้เก็บข้อมูลแบบจับคู่ เช่น

- (row, col)
- (width, height)
- (rect, text)

◆ 5. Boolean

ใช้ควบคุมสถานะ เช่น

- IS_SINGLE_PLAYER
- dropdown_active
- fullscreen_active

◆ 6. Primitive Types

- Integer → คะแนน, turn, index
- Float → volume
- String → game_state, message

2. Algorithms พื้นฐานที่ใช้

◆ 1. Traversal (การวนลูป)

ใช้ for loop ในหลายส่วน เช่น

- วนลูปกระดาน
- วนลูปเบี้ยในมือ
- วนลูปค้นหาเบี้ยในถุง

ลักษณะ: Sequential Traversal

◆ 2. Linear Search

ตัวอย่าง:

```
for i, val in enumerate(tile_bag):
```

ใช้ค้นหาเบี้ยตามประเภท

Time Complexity: $O(n)$

◆ 3. Counting Algorithm

```
sum(1 for v in values if v.isdigit())
```

ใช้ตัวนับ:

- จำนวนตัวเลข
 - จำนวนเครื่องหมาย
 - จำนวนเครื่องหมายเท่ากับ
-

◆ 4. Conditional Decision (If–Else)

ใช้ตัดสินใจเงื่อนไข เช่น

- ตรวจสอบคะแนน
 - ตรวจสอบ Game Over
 - ตรวจสอบความถูกต้องของการวางเบี้ย
-

◆ 5. Finite State Machine (FSM)

เกมทำงานตามสถานะ เช่น:

- MENU

- MODE_SELECT
- SETTINGS
- GAME
- PAUSED
- GAME_OVER

ควบคุมด้วยตัวแปร game_state

◆ **6. Randomization Algorithm**

random.shuffle(tile_bag)

ใช้สุ่มลำดับเบี้ย

◆ **7. Turn Switching Technique**

turn = 3 - turn

สลับผู้เล่นระหว่าง 1 และ 2

◆ **8. Grid Mapping Algorithm**

gx = rel_x // CELL_SIZE

gy = rel_y // CELL_SIZE

แปลงพิกัด Pixel → Grid Position

◆ **9. Validation Algorithm**

validate_move(placed_tiles, grid_logic)

ตรวจสอบความถูกต้องของสมการ

คำนวณคะแนน