



รายงาน

เอกสารประกอบมินิโปรเจค

เสนอ

อาจารย์ประภาส ผ่องสนาม

จัดทำโดย

นายชินชนกนันทร์ พรมศรี 67332110058-4 ECP2N

นายภัทรภณ ตันนารัตน์ 67332110083-8 ECP2N

วิชาโครงสร้างข้อมูล และอัลกอริทึม [31-407-102-202]

คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน วิทยาเขตขอนแก่น
ภาคเรียนที่ 2 ปีการศึกษา 2568

คำนำ

รายงานฉบับนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของการศึกษาในรายวิชา โครงสร้างข้อมูลและอัลกอริทึม (Data Structures & Algorithms) โดยมีวัตถุประสงค์หลักเพื่อนำทฤษฎีและหลักการจัดการข้อมูลที่ได้รับจากการเรียนในห้องเรียน มาประยุกต์ใช้ในการพัฒนาซอฟต์แวร์จริงผ่านการสร้างสรรค์โปรเจกต์เกมแนวแก้ไขปัญหาเชิงตรรกะ (Logic Puzzle Game) ภายใต้ชื่อ Sky Roller ซึ่งเป็นตัวอย่างการนำความรู้เรื่องระบบพิกัดตาราง (Grid System) และการบริหารจัดการสถานะวัตถุ (State Management) มาประมวลผลในรูปแบบสามมิติที่ซับซ้อน

เนื้อหาในรายงานฉบับนี้ ครอบคลุมตั้งแต่แนวคิดการออกแบบ กลไกการทำงานของอัลกอริทึมที่ใช้ควบคุมการเคลื่อนที่ของบล็อก การตรวจสอบเงื่อนไขทางพิสิกส์ (Collision Detection) ตลอดจนการจัดการโครงสร้างข้อมูลแบบ Array เพื่อให้ระบบสามารถตัดสินใจและตอบสนองต่อผู้เล่นได้อย่างมีประสิทธิภาพ

ผู้จัดทำขอขอบพระคุณอาจารย์ประจำวิชาที่ได้ถ่ายทอดความรู้ ให้คำปรึกษา และชี้แนะแนวทางในการนำทฤษฎีจากห้องเรียนมาสร้างสรรค์เป็นผลงานในทางปฏิบัติอย่างเป็นรูปธรรม ผู้จัดทำหวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจศึกษาเกี่ยวกับการพัฒนาเกม และการประยุกต์ใช้โครงสร้างข้อมูลในซอฟต์แวร์ หากรายงานฉบับนี้มีข้อบกพร่องหรือข้อผิดพลาด ประการใด ผู้จัดทำขออนุญาต 수정และขออภัยมา ณ ที่นี้

สารบัญ

เรื่อง	หน้า
คำนำ	ก
สารบัญ	ข
สารบัญ(ต่อ)	ค
บทที่ 1 บทนำ (Introduction)	
1.1 ที่มาและความสำคัญ	1
1.2 วัตถุประสงค์ (Objectives)	1
1.3 ขอบเขตของโครงการ (Project Scope)	1
1.3.1 ขอบเขตด้านเนื้อหา (Functional Scope)	1
1.3.2 ขอบเขตด้านเทคนิค (Technical Scope)	2
1.3.3 ขอบเขตด้านผู้ใช้งาน (User Scope)	2
1.3.4 ขอบเขตด้านอินเทอร์เฟซ (Interface Scope)	2
1.3.5 ขอบเขตข้อจำกัดของโครงการ (Limitations)	2
1.3.6 ระยะเวลาและขอบเขตการพัฒนา	2
บทที่ 2 ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง	
2.1 โปรแกรม Unity Engine และภาษา C#	3
2.2 โครงสร้างข้อมูลแบบอาร์เรย์ (Array Data Structure)	3
2.3 ระบบพิกัดในโลก 3 มิติ (3D Coordinate System)	3
2.4 อัลกอริทึมการหมุนรอบจุดหมุน (Rotation Around Pivot)	4
2.5 การตรวจจับการชนและขอบเขตวัตถุ (Collision Detection)	4
2.6 การจัดการสถานะ (State Management)	4
บทที่ 3 ขั้นตอนการดำเนินงาน (Methodology)	
การออกแบบโครงสร้างฉากและพิกัด (Level & Grid Design)	5
การสร้างตัวละครและตั้งค่าพิสิกส์ (Player Setup)	5
การพัฒนาระบบควบคุมและการกลิ้ง (Movement & Pivot Logic)	5
ระบบการตรวจสอบการตกสนามและจุดเป้าหมาย (Collision Detection)	6
การสร้างส่วนประสานงานผู้ใช้ (Interactive UI & Manager)	6

สารบัญ(ต่อ)

เรื่อง	หน้า
วิธีการเล่นเกม SKY ROLLER (How to Play)	
1. เริ่มต้นเข้าสู่ด่าน (Game Start)	7
2. การควบคุมทิศทาง (Controls)	7
3. สถานะของบล็อก (Block States)	8
4. กฎการแพ้-ชนะ (Winning & Losing Conditions)	8
ความรู้ที่ได้รับจากวิชา Data Structure & Algorithms	
1. ความเข้าใจเรื่องประสิทธิภาพของอัลกอริทึม (Big O Notation)	9
2. การจัดการหน่วยความจำแบบต่อเนื่อง (Contiguous Memory)	9
3. การประยุกต์ใช้พิกัด 2 มิติ (2D Grid Mapping)	9
4. การจัดการสถานะ (State Management & Logic)	9
5. การทำงานแบบ Coroutine และการรอคิวย (Sequence Logic)	9
วิธีการดาวน์โหลดและเข้าใช้งานเกม SKY ROLLER	
1. ขั้นตอนการดาวน์โหลด (Download)	10
2. ขั้นตอนการติดตั้งและเข้าเล่น (Installation & Play)	10

บทที่ 1 บทนำ (Introduction)

1.1 ที่มาและความสำคัญ

ในการศึกษาวิชาที่เกี่ยวข้องกับอัลกอริทึม การทำความเข้าใจเรื่องพิกัดและการเคลื่อนที่ของวัตถุในระบบ Grid มากเป็นเรื่องที่ซับซ้อน "Sky Roller" จึงถูกพัฒนาขึ้นเพื่อเปลี่ยนทฤษฎีการจัดการสถานะ (State Management) ให้เป็นสื่อการเรียนรู้แบบโต้ตอบ (Interactive Learning) ในรูปแบบเกมเชิงปริศนา โดยเน้นฝึกทักษะการวางแผนเชิงตรรกะและการนึกภาพสามมิติบนพื้นที่จำกัด ผู้เล่นจะต้องประมวลผลการพลิกของบล็อกสี่เหลี่ยมเพื่อลุ่มเป้าหมายให้แม่นยำภายใต้เงื่อนไขของเวลา เพื่อเสริมสร้างสมาร์ตและการคิดอย่างเป็นระบบผ่านการลงมือทำจริง

1.2 วัตถุประสงค์ (Objectives)

- เพื่อสร้างสื่อการเรียนรู้แบบโต้ตอบ (Interactive Learning) สำหรับฝึกทักษะการวางแผนและการแก้ปัญหาเชิงตรรกะผ่านระบบพิกัดตาราง
- เพื่อฝึกทักษะการนึกภาพเชิงพื้นที่ (Spatial Reasoning) และรับต้นความสามารถในการลำดับขั้นตอนการเคลื่อนที่อย่างเป็นระบบ (Step-by-Step Logic)
- เพื่อศึกษาการประยุกต์ใช้โครงสร้างข้อมูลแบบ Array และอัลกอริทึมการตรวจสอบเงื่อนไข (Collision Detection) ในการพัฒนาเกม

1.3 ขอบเขตของโครงการ (Project Scope)

1.3.1 ขอบเขตด้านเนื้อหา (Functional Scope)

- ระบบการเล่นหลัก: พัฒนาเกมแนวพัชเซล 3 มิติ ที่ผู้เล่นต้องควบคุมบล็อกสี่เหลี่ยมให้กลิ้งไปบนพื้นที่จำกัด (Grid) เพื่อลุ่มเป้าหมาย
- ระบบตรวจสอบสถานะ รองรับการตรวจสอบสถานะของบล็อก 3 รูปแบบ คือ แนวตั้ง (Standing), แนวนอนแกน X และแนวนอนแกน Z เพื่อใช้ในการคำนวณการเคลื่อนที่และจุดลงจอด
- ระบบเงื่อนไขการแพ้ - ชนะ ตรวจสอบการลงหลุมในแนวตั้ง (Win) และเวลาหมด (Lose)
- ระบบสถิติการเล่น มีการนับจำนวนก้าว (Move Count) และการนับเวลาถอยหลัง (Timer) ในแต่ละด่าน

1.3.2 ขอบเขตด้านเทคนิค (Technical Scope)

- Engine พัฒนาด้วย Unity Game Engine (C# Scripting)
- Data Structure ประยุกต์ใช้ Array ในการจัดเก็บพิกัดพื้นสนาม (Grid Mapping) และใช้ Collider Array ในการตรวจสอบวัตถุฟิสิกส์ได้ตัวละคร
- Algorithm * ใช้ State Management ในการคุ้มการพลิกของวัตถุ 3 มิติ
 - ใช้ Condition Checking ($O(N)$) ในการวนลูปตรวจสอบ Tag ของวัตถุ
 - ใช้ Data Normalization (Snap-to-Grid) เพื่อรักษาความแม่นยำของพิกัดพิกัดตำแหน่ง
- Physics ใช้ระบบ Overlap Sphere ในการตรวจจับพื้นผิวและแรงโน้มถ่วง

1.3.3 ขอบเขตด้านผู้ใช้งาน (User Scope)

- กลุ่มเป้าหมาย นักเรียน หรือบุคคลทั่วไปที่สนใจเกมแนวฟีกทักษะการวางแผนและตระรักษางานพื้นที่
- ระดับความยาก ออกแบบด่านที่มีความซับซ้อนเพิ่มขึ้นตามลำดับ เพื่อทดสอบการคิดวิเคราะห์ของผู้เล่น

1.3.4 ขอบเขตด้านอินเทอร์เฟซ (Interface Scope)

- Main Menu หน้าจอเริ่มต้นที่มีปุ่ม Start เพื่อเข้าสู่เกม และปุ่ม Quit เพื่อออกจากโปรแกรม
- In-game HUD แสดงผลจำนวนก้าว (Moves) และเวลา (Time) แบบ Real-time บนหน้าจอขณะเล่น
- Interactive UI ระบบปุ่มกดที่มี Hover Effect (เปลี่ยนสีเมื่อมาส์ช์) เพื่อตอบสนองต่อผู้ใช้งาน
- Overlay System หน้าจอแจ้งเตือนเมื่อจบเหตุการณ์ เช่น "YOU WIN" หรือ "GAME OVER" เพื่อสื่อสารกับผู้เล่น

1.3.5 ขอบเขตข้อจำกัดของโปรเจกต์ (Limitations)

- การควบคุม รองรับการเล่นผ่านคีย์บอร์ด (Keyboard) เท่านั้น ยังไม่รองรับระบบสัมผัส (Touch Screen) หรือจอยสติ๊ก
- แพลตฟอร์ม ออกแบบมาเพื่อใช้งานบนระบบปฏิบัติการ Windows ในรูปแบบ Desktop Application (.exe)
- ด่านการเล่น ในเวอร์ชันเริ่มต้นนี้จะมีจำนวนด่านที่จำกัดเพื่อใช้ในการสาธิตอัลกอริทึมหลักของเกม
- การบันทึกข้อมูล ไม่มีระบบ Save/Load ข้อมูลผู้เล่น หากปิดโปรแกรมจะต้องเริ่มเล่นใหม่ตั้งแต่ด่านแรก

1.3.6 ระยะเวลาและขอบเขตการพัฒนา

- ระยะเวลาดำเนินงาน ประมาณ 1 อาทิตย์ ของภาคเรียนที่ 2 ปีการศึกษา 2568
- ขอบเขตการพัฒนา ครอบคลุมตั้งแต่การออกแบบ Logic การเขียนโค้ดควบคุม (Scripting), การจัดทำฉาก (Level Design), การทดสอบระบบ (Testing) จนถึงการจัดทำคู่มือรายงานฉบับนี้

บทที่ 2 ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง

ในการพัฒนาเกม SKY ROLLER ผู้จัดทำได้ศึกษาทฤษฎีและเทคโนโลยีต่าง ๆ ที่นำมาประยุกต์ใช้ในการดำเนินงาน ดังนี้

2.1 โปรแกรม Unity Engine และภาษา C#

Unity เป็นโปรแกรมสำหรับการพัฒนาเกม (Game Engine) แบบข้ามแพลตฟอร์มที่ได้รับความนิยมสูง รองรับการทำงานในรูปแบบ Component-based ซึ่งช่วยให้ผู้พัฒนาสามารถจัดการวัตถุในเกม (GameObject) แยกจากกันได้ชัดเจน โดยใช้ภาษา C# (C-Sharp) เป็นภาษาหลักในการเขียนสคริปต์ควบคุมการทำงาน (Logic) ของวัตถุ

2.2 โครงสร้างข้อมูลแบบอาร์เรย์ (Array Data Structure)

อาร์เรย์ (Array) คือโครงสร้างข้อมูลที่เป็นพื้นฐานที่สุดในการจัดเก็บข้อมูลชุดที่มีขึ้นเดียวกันไว้ต่อเนื่องกัน ในหน่วยความจำ

- การประยุกต์ใช้ ในโปรเจกต์นี้มีการใช้ Array เพื่อจัดเก็บพิกัดของพื้นสนาม (Grid Mapping) โดยใช้พิกัด แ雷และคอลัมน์เทียบเคียงกับดัชนี (Index) ของ Array
- ประสิทธิภาพ (Complexity) การใช้ Array ช่วยให้การเข้าถึงข้อมูลพิกัด (Access) มีประสิทธิภาพสูงใน ระดับ O(1) หรือ Constant Time เนื่องจากสามารถระบุตำแหน่งดัชนีที่ต้องการเข้าถึงได้ทันที

2.3 ระบบพิกัดในโลก 3 มิติ (3D Coordinate System)

การทำงานในสภาพแวดล้อม 3 มิติ จะอ้างอิงตามระบบพิกัดคาร์ทีเซียน (Cartesian Coordinate System) ซึ่งประกอบด้วย 3 แกนหลัก

- แกน X (Horizontal) แนวระนาบซ้าย-ขวา
- แกน Y (Vertical) แนวตั้ง (ใช้ตรวจสอบความสูงและการร่วงหล่น)
- แกน Z (Depth) แนวลึก หน้า-หลัง

2.4 อัลกอริทึมการหมุนรอบจุดหมุน (Rotation Around Pivot)

ในการพลิกตัวของวัตถุทรงสี่เหลี่ยม การหมุนรอบจุดศูนย์กลาง (Center) จะทำให้วัตถุเคลื่อนที่ไปตามทิศทางที่กำหนด แต่ถ้าเราต้องใช้ทฤษฎีการหมุนรอบ จุดหมุน (Pivot) ที่อยู่บริเวณขอบของวัตถุในทิศทางที่จะเคลื่อนที่ไป เพื่อให้บล็อกพลิกตัวได้อย่างแนบสนิทกับพื้นสนาม

2.5 การตรวจจับการชนและขอบเขตวัตถุ (Collision Detection)

ระบบฟิสิกส์ใน Unity ใช้สิ่งที่เรียกว่า **Collider** ในการระบุขอบเขตของวัตถุ

- **Physics.OverlapSphere** เป็นฟังก์ชันที่สร้างขอบเขตทรงกลมจำลองที่ตำแหน่งที่ระบุ เพื่อตรวจสอบว่า มีวัตถุใดอยู่ในบริเวณนั้นบ้าง ข้อมูลที่ตรวจพบจะถูกส่งออกมาในรูปแบบของ **Array** เพื่อนำไปวนลูปเช็คสถานะการเล่นต่อไป

2.6 การจัดการสถานะ (State Management)

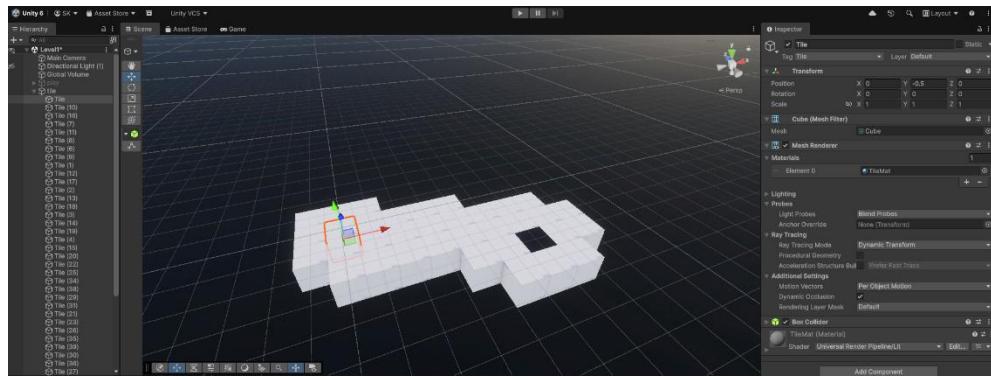
สถานะของบล็อกในเกมมีความสำคัญต่อการคำนวณระยะการเคลื่อนที่ โดยแบ่งออกเป็น:

- **Standing State:** บล็อกตั้งตรง (กินพื้นที่ 1x1 ช่อง)
- **Horizontal State:** บล็อกนอนราบ (กินพื้นที่ 2x1 หรือ 1x2 ช่อง) การจัดการสถานะนี้ช่วยให้ระบบรู้ว่า ในก้าวถัดไปต้องเช็คพื้นสนามที่ตำแหน่งพิกัดใดบ้างใน Array

บทที่ 3 ขั้นตอนการดำเนินงาน (Methodology)

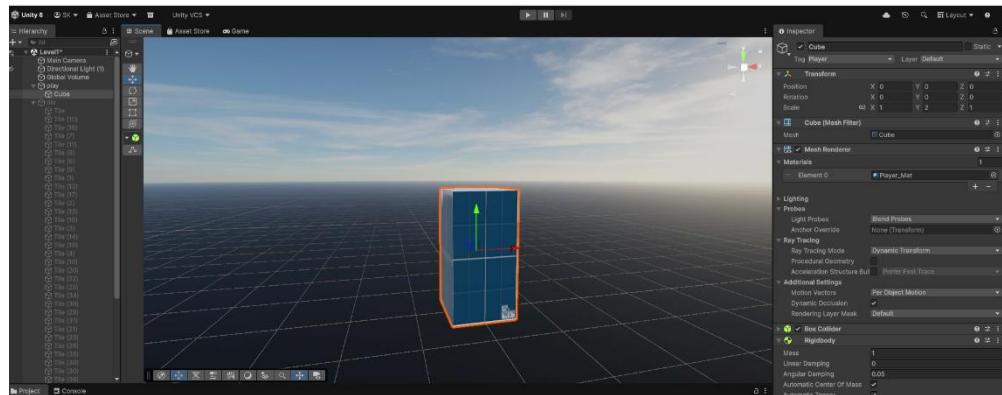
การออกแบบโครงสร้างจากและพิกัด (Level & Grid Design)

- คำอธิบาย เริ่มจากการสร้างพื้นสนามโดยใช้ระบบตาราง (Grid) โดยวางแผ่นพื้น (Tiles) ขนาด 1×1 ในตำแหน่งที่สอดคล้องกับพิกัด X และ Z เพื่อให้ง่ายต่อการเก็บข้อมูลในรูปแบบ Array



การสร้างตัวละครและตั้งค่าฟิสิกส์ (Player Setup)

- คำอธิบาย สร้างบล็อก 3 มิติ (Cube) และปรับสเกลให้เป็นทรงสี่เหลี่ยมผืนผ้า ใส่คอมโพเนนต์ Rigidbody สำหรับฟิสิกส์ และกำหนด Tag เป็น "Player" เพื่อใช้ในการตรวจสอบการชน



การพัฒนาระบบควบคุมและการกลิ้ง (Movement & Pivot Logic)

- คำอธิบาย เขียนสคริปต์ควบคุมการหมุนรอบจุดหมุน (Pivot) 90 องศา โดยคำนวณจากสถานะของบล็อก (ตั้งหรือนอน) เพื่อให้การเคลื่อนที่ลงล็อกกับตารางพื้นสนามพอดี

```
// ส่วนที่ 4: พัฒนาการเคลื่อนที่ (Move Algorithm)
IEnumerator Move(Vector3 dir)
{
    isMoving = true;
    moveCount++;
    lastMoveDir = dir;

    // เช็คเสียง
    if (playerAudio != null && moveSFX != null) {
        playerAudio.PlayOneShot(moveSFX);
    }

    // สำหรับการหมุน และ ระดมความเร็วขณะลงตื้น (ล็อกหรือนอน)
    bool wasStanding = transform.position.y > 0.8f;
    float distToEdge = wasStanding ? 0.5f : Mathf.Abs(Vector3.Dot(dir, transform.up)) > 0.1f ? 1.0f : 0.5f;
    Vector3 anchor = transform.position + (dir * distToEdge);
    anchor.y = 0f;

    // คำนวณแนวตั้ง (Axis)
    Vector3 axis = Vector3.Cross(transform.up, dir);
    float totalAngle = 0;

    // ล็อกเมื่อยกตื้นเมื่อต้อง 90 องศา เพื่อให้สามารถเคลื่อนที่ลง
    while (totalAngle < 90)
    {
        float angle = rollSpeed * Time.deltaTime;
        if (totalAngle + angle > 90) angle = 90 - totalAngle;
        transform.RotateAround(anchor, axis, angle);
        totalAngle += angle;
        yield return null;
    }
}
```

ระบบการตรวจสอบการตกสนามและจุดเป้าหมาย (Collision Detection)

- คำอธิบาย ใช้ฟังก์ชัน OverlapSphere สร้างทรงกลมจำลองเพื่อสแกนวัตถุใต้เท้าบล็อก และเก็บข้อมูลลงใน Array เพื่อเช็ค Tag ว่าเป็นพื้นสนาม (Tile) หรือหدหมาย (Goal)

```
// กรณีที่ ๑: ที่อยู่บนพื้นสนาม
bool CheckIfFalling() {
    bool isStanding = Mathf.Abs(transform.up.y) > 0.8f;
    Vector3 pos = transform.position;

    // กรณีที่ ๑: น้ำตกลง
    if (isStanding) {
        Vector3 checkPos = pos + Vector3.down * 0.9f;
        // สำหรับ检测พื้นที่ดินที่อยู่ใต้เท้า (Tile) หรือหดหมาย (goal)
        Collider[] hitColliders = Physics.OverlapSphere(checkPos, 0.3f);
        foreach (var col in hitColliders) {
            if (col.CompareTag("Tile") || col.CompareTag("Goal")) {
                // ถ้ามีหดหมายอยู่ใต้เท้า -> หยุดลง
                if (col.CompareTag("Goal")) { StartCoroutine(WinSequence(col.transform.position)); return true; }
                return false; // เอฟเฟกต์ None
            }
        }
        // ถ้าไม่พบอะไรเลย -> เช็คการเมืองของสนาม
        StartCoroutine(FallUntil(pos + (lastMoveDir * 0.5f), lastMoveDir)); return true;
    }
    // กรณีที่ ๒: น้ำตกลง
    else {
        Vector3 offset = (Mathf.Abs(transform.up.x) > 0.1f) ? Vector3.right * 0.5f : Vector3.forward * 0.5f;
        // น้ำตกไป ๒ ด้าน (ด้านซ้ายและขวา)
        bool g1 = CheckGroundAtPoint(pos + offset);
        bool g2 = CheckGroundAtPoint(pos - offset);

        if ((g1 && g2) { // ถ้ามีพื้นที่สองด้าน -> ผ่านลง
            StartCoroutine(FallUntil(pos + (lastMoveDir * 0.5f), lastMoveDir)); return true;
        }
        else if (!g1 || !g2) { // ถ้ามีพื้นที่ด้านเดียว -> เมืองลง
            Vector3 safeSide = g1 ? (pos + offset) : (pos - offset);
            StartCoroutine(FallUntil(safeSide + (lastMoveDir * 0.5f), !g1 ? offset : -offset)); return true;
        }
        return false;
    }
}

// กรณีที่ ๓: ใช้จังหวะเมื่อพื้นดินที่มีพื้นที่น้ำ
bool CheckGroundAtPoint(Vector3 point) {
    Collider[] cols = Physics.OverlapSphere(point + Vector3.down * 0.4f, 0.2f);
    foreach (var c in cols) { if (c.CompareTag("Tile") || c.CompareTag("Goal")) return true; }
    return false;
}
```

การสร้างส่วนประสานงานผู้ใช้ (Interactive UI & Manager)

- คำอธิบาย: ออกแบบหน้าจอแสดงผล (Canvas) เพื่อบอกเลขด่าน จำนวนก้าว และเวลาที่เหลือ รวมถึงหน้าจอแจ้งเตือนเมื่อจบด่าน (Win/Lose) โดยมี GameManager เป็นตัวควบคุมสถานะ

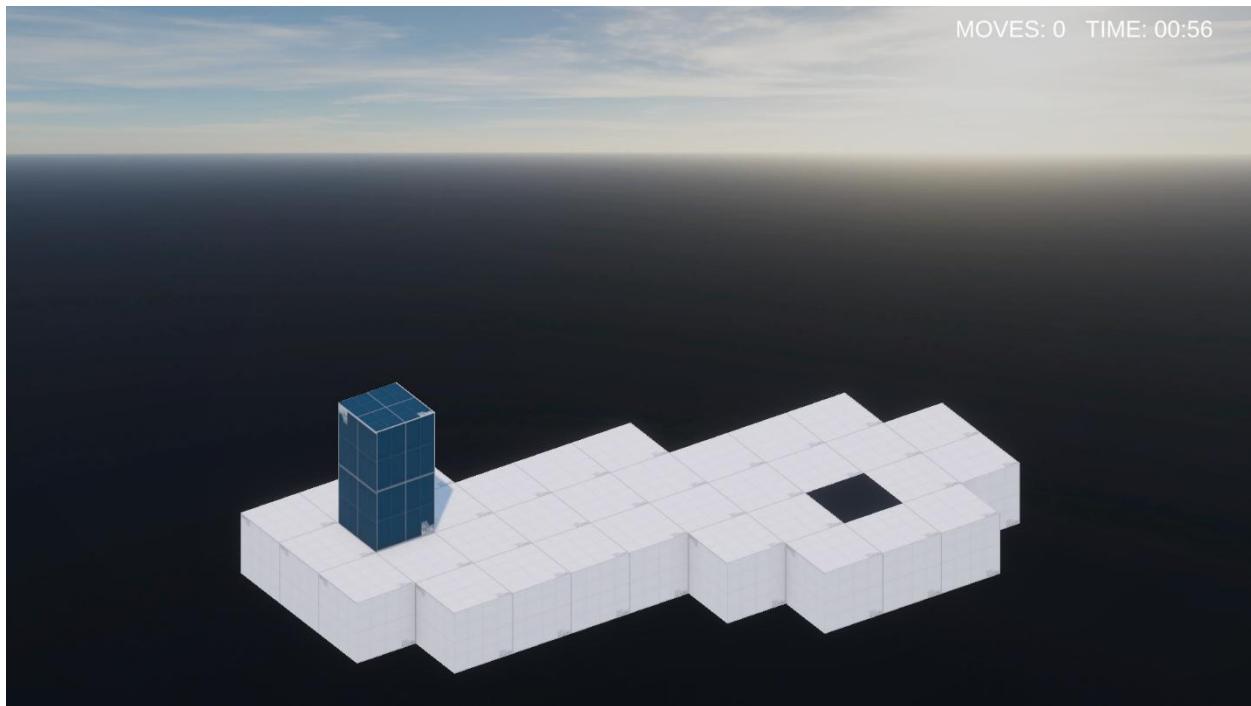


วิธีการเล่นเกม SKY ROLLER (How to Play)

ในโปรเจกต์นี้ ผู้เล่นจะต้องใช้ทักษะการวางแผนเชิงพื้นที่เพื่อควบคุมบล็อก 3D โดยมีขั้นตอนการเล่นดังนี้

1. เริ่มต้นเข้าสู่ด่าน (Game Start)

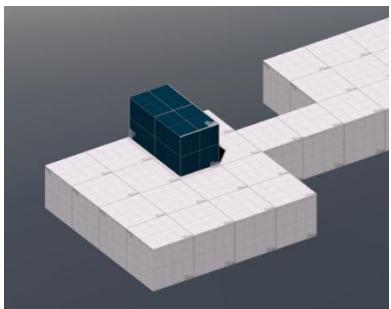
- ทันทีที่กดเริ่มเกม บล็อกจะร่วงลงมาจากท้องฟ้าเพื่อจอดบนจุดเริ่มต้น (Landing)
- ข้อควรรู้ ระบบ Timer จะเริ่มนับถอยหลังทันทีที่บล็อกสัมผัสพื้น ผู้เล่นจึงต้องรีบสังเกตตำแหน่งของ "หลุมเป้าหมาย" ให้ไวที่สุด



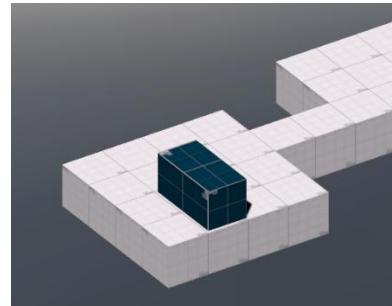
2. การควบคุมทิศทาง (Controls)

ใช้ปุ่มลูกศร 4 ทิศทางบนคีย์บอร์ดในการเล่น

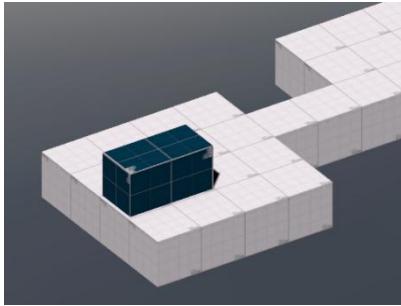
- ลูกศรขึ้น (Up) กลิ้งบล็อกไปข้างหน้า
- ลูกศรลง (Down) กลิ้งบล็อกถอยหลัง
- ลูกศรซ้าย (Left) กลิ้งบล็อกไปทางซ้าย
- ลูกศรขวา (Right) กลิ้งบล็อกไปทางขวา



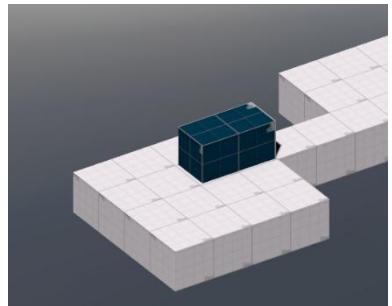
ลูกศรขึ้น (Up)



ลูกศรลง (Down)



ลูกศรซ้าย (Left)



ลูกศรขวา (Right)

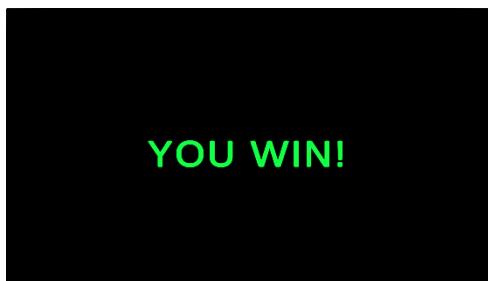
3. สถานะของบล็อก (Block States)

บล็อกจะมีการเปลี่ยนรูปร่างตามจังหวะการกลิ้ง ซึ่งส่งผลต่อการเช็คพิกัดบนตาราง

- แนวตั้ง (Standing) บล็อกกินพื้นที่เพียง 1 ช่อง (เป็นท่าเดียวที่ใช้ลงหลุมเพื่อชนะได้)
- แนวนอนแกน X (Horizontal-X) บล็อกกินพื้นที่ 2 ช่องในแนวซ้าย-ขวา
- แนวนอนแกน Z (Horizontal-Z) บล็อกกินพื้นที่ 2 ช่องในแนวหน้า-หลัง

4. กฎการแพ้-ชนะ (Winning & Losing Conditions)

- เงื่อนไขการชนะ (Win) ผู้เล่นต้องกลิ้งบล็อกให้ไปหยุดอยู่เหนือหลุมเป้าหมายใน "แนวตั้ง" เท่านั้น บล็อกจะจะตกลงไปในหลุมและผ่านเข้าสู่ด้านลับด้านลับ และเมื่อผ่านด่านทั้งหมด ระบบจะตัดเข้าหน้าจอ “YOU WIN!” ทันที
- เงื่อนไขการแพ้ (Lose) เวลาหมด (Time Out) หากเข้มนาฬิกาบนหน้าจอ (Timer) กลับเป็น 0 ก่อนที่บล็อกจะลงหลุม ระบบจะตัดเข้าหน้าจอ “YOU LOSE” ทันที



เงื่อนไขการชนะ



เงื่อนไขการแพ้

ความรู้ที่ได้รับจากวิชา Data Structure & Algorithms

1. ความเข้าใจเรื่องประสิทธิภาพของอัลกอริทึม (Big O Notation)

- การใช้ Array เมื่อเราต้องการเช็คพื้นสนาม การใช้ Array ช่วยให้เข้าถึงข้อมูล (Access) ได้ในระดับ $O(1)$ เพราะเราต้องหา Index ที่ต้องการ ทำให้เกมไม่กระตุกแม้ด่านจะใหญ่ขึ้น
- Linear Search ในส่วนของ foreach ที่ใช้เช็คข้อมูลว่าติดใน hitColliders (Array ของ Collider) เป็นการเรียนรู้เรื่องการค้นหาข้อมูลแบบเส้นตรง ซึ่งหมายความว่าต้องส่องทุกๆ ข้อมูลขนาดเล็ก

2. การจัดการหน่วยความจำแบบต่อเนื่อง (Contiguous Memory)

ได้เห็นภาพจริงว่า Array มีการจัดพื้นที่ในหน่วยความจำแบบเรียงต่อกัน ซึ่งแตกต่างจาก Linked List

- ความรู้ที่ได้ในเกม SKY ROLLER พื้นสนามถูกวางเป็นตารางคงที่ การใช้ Array จึงมีประสิทธิภาพสูงสุดในการประหยัด RAM และทำให้ CPU ดึงข้อมูลไปคำนวณพิสิกส์ได้รวดเร็ว

3. การประยุกต์ใช้พิกัด 2 มิติ (2D Grid Mapping)

วิชา Data Structure สอนเรื่อง Array 2 มิติ ซึ่งในโปรเจกต์นี้พัฒนามาแปลงเป็นโลก 3 มิติ

- Logic ที่ได้ การจับคู่ตำแหน่งของวัตถุใน Unity เข้ากับ Index ของ Array โดยการใช้ฟังก์ชัน SnapToGrid (การปัดเศษทศนิยม) เพื่อให้ข้อมูลในคอมพิวเตอร์กับสิ่งที่ผู้เล่นเห็นตรงกัน 100%

4. การจัดการสถานะ (State Management & Logic)

- ความรู้ที่ได้ การเขียนฟังก์ชัน CheckIfFalling คือการสร้าง Algorithm เพื่อตัดสินใจผลลัพธ์ของเกม โดยแบ่งกรณี (Case Study) ตามสถานะของบล็อก (ตั้ง/นอน) ซึ่งช่วยฝึกทักษะการคิดแบบเป็นระบบ (Systematic Thinking)

5. การทำงานแบบ Coroutine และการรอคอย (Sequence Logic)

- ความรู้ที่ได้ การใช้ IEnumerator ทำให้เข้าใจว่าบางอัลกอริทึมไม่สามารถทำให้จบได้ในเฟรมเดียว (เช่น การค่อยๆ พลิกบล็อก) แต่ต้องมีการแบ่งขั้นตอนการทำงานออกเป็นส่วนย่อย ๆ เพื่อให้เกิดความต่อเนื่อง

วิธีการดาวน์โหลดและเข้าใช้งานเกม SKY ROLLER

ผู้ใช้งานสามารถเข้าถึงและติดตั้งเกมได้ตามขั้นตอนดังนี้



QR Code Download Game

1. ขั้นตอนการดาวน์โหลด (Download)

- เข้าสู่ลิงก์โปรเจกต์ ไปที่หน้า GitHub Repository ของโครงการ (ตามภาพที่ปรากฏในรายงาน)
- การดาวน์โหลดไฟล์
 - กดที่ปุ่มสีเขียวที่เขียนว่า "<> Code"
 - เลือกเมนู "Download ZIP" เพื่อทำการดาวน์โหลดไฟล์ทั้งหมดลงเครื่องคอมพิวเตอร์

2. ขั้นตอนการติดตั้งและเข้าเล่น (Installation & Play)

- การเตรียมไฟล์ เมื่อดาวน์โหลดเสร็จสิ้น ให้ทำการแตกไฟล์ (Extract ZIP) ออกมายเป็นไฟล์เดอร์ปกติ
- การเข้าเกม
 - เข้าไปในไฟล์เดอร์โครงการที่แตกไฟล์แล้ว
 - มองหาไฟล์ที่ชื่อว่า My project 3D.exe (หรือไฟล์ที่มีนามสกุล .exe ตามที่ปรากฏในหน้า GitHub)
 - ดับเบิลคลิกที่ไฟล์ดังกล่าวเพื่อเริ่มรันเกมทันที โดยไม่ต้องติดตั้งโปรแกรมอื่นเพิ่มเติม