



เกมเอาชีวิตรอดหน้าท่วม

โดย

นาย พชร ตรียัง

นาย ชานนท์ สนนท์

โครงการพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิทยาศาสตร์บัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์

ปีการศึกษา 2567

ลิขสิทธิ์ของมหาวิทยาลัยธรรมศาสตร์

เกมเอาชีวิตรอดน้ำท่วม

โดย

นาย พชร ตีรัมย์

นาย ชานนท์ สนนท์

โครงการพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิทยาศาสตร์บัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์

ปีการศึกษา 2567

ลิขสิทธิ์ของมหาวิทยาลัยธรรมศาสตร์

The floor is flooding

BY

Mr. Potchara Treeyung

Mr. Chanon Sahanon

**A FINAL-YEAR PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE**

COMPUTER SCIENCE

FACULTY OF SCIENCE AND TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2024

COPYRIGHT OF THAMMASAT UNIVERSITY

มหาวิทยาลัยธรรมศาสตร์
คณะวิทยาศาสตร์และเทคโนโลยี

รายงานโครงการพิเศษ

ของ

นาย พชร ตีรียัง
นาย ชานนท์ สอนนท์

เรื่อง
เกมเอาชีวิตรอดน้ำท่วม

ได้รับการตรวจสอบและอนุมัติ ให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
เมื่อ วันที่ 30 พฤษภาคม 2568

อาจารย์ที่ปรึกษา



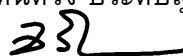
(ผศ.ดร.อรจิรา สิทธิศักดิ์)

กรรมการสอบโครงการพิเศษ



(ผศ.ดร.เด่นดวง ประดับสุวรรณ)

กรรมการสอบโครงการพิเศษ



(อ.สิริกัญญา นิลพานิช)

มหาวิทยาลัยธรรมศาสตร์
คณะวิทยาศาสตร์และเทคโนโลยี

รายงานโครงการพิเศษ

ของ


นาย พชร ตริยง
นายชานนท์ สหนนท์

เรื่อง

เกมเอาชีวิตรอดน้ำท่วม

ได้รับการตรวจสอบและอนุมัติ ให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
เมื่อ วันที่ 30 พฤษภาคม 2568

อาจารย์ที่ปรึกษา



(ผศ.ดร.อรจิรา สิทธิศักดิ์)

กรรมการสอบโครงการพิเศษ



(ผศ.ดร.เต๋นดวง ประดับสุวรรณ)

กรรมการสอบโครงการพิเศษ



(อ.สิริกัญญา นิลพานิช)

หัวข้อโครงการพิเศษ	เกมเอาชีวิตรอดน้ำท่วม
ชื่อผู้เขียน	นาย พชร ตรียัง
ชื่อผู้เขียน	นาย ชานนท์ สหนนท์
ชื่อปริญญา	วิทยาศาสตร์บัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
สาขาวิชา/คณะ/มหาวิทยาลัย	สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์
อาจารย์ที่ปรึกษาโครงการพิเศษ	ผศ.ดร.อรจิรา สิทธิศักดิ์
ปีการศึกษา	2567

บทคัดย่อ

ในปัจจุบันเกมเป็นสื่อความบันเทิงที่ได้รับความนิยมอย่างแพร่หลาย และนำมาประยุกต์ใช้เป็นสื่อการเรียนรู้ที่นอกจากจะสร้างความสนุกสนานบันเทิงแล้วนั้น ยังให้ความรู้ควบคู่ไปด้วย โดยเกมเอาชีวิตรอดน้ำท่วมนั้นมีจุดประสงค์เพื่อสร้างเกมที่แสดงความแตกต่างการทำงานระหว่าง Dijkstra กับ A* Algorithms เพื่อให้ผู้เล่นได้สนุกสนานไปกับตัวเกมและได้เห็นความแตกต่างการทำงานของ Algorithm ทั้ง 2 ตัวที่กล่าวไปข้างต้น ตัวเกมจะพัฒนาผ่านโปรแกรม Godot Engine ที่เป็นโปรแกรมที่นิยมในการทำเกมทั้ง 2D และ 3D

ตัวเกมจะเป็นการเล่นในรูปแบบ Tower Defense เกม ที่ผู้เล่นจะมีจุดประสงค์ในการป้องกันฐานของตัวเอง ด้วยการวางบ้อมปราการตามจุดต่างๆ เพื่อป้องกันศัตรูที่จะเคลื่อนที่มาโจมตีฐาน ศัตรูจะมี 2 ประเภทคือ ศัตรูที่เคลื่อนที่โดยใช้ Dijkstra กับ ศัตรูที่ใช้ A* ในการเคลื่อนที่ ผู้เล่นจะมีทรัพยากรที่ใช้ในการวางบ้อมปราการอย่างจำกัด ผู้เล่นจะต้องวางแผนการใช้ทรัพยากรให้เหมาะสม และตัวเกมจะมีการสุ่มเกิดน้ำท่วมตามจุดต่างๆที่จะทำให้ศัตรูเปลี่ยนทิศทางการเคลื่อนที่ เพื่อให้ผู้เล่นต้องวางแผนรับมือกับเหตุการณ์ที่เกิดขึ้นอย่างฉับพลัน

คำสำคัญ: Dijkstra, A*, Godot Engine, Tower Defense

Thesis Title	The floor is flooding
Author	Mr. Potchara Treeyung
Author	Mr. Chanon Sahanon
Degree	Bachelor of Science
Major Field/Faculty/University	Computer Science Faculty of Science and Technology Thammasat University
Project Advisor	ASST. PROF. DR. ONJIRA SITTHISAK
Academic Years	2024

Abstract

Nowadays, games are a widely popular form of entertainment and have been adapted as an effective learning medium. Not only do they provide enjoyment and fun, but they also deliver knowledge in an engaging way. The Floor is Flooding is designed with the purpose of illustrating the differences in performance between Dijkstra's Algorithm and A. This allows players to enjoy the game while observing the distinct behaviors of these two algorithms in action. The game will be developed using the Godot Engine, a widely used tool for creating both 2D and 3D games.

The game will follow a Tower Defense format, where players aim to defend their base by strategically placing turrets at various points to fend off enemies moving toward the base. There will be two types of enemies: those that navigate using Dijkstra's Algorithm and those that use A*. Players will have a limited number of resources to place their turrets, requiring careful planning and resource management. Additionally, the game will feature randomly occurring flood events at different locations. These floods will alter enemy movement paths, forcing players to adapt their strategies and respond to sudden changes in the environment.

Keyword: Dijkstra, A*, Godot Engine, Tower Defense

กิตติกรรมประกาศ

การจัดทำโครงการฉบับนี้ที่สำเร็จลุล่วงไปได้ด้วยดี ต้องขอขอบพระคุณ ผศ.ดร.อรจิรา สิริศักดิ์ ผู้ที่เป็นอาจารย์ที่ปรึกษาโครงการเกมเอาชีวิตรอดน้ำท่วม ที่ได้สละเวลาอันมีค่าให้คำแนะนำ ชี้แนะแนวทางในการทำวิจัยและให้ความช่วยเหลือได้อย่างดียิ่งในการให้คำปรึกษาในการทำงานให้สำเร็จลุล่วงไปได้ด้วยดี ขอกราบขอพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

ในการทำโครงการครั้งนี้ ขอขอบคุณเพื่อนร่วมงานที่ได้สละเวลามาค้นคว้าหาข้อมูลเพื่อนำมาประกอบโครงการร่วมกัน ขอขอบคุณเพื่อนร่วมงานที่ได้อยู่ด้วยกันจนถึงวันทำงานสำเร็จลุล่วงไปได้ด้วยดี

ในการทำโครงการครั้งนี้ ขอขอบพระคุณครอบครัวที่ให้การสนับสนุนและคอยให้กำลังใจในการทำโครงการฉบับนี้ ที่คอยจัดหาอาหารและเครื่องดื่มไว้รับประทานอยู่เสมอขอขอบพระคุณผู้ที่ไม่ได้กล่าวถึงที่มีส่วนช่วยให้การทำโครงการครั้งนี้ให้สำเร็จลุล่วงไปได้ด้วยดี ขอขอบพระคุณยิ่ง

นาย พชร ตริยง

นาย ชานนท์ สหนนท์

สารบัญ

หน้า

บทคัดย่อ

ABSTRACT

กิตติกรรมประกาศ

สารบัญ

สารบัญตาราง

สารบัญภาพ

รายการสัญลักษณ์และคำย่อ

บทที่ 1 บทนำ	1
1.1 ที่มาและความสำคัญของโครงงาน	1
1.2 วัตถุประสงค์	2
1.3 ขอบเขตของโครงงาน	2
1.4 ประโยชน์ของโครงงาน	3
1.5 ข้อจำกัดของโครงงาน	3
บทที่ 2 วรรณกรรมและงานวิจัยที่เกี่ยวข้อง	4
2.1 แนวคิดและทฤษฎีที่เกี่ยวข้อง	4
2.1.1 แนวคิดที่ข้องกับอัลกอริทึมค้นหาเส้นทาง	4
2.1.2 แนวคิดที่เกี่ยวกับการพัฒนาแอปพลิเคชัน	5
2.1.3 ทฤษฎีที่เกี่ยวข้องกับ Dijkstra Algorithm	6
2.1.4 ทฤษฎีที่เกี่ยวข้องกับ A* Algorithm	7
2.2 ตัวอย่างแอปพลิเคชันที่เกี่ยวข้อง	8
2.1.2 Arknights	8
2.2.2 911 Operator	9
2.3 งานวิจัยที่เกี่ยวข้อง	10
บทที่ 3 วิธีการวิจัย	12

3.1 ภาพรวมของโครงการ	12
3.1.1 Software Architecture Diagram	12
3.2 การวิเคราะห์ขอบเขตและความต้องการของระบบ	13
3.2.1 Functional Requirement	13
3.2.2 Non Functional Requirement	17
3.3 ประเด็นที่น่าสนใจและสิ่งที่ทำ	17
3.3.1 ประเด็นที่น่าสนใจ	17
3.3.2 สิ่งที่ทำหาย	17
3.4 ผลลัพธ์ที่คาดหวัง	18
บทที่ 4 ทฤษฎีการและแผนการดำเนินงาน	19
4.1 การจัดเตรียมฮาร์ดแวร์และซอฟต์แวร์	19
4.1.1 ฮาร์ดแวร์	19
4.1.2 ซอฟต์แวร์	19
4.2 แผนการดำเนินงาน	20
4.3 ผลการดำเนินงาน	21
4.4 Test case	36
บทที่ 5 สรุป	38
5.1 สรุปผลการดำเนินงาน	38
5.2 ข้อเสนอแนะ	38
รายการอ้างอิง	39

สารบัญตาราง

	หน้า
ตารางเปรียบเทียบในแต่ละแอปพลิเคชัน	9
ตารางTest case	36

สารบัญรูปภาพ

	หน้า
รูปภาพที่ 2.1.1.1 ตารางตัวอย่างเกม The prisoner's dilemma	6
รูปภาพที่ 2.1.3.1 รูปภาพแสดงการทำงานของDijkstra Algorithms	7
รูปภาพที่ 2.1.3.2 สมการ Heuristics	8
รูปภาพที่ 2.2.1.1 รูปภาพสถานการณ์ในเกมที่ผู้เล่นต้องเผชิญในด้านตัวอย่าง	9
รูปภาพที่ 2.2.2.1 รูปภาพสถานการณ์ในเกมที่ผู้เล่นต้องเผชิญในด้านตัวอย่าง	9
รูปภาพ 2.3.1.1 การทดสอบอัลกอริทึมในระดับความยากที่ 1	11
รูปภาพ 2.3.1.2 การทดสอบอัลกอริทึมในระดับความยากที่ 2	11
รูปภาพ 2.3.1.3 การทดสอบอัลกอริทึมในระดับความยากที่ 3	11
รูปภาพที่ 3.1.1 Software Architecture Diagram	13
รูปภาพที่ 4.3.1 หน้าเกมเมนูหลัก	21
รูปภาพที่ 4.3.2 ภาพไอคอนของบ้อมปราการทั้ง 2 ชนิด	21
รูปภาพที่ 4.3.3 ภาพจำนวนพลังชีวิตเริ่มต้นของผู้เล่นและจำนวนเงินเริ่มต้น	22
รูปภาพที่ 4.3.4 แสดงการค้นหาเส้นทางของอัลกอริทึม(1)	22
รูปภาพที่ 4.3.5 แสดงการค้นหาเส้นทางของอัลกอริทึม(2)	23
รูปภาพที่ 4.3.6 แสดงการค้นหาเส้นทางของอัลกอริทึม(3)	23
รูปภาพที่ 4.3.7 แสดงการค้นหาเส้นทางของอัลกอริทึม(4)	24
รูปภาพที่ 4.3.8 แสดงการค้นหาเส้นทางของอัลกอริทึม(5)	24
รูปภาพที่ 4.3.9 แสดงการค้นหาเส้นทางของอัลกอริทึม(6)	25
รูปภาพที่ 4.3.10 แสดงการค้นหาเส้นทางของอัลกอริทึม(7)	25
รูปภาพที่ 4.3.11 แสดงการค้นหาเส้นทางของอัลกอริทึม(8)	26
รูปภาพที่ 4.3.12 A* คือเส้นสีแดง	26
รูปภาพที่ 4.3.13 Dijkstra คือเส้นสีฟ้า	27
รูปภาพที่ 4.3.14 รถถังกำลังเดินทางไปตามทางที่อัลกอริทึมค้นหาไว้	27
รูปภาพที่ 4.3.15 ผู้เล่นเริ่มวางบ้อมปราการ	28
รูปภาพที่ 4.3.16 บ้อมปราการที่ผู้เล่นวางไว้โจมตีรถถัง	28

รูปภาพที่ 4.3.17 เริ่มต้นwaveใหม่	29
รูปภาพที่ 4.3.18 รถถังในwaveที่ 2 เริ่มเคลื่อนที่ไปโจมตีบ้านของผู้เล่น	29
รูปภาพที่ 4.3.19 รถถังในwaveที่ 3 เริ่มเคลื่อนที่ไปโจมตีบ้านของผู้เล่น	30
รูปภาพที่ 4.3.20 ผู้เล่นชนะเกม	30
รูปภาพที่ 4.3.21 พลังชีวิตผู้เล่นเหลือ 0	31
รูปภาพที่ 4.3.22 แสดงให้เห็นหน้าการเลือกระดับความยาก	31
รูปภาพที่ 4.3.23 รูปภาพแผนที่ระดับความยากที่ 2	32
รูปภาพที่ 4.3.24รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 2	32
รูปภาพที่ 4.3.25 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 2 หลังจากน้ำท่วมจุดที่ 1	33
รูปภาพที่ 4.3.26 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 2 หลังจากน้ำท่วมจุดที่ 2	33
รูปภาพที่ 4.3.27 รูปภาพแผนที่ระดับความยากที่ 3	34
รูปภาพที่ 4.3.28 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 3	34
รูปภาพที่ 4.3.29 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 3 หลังจากน้ำท่วมจุดที่ 1	35
รูปภาพที่ 4.3.30 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 3 หลังจากน้ำท่วมจุดที่ 2	35

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงการ

ในยุคปัจจุบันเกมได้เข้ามามีบทบาทกับมนุษย์หลายคน ไม่ว่าจะเป็นการเล่นเกมนี้ออนไลน์ การเล่นเกมความสนุก หรือแม้กระทั่งบางคนเล่นเกมเพื่อไปแข่งขันทำเป็นอาชีพ โดยหนึ่งสิ่งที่หลีกเลี่ยงไม่ได้เลยคือการที่เกมนั้นจะเกิดขึ้นได้ก็ต้องมีผู้เล่นทั้ง 2 ฝ่ายในการพยายามหาทางเอาชนะกันในกฎกติกาที่สร้างขึ้นมา แต่วงการอุตสาหกรรมเกมปัจจุบันนี้มีเกมที่ออกแบบมาเพื่อเล่นคนเดียวหรือที่รู้จักกันในชื่อ เกมแนว “Single Player” นั้นเอง โดยจะมีการจำลองผู้เล่นอีกคนขึ้นมาที่มีพฤติกรรมคล้ายกับผู้เล่นคนที่ 2 ที่ผู้เล่นจะต้องเอาชนะ และหนึ่งในพฤติกรรมของโครงการนี้ที่ผู้จัดทำหยิบยกมาคือการค้นหาเส้นทางนั่นเอง

อัลกอริทึมการค้นหาเส้นทางก็เป็นอัลกอริทึมที่นำมาใช้กับเกมในการควบคุมให้สิ่งต่างๆ ให้หาเส้นทางในการเดินทางด้วยตัวเอง แล้วอัลกอริทึมการค้นหาเส้นทางอะไรที่มีประสิทธิภาพมากกว่ากัน ผู้จัดทำโครงการจึงได้ไปศึกษาแล้วพบว่าอัลกอริทึมที่เข้าใจง่ายและน่าสนใจนั้นมี A* และ Dijkstra ซึ่งเป็นอัลกอริทึมที่เข้าใจได้ง่าย โดยที่ทั้ง 2 อัลกอริทึมนี้มีการทำงานที่ต่างกันดังนั้นเราจึงต้องการนำเสนออัลกอริทึมที่แตกต่างกันในรูปแบบของเกมแนว Tower Defense ที่ในตัวละครนั้นจะมีการใช้อัลกอริทึมที่แตกต่างกันทั้ง 2 มาเป็นอัลกอริทึมที่ใช้ในการเคลื่อนที่ไปเพื่อโจมตีตัวผู้เล่น

โครงการนี้นำเสนอผ่านรูปแบบของเกมแนววางแผนป้องกันฐาน (Tower Defense Game) มีเรื่องราวที่ผู้เล่นจะต้องทำการป้องกันบ้านเพื่อไม่ให้รถถังเดินทางมาโจมตีทำลายบ้านของผู้เล่น โดยมีการนำเสนออัลกอริทึมหาเส้นทางผ่านตัวของรถถังที่จะใช้อัลกอริทึมในการหาเส้นทางเพื่อไปโจมตีบ้านของผู้เล่น ที่จะมีรถถัง 2 แบบแบ่งตามอัลกอริทึมที่ใช้ในการค้นหาเส้นทางของรถถังคันนั้นๆ เช่น รถถังสีฟ้าจะใช้เป็นอัลกอริทึม A* รถถังสีแดงจะใช้อัลกอริทึม Dijkstra

1.2 วัตถุประสงค์

โครงการนี้มีเป้าหมายเพื่อสร้างแอปพลิเคชันสำหรับเกมวางแผนในรูปแบบของการป้องกันฐานของผู้เล่นเพื่อนำเสนอการนำอัลกอริทึมมาใช้งานในเกม และผู้จัดทำได้มีการกำหนดเป้าหมายไว้ดังนี้

1. เพื่อศึกษาข้อมูลเกี่ยวกับการทำงานของอัลกอริทึมที่ใช้ในการค้นหาเส้นทาง
2. เพื่อแสดงให้เห็นถึงการทำงานของอัลกอริทึมค้นหาเส้นทางและฝึกสมองใน ส่วนของการวางแผนและตัดสินใจไปพร้อมกัน
3. เพื่อออกแบบระบบการทำงานของเกมแอปพลิเคชันสำหรับอัลกอริทึม

1.3 ขอบเขตของโครงการ

ผู้จัดทำได้กำหนดขอบเขต เพื่อให้บรรลุวัตถุประสงค์ในการสร้างแอปพลิเคชันเกมวางแผนเอาชีวิตรอดจากน้ำท่วมไว้ดังนี้

1. ผู้เล่นมี 3 ชีวิตหากชีวิตหมดผู้เล่นต้องเริ่มเล่นด้านนั้นใหม่
2. ตัวเกมมี 3 ด้านแต่ละด้านจะมีรูปแบบของแผนที่ไม่เหมือนกัน
3. ตัวเกมจะมีบ่อม 3 ชนิด
 - 3.1 บ่อมโจมตีระดับที่ 1
 - 3.2 บ่อมโจมตีระดับที่ 2
 - 3.3 บ่อมโจมตีด้วยจรวด
4. ตัวเกมนั้นจะมีทรัพยากรเงินที่มากำกัฏการที่ผู้เล่นสามารถวางบ่อมได้
5. ทุกครั้งที่บ่อมสามารถกำจัดศัตรูได้ผู้เล่นจะได้รับทรัพยากรเงินเพื่อสร้างบ่อมต่อไป
6. ตัวเกมมีทรัพยากรที่ต้องบริหาร
 - 6.1 เงิน
 - 6.2 พลังชีวิต
7. แอปพลิเคชันสามารถใช้งานได้เพียงในDesktopเท่านั้น
8. แอปพลิเคชันไม่ต้องใช้ระบบอินเทอร์เน็ตในการเชื่อมต่อ

1.4 ประโยชน์ของโครงการ

ในการดำเนินโครงการ หากบรรลุตามเป้าหมายแล้ว จะเกิดประโยชน์จากการใช้งานแอปพลิเคชันดังนี้

1. สามารถสนุกไปพร้อมกับเรียนรู้อัลกอริทึมผ่านตัวแอปพลิเคชัน
2. สามารถเข้าใจถึงการทำงานของอัลกอริทึมที่แตกต่างกันในด้านของระยะทาง กับ หน่วยความจำ
3. สามารถฝึกการบริหารทรัพยากรและการวางแผน

1.5 ข้อจำกัดของโครงการ

1. เนื่องจากแอปพลิเคชันไม่สามารถเข้าผ่านทาง Browser ได้จึงจำเป็นต้องดาวน์โหลดเครื่อง

บทที่2

วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎีที่เกี่ยวข้อง

ในการทำโครงงาน “เกมเอาชีวิตรอดจากน้ำท่วม” ผู้ศึกษาได้ค้นคว้าข้อมูล ทฤษฎีเอกสาร รวมทั้งงานวิจัยที่เกี่ยวข้อง เพื่อนำแนวคิดและทฤษฎีบทมาปรับใช้ในการศึกษาให้เกิดประโยชน์ โดยจะเสนอตามลำดับต่อไปนี้

2.1.1 แนวคิดที่เกี่ยวข้องกับอัลกอริทึมค้นหาเส้นทาง

Hybesis H.urna (2020) อัลกอริทึมในการค้นหาเส้นทาง(Pathfinding Algorithm) คือ กระบวนการหรือขั้นตอนวิธีที่ใช้ในการค้นหาเส้นทางที่เหมาะสมที่สุดระหว่างจุดเริ่มต้นและจุดปลายทางบนกราฟหรือบนแผนที่ โดยที่ใช้ทรัพยากรน้อยที่สุด เช่น เวลา เส้นทาง ราคา ความเสี่ยง น้ำมัน เป็นต้น มีการนำไปใช้งานในหลากหลายด้าน เช่น ระบบนำทาง GPS การควบคุมหุ่นยนต์ AI และเกมบนอุปกรณ์ทั้งหลาย โดยหลักการของ Pathfinding Algorithm จะใช้โครงสร้างกราฟในการแทนข้อมูล และโหนดจะใช้แทนจุดต่างๆและเส้นเชื่อมใช้แทนเส้นทางระหว่างโหนดที่มีระยะทางหรือค่าใช้จ่ายตามที่กำกับ เป้าหมายของการค้นหาเส้นทาง โดยจะเน้นที่ทางที่สั้นที่สุดหรือมีการใช้ทรัพยากรที่คุ้มค่าที่สุด

สมเกียรติ สุนทรสวัสดิ์ (2560) ทฤษฎีกราฟเบื้องต้น(Graph Theory) ในชีวิตประจำวันนั้นกิจกรรมมากมายที่เกิดขึ้นนั้น มักจะเกิดปัญหขึ้น หากแต่ว่าปัญหาเหล่านั้นสามารถแก้ไขได้ด้วยการสร้างแบบจำลองที่เหมาะสมแทนปัญหาจะทำให้เราสามารถเข้าใจปัญหาได้ง่ายมากขึ้น และสามารถแก้ปัญหาโดยการหาคำตอบผ่านแบบจำลอง จากนั้นจึงนำคำตอบที่ได้จากแบบจำลองมาอธิบายผลที่เกิดขึ้นกับปัญหาจริง โดยเราสามารถใชจุดและเส้นเข้ามา หรือก็คือสมการกราฟนั่นเอง

น้ำเพชร รอดประเสริฐ (2022) ทฤษฎีการหาค่าที่เหมาะสมที่สุด (Optimization Theory) เป็นทฤษฎีที่เกี่ยวข้องกับการหาคำตอบที่ดีที่สุดหรือเหมาะสมที่สุดในทางคณิตศาสตร์ โดยจะพิจารณาจากเป้าหมายและขอบเขตของข้อมูลของเป้าหมายที่ถูกกำหนดไว้ และยังถูกนำมาประยุกต์ใช้หลายๆศาสตร์รวมถึงวิทยาการคอมพิวเตอร์ สามารถนำมาใช้ในการปรับแต่งอัลกอริทึม อย่าง A* และDijkstra ในการคำนวณหาเส้นทางที่มีประสิทธิภาพมากที่สุดในเกม

Walter Dean (2015) ทฤษฎีความซับซ้อนในการคำนวณ (Computational Complexity Theory) เป็นสาขาหนึ่งของวิทยาการคอมพิวเตอร์เชิงทฤษฎีโดยมีเป้าหมายหลักในการจัดประเภทและเปรียบเทียบความยากเชิงปฏิบัติในการแก้ปัญหาที่เกี่ยวข้องกับวัตถุเชิงการจัดหมวดหมู่แบบจำกัด (finite combinatorial objects)

สามารถนำมาใช้วิเคราะห์ข้อจำกัดของอัลกอริทึม Dijkstra และ A* ในเกมเอาชีวิตรอดน้ำท่วม โดยมุ่งเน้นไปที่ทรัพยากรของอัลกอริทึม เช่น เวลา(Time Complexity) และหน่วยความจำ (Space Complexity)

2.1.2 ทฤษฎีที่เกี่ยวกับการพัฒนาแอปพลิเคชัน

Don ross (1997) ทฤษฎีเกม (Game Theory) คือการศึกษาแนวทางการมีปฏิสัมพันธ์ระหว่างบุคคลผ่านการตัดสินใจของแต่ละบุคคล โดยจะก่อให้เกิดผลลัพธ์ต่างๆ ตามแต่บริบทของแต่ละสถานการณ์ที่แตกต่างกัน โดยที่ผลลัพธ์จะเป็นผลจากการตัดสินใจของบุคคลซึ่งผ่านการคิดวิเคราะห์ออกมาอย่างรอบคอบว่าจะได้ผลลัพธ์ที่ดีที่สุดเสมอ. ผู้คิดค้นทฤษฎีนี้คือ John von Neumann และ Oskar Morgenstern โดยได้เขียนหนังสือที่มีชื่อว่า “Theory of Games and Economic Behavior” ในทางทฤษฎีเกม “เกม” หมายถึงสถานการณ์ใดๆที่เกิดจากการตัดสินใจจากทั้ง 2 ฝ่าย โดยผู้ตัดสินใจแต่ละฝ่ายจะมีเป้าหมายและผลลัพธ์ที่ตัวเองต้องการขึ้นอยู่กับแต่ละฝ่าย ผู้ตัดสินใจแต่ละฝ่ายเราสามารถนิยามได้ว่าเป็น “ผู้เล่น” โดยทฤษฎีนี้ได้เป็นพื้นฐานสำหรับการศึกษาในหลากหลายสาขาวิชา เช่น เศรษฐศาสตร์ การเมือง ชีววิทยา และวิทยาการคอมพิวเตอร์ ตัวอย่างเกมที่นำทฤษฎีเกมมาใช้ “The prisoner’s dilemma” Merrill Flood and Melvin Dresher(1950) เกมจะนำเสนอสถานการณ์ที่ บุคคล 2 คน ถูกแยกออกจากกันและห้ามติดต่อสื่อสารกัน และจะต้องตัดสินใจเลือกว่าจะร่วมมือกับอีกฝ่ายหรือไม่ โดยผลตอบแทนสูงสุดสำหรับแต่ละฝ่ายจะเกิดจากการที่ทั้ง 2 ฝ่ายร่วมมือกัน

คนร้าย A กับ B ร่วมมือกันขโมยของแล้วถูกจับได้ ทั้ง 2 คนถูกแยกไปสอบปากคำ คนร้ายทั้ง 2 คนมีทางเลือกแค่สารภาพหรือไม่สารภาพ ถ้าคนใดคนหนึ่งสารภาพแต่อีกคนไม่รับสารภาพ ตำรวจจะปล่อยตัวคนที่รับสารภาพไว้เป็นพยาน และจะดำเนินคดีกับคนที่ไม่รับสารภาพ จำคุก 20 ปี ถ้าทั้ง 2 คนรับสารภาพจะถูกโทษจำคุก 10 ปี แต่ถ้าทั้ง 2 คนไม่รับสารภาพจะสามารถดำเนินคดีได้ในข้อหาเล็กน้อย ถูกทำโทษจำคุก คนละ 1 ปี

	รับสารภาพ	ไม่รับสารภาพ
รับสารภาพ	-10, -10	0, -20
ไม่รับสารภาพ	-20, 0	-1, -1

รูปภาพที่ 2.1.1.1 ตารางตัวอย่างเกม The prisoner’s dilemma

โดยเราจะเห็นได้ว่ากลยุทธ์ที่โดดเด่นของทั้ง 2 ฝ่ายคือการรับสารภาพ เพราะไม่ว่าผู้เล่นจะตัดสินใจอย่างไร ก็จะได้ผลตอบแทนที่ดีกว่าเสมอ แต่เมื่อทั้ง 2 ฝ่ายเลือกเส้นทางนี้ กลับ

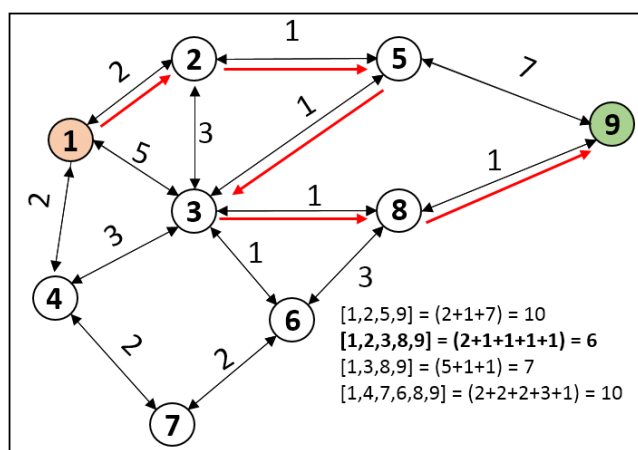
ไม่ได้ให้ผลลัพธ์ที่ดีที่สุด เราสามารถนำแนวคิดนี้มาประยุกต์ใช้ในเกม เพื่อให้ผู้เล่นรู้สึกถึงความพึงพอใจในการตัดสินใจของผู้เล่นว่าได้ตัดสินใจมาอย่างดีที่สุดแล้ว แต่กลับพบว่าอาจจะไม่ใช่ทางเลือกที่ดีที่สุดจริงๆ เพื่อให้เกมมีความซับซ้อนมากขึ้น

“What is Data Structures” (n.d.) โครงสร้างข้อมูล (Data Structures) คือการจัดระเบียบและเก็บข้อมูลในคอมพิวเตอร์ ให้มีประสิทธิภาพต่อการเข้าถึงและประมวลผลในโปรแกรมและอัลกอริทึมต่างๆ โครงสร้างข้อมูลที่เหมาะสมจะสามารถทำให้การดำเนินการต่างๆ เช่น การเพิ่ม การลบ การค้นหา หรือการเรียงลำดับข้อมูลเป็นไปได้อย่างรวดเร็วยิ่งขึ้น การจัดการระเบียบหรือการเก็บข้อมูลต่างๆ ในเกมให้เหมาะสมจะทำให้เกมมีประสิทธิภาพและการประมวลผลที่ดี

Masterclass (2021) เกมแนว Tower Defense หรือเกมแนว TD เป็นเกมการวางแผนที่ผู้เล่นคนเดียววางแผนป้องกันพื้นที่หรือป้องกันทรัพยากรของผู้เล่น โดยการสร้างโครงสร้างหรือสิ่งกีดขวางต่างๆ ไม่ให้ศัตรูเคลื่อนที่หรือกำจัดก่อนที่ศัตรูจะเข้าถึงพื้นที่หรือทรัพยากรของผู้เล่น เกมประเภทนี้สามารถเล่นได้แบบหลายคน หรือเล่นคนเดียวก็ได้ เกมประเภทนี้ไม่จำเป็นจะต้องเป็นปราสาทและหอคอยแต่เพียงอย่างเดียว ผู้พัฒนาเกมสามารถออกแบบสิ่งต่างๆ ให้มีความน่าดึงดูดได้

2.1.3 ทฤษฎีที่เกี่ยวข้องกับ Dijkstra search Algorithms

Fatima Rubio (2023) Dijkstra Algorithms คืออัลกอริทึมที่ถูกใช้ในการหาเส้นทางที่สั้นที่สุดระหว่างจุด 2 จุด ในกราฟ โดยประเมินแต่ละโหนดในกราฟและคำนวณระยะทางจากโหนดเริ่มต้นไปยังโหนดอื่นๆ ขั้นตอนเริ่มต้นการทำงานของอัลกอริทึมจะประเมินจุดเริ่มต้น ให้กำหนดระยะทางไว้เป็น 0 จากนั้นจะประเมินโหนดที่อยู่ติดกัน และคำนวณระยะทางไปยังแต่ละโหนด อัลกอริทึมจะเลือกโหนดที่มีระยะทางที่สั้นที่สุด และเพิ่มเข้าไปในบันทึกในรายการว่าได้เดินผ่านเยี่ยมชมไปแล้ว หลังจากนั้นจะประเมินโหนดที่อยู่ติดกันกับโหนดที่ถูกเพิ่มเข้าไปใหม่ในรายการ อัลกอริทึมจะคำนวณ เลือกโหนดที่มีระยะทางที่สั้นที่สุดและจะบันทึกไว้ในรายการว่าได้เยี่ยมชมไปแล้วกระบวนการทั้งหมดนี้จะดำเนินต่อไปเรื่อย ๆ จนกว่าจะถึงโหนดปลายทาง ในขณะที่ประเมินแต่ละโหนด อัลกอริทึมจะทำการบันทึกระยะทางที่ใช้เดินทางในแต่ละโหนด หากโหนดได้รับการประเมินซ้ำอีกครั้งและพบว่าใช้ระยะทางที่สั้นกว่าก็จะทำการปรับปรุงระยะทาง ให้เป็นระยะทางที่สั้นที่สุดจากการประเมินในรอบปัจจุบันเพื่อแสดงเส้นทางที่ดีกว่า Dijkstra Algorithms ถูกใช้อย่างแพร่หลายในระบบนำทาง เช่น การวางแผนการขนส่ง ระบบ GPS นำทาง เป็นต้น



รูปภาพที่ 2.1.3.1 รูปภาพแสดงการทำงานของDijkstra Algorithms

2.1.4 ทฤษฎีที่เกี่ยวข้องกับ A* search Algorithms

Fatima Rubio (2023) A* Algorithms เป็นอัลกอริทึมสำหรับการค้นหาเส้นทางและกำลัสำรวจกราฟที่ใช้ Heuristics ในการประมาณเส้นทางที่มีประสิทธิภาพมากที่สุด A* เป็นอัลกอริทึมที่ทรงพลังมากในการหาเส้นทางที่สั้นที่สุดระหว่างจุด 2 จุดในกราฟ ด้วยการสำรวจกราฟและใช้ Function Heuristics ในการหาเส้นทางไปยังจุดหมาย Function Heuristics จะประมาณระยะทางระหว่างโหนดปัจจุบันและโหนดปลายทาง ซึ่งเป็นจุดเด่นที่ทำให้ A* Algorithms แตกต่างจากอัลกอริทึมค้นหาเส้นทางอื่นๆ ช่วยให้อัลกอริทึมมุ่งไปยังปลายทางได้อย่างมีประสิทธิภาพได้มากขึ้น โดยให้ความสำคัญกับโหนดที่อยู่ใกล้ปลายทางมากกว่า นิยมใช้กันอย่างแพร่หลายกับ หุ่นยนต์ วิดีโอเกม และแอปพลิเคชันที่ต้องการการนำทางที่มีประสิทธิภาพ

Kristen M. Meiburger and Filippo Molinari (2018) Heuristics Function หรือที่เรียกว่าHeuristics เป็นวิธีการประเภทหนึ่งที่ใช้ลำดับการเลือกในอัลกอริทึมการค้นหาเส้นทาง โดยอาศัยข้อมูลที่มีอยู่ในการตัดสินใจเลือกเส้นทางใด ข้อดีของการนำ Heuristic มาใช้คือมีความรวดเร็วในการตัดสินใจเลือกเส้นทาง ข้อเสียคือ หากไม่กำหนดค่า Heuristics ให้เหมาะสม อาจจะทำให้เสียค่าใช้จ่ายมากกว่าการใช้อัลกอริทึมค้นหาเส้นทางแบบปกติ A* จะประเมินโหนดแต่ละโหนดโดยใช้ Function " $f(n) = g(n) + h(n)$ " " $g(n)$ " คือระยะทางระหว่างโหนดเริ่มต้นถึงโหนดปัจจุบันมีค่า n " $h(n)$ " คือค่า heuristics หรือการประมาณค่าระยะทางจากโหนดปัจจุบัน n ไปยังโหนดเป้าหมาย " $f(n)$ " คือผลรวมของค่าทั้ง 2 และเป็นค่าเพื่อตัดสินใจเลือกเส้นทาง

$$f(n) = g(n) + h(n)$$

รูปภาพที่ 2.1.3.2 สมการ Heuristics

2.2 ตัวอย่างแอปพลิเคชันที่เกี่ยวข้อง

2.2.1 Arknights

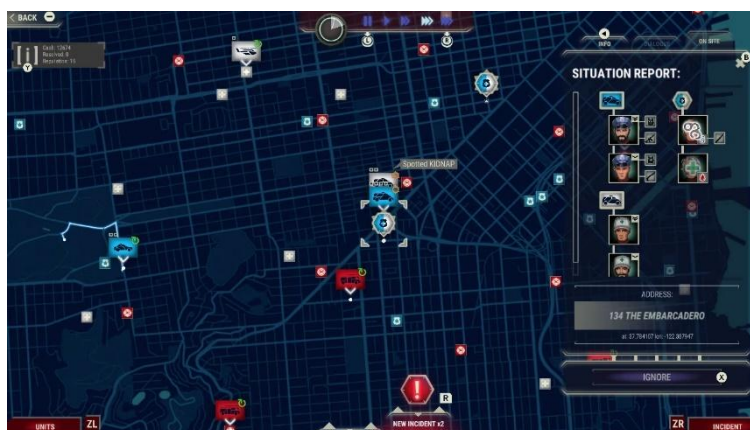
เป็นเกมมือถือ เล่นสมมติบทบาทแนวยุทธวิธีเป็นเกมแนว Tower Defense ที่สามารถเล่นได้ฟรี พัฒนาโดย Hypergryph ผู้พัฒนาเกมจากประเทศจีน ที่เปิดให้เล่นวันที่ 1 พฤษภาคม ปี พ.ศ.2562 ที่ประเทศจีน และเปิดให้บริการในประเทศอื่นๆในวันที่ 16 มกราคม พ.ศ.2563 และที่ไต้หวันในวันที่ 29 มิถุนายน พ.ศ. 2563 มีให้บริการทั้งระบบปฏิบัติการแบบ Android และ IOS การเล่นเกมคือผู้เล่นจะได้เข้าไปในด่านด่านใดด่านหนึ่งและจะต้องบริหารทรัพยากรที่มีอยู่อย่างจำกัดในการวางยูนิตลงในจุดต่างๆเพื่อป้องกันฐานตัวเอง ไม่ให้ศัตรูเข้ามาที่ฐานและทำให้ค่าพลังชีวิตของฐานเหลือ 0 หากทำไม่สำเร็จหรือพลังชีวิตฐานของผู้เล่นเหลือ 0 เกมก็จะจบลงที่ผู้เล่นเป็นฝ่ายแพ้ หากผู้เล่นสามารถกำจัดศัตรูได้หมด ผู้เล่นจะเป็นฝ่ายชนะ



รูปภาพที่ 2.2.1.1 รูปภาพสถานการณ์ในเกมที่ผู้เล่นต้องเผชิญในด้านตัวอย่าง

2.2.2 911 Operator

เป็นเกมจำลองสถานการณ์ที่ถูกพัฒนาโดย ค่ายเกม Studio Justsu Games ของประเทศโปแลนด์และจัดจำหน่ายในปี พ.ศ.2560 โดยที่ผู้เล่นจะได้รับบทบาทเป็นเจ้าหน้าที่กู้ภัยที่คอยรับโทรศัพท์และจะต้องคอยส่งเจ้าหน้าที่ในแต่ละแผนกไประงับเหตุการณ์ทั้งหลายเพื่อเก็บคะแนน โดยที่เวลาผู้เล่นส่งเจ้าหน้าที่ออกไปนั้นเพียงแคกดที่จุดหมายตัวเกมจะส่งเจ้าหน้าที่ออกไปโดยที่มีการคำนวณเส้นทางไว้แล้ว โดยในตัวเกมมีการใช้เพียงอัลกอริทึม A* มาใช้ในการหาเส้นทางเท่านั้น



รูปภาพที่ 2.2.2.1 รูปภาพสถานการณ์ในเกมที่ผู้เล่นต้องเผชิญในด้านตัวอย่าง

ตารางเปรียบเทียบแต่ละแอปพลิเคชัน

ระบบการเล่นหรือพีเจอร์ในเกม	Arknights	911Operator	เกมเอาชีวิตรอดน้ำท่วม
การเดินทางของศัตรูมีรูปแบบที่ชัดเจน	มี	ไม่มี	ไม่มี
การแสดงผลเส้นทางที่อัลกอริทึมคำนวณไว้	ไม่มี	มี	มี
มีการใช้อัลกอริทึมในการหาเส้นทาง	ไม่มี	มี	มี
A* Algorithm	ไม่มี	มี	มี
Dijkstra Algorithm	ไม่มี	ไม่มี	มี
เกมแนวTower Defense	มี	ไม่มี	มี

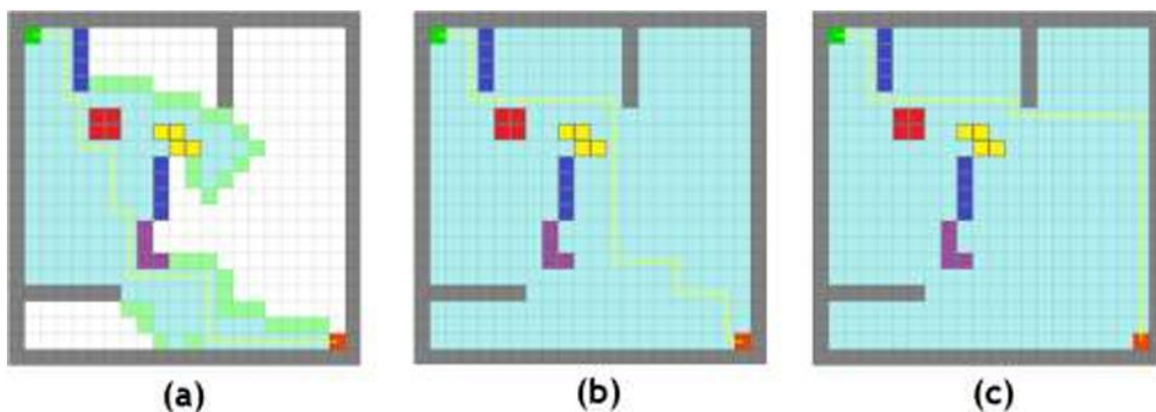
2.3 งานวิจัยที่เกี่ยวข้อง

2.3.1 Ade Syahputra, Budi Arifitama, Ketut bayu yogha Bintoro and Silvester

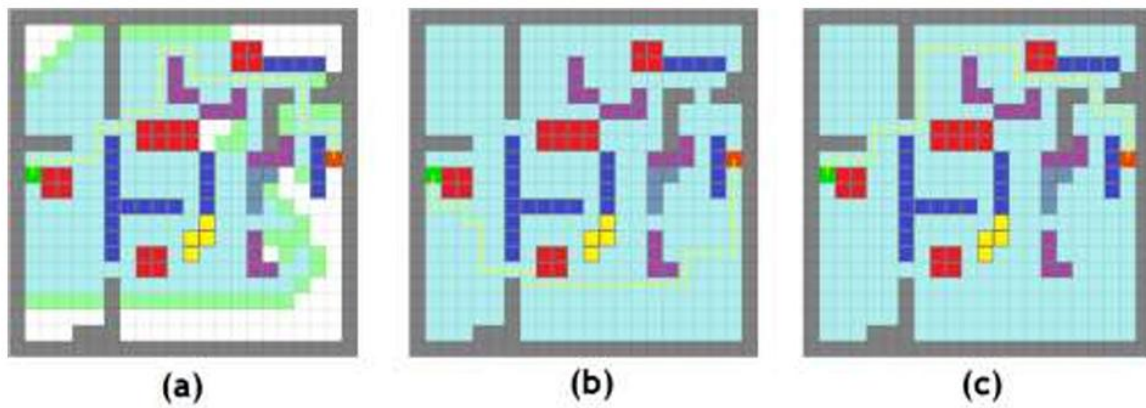
Dian Handy Permana (2018) ได้ศึกษาการเปรียบเทียบระหว่าง อัลกอริทึม Dijkstra a* และ Breadth First Search (BFS) อัลกอริทึมแบบใดนั้นมีความเหมาะสมที่จะนำมาใช้ในเกม Maze Runner Game โดยวิธีที่ใช้การวิเคราะห์ว่าอัลกอริทึมใดมีความเหมาะสมมากกว่ากันที่จะนำมาใช้เกม Maze Runner โดยวิธีที่ใช้ในการวิจัยคือ กำหนดระดับความยากของด่านไว้ที่ 3 ระดับ ในแต่ละAlgorithm ที่ทดสอบจะมีสิ่งกีดขวางที่เหมือนกัน การวัดผลจะใช้เวลา ความยาวของเส้นทางและ จำนวนช่องที่ใช้ในการคำนวณในการเดินทางไปยังจุดหมาย ผู้วิจัยจะมี 3 ระดับ โดยที่แต่ละระดับจะมีสิ่ง กีดขวางเป็นบล็อก tretis ที่แตกต่างกัน หลังจากทดสอบเสร็จทั้ง 3 ระดับจึงจะทำการสรุปผล ว่า Algorithm ใด เหมาะสมกับการนำไปใช้เกม Maze Runner

โดยสรุปผลการทดลองไว้ดังนี้

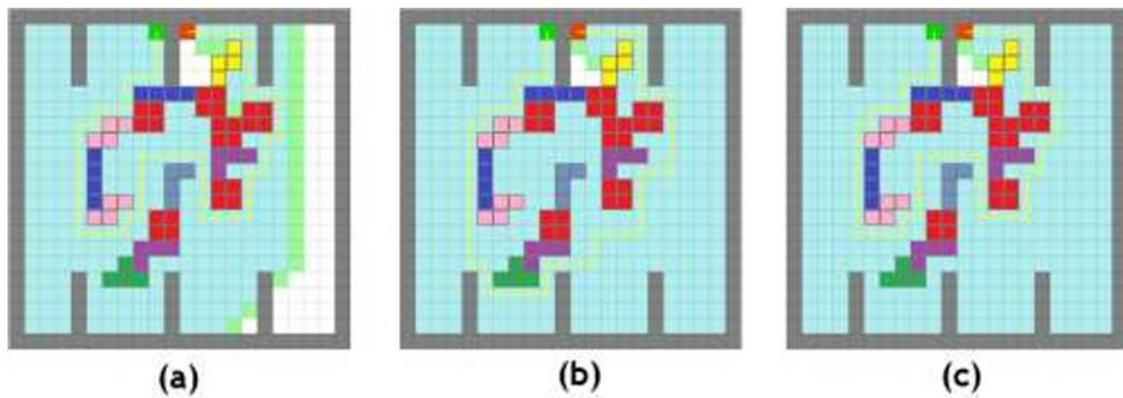
1. A*, Dijkstra และ BFS สามารถนำมาใช้ในการหาเส้นทางในเกมเขาวงกต ได้ทั้งหมด เพราะสามารถหาเส้นทางที่สั้นที่สุด
2. A* คืออัลกอริทึมที่ดีที่สุดในเกมรูปแบบGrid เนื่องจากใช้หน่วยความจำน้อย และใช้เวลาในการค้นหาเส้นทางน้อย
3. การเลือกใช้อัลกอริทึมที่เหมาะสมจะทำให้เกมมีประสิทธิภาพที่ดีขึ้นได้



รูปภาพ2.3.1.1 การทดสอบอัลกอริทึมในระดับความยากที่ 1 (a)แทน A* (b)แทนDijkstra และ (c) แทน BFS



รูปภาพ2.3.1.2 การทดสอบอัลกอริทึมในระดับความยากที่2 (a)แทน A* (b)แทนDijkstra และ (c) แทน BFS



รูปภาพ2.3.1.3 การทดสอบอัลกอริทึมในระดับความยากที่3 (a)แทน A* (b)แทนDijkstra และ (c) แทน BFS

บทที่3

วิธีการวิจัย

3.1 ภาพรวมของโครงงาน

ในการพัฒนาเว็บแอปพลิเคชัน เพื่อการแสดงผลและเปรียบเทียบการทำงานของอัลกอริทึมทั้งสองนั้น มีขั้นตอนการดำเนินงานดังนี้

3.1.1 Software Architecture Diagram

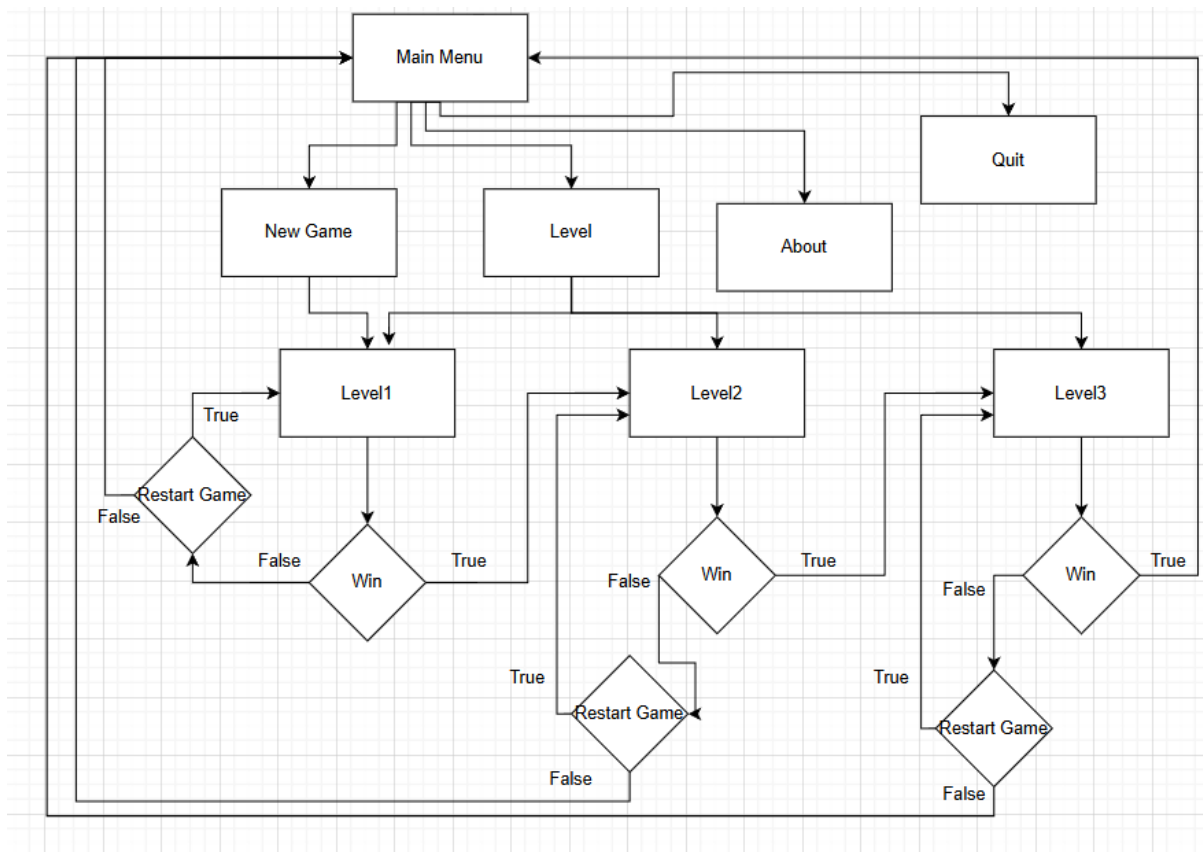
ในเกมเอาชีวิตรอดน้ำท่วมเราจะใช้ Godot Engine ในการพัฒนาเกม จะมีการแบ่งการออกแบบระบบการทำงานไว้ดังนี้

เมื่อผู้เล่นเข้าเกมจะเริ่มต้นด้วยหน้าMain Menu ที่จะมีตัวเลือก

- New Game เริ่มเกมตั้งแต่Level 1
- Level ผู้เล่นจะสามารถเลือกระดับความยากก่อนเริ่มเกม
- About ผู้เล่นจะได้เห็นข้อมูลเกี่ยวกับตัวเกม
- Quit ปิดเกม

เมื่อเลือก New Game ผู้เล่นจะเข้าสู่ เกม Level 1 ถ้าผู้เล่นเล่นเกมในเกมนั้นชนะจะสามารถเลือกได้ว่าจะไปยังความยากระดับถัดไปหรือจะกลับไปหน้าMain Menu ก็ได้ หากผู้เล่นแพ้จะเลือกได้ว่าจะเริ่มเกมใหม่อีกครั้งที่ระดับความยากเดิมหรือกลับออกไปที่หน้าMain Menu ก็ได้

เมื่อผู้เล่นเลือก Level ผู้เล่นจะเข้าสู่หน้าเลือกระดับความยาก จะมีให้เลือกทั้งหมด 3 ระดับ ไม่ว่าผู้เล่นจะเลือกระดับความยากใดเกมจะมีความทำงานเหมือนกันคือ ถ้าผู้เล่นเล่นเกมในเกมนั้นชนะจะสามารถเลือกได้ว่าจะไปยังความยากระดับถัดไปหรือจะกลับไปหน้าMain Menu ก็ได้ หากผู้เล่นแพ้จะเลือกได้ว่าจะเริ่มเกมใหม่อีกครั้งที่ระดับความยากเดิมหรือกลับออกไปที่หน้าMain Menu ก็ได้



รูปภาพที่ 3.3.1 Software Architecture Diagram

3.2 การวิเคราะห์ขอบเขตและความต้องการของระบบ

3.2.1 Functional Requirement

3.2.1.1 Use Case

(1) ID and Name : UC-1 ผู้เล่นกดหน้าเลือกด่าน

Primary Actor : ผู้เล่น Secondary Actor: -

Description : ผู้เล่นกดหน้าเลือกด่านจากหน้าจอ

Trigger : ผู้เล่นต้องการเลือกด่านเพื่อเริ่มเล่นเกม

Preconditions : PRE-1 ผู้เล่นต้องเปิดแอปเกมขึ้นมาก่อน

PRE-2 ผู้เล่นต้องมีเป้าหมายด่านที่ต้องการเล่น

Postconditions : POST-1 ผู้เล่นจะเข้าไปในด่านที่ตนเลือก

Normal Flow : 1.0 แสดงหน้าต่างเลือกด่าน

1. ผู้เล่นต้องเปิดแอปพลิเคชันเกม

2. ผู้เล่นกดปุ่ม เลือกด่าน

3. ผู้เล่นกดเลือกด้าน

4.ระบบโหลดด้านที่ผู้เล่นเลือก

Priority: High

(2) ID and Name : UC-2 ผู้เล่นได้รับคะแนน

Primary Actor : ป้อม Secondary Actor: ผู้เล่น

Description : ป้อมทำการกำจัดศัตรูในระยะทำให้ผู้เล่นได้คะแนน

Trigger : ศัตรูเดินเข้ามาในระยะป้อมที่ผู้เล่นวางไว้

Preconditions : PRE-1 มีป้อมที่ถูกสร้างโดยผู้เล่น

PRE-2 มีศัตรูอยู่ในระยะโจมตีของป้อม

Postconditions : POST-1 ศัตรูจะถูกกำจัดและผู้เล่นจะได้รับคะแนน

Normal Flow : 1.0 ศัตรูปรากฏบนแผนที่

1. ศัตรูเดินเข้ามาในระยะโจมตีของป้อม

2. ป้อมตรวจพบศัตรูในระยะ

3. ป้อมทำการโจมตีศัตรู

4.ศัตรูถูกกำจัด

5.ผู้เล่นได้รับคะแนนเมื่อเลือดของศัตรูเหลือ 0

Priority: High

(3) ID and Name : UC-3 ศัตรูทำลายป้อมของผู้เล่น

Primary Actor : ศัตรู Secondary Actor: ป้อมที่ผู้เล่นสร้าง

Description : ศัตรูโจมตีป้อมที่ผู้เล่นสร้างเมื่อป้อมอยู่ในระยะโจมตีจนป้อมถูกทำลาย

Trigger : ป้อมที่ผู้เล่นสร้างไว้อยู่ในระยะโจมตีของศัตรู

Preconditions : PRE-1 มีป้อมที่ถูกสร้างไว้วางอยู่ในพื้นที่

PRE-2 มีศัตรูอยู่ในแผนที่

Postconditions : POST-1 ศัตรูจะถูกกำจัดและผู้เล่นจะได้รับคะแนน

Normal Flow : 1.0 ศัตรูปรากฏบนแผนที่

1. ศัตรูเดินเข้ามาในระยะโจมตีของป้อม
2. ป้อมตรวจพบศัตรูในระยะ
3. ป้อมทำการโจมตีศัตรู
4. ศัตรูถูกกำจัด
5. ผู้เล่นได้รับคะแนนเมื่อเลือดของศัตรูเหลือ 0

Priority: High

(4) ID and Name : UC-4 ผู้เล่นได้รับเงิน

Primary Actor : ป้อม Secondary Actor: ผู้เล่น

Description : ป้อมทำการกำจัดศัตรูในระยะทำให้ผู้เล่นได้คะแนน

Trigger : ศัตรูเดินเข้ามาในระยะป้อมที่ผู้เล่นวางไว้

Preconditions : PRE-1 มีป้อมที่ถูกสร้างโดยผู้เล่น

PRE-2 มีศัตรูอยู่ในระยะโจมตีของป้อม

Postconditions : POST-1 ศัตรูจะถูกกำจัดและผู้เล่นจะได้รับคะแนน

Normal Flow : 1.0 ศัตรูปรากฏบนแผนที่

1. ศัตรูเดินเข้ามาในระยะโจมตีของป้อม
2. ป้อมตรวจพบศัตรูในระยะ
3. ป้อมทำการโจมตีศัตรู
4. ศัตรูถูกกำจัด
5. ผู้เล่นได้รับเงินเมื่อเลือดของศัตรูเหลือ 0

Priority: High

(5) ID and Name : UC-5 ชนะเกม

Primary Actor : ระบบของเกม Secondary Actor: -

Description : เกมจบหลังจากที่ศัตรูทั้งหมดในทุกระลอกถูกกำจัด

Trigger : ไม่มีศัตรูเหลือในแผนที่

Preconditions : PRE-1 เป็นระลอกของศัตรูระลอกสุดท้าย

PRE-2 มีศัตรูอยู่ในแผนที่

Postconditions : POST-1 เกมจะจบและระบบจะเริ่มนับคะแนน

Normal Flow : 1.0 หน้าต่างแสดงผลสรุป

1. ศัตรูเหลือตัวสุดท้าย
2. บั๊อมตรวจพบศัตรูในระยะ
3. บั๊อมทำการโจมตีศัตรู
4. ศัตรูตัวสุดท้ายถูกกำจัด
5. เกมทำการเปิดหน้าต่างสรุปคะแนน

Priority: High

(7) ID and Name : UC-7 แพ้เกม

Primary Actor : ระบบของเกม Secondary Actor: -

Description : เกมจบหลังจากที่พลังชีวิตของผู้เล่นเหลือ 0

Trigger : พลังชีวิตของผู้เล่นเหลือ 0

Preconditions : PRE-1 ผู้เล่นมีพลังชีวิตเหลือ 1

PRE-2 มีศัตรูอยู่ในแผนที่

Postconditions : POST-1 เกมจะจบและระบบจะเริ่มนับคะแนน

Normal Flow : 1.0 หน้าต่างแสดงผลสรุป

1. ศัตรูอยู่ในแผนที่
2. ผู้เล่นมีพลังชีวิตเหลือ 1
3. ศัตรูเดินเข้าไปในบ้านของผู้เล่น
4. ผู้เล่นมีพลังชีวิตเหลือ 0
5. เกมทำการเปิดหน้าต่างแพ้ในตานัน

Priority: High

3.2.2 Non-Functional Requirement

3.2.2.1 Usability

เกมเอาชีวิตรอดจากน้ำท่วม เอาชีวิตรอดจากน้ำท่วม เนื่องจากตัวเกมต้องการมุ่งเน้นไปที่การแสดงอัลกอริทึมเป็นหลัก จึงออกแบบหน้าตามาให้ใช้งานง่ายภาพไม่ดูยุ่งยาก และสามารถเปรียบเทียบอัลกอริทึมได้ชัดเจน

3.2.2.2 Portability

เกมเอาชีวิตรอดจากน้ำท่วม เป็นแอปพลิเคชันที่ต้องติดตั้งบนคอมพิวเตอร์เท่านั้น**

3.2.2.3 Performance

เกมเอาชีวิตรอดจากน้ำท่วม เนื่องจากเป็นเกมที่ต้องมีการโหลดศัตรูออกมาจำนวนมาก จึงได้มีการเลือกอัลกอริทึมที่ไม่สิ้นเปลืองทรัพยากรในการเปรียบเทียบกับ Dijkstra ซึ่งก็คือ A*

3.3 ประเด็นที่น่าสนใจและสิ่งท้าทาย

3.3.1 ประเด็นที่น่าสนใจ

ในปัจจุบันเกมเป็นสื่อความบันเทิงที่ได้รับความนิยมอย่างแพร่หลาย การสร้างสื่อการเรียนรู้ต่าง ๆ ผ่านการเล่นเกมจึงเป็นที่นิยมนำมาใช้ในการเรียนการสอน อย่างไรก็ตาม การสร้างเกมขึ้นมานั้นไม่ใช่เรื่องง่ายที่จะสามารถสร้างความสนุกสนานไปพร้อมกับการเรียนรู้ได้ โดยเกมเอาชีวิตรอดน้ำท่วมจะแสดงให้เห็นถึงความแตกต่างระหว่างการทำงานของ Dijkstra กับ A* อย่างชัดเจนผ่านการแสดงการเคลื่อนที่ของศัตรู 2 ประเภทที่ใช้ Algorithm ที่แตกต่างกัน

3.3.2 สิ่งท้าทาย

การออกแบบเกมให้มีความน่าสนใจ และสร้างสรรค์ ที่ผู้เล่นจะรู้สึกสนุกสนานไปกับสิ่งที่เกมพยายามจะนำเสนอ โดยที่จะต้องได้ทั้งความสนุกสนาน เกิดการเรียนรู้ความแตกต่างระหว่าง Dijkstra กับ A* และทำให้ผู้เล่นรู้สึกไม่เบื่อ

3.4 ผลลัพธ์ที่คาดหวัง

3.4.1 ตัวเกมสามารถเล่นได้อย่างมีประสิทธิภาพ ตัวเกมไม่มีปัญหาเรื่องการประมวลผลซ้ำ

3.4.2 ผู้เล่นได้รับความสนุกสนานพร้อมกับเรียนรู้ความแตกต่างระหว่าง Dijkstra Algorithm กับ A* Algorithm

บทที่ 4

ทรัพยากรและแผนการดำเนินงาน

4.1 การจัดเตรียมฮาร์ดแวร์และซอฟต์แวร์

4.1.1 ฮาร์ดแวร์ที่ใช้

ตารางแสดงคุณสมบัติของคอมพิวเตอร์สำหรับพัฒนาเกม

	คอมพิวเตอร์
Processor	12th Gen Intel(R) Core(TM) i9-12900K 3.2GHz
Installed memory(RAM)	48 GB
System type	64-bit
Operating System	Windows 10
Graphic Card	NVIDIA GeForce RTX3050

ตารางที่ 4.1 แสดงคุณสมบัติของคอมพิวเตอร์สำหรับพัฒนาเว็บไซต์

4.1.2 ซอฟต์แวร์ที่ใช้

1. Microsoft Word
2. Godot
3. Draw.io
4. Visual Studio Code
5. Figma

4.2 แผนการดำเนินงาน

4.2.1 ตารางการดำเนินงานส่วนที่ 1

ขั้นตอน	ส.ค.67				ก.ย.67				ต.ค.67				พ.ย.67				ธ.ค.67			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. กำหนดหัวข้อโครงการ																				
2. กำหนดขอบเขตปัญหาและเป้าหมาย ของงาน																				
3. ทบทวนวรรณกรรม และ ศึกษาาระบบที่มี มาก่อนหน้า																				
4. ออกแบบโปรโตไทป์ของเกม																				
5. ออกแบบ Diagram																				
6. ส่งรายงานฉบับสมบูรณ์																				
7. นำเสนอโครงการ																				

4.2.2 ตารางการดำเนินงานส่วนที่ 2

ขั้นตอน	ม.ค.68				ก.พ.68				มี.ค.68				เม.ย.68				พ.ค.68			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
8. ศึกษาการใช้โปรแกรมGodot																				
9. พัฒนาเกมตัวเบื้องต้น																				
10. ศึกษาการใช้อัลกอริทึม A*																				
11. ศึกษาการใช้อัลกอริทึม Dijkstra																				
12. พัฒนาตัวเกมแบบสมบูรณ์																				
13. ตรวจสอบปัญหาที่ต้องแก้ไข																				
14. นำเสนอตัวเกมต่อกรรมการ																				

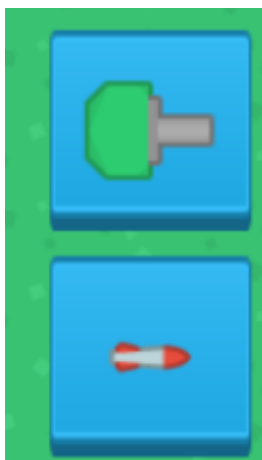
4.3 ผลการดำเนินงาน

ผู้จัดทำโครงการได้พัฒนาเกมเล่นคนเดียวที่ผู้เล่นจะมีหน้าที่ในการวางป้อมปราการเพื่อป้องกันศัตรูบุกเข้ามาโจมตีฐานของผู้เล่น โดยศัตรูจะใช้เส้นทางการเคลื่อนที่ด้วยอัลกอริทึม A* และ อัลกอริทึม Dijkstra



รูปภาพที่ 4.3.1 หน้าเกมเมนูหลัก

เมื่อผู้เล่นกด New Game จะเป็นการเข้าเล่นเกมตั้งแต่ level1 ผู้เล่นจะได้รับพลังชีวิต 100 หน่วย พร้อมกับเงินในการสร้างป้อมปราการอีกเป็นจำนวน 200 หน่วย ในเกมจะมีป้อมปราการอยู่ทั้งหมด 2 ชนิดได้แก่ ป้อมปราการยิงกระสุนธรรมดาแทียร์1 และป้อมปราการยิงจรวดมิสไซล์ โดยป้อมปราการยิงกระสุนธรรมดาแทียร์1 จะมีพลังโจมตีอยู่ที่ 20 หน่วย และป้อมปราการจรวดมิสไซล์จะมีพลังโจมตีอยู่ที่ 30 หน่วย โดยป้อมปราการจะไม่สามารถวางบนพื้นที่อื่นๆได้ นอกจากพื้นที่หญ้าสีเขียว

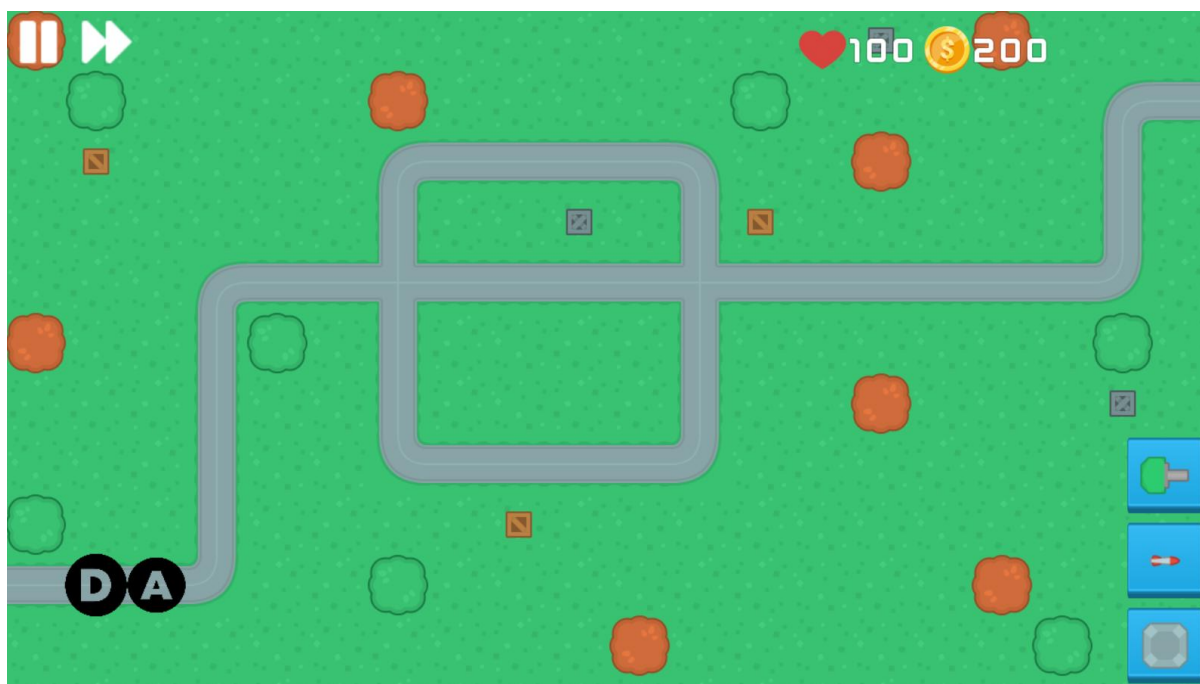


รูปภาพที่ 4.3.2 ภาพไอคอนของป้อมปราการทั้ง 2 ชนิด

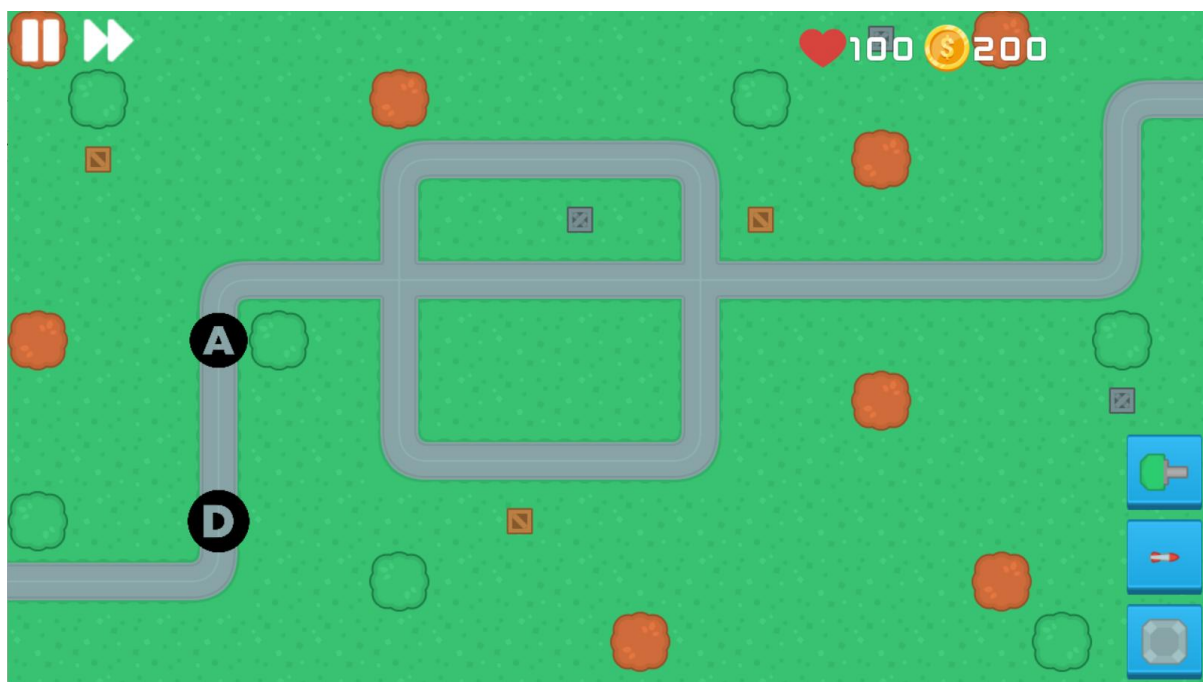


รูปภาพที่ 4.3.3 ภาพจำนวนพลังชีวิตเริ่มต้นของผู้เล่นและจำนวนเงินเริ่มต้น

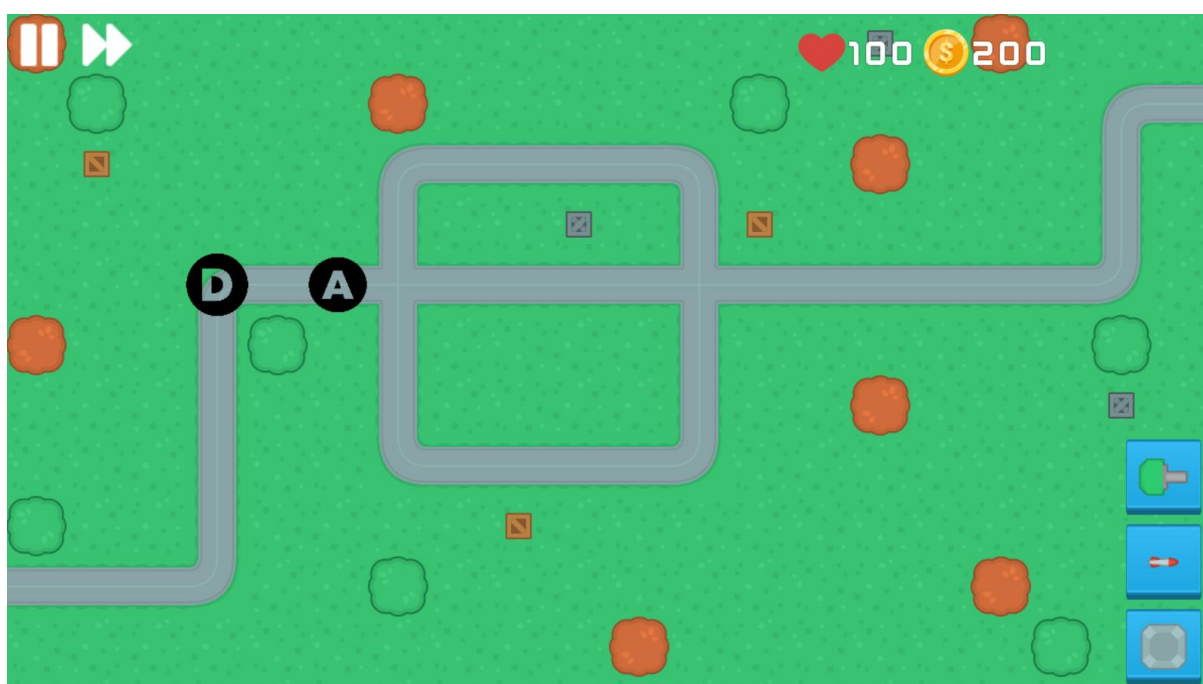
ผ่านไปสักพักเกมจะเริ่มทำการคำนวณเส้นทางที่ใช้ในการไปหาผู้เล่นโดยใช้เส้นทางที่สั้นที่สุด โดยผู้เล่นจะเห็นการทำงานของอัลกอริทึมผ่านวงกลมที่เกิดขึ้นที่กำลังเดินทางไปสำรวจตามจุดต่างๆ ก่อนที่จะเริ่มวาดเส้นทางที่สั้นที่สุด



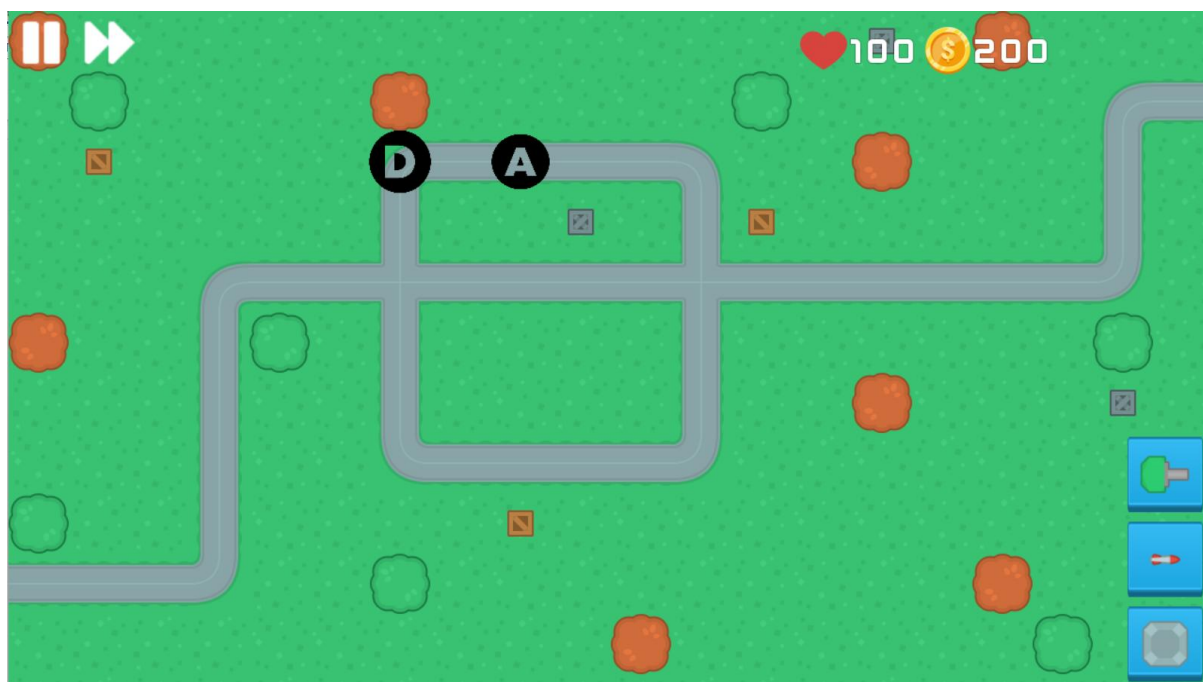
รูปภาพที่ 4.3.4 แสดงการค้นหาเส้นทางของอัลกอริทึม(1)



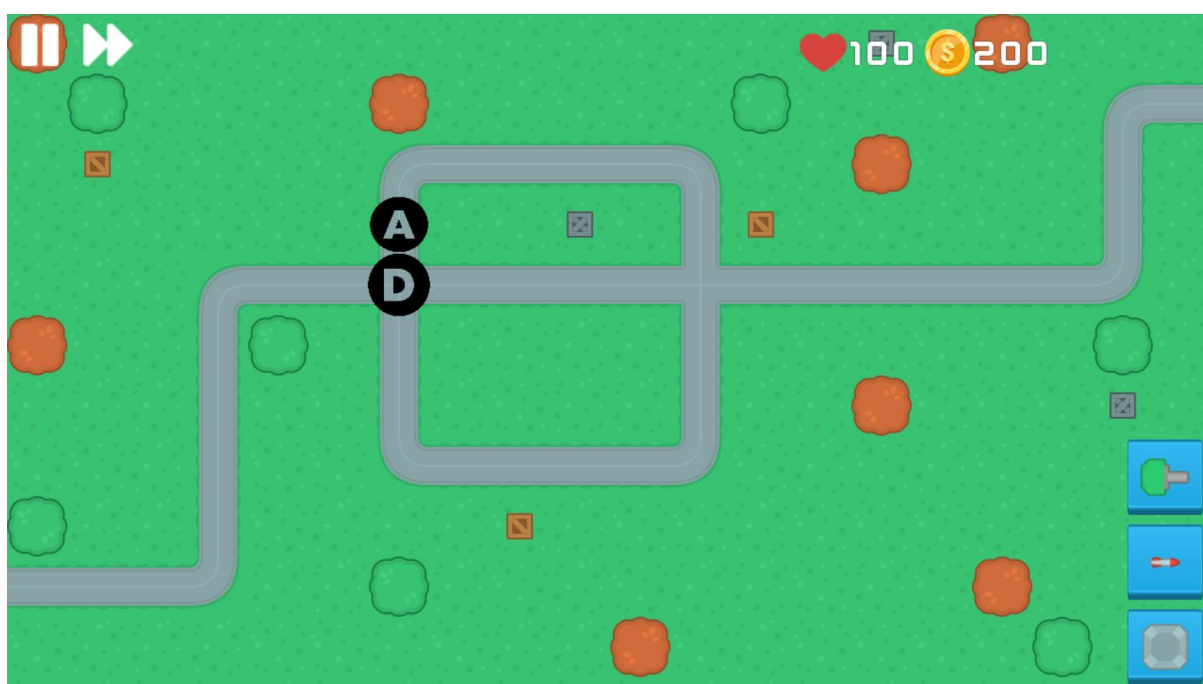
รูปภาพที่ 4.3.5 แสดงการค้นหาเส้นทางของอัลกอริทึม(2)



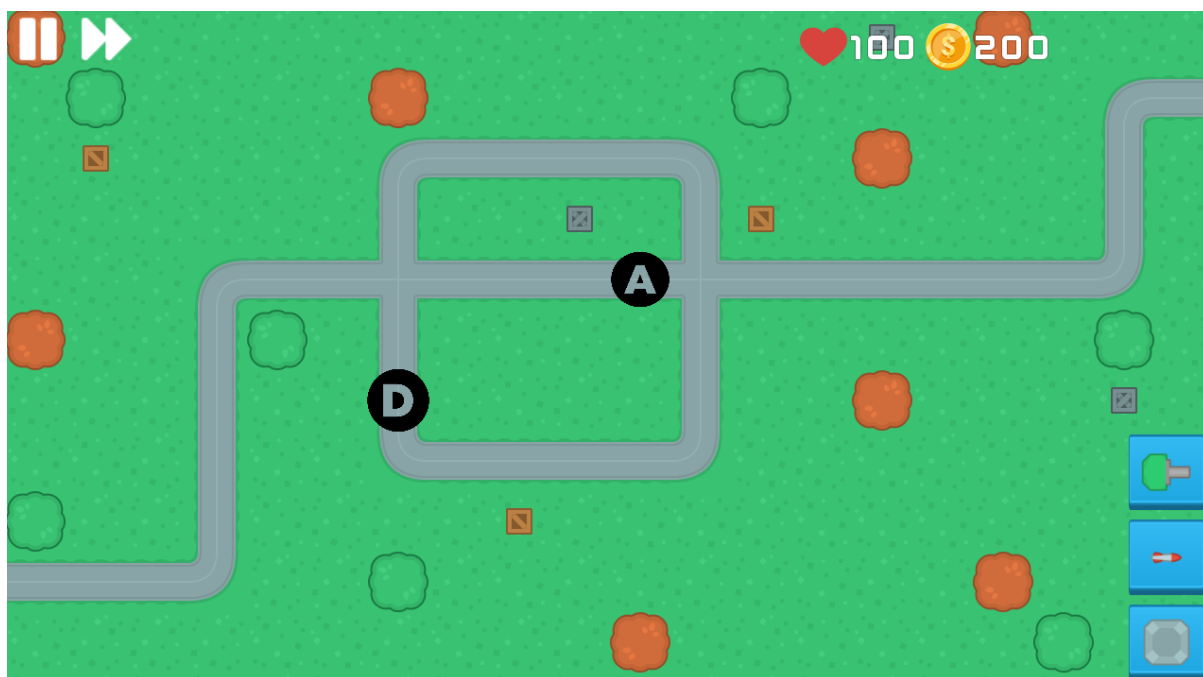
รูปภาพที่ 4.3.6 แสดงการค้นหาเส้นทางของอัลกอริทึม(3)



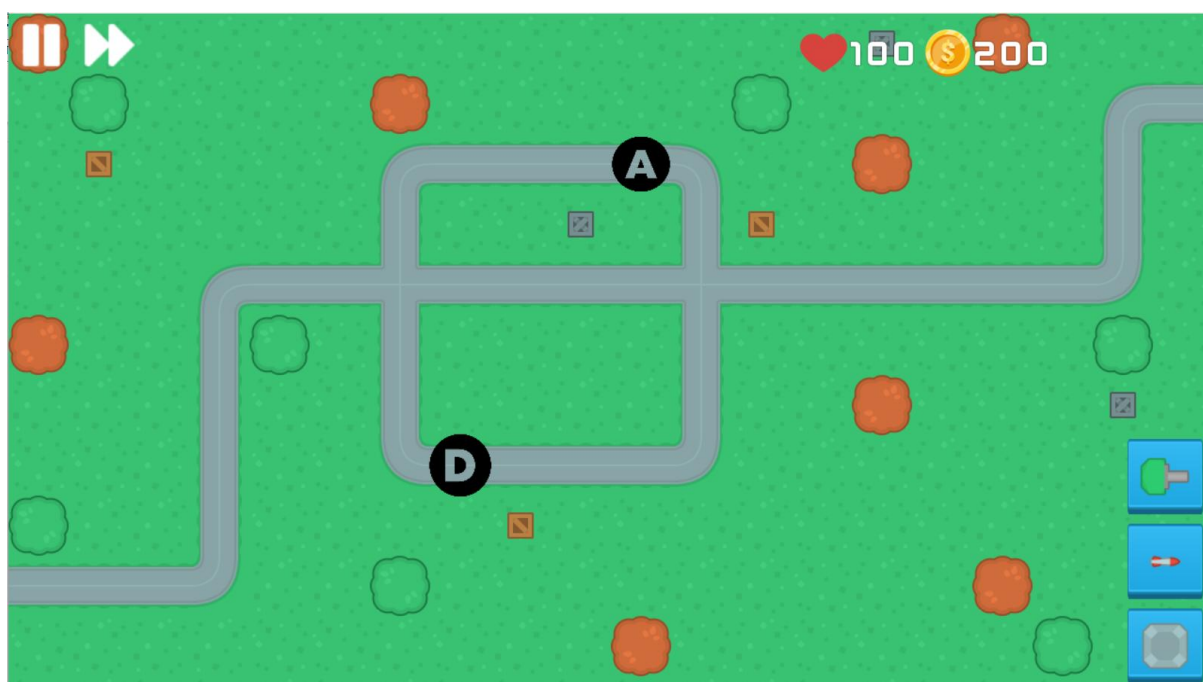
รูปภาพที่ 4.3.7 แสดงการค้นหาเส้นทางของอัลกอริทึม(4)



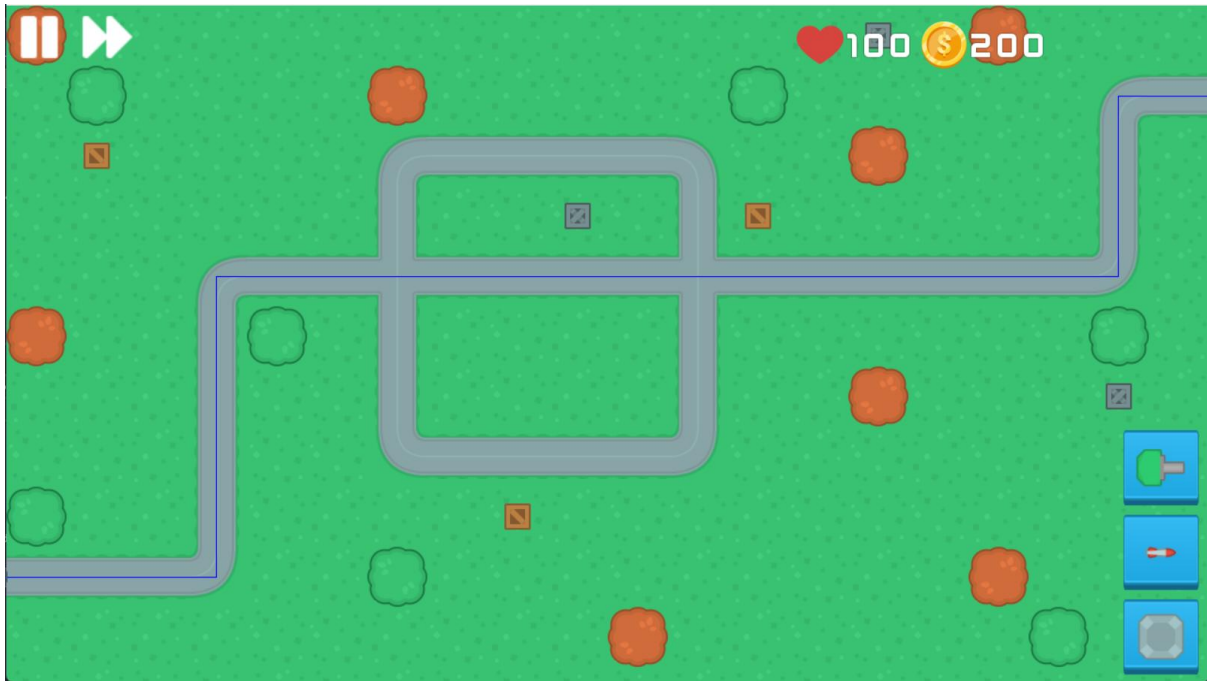
รูปภาพที่ 4.3.8 แสดงการค้นหาเส้นทางของอัลกอริทึม(5)



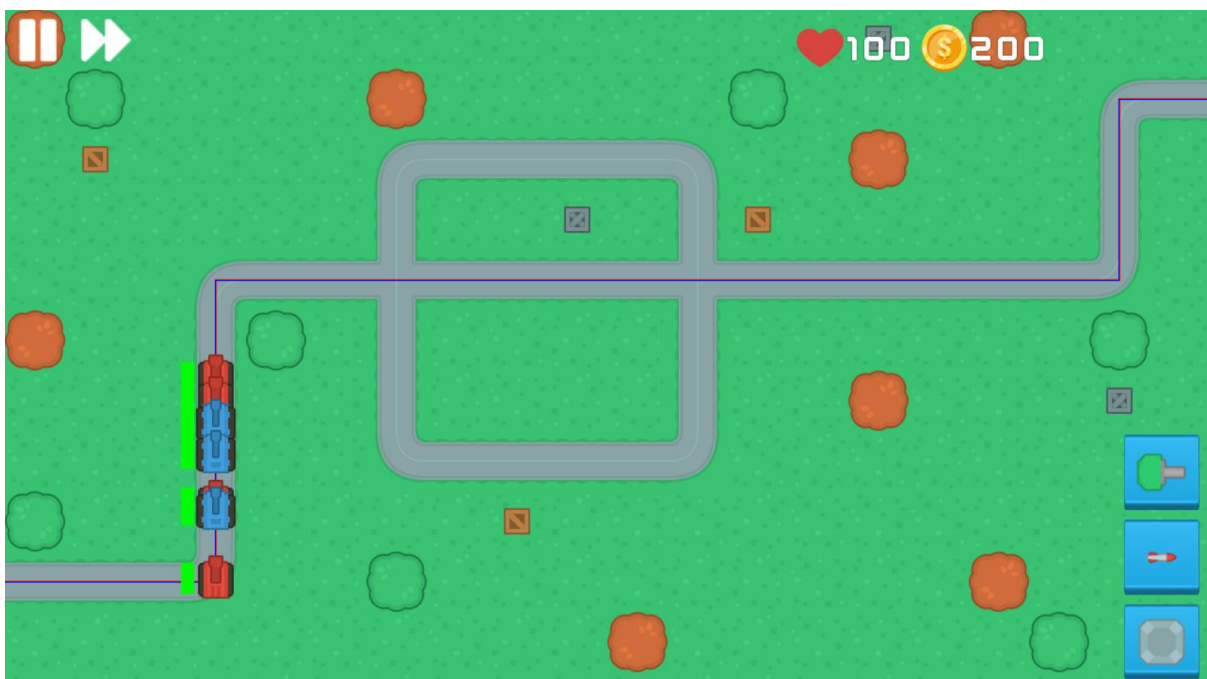
รูปภาพที่ 4.3.9 แสดงการค้นหาเส้นทางของอัลกอริทึม(6)



รูปภาพที่ 4.3.10 แสดงการค้นหาเส้นทางของอัลกอริทึม(7)

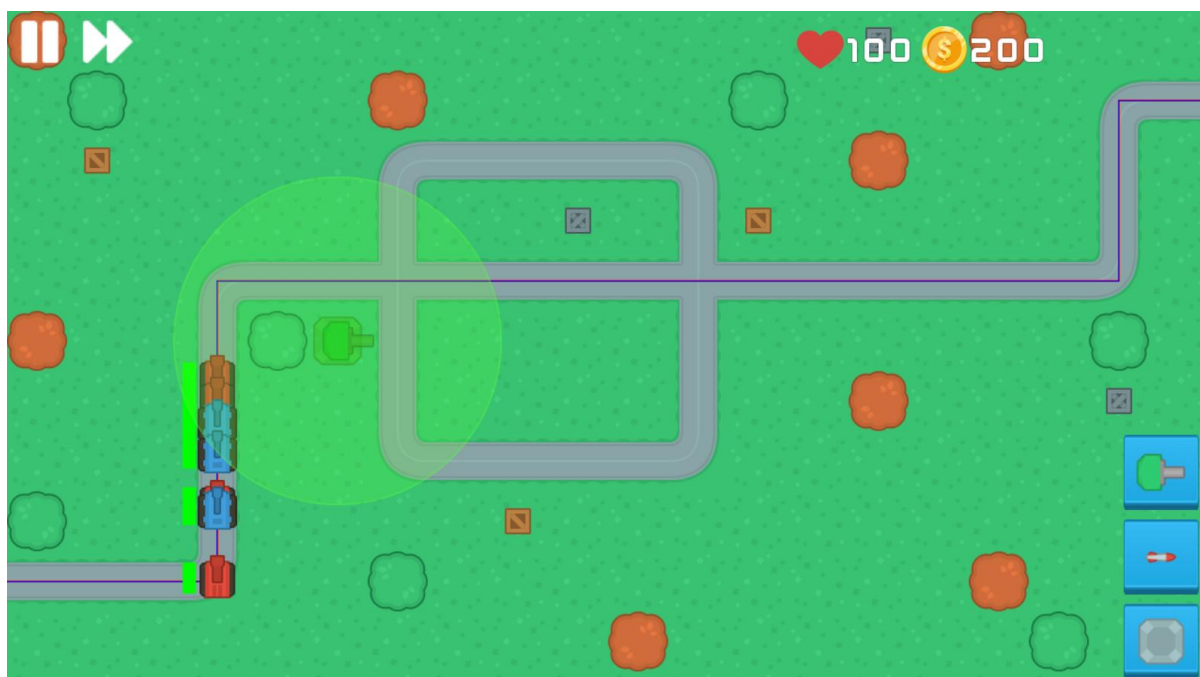


รูปภาพที่ 4.3.13 Dijkstra คือเส้นสีฟ้า



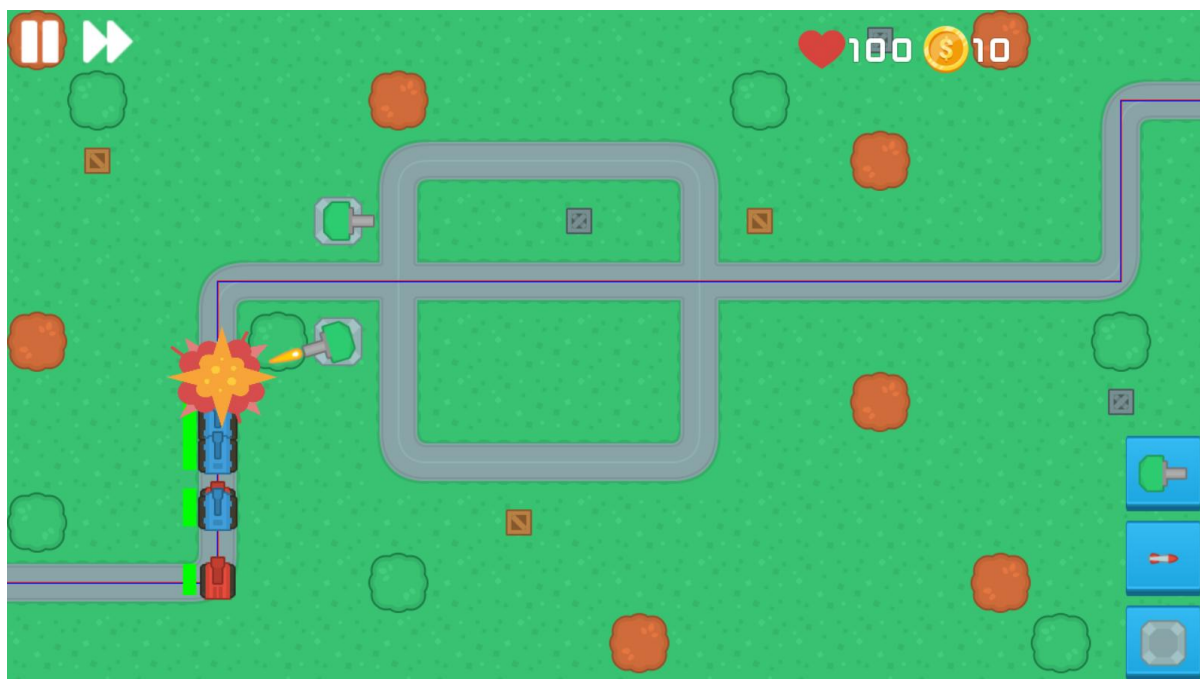
รูปภาพที่ 4.3.14 รถถึงกำลังเดินทางไปตามทางที่อัลกอริทึมค้นหาไว้

เมื่อเส้นทางของทั้ง 2 อัลกอริทึมถูกวาดเสร็จแล้วรถถึงจึงจะเริ่มเคลื่อนที่ โดยที่รถถึงสีแดงจะเป็นรถถึงที่ใช้เส้นทางของ A* ในการเคลื่อนที่ และรถถึงสีฟ้าจะใช้เส้นทางของ Dijkstra ในการเคลื่อนที่ ผู้เล่นจะต้องใช้เงินในการสร้างป้อมปราการเพื่อขวางไม่ให้รถถึงเข้าไปโจมตีบ้านของผู้เล่น

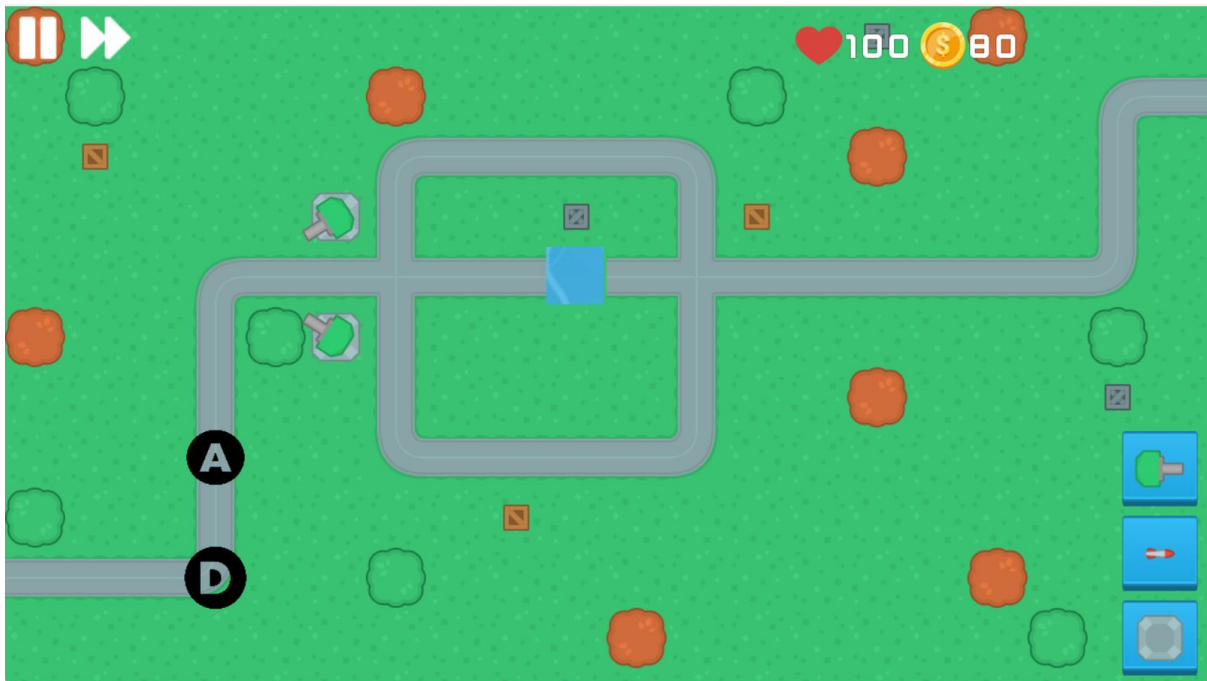


รูปภาพที่ 4.3.15 ผู้เล่นเริ่มวางบ้อมปราการ

บ้อมปราการจะทำการโจมตีรถถังอัตโนมัติ โดยบ้อมปราการจะเล็งไปที่รถถังที่อยู่ใกล้กับบ้อมปราการมากที่สุดก่อนเสมอ เมื่อบ้อมปราการของผู้เล่นสามารถทำลายรถถังได้จะได้รับเงินมา 10 หน่วย

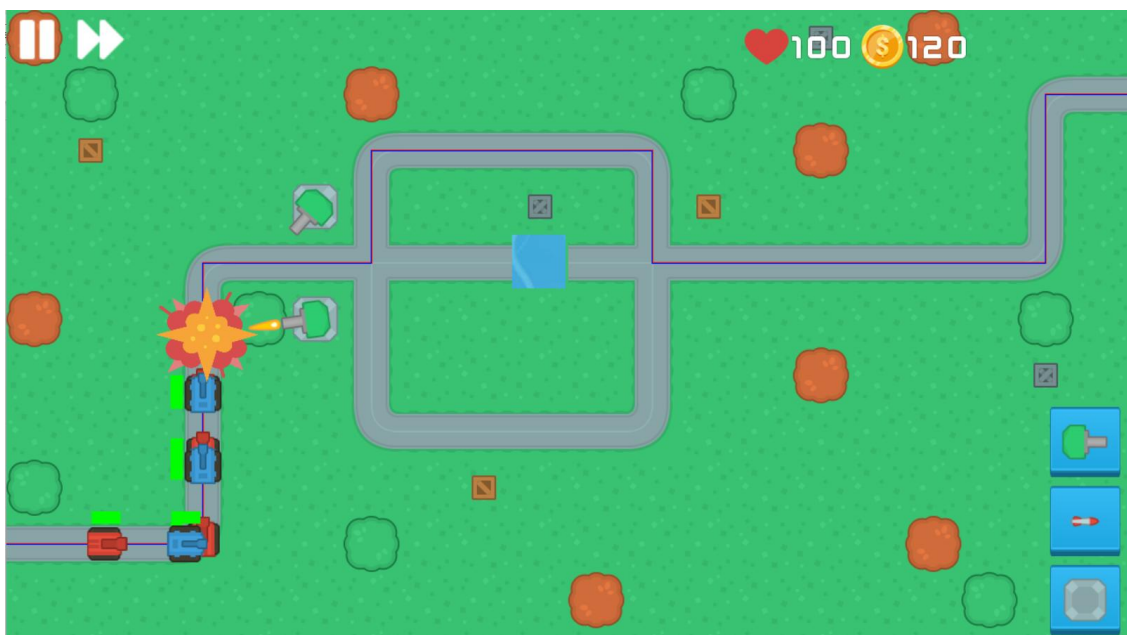


รูปภาพที่ 4.3.16 บ้อมปราการที่ผู้เล่นวางไว้โจมตีรถถัง

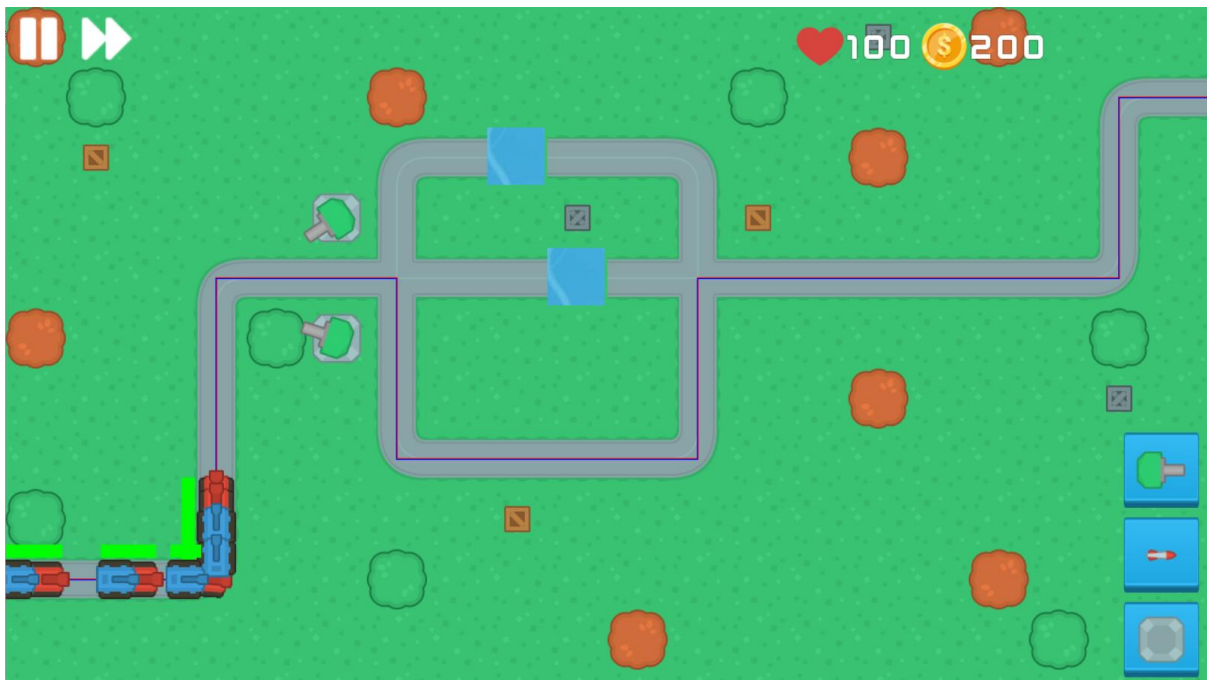


รูปภาพที่ 4.3.17 เริ่มต้นwaveใหม่

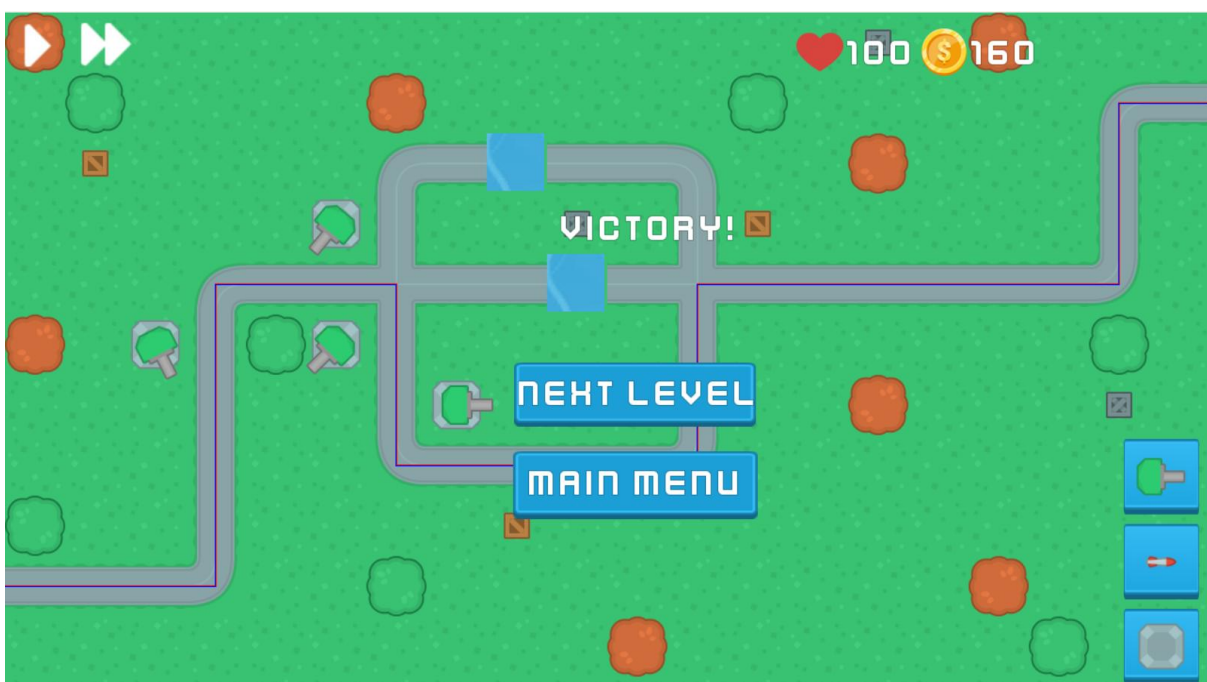
เมื่อผู้เล่นทำลายรถถังจนถึงจำนวนที่กำหนดแล้วจะเกิดน้ำท่วมขึ้นมาปิดเส้นทางที่สั้นที่สุดไว้ในจุดใดจุดหนึ่งแล้วตัวเกมจะเริ่มหาเส้นทางที่สั้นที่สุดใหม่อีกครั้ง หลังจากนั้นรถถังจะเคลื่อนที่ออกมาโจมตีผู้เล่นเป็นwaveที่ 2 หลังจากผู้เล่นทำลายรถถังในwaveที่ 2 หมดจะเริ่มเกิดน้ำท่วมอีกครั้ง ตัวเกมจะเริ่มหาเส้นทางที่สั้นที่สุดใหม่อีกครั้ง แล้วหลังจากหาเส้นทางที่เสร็จก็จะเริ่ม wave ที่ 3



รูปภาพที่ 4.3.18 รถถังในwaveที่ 2 เริ่มเคลื่อนที่ไปโจมตีบ้านของผู้เล่น

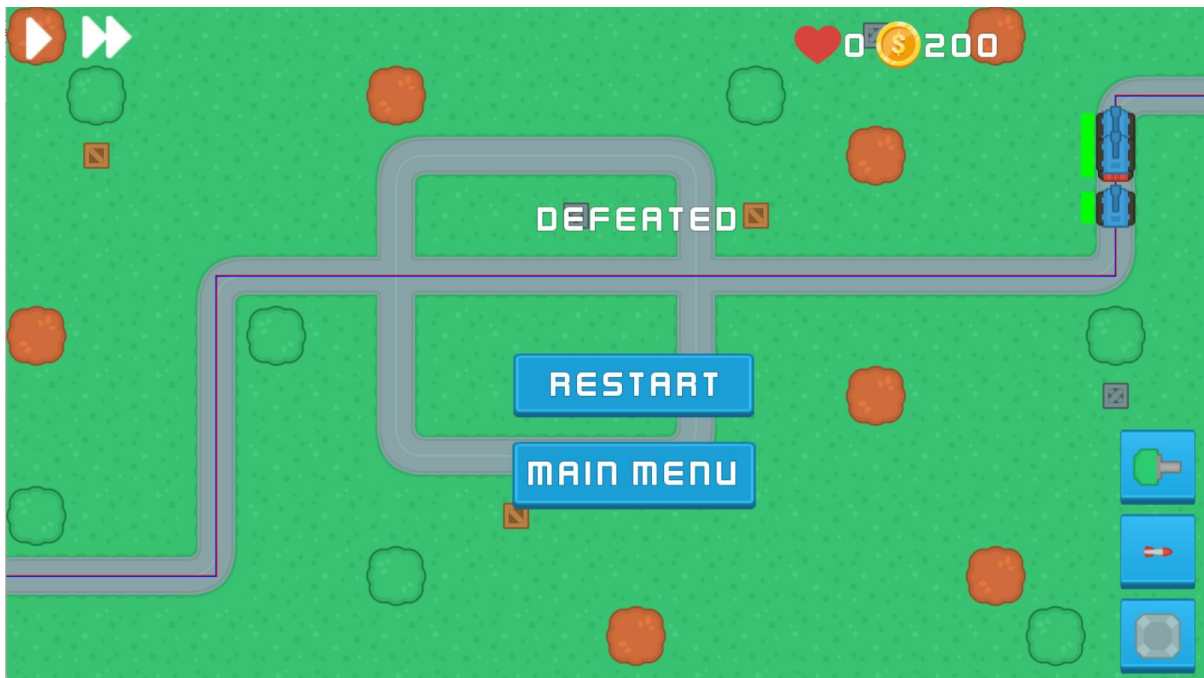


รูปภาพที่ 4.3.19 รถถึงในwaveที่ 3 เริ่มเคลื่อนที่ไปโจมตีบ้านของผู้เล่น



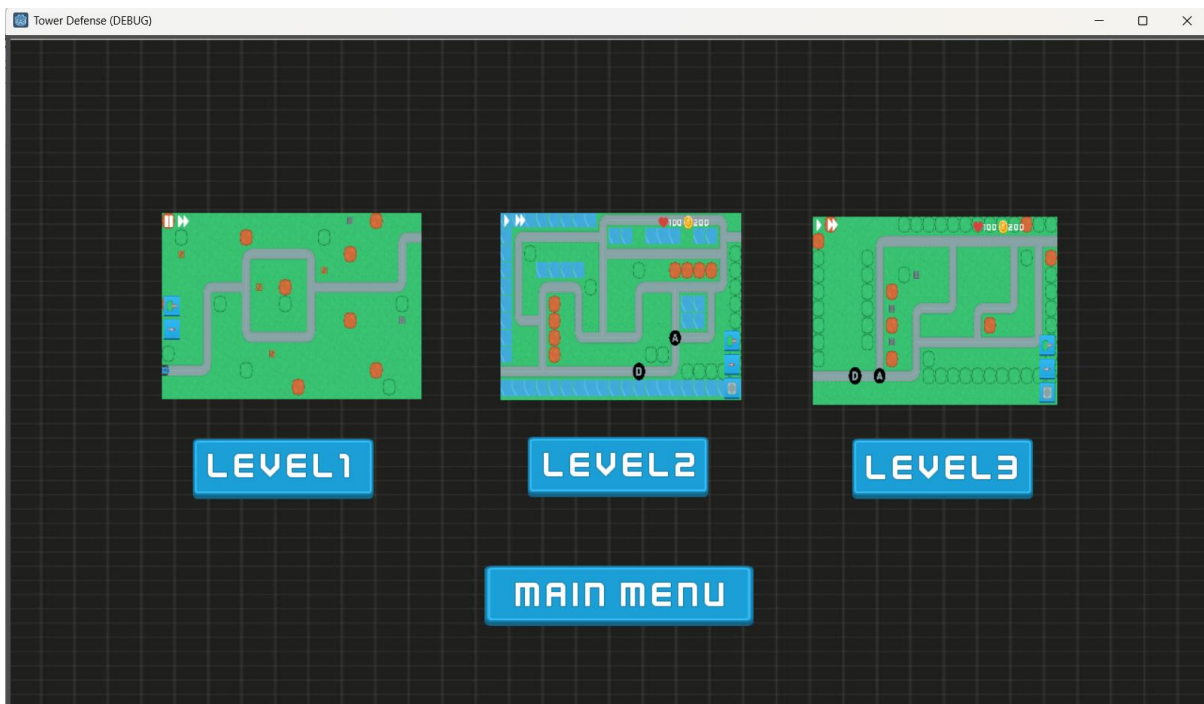
รูปภาพที่ 4.3.20 ผู้เล่นชนะเกม

เมื่อเล่นสามารถทำลายรถถึงได้ทั้งหมดทั้ง 3 wave แล้วผู้เล่นจะชนะในด่านนั้นผู้เล่นจะสามารถเลือกได้ว่าจะไปยังระดับความยากถัดไปหรือจะกลับไปหน้าจอ Main Menu ก็ได้



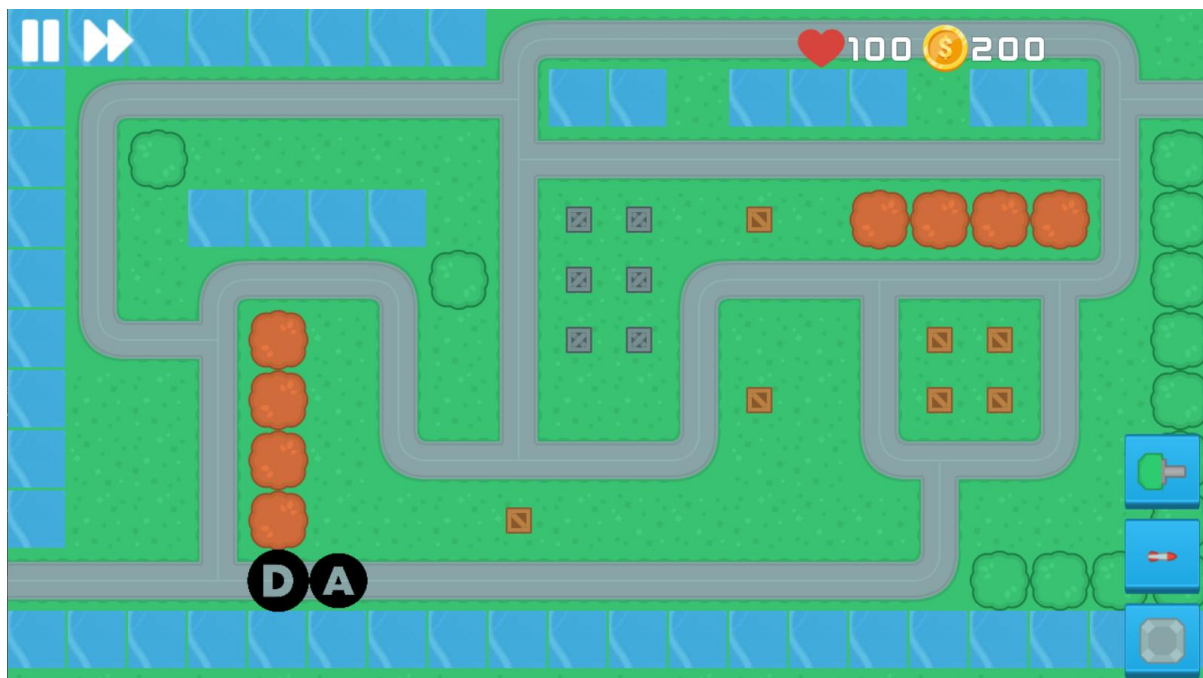
รูปภาพที่ 4.3.21 พลังชีวิตผู้เล่นเหลือ 0

หากผู้เล่นไม่สามารถป้องกันรถไม่ให้ไปโจมตีบ้านเราได้จนพลังชีวิตเราเหลือ 0 ผู้เล่นจะแพ้ในเกมนั้น แล้วผู้เล่นจะสามารถเลือกได้ว่าจะ Restart เกมหรือจะกลับไปหน้าจอ Main Menu

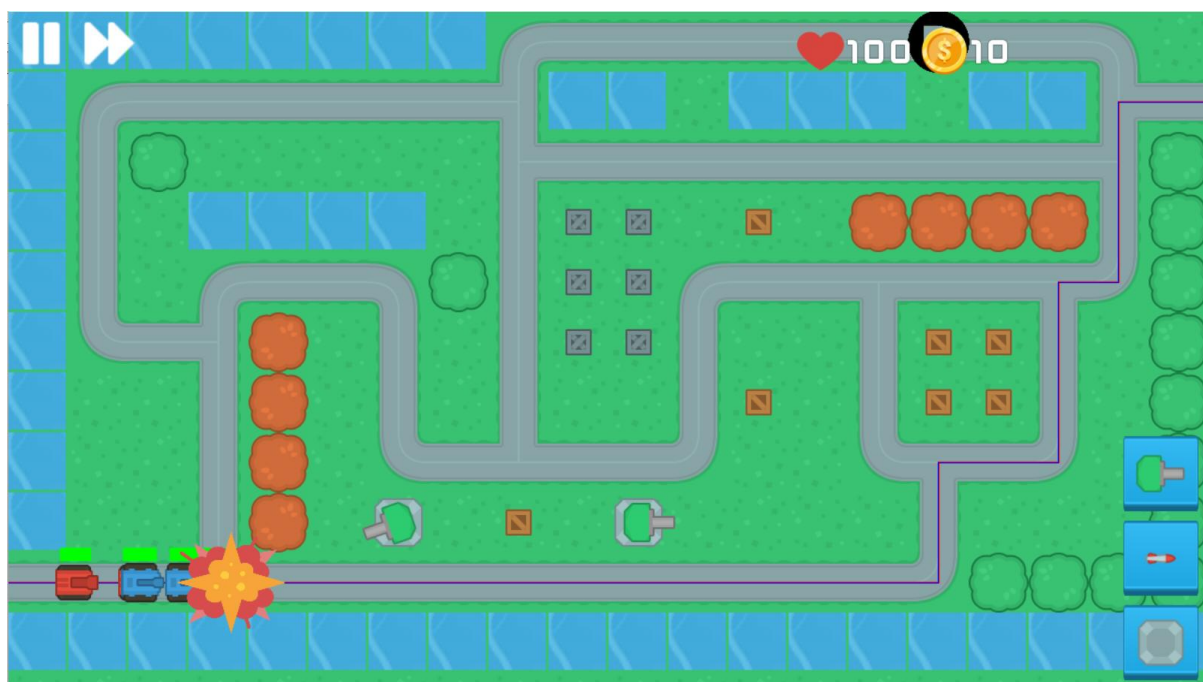


รูปภาพที่ 4.3.22 แสดงให้เห็นหน้าการเลือกระดับความยาก

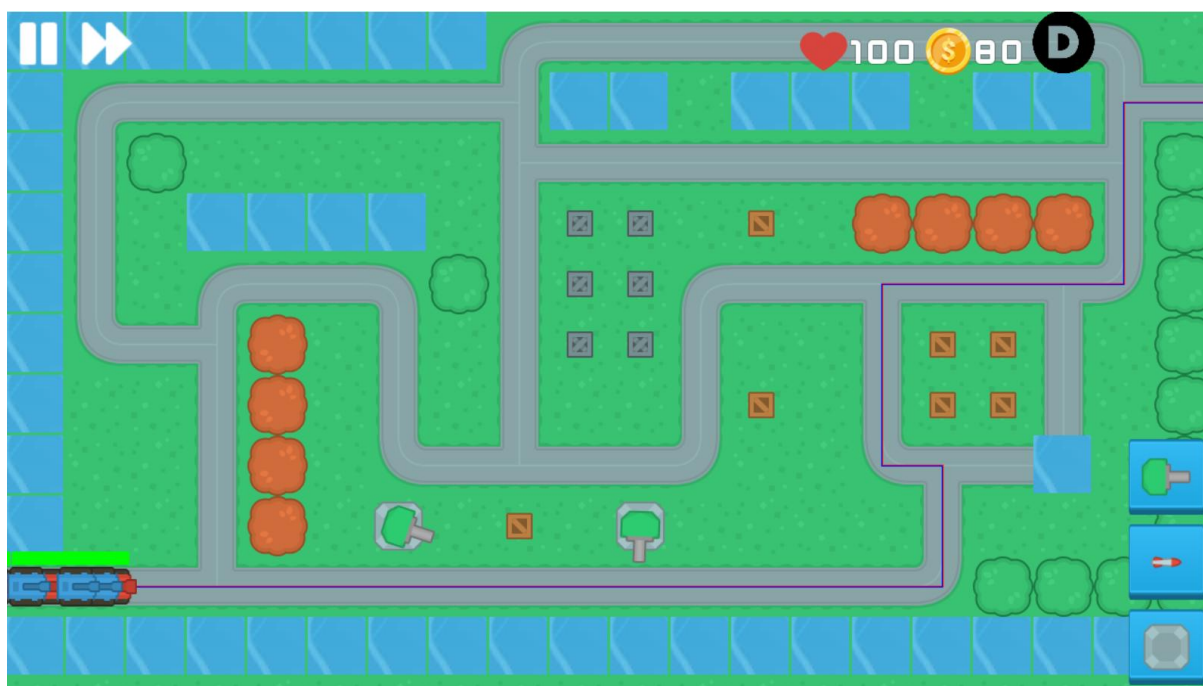
ผู้เล่นจะสามารถเลือกระดับความยากได้ที่ Level โดยจะมีทั้งหมด 3 ระดับ โดยที่แต่ละระดับจะมีเส้นทางไม่เหมือนกันและสภาพแวดล้อมที่ไม่เหมือนกัน



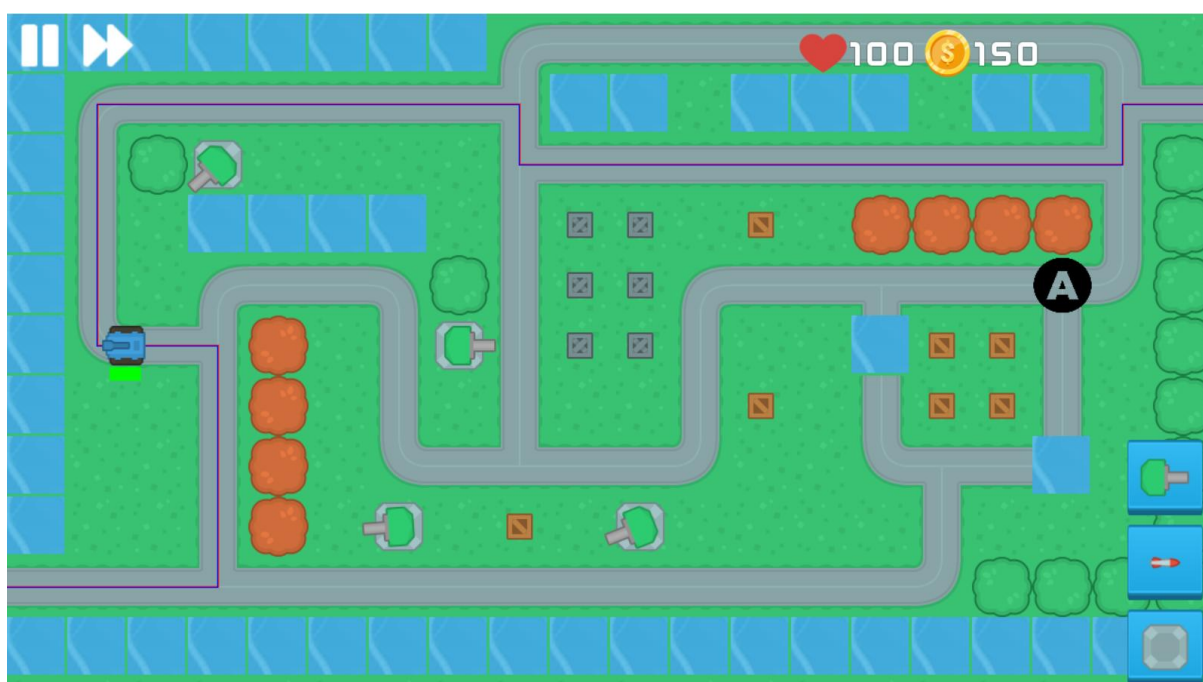
รูปภาพที่ 4.3.23 รูปภาพแผนที่ระดับความยากที่ 2



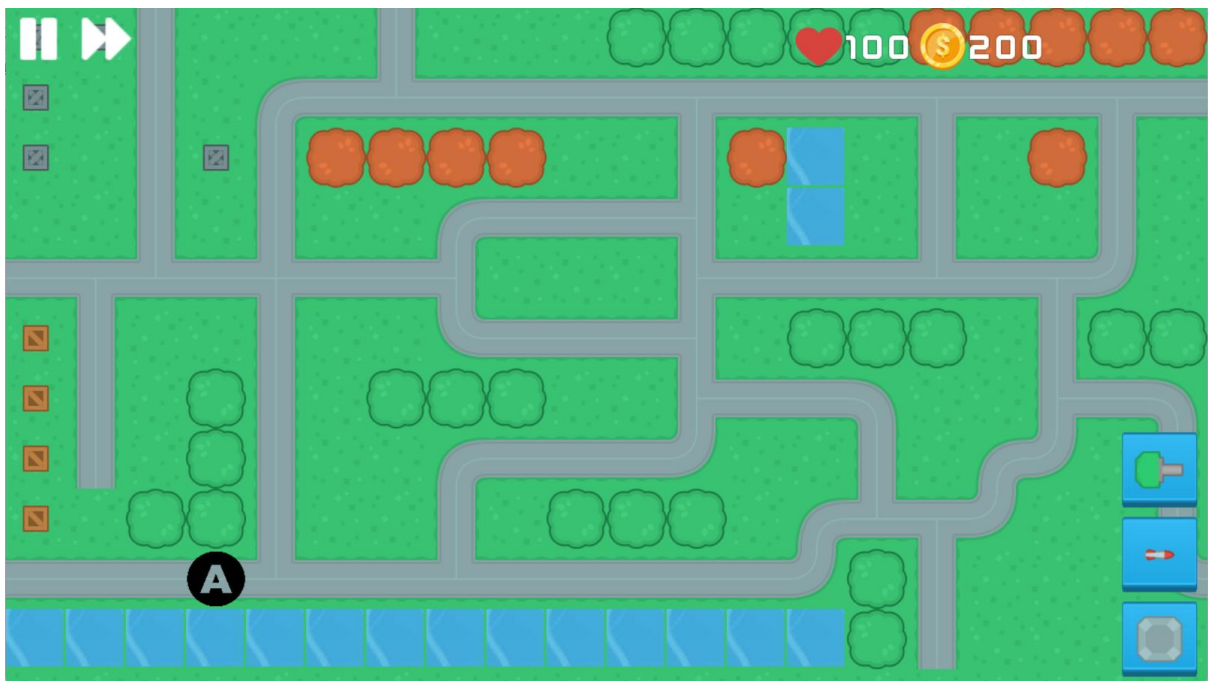
รูปภาพที่ 4.3.24 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 2



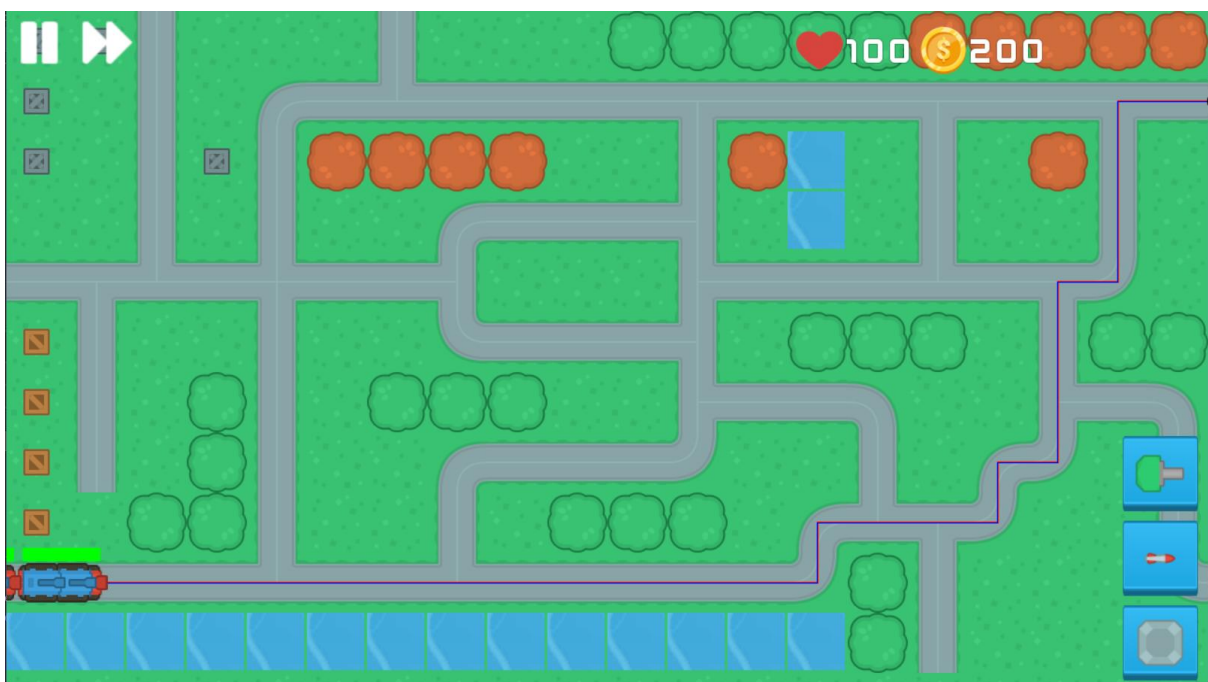
รูปภาพที่ 4.3.25 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 2 หลังจากน้ำท่วมจุดที่ 1



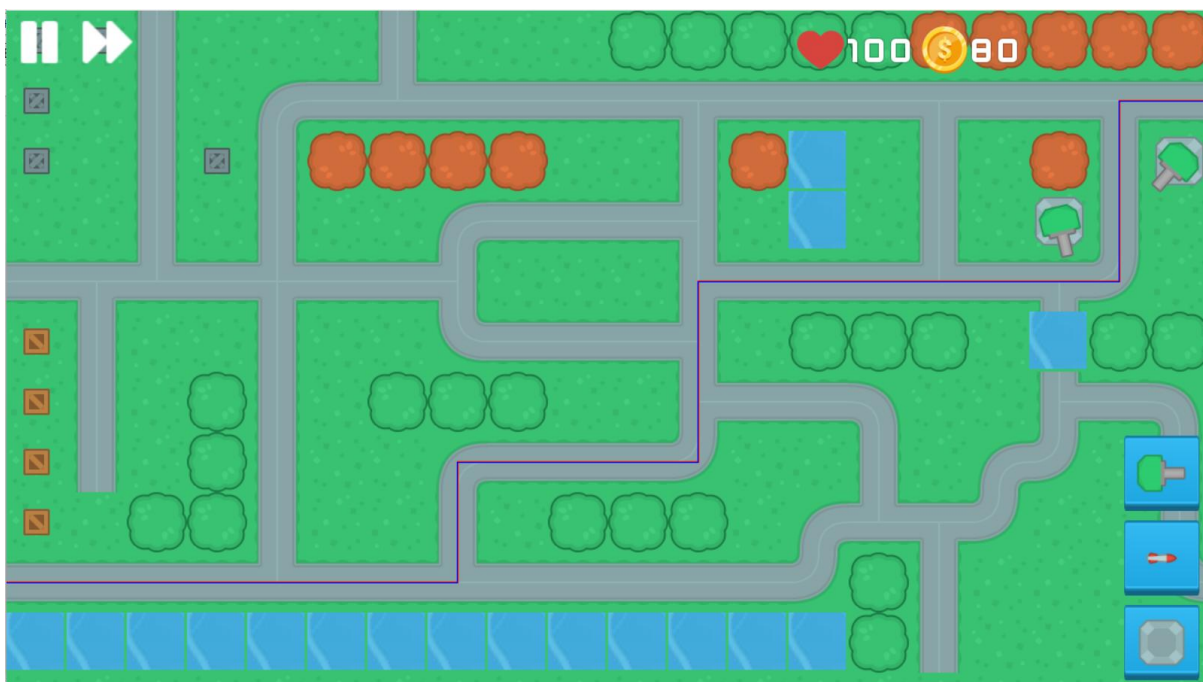
รูปภาพที่ 4.3.26 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 2 หลังจากน้ำท่วมจุดที่ 2



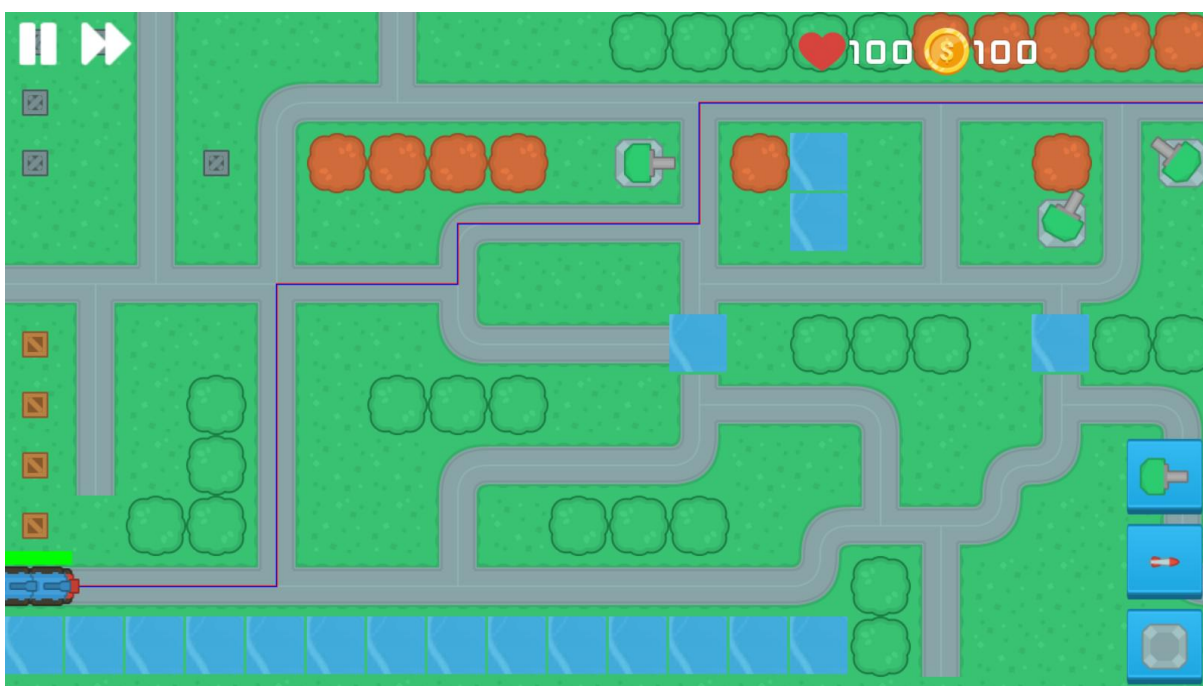
รูปภาพที่ 4.3.27 รูปภาพแผนที่ระดับความยากที่ 3



รูปภาพที่ 4.3.28 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 3



รูปภาพที่ 4.3.29 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 3 หลังจากน้ำท่วมจุดที่ 1



รูปภาพที่ 4.3.30 รูปภาพเส้นทางที่สั้นที่สุดในระดับความยากที่ 3 หลังจากน้ำท่วมจุดที่ 2

4.4 Test case

Test Case ID	Description	input	Expected Result	Result
TCTower1	บ้อมปราการสามารถวางได้บนพื้นหญ้า	ตำแหน่ง x , y	สามารถวางบ้อม ปราการได้	ผ่าน
TCTower2	บ้อมปราการไม่สามารถวางได้บนพื้นที่อื่น นอกจากพื้นหญ้า	ตำแหน่ง x , y	ไม่สามารถวาง บ้อมปราการได้	ผ่าน
TCTower3	บ้อมปราการสามารถสร้างความเสียหายกับ รถถังได้	รถถัง เคลื่อนที่ ผ่าน	สามารถทำความ เสียหายกับรถถัง ได้ตามพลังโจมตี ของบ้อมปราการ	ผ่าน
TCTower4	บ้อมปราการวางลงไปแล้วเงินต้องลดลงตาม ราคาบ้อมปราการ	ตำแหน่ง x , y	เงินลดตามราคา ของบ้อมปราการ	ผ่าน
TCTower5	บ้อมปราการจะวางไม่ได้ถ้าเงินไม่เพียงพอ	ตำแหน่ง x , y	ไม่สามารถวาง บ้อมปราการได้	ผ่าน
CTank1	พลังชีวิตของรถถังลดลงตามความเสียหายที่ เกิดขึ้น	ให้พลัง ชีวิต รถถังอยู่ ที่ 50	พลังชีวิตรถถัง เปลี่ยนสีเป็นสี เหลืองสีแดง ตามลำดับของ พลังชีวิต	ผ่าน
CTank2	รถถังจะต้องถูกทำลายเมื่อพลังชีวิตเหลือ 0	ให้พลัง ชีวิต รถถังอยู่ ที่ 0	รถถังจะระเบิด และถูกทำลาย ออกไปจากเกม	ผ่าน
CTank3	รถถังจะต้องสร้างความเสียหายกับพลังชีวิต ของผู้เล่นเมื่อถึงปลายทาง	ให้รถถังมี พลัง โจมตีที่ 25 หน่วย	พลังชีวิตของผู้เล่น ลดลง 25 หน่วย	ผ่าน
CTank4	เมื่อรถถังถูกทำลายผู้เล่นจะต้องได้รับเงิน จำนวน10หน่วย	ตั้งราคา รถถังที่	ผู้เล่นได้รับเงิน จำนวน 10	ผ่าน

		ถูกทำลาย 10 หน่วย	หน่วยหลังจากที่ รถถึงถูกทำลาย	
TCWave1	เมื่อรถถึงในแต่ละwaveถูกทำลายจนหมดจะมีน้ำท่วมเกิดขึ้นมา	ตำแหน่ง น้ำท่วม x, y	มีน้ำท่วมเกิดขึ้น ตามจุดที่กำหนด	ผ่าน
TCVictory 1	เมื่อรถถึงถูกทำลายจนหมดทุกwaveแล้วจะต้องมีหน้าต่างขึ้นมาเพื่อบอกผู้เล่นว่าชนะเกม	จำนวน รถถึงทุก wave เหลือ 0	มีหน้าต่างแสดง ความยินดีกับผู้เล่นขึ้นมา	ผ่าน
TCDefeat ed1	เมื่อพลังชีวิตของผู้เล่นเหลือ 0	พลังชีวิต ผู้เล่น เหลือ 0	มีหน้าต่างแสดง ให้กับผู้เล่นว่า แพ้ในเกมนั้น	ผ่าน

บทที่ 5

สรุป

5.1 สรุปผลดำเนินงาน

เกมเอาชีวิตรอดจากน้ำท่วม เป็นแอปพลิเคชันเกมที่พัฒนาจากการนำ อัลกอริทึม Pathfinding มาประยุกต์ใช้ในการแสดงหาเส้นทางที่สั้นที่สุด ที่มีการใช้อัลกอริทึม A* และ Dijkstra ผ่านการใช้ Godot เป็นโปรแกรมหลักที่ใช้ในการพัฒนาเกมโดยแอปพลิเคชันเกม สามารถแสดงการเปรียบเทียบการใช้งานและความเหมาะสมสำหรับการนำอัลกอริทึม Pathfinding มาประยุกต์ใช้กับการแสดงพฤติกรรมการหาเส้นทางของตัวละครสมมติในเกม เราจะเห็นได้ว่าการเดินทางไปสำรวจเส้นทางต่างๆของอัลกอริทึมทั้ง 2 ตัวนั้นมีความแตกต่างกันอย่างชัดเจน Dijkstra จะมีการเดินทางไปสำรวจเส้นทางทุกเส้นทางที่เป็นไปได้ ในขณะที่ A* ที่มี heuristic function เข้ามาช่วยในการคาดการณ์เส้นทางจึงจะเห็นได้ว่า A* จะใช้การสำรวจเส้นทางที่น้อยกว่าและสามารถวาดเส้นทางให้รถถึงเคลื่อนที่ได้เร็วกว่าอย่างเห็นได้ชัด รวมถึงการใช้ทรัพยากรในการสำรวจจะน้อยกว่า Dijkstra ไปด้วย

5.2 ข้อเสนอแนะ

เกมยังขาดการแสดงผลที่ชัดเจนกับผู้เล่น การเพิ่มการแสดงผลรายละเอียดต่างๆให้กับตัวเกม การเพิ่มระบบแจ้งเตือนเมื่อรถถึงเข้าใกล้บ้านของผู้เล่น การปรับปรุงกราฟิกของพื้นผิวน้ำให้มีรายละเอียดชัดเจนและสวยงามมากยิ่งขึ้น มีการบอกรายละเอียดของสิ่งปลูกสร้าง ราคาของสิ่งปลูกสร้าง ราคาเท่าไร พลังโจมตีของบ้อมปราการเท่าไร หากผู้เล่นมีเงินไม่พอจะต้องแสดงออกมาให้ผู้เล่นเห็นว่าไม่มีเงินเพียงพอที่จะซื้อสิ่งปลูกสร้างนั้นๆ ความแตกต่างของแต่ละบ้อมปราการว่าต่างกันอย่างไรบ้าง มีการนำอัลกอริทึมค้นหาเส้นทางอื่นๆมาใช้เพิ่มเติม เพื่อเพิ่มความหลากหลายให้กับตัวเกม ตัวเกมมีการแสดงการค้นหาเส้นทางของอัลกอริทึมที่เร็วเกินไปทำให้ผู้เล่นไม่สามารถเรียนรู้ได้ว่าจะเกิดขึ้นกับการทำงานของอัลกอริทึม ควรทำให้การแสดงผลการค้นหาเส้นทางของอัลกอริทึมแสดงออกมาช้ากว่านี้ เพื่อให้ผู้เล่นได้เห็นอย่างชัดเจนถึงการค้นหาเส้นทางของอัลกอริทึม

รายการอ้างอิง

น้ำเพชร รอดประเสริฐ. (2022). ทำความเข้าใจ Optimization ศาสตร์ที่ใช้แก้ปัญหาเพื่อให้ได้ค่าที่เหมาะสมที่สุด. <https://bigdataexperience.org/ทำความเข้าใจ-optimization-ศาสตร์ที่/>

สมเกียรติ สุนทรสวัสดิ์. (2560). ทฤษฎีกราฟเบื้องต้น. <https://www.scimath.org/lesson-mathematics/item/7334-2017-06-17-14-37-32>

Ade Syahputra, Budi Arifitama, Ketut bayu yogha Bintoro and Silvester Dian Handy

Permana. (2018). Comparative Analysis of Pathfinding Algorithms A *, Dijkstra, and BFS on Maze Runner Game.

https://www.researchgate.net/publication/325368698_Comparative_Analysis_of_Pathfinding_Algorithms_A_Dijkstra_and_BFS_on_Maze_Runner_Gam

Arknights Gameplay. (2020). จากWikipedia.

https://en.wikipedia.org/wiki/Arknights#/media/File:Arknights_gameplay.png

Data Structures Tutorial. (n.d.). สืบค้นจาก <https://www.geeksforgeeks.org/data-structures/>

Don ross. (1997). Game Theory. สืบค้นจาก <https://plato.stanford.edu/entries/game-theory/>

Elisha Deogracias. (2018). What's your emergency? 911 Operator heads to Switch this month
สืบค้นจาก <https://gamingtrend.com/news/whats-your-emergency-911-operator-heads-to-switch-this-month/>

Fatima Rubio. (2023). The 5 Most Powerful Pathfinding Algorithm.

<https://www.graphable.ai/blog/pathfinding-algorithms/>

Hybasis H.urna. (2020). Pathfinding algorithms: the four Pillars.

<https://medium.com/@urna.hybasis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b>

Kristen M. Meiburger and Filippo Molinari. (2018). Automated localization and segmentation techniques for B-mode ultrasound images: A review.

<https://www.sciencedirect.com/topics/computer-science/heuristic-function>

Masterclass. (2021). Tower Defense Game Genre: 6 Characteristics of TD Games.

<https://www.masterclass.com/articles/tower-defense-game-video-game-guide>

Walter Dean. (2015). Computational Complexity Theory. สืบค้นจาก

<https://plato.stanford.edu/entries/computational-complexity/>