พัฒนาเกมที่สามารถสร้างฉากแบบสุ่มอัตโนมัติด้วยยูนิตี้

GAME DEVELOPMENT WITH RANDOMLY GENERATED SCENE VIA UNITY

พีร์ บุญมาเลิศ

วสุต พงษ์สัตยาพิพัฒน์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

พัฒนาเกมที่สามารถสร้างฉากแบบสุ่มอัตโนมัติด้วยยูนิตี้

GAME DEVELOPMENT WITH RANDOMLY GENERATED SCENE VIA UNITY

พีร์ บุญมาเลิศ
วสุต พงษ์สัตยาพิพัฒน์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

GAME DEVELOPMENT WITH RANDOMLY GENERATED SCENE VIA UNITY

PEE BOONMALERT
VASUT PONGSATTAYAPIPAT

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)
DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2015

หัวข้อปัญหาพิเศษ	พัฒนาเกมที่สามารถสร้างฉากแบบสุ่มอัตโนมัติด้วยยูนิตี้ Game Development With Randomly Generated Scene Via Unity
ชื่อนักศึกษา	นายพีร์ บุญมาเลิศ รหัสนักศึกษา 55050409 นายวสุต พงษ์สัตยาพิพัฒน์ รหัสนักศึกษา 55050453
ปริญญา	วิทยาศาสตร์บัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2558
อาจารย์ที่ปรึกษา	ผศ.ธีระ ศิริธีรากุล

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ประจำปีการศึกษา 2558

คณะกรรมการสอบ	ลายมือชื่อ
อ.สันธนะ อู่อุดมยิ่ง ประธานกรรมการ	
ดร.สายชล ใจเย็น กรรมการ	
ผศ.ธีระ ศิริธีรากุล กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อปัญหาพิเศษ พัฒนาเกมที่สามารถสร้างฉากแบบสุ่มอัตโนมัติด้วยยูนิตี้
Game Development With Randomly Generated Scene Via Unity

ชื่อนักศึกษา นายพีร์ บุญมาเลิศ รหัสนักศึกษา 55050409
นายวสุต พงษ์สัตยาพิพัฒน์ รหัสนักศึกษา 55050453

ปริญญา วิทยาศาสตร์บัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา วิทยาการคอมพิวเตอร์
ปีการศึกษา 2558
อาจารย์ที่ปรึกษา ผศ.ธีระ ศิริธีรากุล

บทคัดย่อ

ในโปรเจคนี้นำเสนอการพัฒนาเกมด้วยยูนิตี้เอนจิ้น โดยใช้การสุ่มฉากมาเกี่ยวข้อง เพื่อให้เกมมีความแปลกใหม่ ไม่น่าเบื่อ การใช้ฉากสุ่มนั้น จะส่งผลต่อดีไซน์ของเลเวล และสามารถนำไปประยุกต์กับเกมอื่น ๆ ที่อาจมีความซับซ้อนและมีระบบที่หลากหลาย กล่าวคือในโปรเจคนี้มุ่งหวังให้งานนี้เป็นแนวคิดในการพัฒนาเกมรูปแบบใหม่ เกมที่พัฒนานั้นมีการออกแบบอย่างเรียบง่าย โดยคำนึงถึงการสื่อสารระหว่างตัวเกมกับผู้เล่น สีสันต่าง ๆ ที่เกิดขึ้นในแต่ละฉากของเกมล้วนมีความหมาย ทำให้ผู้เล่นมีอารมณ์ร่วมไปกับเกม นอกจากนี้ระบบต่าง ๆ ในเกมยังเป็นตัวอย่างของการนำทฤษฎีต่าง ๆ มาใช้ร่วมกัน อาทิเช่น เซลลูลาร์ออโตมาตา ซึ่งเป็นแบบจำลองการสุ่มที่ไม่ต่อเนื่องอย่างหนึ่งที่ถูกใช้อย่างแพร่หลาย ทั้งในทฤษฎีของการคำนวณ ฟิสิกส์ คณิตศาสตร์ วิทยาศาสตร์ ชีววิทยา เป็นต้น นอกจากนี้นำเสนอเทคนิคดีไซน์เลเวลที่ใช้ในการพัฒนาเกม เพื่อให้แต่ละฉากมีความยากง่ายที่เหมาะสม และในอนาคตอาจนำเทคนิค มาร์ชชิ่งสแควร์ เข้ามาใช้ โดยเทคนิคทั้งหมดจะถูกนำมาอธิบายอย่างง่ายเพื่อให้เหมาะสำหรับผู้ที่ต้องการศึกษา เนื่องจากผู้พัฒนาเลือกที่จะพัฒนาเกมในรูปแบบที่สามารถทำความเข้าใจได้ง่าย แม้ผู้ที่พึ่งเคยเริ่มใช้ยูนิตี้ ก็สามารถทำความเข้าใจจากตัวอย่างสคริปที่ผู้จัดทำเขียนได้อย่างไม่ยาก

คำสำคัญ : เซลลูลาร์ออโตมาตา เกมสองมิติ ภาษาซีชาร์ป ยูนิตี้เอนจิ้น สุ่มสร้างฉาก

Title	Game Development With Randomly Generated Scene Via Unity
Students	Mr. Pee Boonmalert Student ID 55050409
	Mr. Vasut Pongsattayapipat Student ID 55050453

Degree	Bachelor of Science (Computer Science)
Department	Computer Science
Faculty	Science
University	King Mongkut's Institute of Technology Ladkrabang (KMITL)
Academic Year	2015
Advisor	Asst.Prof.Teera Siriteerakul

Abstract

In this project we propose game development via Unity engine with an implementation of scene randomization, in which will affect the level design and such implementation can be use as an underlying component for many different kind of game project. By introducing randomization, not only the system side of the game, but also the communication between the player and the game which will in turn create flow state for the player who play the game. Each individual feedback that the game sends back to the player is meaningful. We also display the example implementation of theory into the build including Cellular Automata which is a controlled randomization theory that has been used widely in many different fields. We also display the thought process into the game level design and in the future may applied the Marching Square theory into the build. All of this will be presented in an easy to understand steps in which beginner to Unity will be able to follow.

Keywords : Cellular Automata, 2D Game, C# language, Unity Game Engine, Procedural Generated Level

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จไปได้ด้วยดี ด้วยความกรุณาของอาจารย์ที่ปรึกษาโครงงาน ผศ.ธีระ ศิริธีรากุล ที่ได้สละเวลาให้ความช่วยเหลือ ให้คำแนะนำที่มีค่ายิ่งในการปรับปรุงแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้น ตลอดจนให้ความรู้และประสบการณ์ที่ดีในการทำงาน

ขอกราบขอบพระคุณคณาจารย์ คณะวิทยาศาสตร์ ภาควิชาวิทยาการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกท่าน ที่ได้ประสิทธิ์ประสาทวิชาให้แก่ข้าพเจ้า คอยให้คำปรึกษาและชี้แนะแนวทางในการแก้ไขปัญหาต่าง ๆ ให้สำเร็จลุล่วงไปได้

ขอกราบขอบพระคุณบิดามารดาและครอบครัวของข้าพเจ้าที่ให้กำลังใจและให้การสนับสนุนทุก ๆ เรื่อง ทำให้ข้าพเจ้ามีกำลังใจ ทำให้โครงงานสำเร็จลุล่วงได้ด้วยดี

คุณค่าและประโยชน์อันพึงมาจากโครงงานนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นายพีร์ บุญมาเลิศ
นายวสุต พงษ์สัตยาพิพัฒน์

สารบัญ

หน้า

สารบัญรูป

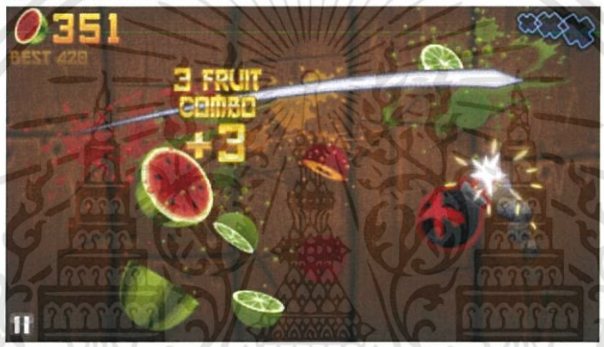
หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เกมอาเขต (Arcade Game) จัดเป็นเกมแนวแอคชั่น (Action Game) ประเภทหนึ่ง ที่เน้นการบังคับทิศทางของตัวละครให้ผ่านแต่ละด่านไปได้ โดยใช้เวลาไม่นานในการจบเกม เน้นความเรียบง่าย ใช้หลักจิตวิทยาในการจูงใจให้ผู้เล่นกลับมาเล่นใหม่โดยบันทึกคะแนนสูงสุดไว้เท่านั้น เพื่อให้ผู้เล่นทำคะแนนได้มากขึ้น ในปัจจุบันได้มีการพัฒนาการเล่นจากเครื่องเกมตู้มาสู่อุปกรณ์เคลื่อนที่อย่างสมาร์ทโฟนในระบบปฏิบัติการต่างๆ เช่น เกมฟรุทนินจา (Fruit Ninja) ที่อาศัยจังหวะ ความว่องไวในการเล่น หรือเกมกราดิอุส (Gradius) เป็นเกมแนวยานอวกาศยิงศัตรูและเก็บไอเทมไปเรื่อยๆ จนเจอบอส และทำลายบอสให้ได้เพื่อผ่านด่าน



รูปที่ 1.1 เกม Fruit Ninja

รูปที่ 1.2 เกม Gradius

เกมอาเขตเหล่านี้เป็นเกมที่เล่นสนุกเพลิดเพลิน เพราะอาศัยทักษะและสมาธิของผู้เล่น ที่ต้องจดจ่ออยู่กับการหลบเลี่ยงศัตรู และเก็บของในเกมอยู่ตลอดเวลา สามารถแข่งกันกับเพื่อนได้โดยวัดกันที่คะแนนสูงสุด ตัวเกมประเภทนี้นั้นต้องใช้ความคิดสร้างสรรค์ตัวละคร ด่านต่างๆ ไอเทม บอส เพราะระดับความยากง่ายขึ้นอยู่กับปัจจัยเหล่านั้นทั้งสิ้น

จากที่กล่าวมาข้างต้น เนื่องจากวงการเกมในปัจจุบันกำลังเติบโต ทางผู้จัดทำจึงพัฒนาเกมขึ้นมาเพื่อตอบสนองต่อความต้องการของวงการในปัจจุบันในแบบของกลุ่มผู้จัดทำเอง ในรูปแบบเกมอาเขตที่สามารถสร้างฉากแบบสุ่มออกมาได้ตั้งแต่เริ่มเกมแบบไม่ซ้ำติดต่อกัน และคาดเดาระดับความยากไม่ได้ตั้งแต่เริ่มต้น เป็นเกมอาเขตประเภท Shoot 'em up' เน้นการควบคุมตัวละครเพื่อยิงทำลายคู่ต่อสู้จากระยะที่ไกลออกไป ทำให้ผู้เล่นตื่นเต้นตั้งแต่เข้าเกม โดยทางผู้จัดทำตั้งใจศึกษาการสร้างเกมนี้ด้วยยูนิตี้เอนจิ้น (Unity Engine) เพราะ Unity มีชุมชนผู้พัฒนาจากทั่วโลก ที่พร้อมให้คำปรึกษาปัญหาที่เกิดขึ้นภายในการพัฒนาหรือศึกษาจากกรณีตัวอย่างจากกรณีที่เคยเกิดขึ้นได้โดยง่าย

1.2 จุดมุ่งหมายและวัตถุประสงค์ของการศึกษา

1) เพื่อศึกษาและพัฒนาเกมโดยใช้ Unity Engine
2) เพื่อศึกษาโครงสร้างและขั้นตอนในการพัฒนาเกมคอมพิวเตอร์
3) เพื่อพัฒนาเกมอาเขตให้มีความแปลกใหม่ต่อผู้เล่นที่เคยเล่นแนวเกมแบบเก่า และสามารถเข้าใจได้ง่ายสำหรับผู้เล่นใหม่มากขึ้น
4) เพื่อออกแบบและพัฒนาเกมไปสู่แพลตฟอร์มอื่นในอนาคตได้สะดวกขึ้นโดยข้อดีของ Unity
   ขยายกลุ่มผู้เล่นได้กว้างขึ้น
5) เพื่อต่อยอด Game Mechanic ที่มีอยู่แล้วให้ดีขึ้น
6) เพื่อแสดงให้เห็นว่าการพัฒนาเกมสามารถเข้าถึงได้ง่ายขึ้น

1.3 ขอบเขตของการพัฒนาระบบ

ระบบที่พัฒนาคือเกมบนคอมพิวเตอร์ทั่วไป ที่มีการสุ่มด่าน สุ่มไอเทมต่างๆ รวมถึงระดับความยากของเกม โดยผู้เล่นต้องต่อสู้กับศัตรูจนบรรลุเป้าหมายที่กำหนดไว้ในแต่ละด่านของเกม ซึ่งมีคุณลักษณะดังนี้

1) พัฒนาในส่วนของไอเทมที่มีคุณลักษณะเพิ่มความสามารถให้กับผู้เล่นได้ต่างกันไป ทำให้ผู้เล่นสามารถชนะศัตรูได้เร็วขึ้น ผ่านด่านไวขึ้น
2) พัฒนาการสุ่มด่าน ไอเทม ศัตรู อย่างละ 3 รูปแบบ
3) ใช้เม้าส์และคีย์บอร์ดในการเล่นเกม

1.4 แผนการดำเนินงาน

ภาคเรียนที่ 1

- ศึกษาทฤษฎีเกมแนว Space Shooter กฎและกติกาต่างๆ
- ศึกษาการใช้งาน Unity Engine รวมถึงภาษา C#
- ศึกษาการออกแบบส่วนติดต่อกับผู้ใช้ และรูปแบบเกม

- พัฒนาเกมต้นแบบ

ภาคเรียนที่ 2
- นำเกมต้นแบบมาศึกษาและแก้ไขข้อผิดพลาดต่างๆ
- พัฒนาเกมต้นแบบ
- ทดสอบจริงและแก้ไขข้อผิดพลาด

1.5 ขั้นตอนการศึกษา

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา จุดมุ่งหมายของการพัฒนาและวัตถุประสงค์ของการศึกษา ขอบเขตของการศึกษา ขั้นตอนการศึกษา และประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 กล่าวถึงทฤษฎีและเทคโนโลยีที่ใช้ในการพัฒนา ประกอบด้วย Unity Engine, Unity 2D Space Shooter, Unity 2D Roguelike, The Art of Screen Shake, โปรแกรมสร้างเสียงเอฟเฟ็ค SFXR, Flow Theory, Enemy Speed Control on Shoot em' Up Game และความเข้าใจในการออกแบบเกม

บทที่ 3 กล่าวถึงการวิเคราะห์และออกแบบระบบ ประกอบไปด้วยกฎและกติกาการเล่นเกม การวิเคราะห์และออกแบบเกม การออกแบบระบบ

บทที่ 4 กล่าวถึงการพัฒนาระบบบ ประกอบด้วยการพัฒนาส่วนของเมนูตัวเลือก, การพัฒนาไอเทม, พัฒนาระบบต่างๆ ภายในเกม

บทที่ 5 กล่าวถึงสรุปผลการพัฒนาเกม และข้อเสนอแนะ ประกอบด้วยสรุปผลการพัฒนา, ปัญหาและข้อจำกัดในการทำงานของระบบ, ข้อเสนอแนะในการพัฒนาระบบต่อไป

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้เรื่องการพัฒนาเกมสองมิติด้วย Unity Engine
2) ได้เรียนรู้แนวคิด ระบบและการออกแบบการพัฒนาเกม
3) ผู้เล่นสามารถนำไปเล่นกับผู้อื่นได้ สร้างความสัมพันธ์และสังคมของผู้เล่นเกมใหม่ๆ
4) สามารถใช้เป็นแนวทางในการพัฒนาเกมให้ได้หลายแพลตฟอร์มมากขึ้น และเป็นประโยชน์ให้กับผู้ที่สนใจการพัฒนาเกมด้วย Unity Engine ต่อไปในอนาคต

บทที่ 2
ทฤษฎีที่เกี่ยวข้องและเทคโนโลยีที่ใช้

2.1 ยูนิตี้ เกม เอนจิ้น (Unity Game Engine)

ยูนิตี้ (Unity) คือเครื่องมือสร้างเกมแบบ 2 มิติ และ 3 มิติ สามารถทำงานบนระบบปฏิบัติการได้ 2 ระบบคือ Windows และ OSX ผลิตเกมออกไปได้หลากหลายแพลตฟอร์ม เช่น แอนดรอยด์, iOS, PC เป็นต้น Unity มีจุดเด่นคือมองทุกอย่างเป็นวัตถุ และในวัตถุต่างๆ ก็จะมีองค์ประกอบของวัตถุนั้นที่ทางานร่วมกัน ไม่ว่าจะเป็นสิ่งของ ไอเทมในเกม ตัวละคร ทุกอย่างต้องการส่วนประกอบที่ทาให้ วัตถุเหล่านั้นเคลื่อนไหว หรือมีเสียงประกอบ นอกจากนั้น Unity มีร้านค้าออนไลน์สำหรับขาย Asset สำเร็จรูป ภาพพื้นหลังในเกม เสียงประกอบต่างๆ ชื่อว่า Asset Store รวมถึงคู่มือในการใช้งานที่ มีความละเอียด ง่ายต่อการศึกษาการใช้งาน นอกจากนั้น Unity มีชุมชนผู้พัฒนาจากทั่วโลก ที่พร้อมให้คำปรึกษาปัญหาที่เกิดขึ้นภายในการพัฒนา หรือ ศึกษาจากกรณีตัวอย่างจากกรณีที่เคยเกิดขึ้นได้โดยง่าย ถ้าต้องการใช้ในเชิงพาณิชย์จำเป็นต้องมีค่าใช้จ่ายค่าลิขสิทธิ์ของโปรแกรมเพิ่มเติมด้วย

รูปที่ 2.1 โลโก้โปรแกรมยูนิตี้

ส่วนประกอบหลักของโปรแกรมมีดังนี้

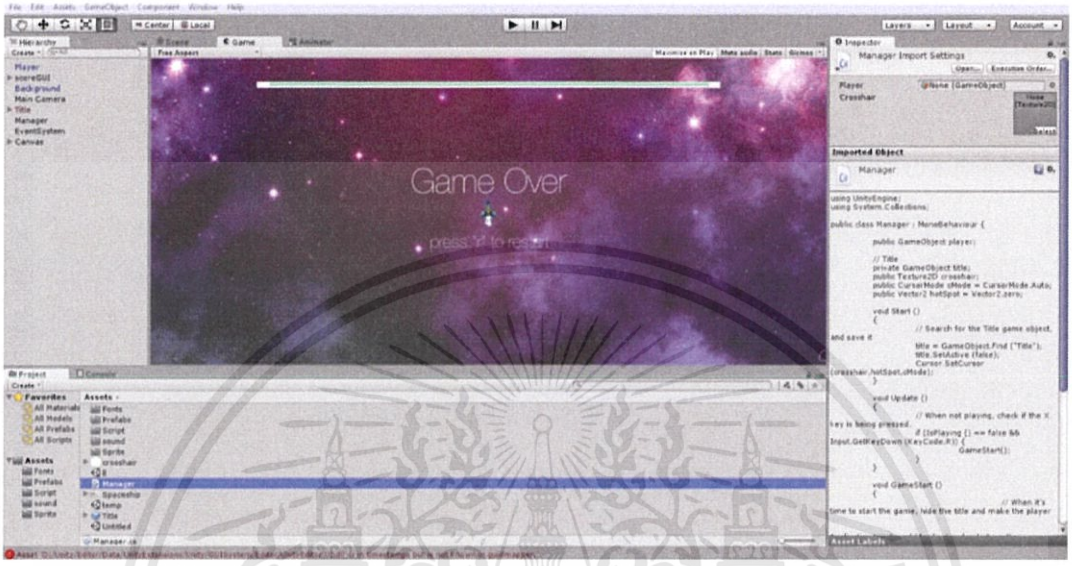
แท็บเครื่องมือ ประกอบด้วยเครื่องมือสำหรับเปลี่ยนมุมมองกล้อง ย้าย หมุน ย่อหรือขยายวัตถุ และปุ่มควบคุมการรันเกมในโหมด Debug

แท็บมุมมอง แบ่งเป็นสองแท็บย่อยคือ แท็บซีน (Scene) สำหรับมุมมองที่เห็นขณะออกแบบเกม และแท็บเกม (Game) สำหรับมุมมองที่ผู้เล่นเห็นจริงขณะเล่นเกม

แท็บอินสเปคเตอร์ (Inspector) ใช้แสดงค่าคุณสมบัติต่างๆ ของวัตถุที่กำลังทำงานอยู่

แท็บลำดับชั้น (Hierarchy) แสดงรายการวัตถุที่กำลังปรากฏอยู่ในฉากที่กำลังแสดงอยู่

แท็บโปรเจ็ค (Project) แสดงรายการวัตถุ โค้ด โปรแกรม ฉาก ไฟล์อ้างอิง และอื่นๆ ที่อยู่ในโปรเจ็ค โดยแท็บแต่ละแท็บนั้นสามารถปรับขนาดหรือลากย้ายไปยังตำแหน่งอื่นๆ ในหน้าจอของโปรแกรมยูนิตี้ได้



รูปที่ 2.2 โปรแกรมยูนิตี้

2.2 The Art of Screen Shake โดย Jan Willem Nijman จากสตูดิโอ Vlambeer

Jan Willem Nijman จากสตูดิโอ Vlambeer ผู้พัฒนาเกมเช่น Nuclear Throne, Ridiculous Fishing, Luftrausers, etc ได้นำเสนอแนวคิดเกี่ยวกับ User Feedback ของ Action Game โดยไอเดียหลักคือ การติดต่อกันระหว่างเกมและผู้เล่น โดยมีประเด็นหลักๆคือ ผู้เล่นนั้นควรจะสามารถที่จะ input ติดต่อกับเกมได้อย่างต่อเนื่อง และ รวดเร็ว ตัวเกมนั้นควรสามารถ ตอบรับโต้ตอบ และ แสดงผล input ของผู้เล่นได้อย่างทันที โดยในประเด็นของตัวเกมนั้นจะแตกย่อยออกไปในเรื่องของ จำนวนศัตรู กล้องตามตัวผู้เล่น Sound Effect และ Feedback ต่อการกระทำต่างๆ

โดยตัวอย่าง ที่ถูกยกมานั้นเป็นเกมแนว 2D Action Platformer ซึ่งมีข้อแนะนำดังต่อไปนี้

01 Basic Animation & Sfx

02 Lower Enemy Hp: HP ของศัตรูนั้นไม่จำเป็นต้องมากนักเนื่องจาก การสร้างศัตรูที่มี HP สูงขึ้นมาทำให้ผู้เล่นต้องเสียเวลาอยู่กับศัตรูตัวเดียว จะทำให้เกมน่าเบื่อ ต่างกับการใส่ศัตรู HP น้อยเข้าไปมากๆ ทำให้มีเหตุการณ์เกิดขึ้นตลอดเวลา ทำให้ผู้เล่นต้อง input ต่างๆ กันไปตลอดเวลา

03 Higher Rate of Fire: ปืนที่ดีไซน์ควรจะยิงรัวๆได้ เพราะวิธีการติดต่อกับโลกของเกมของผู้เล่นนั้น มีเพียงการยิง เพราะฉะนั้นเมื่อสามารถ input ได้รัวๆก็จะทำให้ user สามารถ input ได้ต่อเนื่องไม่ติดขัด

04 More Enemies: เมื่อสามารถยิงต่อเนื่องได้ จำนวนศัตรูควรเยอะขึ้นเพื่อให้เกมมีความสมดุล ไม่ง่ายเกินไป

05 Muzzle Flash: มีแสงปลายกระบอกปืน เพื่อสื่อต่อผู้เล่นว่าได้เกิด Action ขึ้นต่อจาก Input ที่ได้รับมา

06 Faster Bullet: เมื่อกระสุนเร็ว ทำให้ผู้เล่นสามารถที่จะ input ได้รัวขึ้น

07 Less Accuracy: ควรทำให้ปืนยิงออกมาไม่แม่นเพื่อเพิ่มความท้าทายให้กับผู้เล่น และ ดึงความสนใจของผู้เล่นไว้

08 Impact Effect: เพื่อให้ผู้เล่นสามารถเห็นได้ชัดเจนว่า กระสุนที่ยิงไปนั้นตกกระทบกับอะไร

09 Hit Animation: ให้รู้ว่าเกิดอะไรขึ้นเมื่อ Input ของ User ได้ส่งผลต่อโลกในเกมนั้นๆ

10 Enemy Knockback: เพราะเมื่อศัตรูถูกยิง ควรมีสถานะบอกให้ผู้เล่นรับทราบว่าที่ยิงไปนั้นกระทบกับตัวศัตรู

11 Permanence: เมื่อฆ่าศัตรูแล้ว ควรเหลือซากทิ้งไว้เพื่อแสดงให้เห็นถึงผลของการกระทำของผู้เล่นที่ได้ทำไป

12 Camera Lerp: กล้องหน่วงค่อยๆเลื่อนตามผู้เล่น

13 Camera Position: กล้องไม่ได้โฟกัสที่ตัวผู้เล่นแต่ โฟกัสที่ Action ที่ผู้เล่นกระทำตามจุดประสงค์ของเกม

14 Screenshake: Screenshake เป็นการโต้ตอบกลับไปสู่ผู้เล่นที่สำคัญอย่างหนึ่งโดยสามารถที่ใช้เป็นตัวบ่งบอกว่าตัวผู้เล่นกำลังตกอยู่ในอันตรายได้

15 Player Knockback: เราควรให้เหตุผลกับผู้เล่นในการที่จะไม่กดปุ่มโจมตีค้าง มิฉะนั้นควรดีไซน์ให้ไม่ต้องกดปุ่มในการยิงไปเลย

16 Sleep: สั่งให้เกมหยุดตัวเอง ไม่กี่ ms โดยตัวผู้เล่นนั้นจะไม่สังเกตเห็นชัดเจน แต่สมองนั้นจะได้มีเวลาในการตอบสนองต่อสิ่งที่ AI ศัตรูจะทำมากขึ้น

17 More Bass in SFX: เพราะว่าเสียงเป็นหนึ่งในส่วนสำคัญในการตอบสนองต่อ input ของผู้เล่น โดยมี case study เรื่องเสียงปืนของเกม Wolfenstein (2000)

18 Super Machine Gun: เพิ่มความหลากหลายใน Input ที่ผู้เล่นสามารถใส่เข้ามาในตัวเกม

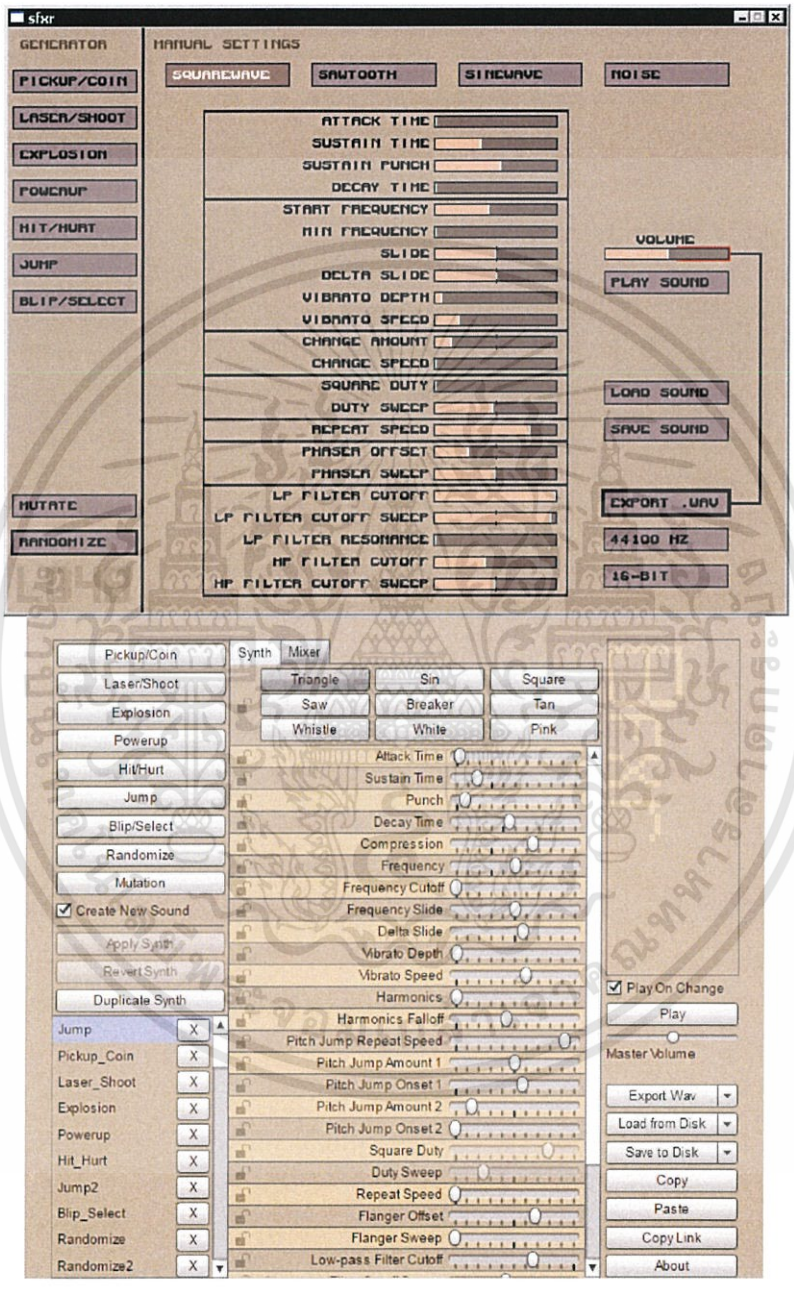
19 Random Explosion: การเพิ่มระเบิดเข้าไปในเกม ในเคสที่ยกตัวอย่างมาอย่าง 2D Side scroller นั้นช่วยเพิ่ม action บนหน้าจออย่างมาก ทำให้ผู้เล่นต้องจดจ่ออยู่กับสิ่งที่เกิดขึ้น

20 Faster Enemies: เมื่อได้ดีไซน์ให้ผู้เล่นมีความสามารถในการกำจัดศัตรูได้รวดเร็ว ก็ควรจะเพิ่มความเร็วของศัตรูด้วยเพื่อให้สมดุลกัน

21 Meaning: เกมที่ดีไซน์ควรมีแรงจูงใจแก่ผู้เล่น ไม่ว่าแรงจูงใจนั้นจะเป็นอะไรก็ตาม

2.3 โปรแกรมสร้างเอฟเฟ็คเสียงอย่างง่าย Sfxr และ Bfxr

Sfxr เป็นโปรแกรมสำหรับสร้างเสียงประกอบในเกมอย่างง่าย โดยตัวโปรแกรมนั้นมีเสียงให้เลือกใช้หลายแบบ สามารถสร้างเสียงตามรูปแบบที่เราต้องการ มีปุ่มให้เลือกชนิดของเสียง ปรับความถี่ ประเภทคลื่นเสียงต่างๆ เก็บเสียงในรูปแบบไฟล์ .wav



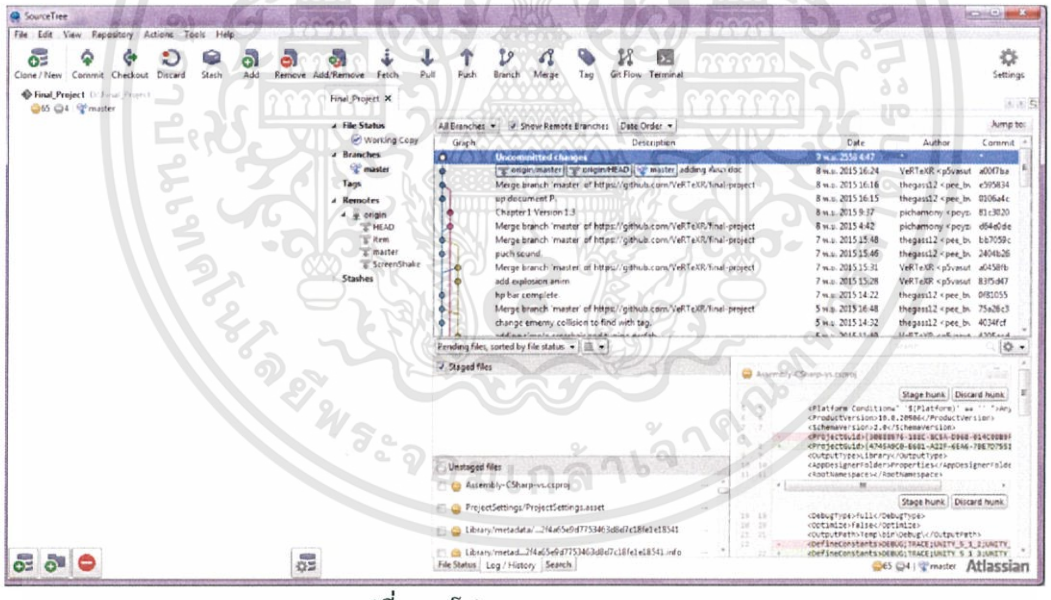
รูปที่ 2.3 โปรแกรม Sfxr

2.4 ควบคุม Version Control ผ่านการใช้งานโปรแกรม Source Tree ร่วมกับ GitHub

ผู้พัฒนาได้เลือกใช้ระบบ Git ซึ่งเป็น Version Control รูปแบบหนึ่ง มีข้อดีในการทำงานร่วมกันเป็นทีม สามารถควบคุมและอัพเดทปรับปรุงงานได้ โดยที่ทราบว่าในแต่ละครั้งคนในทีมเปลี่ยนแปลง แก้ไขอะไรไปบ้าง ถ้าหากว่ามีข้อผิดพลาด เวอร์ชั่นแรกนั้นดีกว่าล่าสุด สามารถกด Revision กลับมาได้

Source Tree เป็นซอฟท์แวร์ที่ช่วยจัดการกับ Git repository ได้ง่ายขึ้น มี GUI ที่ใช้งานง่าย สะดวกกว่าควบคุมผ่านระบบ command line

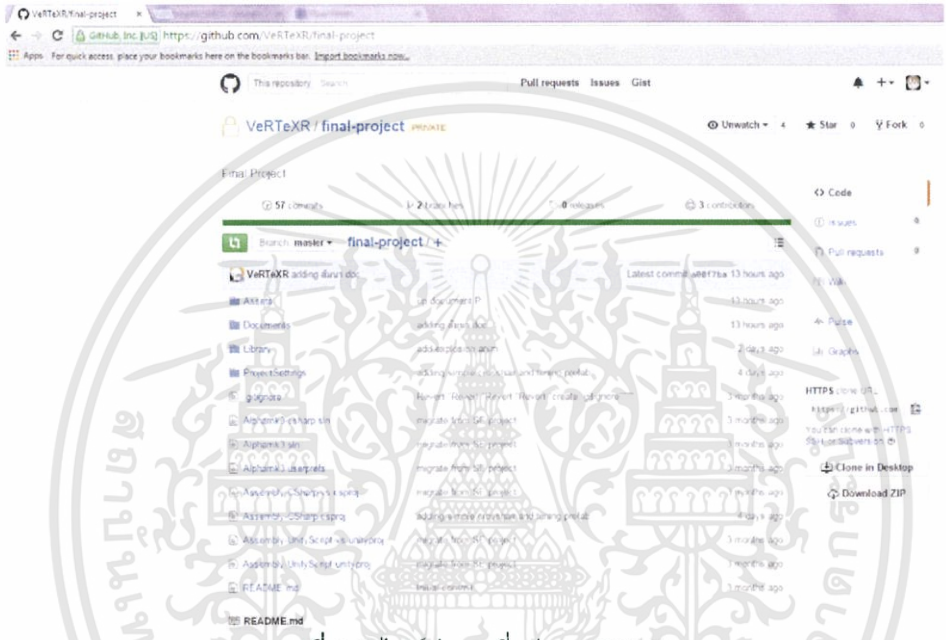
รูปที่ 2.4 โลโก้โปรแกรม SourceTree



รูปที่ 2.5 โปรแกรม SourceTree

GitHub เป็นเว็บเซิฟเวอร์ที่ให้บริการในการฝากไฟล์ Git โดยเราใช้เพื่อเก็บโปรเจ็คงานร่วมกันไว้ที่นี่ และใช้งาน อัพโหลด แก้ไขผ่าน SourceTree

รูปที่ 2.6 โลโก้ Github



รูปที่ 2.7 ไฟล์ต่างๆ ที่อยู่บน GitHub

2.5 การควบคุมความเร็วของศัตรูของเกมแนว Shoot em' Up (Enemy Speed Control on Shoot em' Up Game With Fuzzy Takagi Sugeno Method)

Shoot em' Up Game เป็นประเภทย่อยประเภทหนึ่งของเกม ต่อสู้ ซึ่งเป็นเกมที่เป็นที่นิยมด้วยกับ การมี ส่วนปฏิสัมพันธ์ที่น่าสนใจ ด้วยกับการที่เกมนี้มีจุดมุ่งหมายในการเล่นคือ การกำจัดศัตรูให้หมดไป ซึ่งเกมชนิดนี้อาจทำให้ผู้เล่นรู้สึกเบื่อหน่ายหากว่าศัตรูนั้นมีพฤติกรรมการเคลื่อนไหวที่ไปในทิศทางเดียว ดังนั้น เกมชนิดนี้จึงต้องการตัวควบคุมสำหรับควบคุมการเคลื่อนไหวที่มีการเปลี่ยนแปลงอยู่เสมอของศัตรู เช่น ปัญญาประดิษฐ์ ดังนั้น งานวิจัยนี้จึงได้นำเสนอ กระบวนการฟัซซี่ของ ทาคากิ ซูกิโนะ ซึ่งจะเป็นตัวจัดการพฤติกรรมการเคลื่อนไหวของศัตรูในเกมเพื่อทำให้เกมดูมีความน่าสนใจมากยิ่งขึ้น

เกมที่ดี ไม่ใช่เกมที่ให้ความสุขต่อผู้เล่นเพียงอย่างเดียว แต่ต้องเป็นเกมที่ให้ประโยชน์ต่อผู้เล่นด้วย และหนึ่งในประโยชน์นั้นก็คือ การที่ผู้เล่นได้มีการพัฒนาสมอง ซึ่งเกมเป็นระบบหนึ่งที่ผู้เล่นสามารถเป็นส่วนหนึ่งในการควบคุมและเผยแพร่วัฒนธรรมได้ ซึ่งผู้เล่นจะมีปฏิสัมพันธ์กับระบบ และความขัดแย้งต่างๆ ในเกม ในรูปแบบของวิศวกรรมเสมือน

Shoot em' Up Game เป็นเกมแนวยิงกันที่สามารถเล่นได้แบบผู้เล่นสองคนเล่นกัน หรือแบบผู้เล่นคนเดียวเล่นกับศัตรูที่เป็นปัญญาประดิษฐ์ซึ่งการเล่นเกมแนวนี้เป็นไปแบบง่ายๆคือ ผู้เล่นจะต้องกำจัดศัตรูให้หมดและในขณะเดียวกันก็ต้องพยายามเอาตัวรอดจากการโจมตีของศัตรูด้วยเช่นกัน ซึ่งในงานวิจัยนี้จะนำเสนอเรื่องของความเร็วในการเคลื่อนที่ของยานอวกาศในขณะที่มันลอยอยู่และทำการโจมตีผู้เล่น เพื่อสร้างให้เกมดูน่าสนใจมากยิ่งขึ้น ศัตรูในเกมจึงถูกสร้างให้เป็นปัญญาประดิษฐ์ และโดยเฉพาะเวลาในการตอบสนองของศัตรูนั้นสามารถเดาการยิงได้ สามรถหลบหลีก และจดจำพฤติกรรมการเคลื่อนไหวของศัตรูใด้ Shoot em' Up Game เป็นเกมที่จะมีเป้าหมายในการออกแบบศัตรูให้ช่วยปัญญาประดิษฐ์ในการควบคุมระดับความยากของเกม

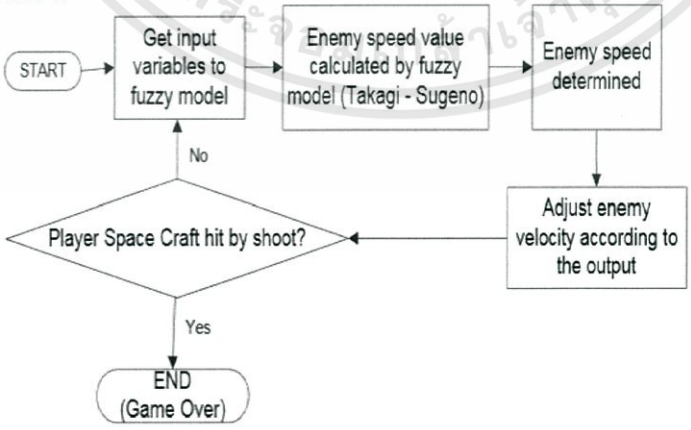
Fuzzy Takagi – Sugeno เป็นตัวช่วยในการควบคุมความเร็วของศัตรูซึ่งจะใช้ตัวแปรต่างๆในเกมที่รับเข้ามาเพื่อประมวลผลและส่งออกคำสั่งที่จะขึ้นอยู่กับตัวแปรที่นำเข้ามาด้วย ซึ่งจะทำให้พฤติกรรมการเคลื่อนไหวของศัตรูนั้นไม่เป็นไปในทิศทางเดียวหรือเส้นตรง
Fuzzification เป็นขั้นตอนการเปลี่ยน ค่าที่ได้รับจากตัวแปรต่างๆเข้าไปเก็บในเซ็ตของเงื่อนไขต่างๆเพื่อนำไปใช้ในการทำงานต่อไป ซึ่งขั้นตอนในการทำ Fuzzification มีดังนี้

1) รับค่าจากตัวแปรที่นำเข้ามา
2) ดำเนินการจัดเก็บรวบรวมข้อมูลที่ได้รับมาจากตัวแปรต่างๆ ที่สอดคล้องกัน
3) ดำเนินการแปลงค่าที่ได้ให้ชัดเจนเพื่อส่งค่าออกไปทำงาน

$$x^* = \frac{\sum_{i=1}^{n} m^i w_i}{\sum_{i=1}^{n} m^i}$$

Defuzzification เป็นการทำค่าฟัซซี่ให้เป็นค่าที่ใช้งานจริง ซึ่งในการนำเสนองานวิจัยนี้ใช้สมการในการแปลงค่าดังต่อไปนี้
m^i เป็นค่าเอาท์พุทที่ได้ของกฎแต่ละข้อ
w_i เป็นค่าเฉลี่ยถ่วงน้ำหนักของกฎแต่ละข้อ

การนำเสนองานวิจัยในครั้งนี้ ฟัซซี่ ลอจิค ที่ใช้ในการคำนวนความเร็วในการเคลื่อนที่ของศัตรูอธิบายโดย flow chart นี้



รูปที่ 2.8 Flow Chart แสดงการทำงาน
ของ กระบวนการฟัซซี่ของ ทาคากิ ซูกิโนะ

จากภาพจะแสดงการทำงานของ Fuzzy Takagi – Sugeno โดยเริ่มจากการรับค่าต่างๆ จากตัวแปร โดยค่าที่รับจะมี ดังนี้

- Distance : ระยะห่างระหว่างตัวศัตรู กับ ผู้เล่น
- Enemy Health Point : พลังชีวิตของศัตรู
- Player Remaining Life : พลังชีวิตที่เหลืออยู่ของผู้เล่น

โดยเมื่อได้ค่าทั้งหมดแล้ว จะนำไปเปรียบเทียบกับค่าในตารางเปรียบเทียบเพื่อหาค่าที่จะส่งกลับไปควบคุมความเร็วในการเคลื่อนที่ของศัตรู ซึ่งตารางเปรียบเทียบเพื่อตำนวนค่าความเร็วมีดังนี้

R1:	IF distance is close and enemy HP is low THEN enemy speed is fast
R2:	IF distance is close and enemy HP is high THEN enemy speed is medium
R3:	IF distance is medium or enemy HP is low THEN enemy speed is slow
R4:	IF distance is medium or enemy HP is high THEN enemy speed is medium
R5:	IF distance is far and enemy HP is THEN low enemy speed is slow
R6:	IF distance is near or player remaining life is low THEN enemy speed is fast
R7:	IF distance is near or player remaining life is medium THEN enemy speed is fast
R8:	IF distance is medium and player remaining life is low THEN enemy speed is fast
R9:	IF distance is medium or player remaining life is medium THEN enemy speed is medium
R10:	IF distance is medium or player remaining life is high THEN enemy speed is slow
R11:	IF distance is far or player remaining life is high THEN enemy speed is slow
R12:	IF enemy HP is low and player remaining life is low THEN enemy speed is fast
R13:	IF enemy HP is low or player remaining life is high THEN enemy speed is slow
R14:	IF enemy HP is medium and player remaining life is high THEN enemy speed is slow
R15:	IF enemy HP is high or player remaining life is low THEN enemy speed is fast
R16:	IF enemy HP is high or player remaining life is medium THEN enemy speed is medium

รูปที่ 2.9 ตารางแสดงรายการกฎของฟัซซี่

ดังในตาราง ใช้การดำเนินการโดยใช้ AND และ OR ในการดำเนินการเปรียบเทียบค่าต่างๆ โดย AND จะใช้ดำเนินการกับค่าขนาดเล็ก และ OR จะใช้ดำเนินการกับค่าขนาดใหญ่

จากการศึกษางานวิจัยในครั้งนี้ทำให้ผู้ศึกษาได้เรียนรู้วิธีจัดการพฤติกรรมการเคลื่อนที่ของศัตรูในเกมแนว Shoot em' Up Game ซึ่งการจัดการความเร็วในการเคลื่อนที่ของศัตรู ทำให้เกมมีความน่าสนใจ และ ท้าทายมากยิ่งขึ้น ทั้งยังทำให้ผู้เล่นสามารถฝึกการใช้สมองได้อีกด้วย ไม่เพียงแค่การจดจำรูปแบบการเคลื่อนไหวเดิมๆ ของศัตรูอีกต่อไป

2.6 ทฤษฎีโฟลว์ วิวัฒนาการ และ ความคิดสร้างสรรค์ หรือความสนุกและเกม

เกมและสื่อต่างๆ ที่เกี่ยวข้อง ในมุมมองการสร้างและการตอบรับของผู้เล่นโดยใช้ระบบโมเดลความคิดสร้างสรรค์ DPFi เพื่อที่จะแสดงให้เห็นถึงวิวัฒนาการของสื่อที่เกิดขึ้น ความสัมพันธ์ที่ส่งผลต่อกันและกัน ทฤษฎีโฟลว์ของความคิดสร้างสรรค์เป็นสิ่งชี้ให้เห็นถึง fun factor ของเกม และทฤษฎี Narrative Transportation ชี้วัดในส่วนของเรื่องราว รวมถึงความสัมพันธ์ของทฤษฎีข้างต้นกับ เนื้อเรื่องของเกม นอกจากนั้นยังใช้ทฤษฎีความคิดสร้างสรรค์ทั่วไปของ Boyd (2009) ว่า ศิลปะคือ การเล่นกับแพทเทิร์น ดังนั้นจึงสามารถยกบรบทที่ว่า เกมเพลย์ ไม่ว่าจะรูปแบบใดก็ตามสามารถที่จะส่งเสริมพัฒนาการความฉลาดของสัตว์ได้ ดังนั้นเกมในฐานะของศิลปะอย่างนึงนั้นอาจจะสามารถที่จะเพิ่มความฉลาดให้กับมนุษย์ได้

ทฤษฎีโฟลว์ ในปี 1996 Csikszentmihalyi ได้นำเสนอปัจจัยต่างๆ ที่เป็นเอกลักษณ์ของโฟลว์ ขึ้นมาได้แก่

1) ในการกระทำใดการกระทำหนึ่งมีเป้าหมายอย่างชัดเจนเป็นขั้นตอน
2) เกิดผลของการกระทำทันทีที่มีการกระทำเกิดขึ้น
3) ความยากของเป้าหมายและความสามารถของผู้ทำงานนั้นๆ เหมาะสมกัน
4) การกระทำและการรับรู้ของผู้กระทำนั้นได้รวมเป็นหนึ่งเดียวกัน
5) สิ่งรบกวนทั้งหลายถูกตัดออกจากความคิด
6) ไม่มีความกลัวที่จะล้มเหลว
7) ผู้กระทำนั้นไม่คำนึงถึงสิ่งที่ตัวเองเป็น
8) การรับรู้เวลาของผู้กระทำนั้นถูกบิดเบือนไป
9) การกระทำนั้นๆ กลายเป็นงานอัตโนมัติ

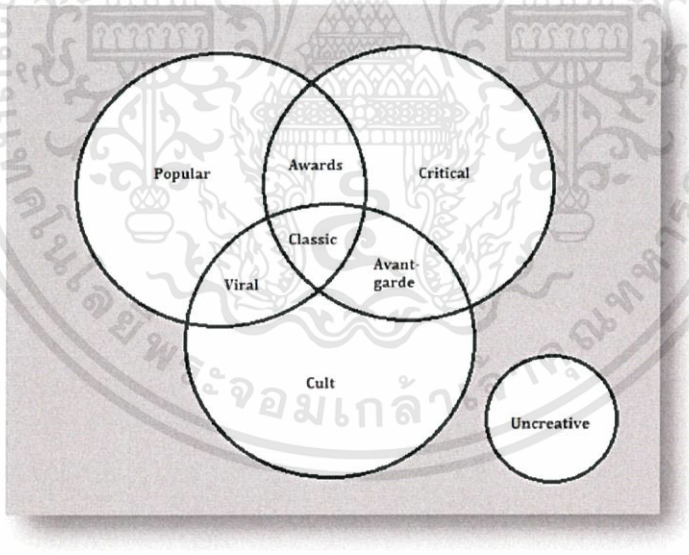
โดยในงานวิจัยของ Sweetser & Wyeth นั้นได้แสดงให้เห็นว่าเกมที่ได้รับการยอมรับว่าเป็นเกมที่สนุกนั้นมีคุณสมบัติเหล่านี้ทั้งหมด การติดอยู่ในโฟลว์นั้นเท่ากับว่ากำลังสนุกกับการเล่นเกม หรือ สามารถใช้สุภาษิตที่ว่า “เวลามักผ่านไปเร็วเสมอเมื่อตนเองรู้สึกสนุก” ได้อย่างชัดเจน โดย Csikszentmihalyi ได้นำเสนอโมเดลของโฟลว์โดยสามารถเห็นได้ชัดว่าจะสามารถเกิดขึ้นได้เมื่อความสามารถของผู้เล่นเหมาะกับการท้าทายที่ได้เผชิญ จะทำให้ผู้เล่นนั้นๆ ตกอยู่ในโฟลว์

ทฤษฎี Narrative Transportation ในเกมส่วนใหญ่นั้นจะมีเรื่องราวที่คอยส่งเสริมเกมเพลย์ของเกมนั้นๆ อยู่ เพื่อที่จะช่วยให้ผู้เล่นสามารถ เป็นส่วนหนึ่งเข้ากับเกมได้ง่ายขึ้น เรื่องราวที่ว่านี้คือสิ่งที่ทำให้เกิด Narrative Transportation โดย Van Laer ได้ให้นิยามไว้ว่า Narrative

Transportation คือกระบวนการที่ผู้บริโภคสื่อถูกดูดเข้าไปในเนื้อเรื่องที่บริโภค โดยจะทำให้เนื้อเรื่องที่เล่านั้นเสมือนเกิดขึ้นจริงในความคิดและจิตใจ ของผู้บริโภคสื่อนั้นเมื่อ เนื้อเรื่องนั้นๆ หรือประสบการณ์ส่วนบุคคลของผู้บริโภคเข้ากันได้

ส่วนในปี 2002 Green and Brock ได้ให้นิยามไว้ว่า Narrative Transportation นั้นเกิดขึ้นเมื่อผู้บริโภคสื่อรู้สึกเหมือนได้เข้าไปอยู่ในโลกที่ถูกสร้างขึ้นในเรื่องราวที่เกิดขึ้น เพราะมีความผูกพันธ์กับตัวละคร และ จินตนาการที่วาดพลอตเรื่องนั้นๆ ไว้

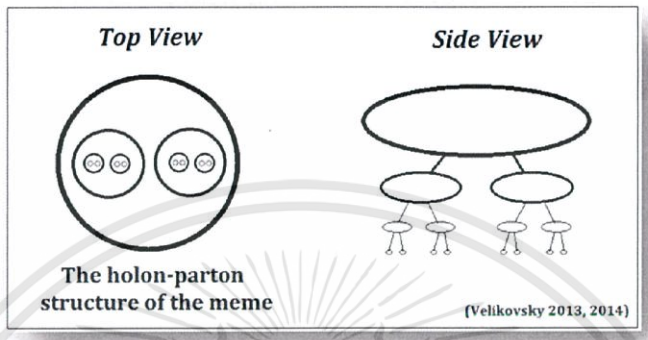
ความเกี่ยวข้องระหว่างนิยามความคิดสร้างสรรค์และ โมเดล DPFi ระบบ DPFi (Domain, Person, Field Interactions) นั้นเป็นระบบที่เชื่อมระหว่างวิทยาศาสตร์และศิลปะ โดยนิยามที่เรียบง่ายที่สุดของความคิดสร้างสรรค์นั้นมาจากงานวิจัยด้านจิตวิทยาซึ่งสามารถนิยามออกมาได้ว่า ความคิดสร้างสรรค์คือไอเดีย กระบวนการ หรือ ชิ้นงานที่ใหม่และเหมาะสม โดยในปี 2006 Csikszentmihalyi ได้อธิบายระบบโมเดลของความคิดสร้างสรรค์ไว้ดัง รูปที่ 2.9 เพื่อที่จะให้เกิดความสร้างสรรค์ขึ้นมา บุคคลต้องสร้างผลิตภัณฑ์ที่เป็นสิ่งแปลกใหม่ไปจากที่มีอยู่ในโดเมน จากนั้นผลิตภัณฑ์จะถูกเลือกโดยฟิลด์เพื่อที่จะให้เข้าไปอยู่ในโดเมน การสร้างสรรค์นั้นเกิดขึ้นเมื่อคนได้สร้างการเปลี่ยนแปลงในโดเมนที่จะถูกส่งต่อไปในเวลาต่อไป โดยกระบวนการข้างต้นนั้นเรียกได้ว่าเป็นอัลกอริทึมแห่งพัฒนาการ มีการคัดเลือก การเปลี่ยนแปลง การส่งต่อข้อดี และข้อเสีย ดังนั้นเมื่อนำมาใช้กับเกม ถ้าหากว่าคนส่วนใหญ่ในฟิลด์เกมนั้นได้เห็นพ้องกัน ว่าเกมใหม่ที่ออกมานั้นสนุกและสร้างสรรค์ (ใหม่ และ เหมาะสม) เกมนั้นก็จะเป็นเกมที่ได้รับการยอมรับ



รูปที่ 2.10 ประเภทของความสร้างสรรค์ หรือการยอมรับของวงการเกม: © JT Velikovsky

Holon-Parton เป็นหน่วยวัดค่าทางวัฒนธรรม Arthur K ได้ให้คำนิยาม holon ไว้ว่า สิ่งใดที่เป็นส่วนหนึ่งแต่ก็เต็มด้วยตัวเองในเวลาเดียวกัน นักฟิสิกส์ควอนตัม Richard P. ได้ใช้คำว่า parton ในการเรียกคอนเซ็ปต์เดียวกันในทางฟิสิกส์ เช่นเดียวกันกับการแข่งขันกันระหว่างสิ่งมีชีวิตต่างสายพันธุ์ในด้านวัฒนธรรม เกม (ในฐานะมีม) ก็ต้องแข่งกันในสภาพแวดล้อมเดียวกันกับเกมอื่นๆ เพื่อเงินและความสนใจของผู้เล่น ในขณะเดียวกันก็อยู่ภายใต้ประเภทเดียวกัน และยังพยายามที่จะควบคุม

และให้คำสั่งแก่ความสนใจของผู้เล่นเกมในฐานะ holon-partons เป็นกลุ่มก้อนของไอเดีย ขั้นตอน และผลลัพธ์ (หรือมีม) สามารถเห็นได้ชัดว่า มีม นั้นอยู่ทั้งชั้นบนสุดและล่างสุด แทนการทำซ้ำในตัวเองเมื่อเวลาผ่านไปเรื่อย อัลกอริทึมของการพัฒนาการ การคัดเลือก, แปลกแยก (ประกอบด้วยการรวมกัน กลายพันธุ์และดัดแปลง) และการส่งต่อโดยพันธุ์กรรมนั้นเกิดขึ้นในลำดับชั้นด้านล่างในฐานะระบบต่อยอดและย้อนทำซ้ำในตัวเองอย่างไม่ลิเนีย



รูปที่ 2.11 รูปแบบ Holon-Parton ของมีม หน่วยทางวัฒนธรรม © JT Velikovsky

รูปที่ 2.12 ลำดับชั้นของโดเมนเกมและ ลำดับชั้นของมีม © JT Velikovsky

Transmedia Storytelling (การเล่าเรื่องข้ามสื่อ)นั้นถูกนิยามโดย Kinder, Jenkin และในกฎหมายโดยว่าเนื้อเรื่องที่เกี่ยวข้องกัน 3 เรื่องเป็นอย่างต่ำโดยเนื้อเรื่องทั้งหมดนั้นอยู่ในจักรวาลเดียวกัน บนแพล็ตฟอร์มต่างๆกันไป โดยสามารถสังเกตได้ว่าสื่อที่ถูกเล่าใหม่นั้นไม่ใช่เพียงแค่การแปลงเนื้อเรื่องเดิมไปบนแพลตฟอร์มอื่น แต่เป็นเนื้อเรื่องแยกไปที่มีบางจุดที่คล้ายคลึง และบางจุดแต่งต่างเป็นเอกลักษณ์ อยู่บนแพลตฟอร์มของตัวเอง ดังนั้น กฎของอัลกอริทึมในการพัฒนาการ (และลำดับชั้น) จะสามารถนำมาใช้ได้เช่นเดียวกันกับที่มีการใช้ในเกม

กระบวนการของการดีไซน์เกมเองก็สามารถมองได้ว่าเป็นการเล่นกับแพทเทิร์น เมื่อเกมดีไซน์เนอร์ สร้างแพทเทิร์นของเกมเพลย์ ถ้าแพทเทิร์นเหล่านี้นั้นไม่มีความแตกต่างจากเกมที่ประสบ

ความสำเร็จก่อนหน้านี้มากพอ เกมนั้นๆ ก็จะถูกตราหน้าว่าไม่มีความเป็นตัวเอง (หรือไม่มีความคิดสร้างสรรค์) ในการเล่นซ้ำ ผู้เล่นเองก็สามารถที่จะเปลี่ยนแพทเทิร์นของการเดินทางของตนเองในเลเวลนั้นๆ ของเกมได้ เพื่อที่จะปรับกลยุทธ์ตัวเองเพื่อที่จะสามารถเล่นเกมจบได้เร็วที่สุด

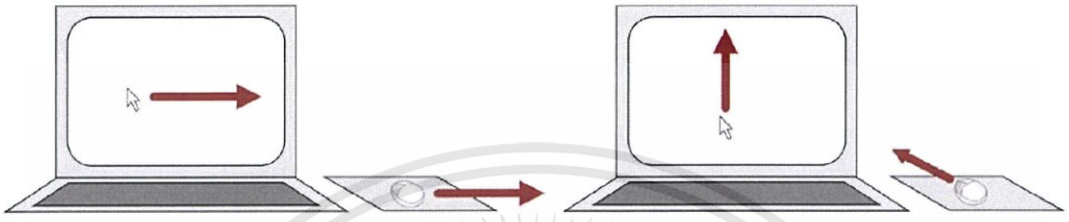
ในงานวิจัยของ Capra & Luisi (2014) นั้นได้นิยามว่าความฉลาดคือความสามารถในการแก้ไขปัญหา เกมส์นั้นได้เพิ่มความฉลาดแก่ผู้เล่นได้มาก เนื่องจากผู้เล่นนั้นต้องแก้ปัญหาใหญ่นั่นคือเป้าหมายหลักของด่าน และเป้าหมายย่อยทั้งหมด โดยจากงานวิจัยของมหาวิทยาลัย Queen Mary University of London และ University College London (UCL) ได้ทดสอบการตอบสนองต่อสิ่งเร้าของอาสาสมัครจำนวน 72 คน โดยได้มีการให้อาสาสมัครนั้นเล่นเกม Starcraft และ The Sims โดยผลสรุปออกมานั้นพบว่าเกมส์บางประเภทนั้นสามารถเพิ่มความฉลาดให้กับผู้เล่นได้มากกว่าเกมส์ประเภทอื่นๆ โดย Starcraft นั้นสามารถเพิ่มความฉลาดให้กับผู้เล่นได้มากกว่า The Sims นอกจากนั้น ในปี 2014 งานวิจัยของ Przybylski et al ได้พบว่าเกมส์ที่ถูกดีไซน์มาแย่สามารถที่จะเพิ่มความก้าวร้าวให้กับผู้เล่นได้ แต่อย่างไรก็ตามเกมส์นั้นๆ ก็เป็นตัวเปิดโอกาสให้กับผู้เล่นได้เรียนรู้วิธีการที่จะเผชิญหน้ากับดีไซน์ที่ย่ำแย่เหล่านั้น ทำให้เกิดการพัฒนาด้านอารมณ์ในตัวผู้เล่น ดังนั้นเราจึงสามารถกล่าวได้ว่าเกมส์ที่แย่นั้นก็สามารถที่จะทำให้ความฉลาดด้านอารมณ์ของผู้เล่นนั้นพัฒนาขึ้นเช่นกัน

ในการพัฒนาเกม ดีไซน์เนอร์มักเลือกไอเดียที่ดีที่สุดจากที่มีอยู่ในปัจจุบันและนำมารวมกับไอเดียอื่น หรือไอเดียของตนเองมีอยู่ เพื่อให้เกิดเป็นไอเดียใหม่ที่แตกต่างจากเดิมขึ้น โดยในมุมมองนี้ความคิดสร้างสรรค์ทั้งหมดได้เกิดมาจากอัลกอริทึมการพัฒนาเช่นนี้ โดยมีแรงกดดันจากการเลือกปะทะกับไอเดียในทุกขั้นและทุกลำดับชั้นของ holon-parton ถ้าหากว่าไม่ใช่เช่นนั้นเกมนั้นๆ สามารถพูดได้ว่าเป็นเกมที่ไม่เต็มเกม และสามารถพิจารณาได้ว่าถูกดีไซน์มาไม่ดีพอ

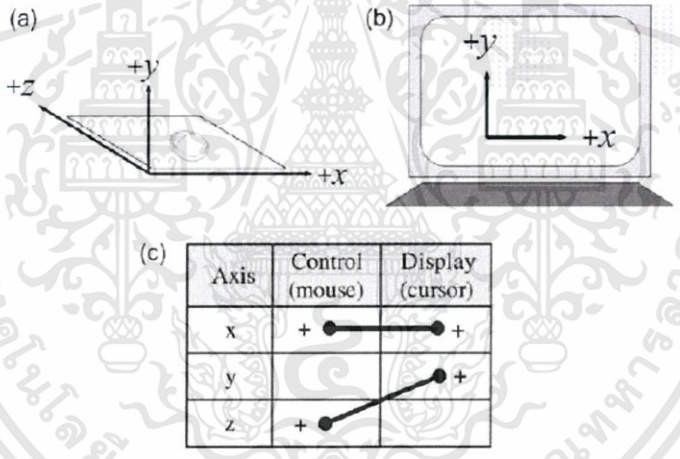
2.7 ความสัมพันธ์ระหว่างการควบคุมและมุมมองที่เห็น

ใช้หลักการของปฏิสัมพันธ์ของมนุษย์กับคอมพิวเตอร์มาช่วยในการออกแบบ Interface การเคลื่อนไหว ควบคุมเกม เพื่อให้ผู้เล่นได้รู้สึกว่าเล่นง่ายและลื่นไหลมากที่สุด

ความสัมพันธ์ในพื้นที่ว่างนั้น ขึ้นอยู่ระหว่างการควบคุมของผู้เล่นจากอุปกรณ์เล่นเกม (Hard Control) และภาพที่มองเห็นในหน้าจอ (mapping) พื้นที่ของเม้าส์ที่ใช้ควบคุมนั้นประกอบด้วยแกน x, y, z ส่วนพื้นที่ในหน้าจอประกอบด้วยแกน x และ y เท่านั้น ดังแสดงในรูปที่ 2.13



รูปที่ 2.13 แสดงให้เห็นทิศทางการเคลื่อนที่ของเม้าส์ที่คนบังคับ กับเม้าส์ที่เคลื่อนไหวบนหน้าจอ



รูปที่ 2.14 (a) พื้นที่ควบคุม (b) พื้นที่แสดงผล (c) แสดงถึงความสัมพันธ์ของการควบคุมเม้าส์และลูกศรบนหน้าจอตามแกน x, y, z

บทที่ 3
การวิเคราะห์และออกแบบระบบ

3.1 การวิเคราะห์และออกแบบรูปแบบเกม (Game Design)

เป็นเกมแอคชั่นอาเขต เกม 2 มิติ มีฉากเป็นอวกาศ ตัวละครหลักเป็นยานอวกาศ ให้ผู้เล่นยิงและหลบกระสุนจากยานของศัตรูให้ได้ ผู้เล่นเห็นมุมมองจากด้านหลังของตัวละคร บังคับทิศทางและโจมตีศัตรูเพื่อให้ผ่านด่าน โดยยิงศัตรูและเก็บไอเทมไปเรื่อยๆ ตามจำนวนที่กำหนด เพื่อผ่านไปด่านต่อไป ถ้าชีวิตของผู้เล่นหมดก็จะแพ้แล้วเริ่มต้นใหม่ตั้งแต่ด่านแรก สามารถเล่นได้ทุกเพศทุกวัย ง่ายต่อการเล่นแม้ว่าจะเพิ่งเริ่มต้น การเคลื่อนไหวลื่นไหล บังคับง่าย

3.1.1 Game Design Document

1) Design History

Version 1.0	1. จำนวนผู้เล่น 1 คน 2. ศัตรูเกิดแบบสุ่มและยิงไปที่ผู้เล่น 3. ผู้เล่นยิงศัตรู ถ้ายิงโดนนับเป็น 1 คะแนน แต่ถ้าโดนศัตรูยิงลบ 1 คะแนน 4. ใส่แถบแสดงสถานะชีวิตผู้เล่น 5. มีเพียงด่านเดียว
Version 2.0	1. เพิ่มด่านขึ้น 2. มีจุดเปลี่ยนฉากให้ผู้เล่นเข้าชนเพื่อเปลี่ยนฉาก 3. มีฉากคั่นระหว่างการเปลี่ยนฉาก 4. เพิ่มศัตรูหลายรูปแบบ 5. มีเสียงประกอบในเกม

1) Vision Statement

ชนิดของเกม	เป็นเกมแอคชั่น Arcade ผู้เล่นเห็นมุมมองจากด้านหลังของตัวละคร บังคับทิศทางและโจมตีศัตรูเพื่อให้ผ่านด่านไปได้
Game logline	ยิงศัตรูและเก็บไอเทมไปเรื่อยๆ ตามจำนวนที่กำหนด เพื่อผ่านไปด่านต่อไป ถ้าชีวิตของผู้เล่นหมดก็จะแพ้แล้วเริ่มใหม่ตั้งแต่ด่านแรก
Gameplay synopsis	**Mechanic:** รูปแบบการเล่นจะเป็นการยิงกระสุนให้โดนศัตรู และเก็บไอเทมเพื่อเพิ่มพลัง และลูกเล่นที่ได้เปรียบในการโจมตีศัตรู **Look and feel:** เกม 2 มิติ มีฉากเป็นอวกาศ ตัวละครหลักเป็นยานอวกาศ ให้ผู้เล่นยิงและหลบกระสุนจากยานของศัตรูให้ได้

2) Marketing Information

กลุ่มเป้าหมาย	สามารถเล่นได้ทุกเพศทุกวัย
Platform	OSX และ Windows
Feature comparison	ง่ายต่อการเล่นแม้ว่าจะเพิ่งเริ่มต้น การเคลื่อนไหวลื่นไหล บังคับง่าย

3) Legal Analysis

Software Commercial	Windows 7 Professional Windows 8 Ultimate Adobe Photoshop CS6
Free Software License	Unity (ถ้าใช้เชิงพาณิชย์ต้องเสียค่า license เพิ่ม) SourceTree GitHub

4) Gameplay

คำอธิบายเกม	เมื่อคลิกเม้าส์ค้างไว้จะสามารถยิงศัตรูได้ต่อเนื่อง เล็งกระสุนไปที่ยานศัตรู และจะมีไอเทมปรากฏขึ้นให้เก็บ เพื่อพลังพิเศษในการโจมตีศัตรู
การควบคุม	- ใช้เม้าส์เลื่อนตัวยาน และคลิกซ้ายเพื่อยิง - ปุ่ม space bar เพื่อเริ่มเกม - ปุ่ม W เคลื่อนที่ไปด้านหน้า - ปุ่ม A เคลื่อนที่ไปด้านซ้าย - ปุ่ม S เคลื่อนที่ไปด้านขวา - ปุ่ม D เคลื่อนที่ไปด้านหลัง - ปุ่ม r เพื่อเริ่มเล่นใหม่
Interfaces	มุมมองการเล่นเกมเป็นแบบ 2 มิติ (Bird eye View) ผู้เล่นเห็นตัวยานเพื่อบังคับ

Gameplay (ต่อ)

กฎ	ยิงให้โดนยานศัตรูและเก็บไอเทมเพื่อนับคะแนนสะสมไปด่านต่อไป ถ้าหากโดนยิงจนเลือดหมดถือว่าผู้เล่นแพ้
โหมดผู้เล่น	ออฟไลน์ มี 1 แบบ คือผู้เล่นเป็นผู้บังคับยาน
ระดับความยากง่าย	ระดับความยากง่ายเพิ่มตามด่าน และการสุ่มไอเทม

5) Game Character

ในเกมประกอบไปด้วยตัวละคร 2 ฝ่าย คือผู้เล่นและศัตรู

ยานผู้เล่น

	เป็นประเภท Player Character เป็นตัวละครที่ผู้เล่นควบคุมได้ โดยผู้เล่นจะต้องควบคุมยานเพื่อหลบกระสุนจากยานศัตรู และยิงยานศัตรู พร้อมกับเก็บไอเทมเพื่อพลังพิเศษด้วย

ยานศัตรู

   	เป็นประเภท Non- Player Character เป็นตัวละครที่ผู้เล่นไม่สามารถควบคุมได้ ยานศัตรูจะยิงกระสุนใส่ผู้เล่น และสุ่มเกิดไปเรื่อยๆ

กระสุน

รูปในเกม	ความสามารถ
	เป็นกระสุนแบบธรรมดาที่มีอยู่ตลอดเกม ไม่มีวันหมด สามารถยิงได้อย่างต่อเนื่อง
	เป็นกระสุนแบบธรรมดาที่ถูกยิงออกมาจากยานศัตรู

ไอเทม

รูปในเกม	ความสามารถ
	เพิ่มความสามารถหลากรูปแบบให้ผู้เล่น

3.2 การออกแบบระบบ

3.2.1 แผนผังการดำเนินเกม (Game Flow Overview)

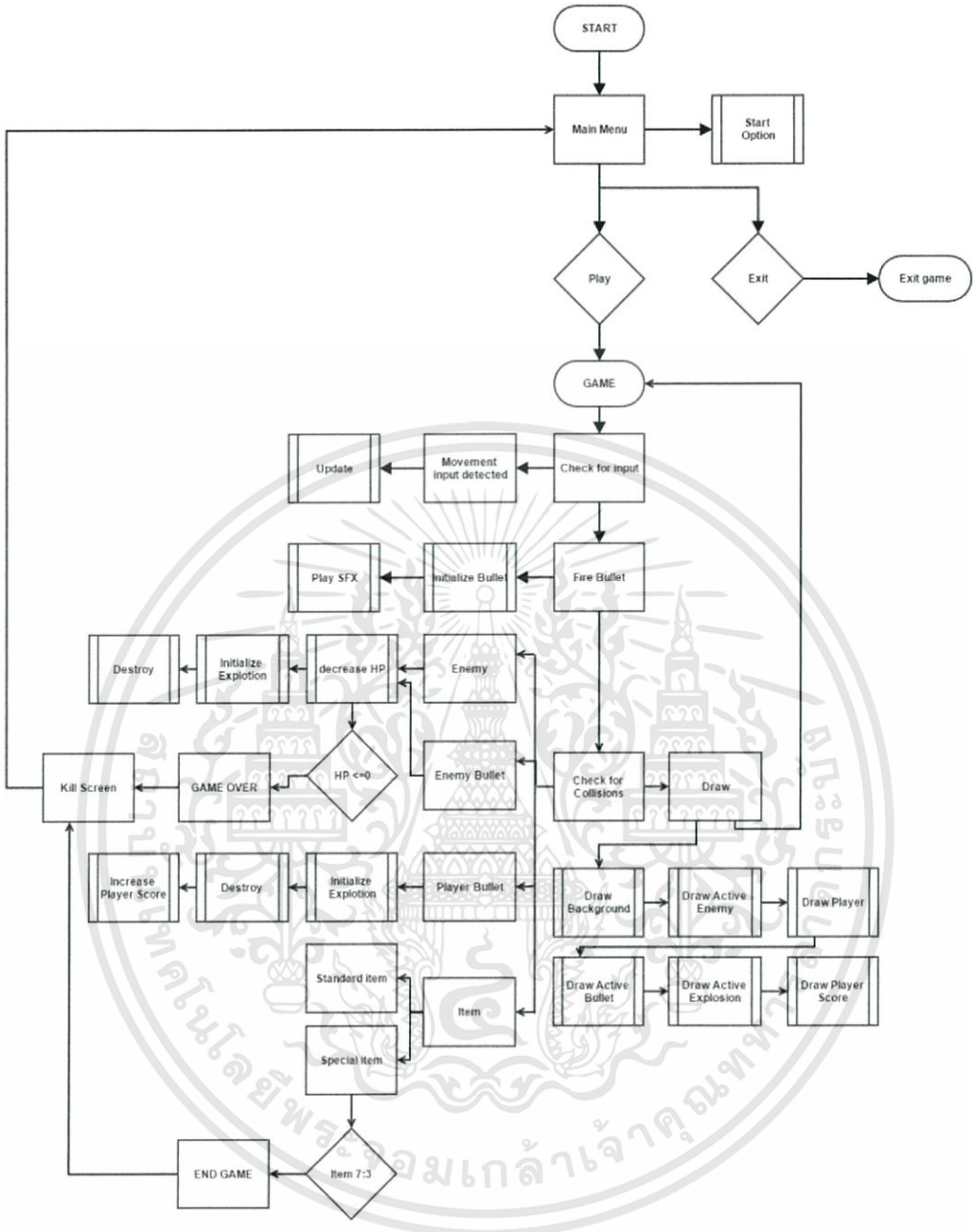
เมื่อเริ่ม Session เกมขึ้นมา ทางตัวเกมจะมีเมนูให้เลือกระหว่าง Start, Option และ Exit โดยหากเลือก Start ตัวเกมจะส่งผู้เล่นเข้าไปในฉาก ซึ่งจะมีการวาดวัตถุต่างๆ รวมถึงตัวผู้เล่นขึ้นมา จากนั้นตัวเกมจะมีคอยเช็ค Input ของผู้เล่น และ วาดตัวผู้เล่นซ้ำตาม Input ที่ได้รับมา ถ้าหากว่าผู้เล่นคลิกเมาส์ซ้าย ตัวเกมจะสร้างกระสุนของผู้เล่นขึ้นมา จากนั้นค่อยๆสร้างศัตรูขึ้น แล้วตัวเกมจะคอยตรวจสอบการชนกันระหว่างวัตถุในเกมว่ากระสุนของศัตรูหรือตัวศัตรูได้กระทบกับตัวผู้เล่นหรือไม่ และตรวจสอบกระสุนของผู้เล่นว่าได้กระทบกับตัวศัตรูหรือไม่

ถ้าหากกระสุนของศัตรูกระทบกับตัวผู้เล่น ตัวเกมจะลดค่า HP ของผู้เล่นตามค่าความรุนแรงของกระสุนประเภทนั้น หาก HP ของผู้เล่นน้อยกว่าหรือเท่ากับ 0 จะถือว่าสิ้นสุดเกม ตัวเกมตัดเข้าหน้ารายงานผล

ถ้าหากศัตรูกระทบกับตัวผู้เล่นตัวเกมจะลด HP ผู้เล่นตามค่าที่กำหนดไว้ของประเภทศัตรูนั้นและทำลายตัวศัตรู หาก HP ของผู้เล่นน้อยกว่าหรือเท่ากับ 0 จะถือว่าสิ้นสุดเกม ตัวเกมตัดเข้าหน้ารายงานผล

ถ้าหากกระสุนของผู้เล่นกระทบกับศัตรู ตัวเกมจะทำลายศัตรูและเพิ่มคะแนนให้กับผู้เล่น ในขณะเดียวกันตัวเกมจะทำการสุ่มสร้างไอเทมขึ้นมา ซึ่งไอเทมที่สร้างขึ้นมามีสองประเภท ได้แก่ ไอเทมธรรมดาและ คีย์ไอเทม ถ้าผู้เล่นสามารถเก็บคีย์ไอเทมได้ 3 อัน เกมจะตัดเข้าหน้ารายงานผล ถ้าหากว่าไอเทมที่ผู้เล่นเก็บมาเป็นไอเทมธรรมดา ผู้เล่นจะได้รับผลประโยชน์ต่างๆ ตามประเภทของไอเทมธรรมดา

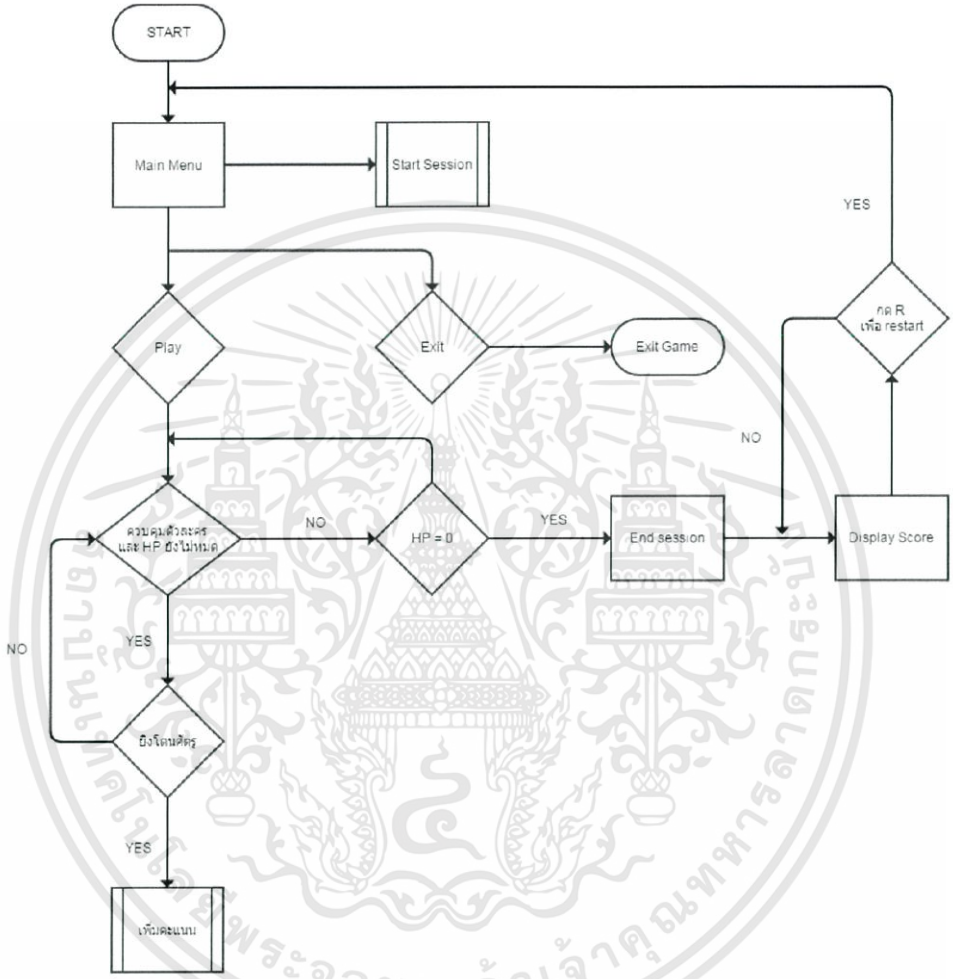
เมื่อผู้เล่น ทำลายศัตรูหมดฉากตัวเกมจะสุ่มฉากใหม่ขึ้นมา



รูปที่ 3.1 แสดง Flow Chart ของเกม

3.2.2 แผนผังกิจกรรมของผู้เล่น (Player Behavior)

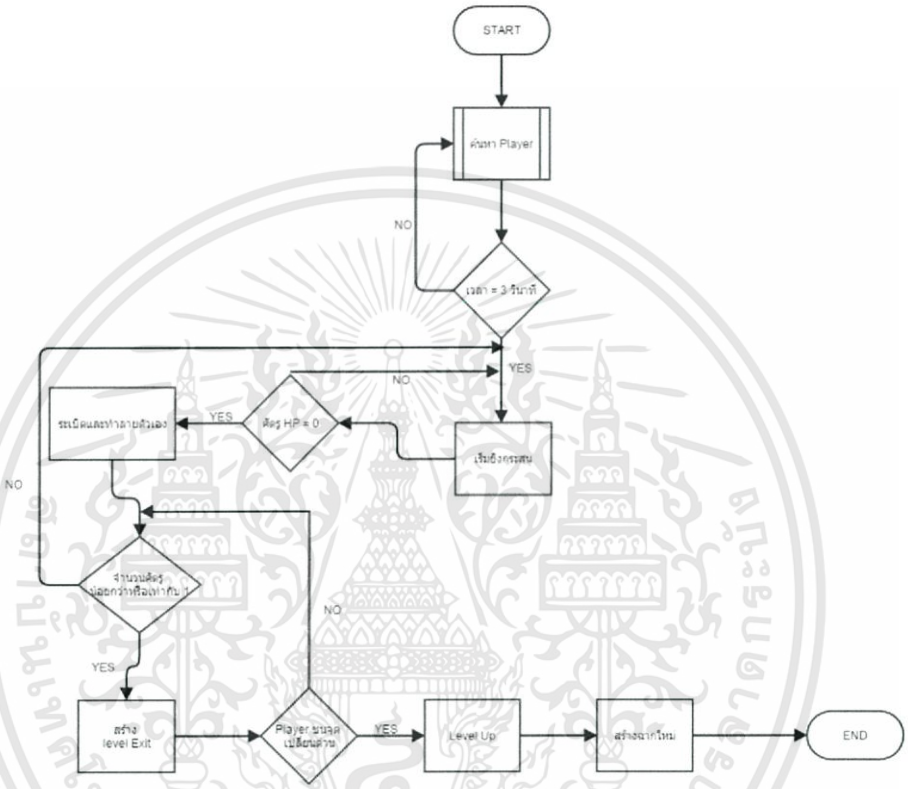
ในส่วนของผู้เล่นนั้น เมื่อเริ่มเกมแล้วต้องคอยควบคุมตัวละครหลบกระสุนจากศัตรู และระวังไม่ให้ HP หมด เมื่อยิงโดนศัตรูก็จะเพิ่มคะแนนให้กับผู้เล่น แต่ถ้าหาก HP หมด เกมจบ และแสดงคะแนนรวมให้เห็น ถ้าต้องการเล่นต่อให้กด r เพื่อเริ่มเล่นใหม่อีกครั้ง



รูปที่ 3.2 Flow Chart ของกิจกรรมผู้เล่น

3.2.3 แผนผังพฤติกรรมของศัตรู (Enemy Behavior)

เมื่อเกมเริ่มขึ้น ศัตรูจะค้นหาตัวผู้เล่น เมื่อเวลาผ่านไป 3 วินาทีเริ่มยิงกระสุน ถ้าหากศัตรูโดนโจมตีจนค่า HP หมด ตัวเองจะระเบิด ถ้าหากว่าศัตรูระเบิดหมดแล้วจะทำการสร้างประตูเพื่อผ่านไปด่านถัดไป โดยให้ผู้เล่นวิ่งเข้ามาชนจุดนี้ เมื่อผู้เล่นชนประตูผ่านด่าน เลเวลผู้เล่นเพิ่มขึ้นและทำการสร้างฉากใหม่ขึ้นมา



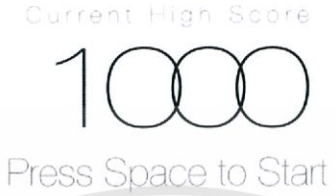
รูปที่ 3.3 Flow Chart ของกิจกรรมศัตรู

3.2.4 การออกแบบส่วนติดต่อกับผู้ใช้ (User Interface)

ตัวเกมทั้งหมดถูกดีไซน์ขึ้นมาด้วยกราฟฟิค แนว Pixel Art โดยทางผู้พัฒนาพยายามที่จะทำให้คอนโทรลของตัวยานที่ผู้เล่นเลือกแต่ละแบบมีความแตกต่างกัน เมื่อ Session ของผู้เล่นดำเนินไป ผู้เล่นได้พบเจอกับไอเทม และความสามารถพิเศษต่างๆ ที่อาจเพิ่มหรือลดความสามารถของยานที่ผู้เล่นเลือกมา โดยผู้เล่นต้องเผชิญหน้ากับศัตรูที่ตัวเกมได้ทำการสุ่มขึ้นโดยผู้เล่นมีสิทธิ์ที่จะสู้หรือหนีโดยการตัดสินใจในการสู้หรือหนี มีผลต่อบอสตัวสุดท้ายที่ผู้เล่นต้องเผชิญ

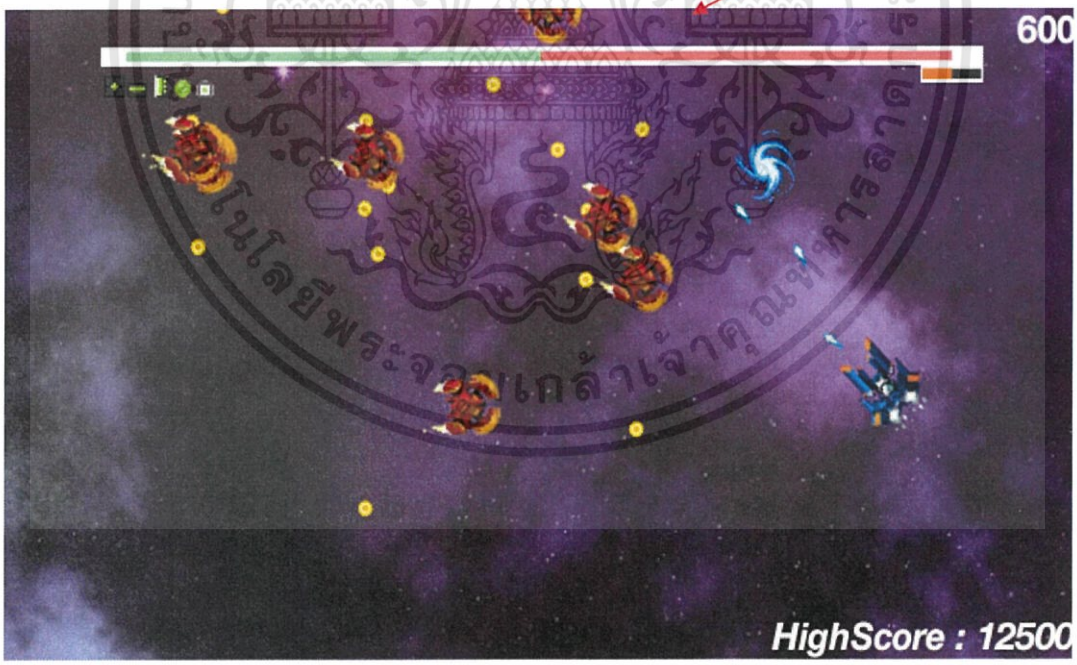
เนื่องจากความต้องการที่ให้ผู้เล่นใช้งานเกมได้ง่ายที่สุด จึงออกแบบหน้าจอให้มีเพียงหน้าเดียว และมีคำแนะนำบอกให้ผู้เล่นเลือกกดปุ่มเพื่อไปต่อ มีสถานะบอกเมื่อเปลี่ยนฉาก และแสดงจุดเปลี่ยนฉากขึ้นมาเมื่อชนะในแต่ละด่าน

Game Mock up

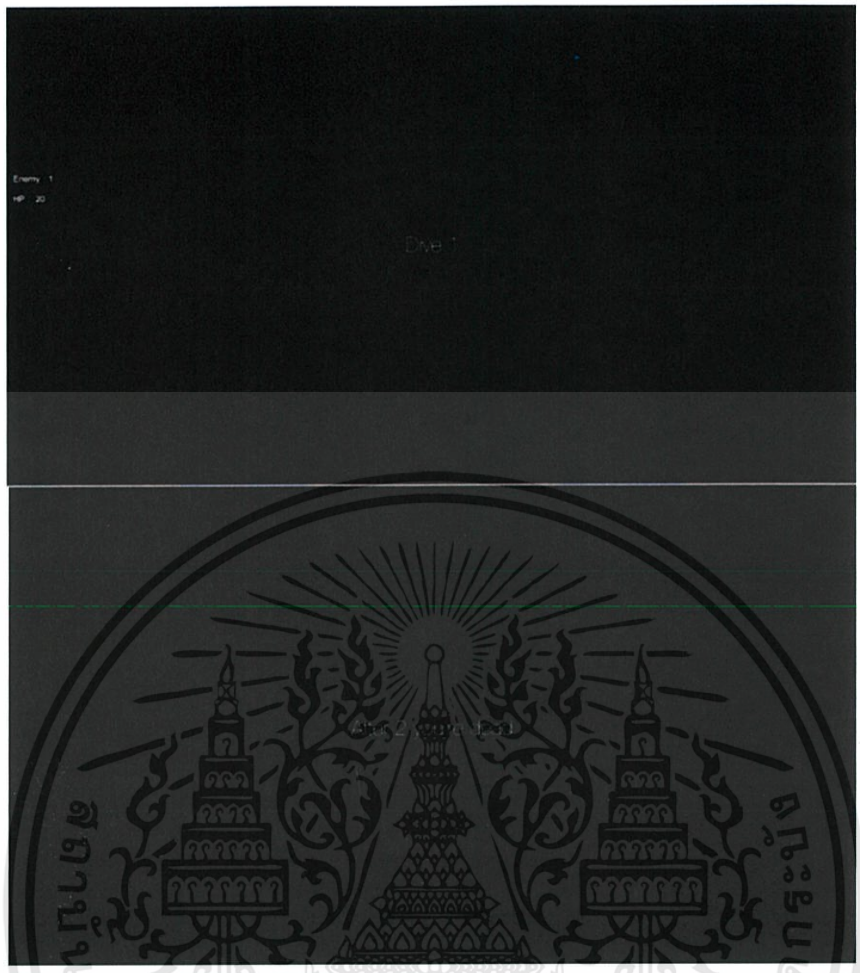


รูปที่ 3.4 หน้าจอเริ่มต้นเข้าสู่เกม

แถบเลือดของผู้เล่น สีเขียว คือเลือดคงเหลือ สีแดง คือเลือดที่เสียไป



รูปที่ 3.5 หน้าจอเมื่อเข้ามาในเกม



รูปที่ 3.6 การเปลี่ยนแปลงระหว่างฉาก

บทที่ 4
การพัฒนาระบบ

ในบทนี้จะอธิบายเกี่ยวกับโครงสร้างของเกมทั้งการพัฒนาในส่วนของตัวเกม สคริปต่าง ๆ และบางส่วนของ User Interface

4.1 การพัฒนาส่วนของตัวเกม (Game Application)

4.1.1 Manager

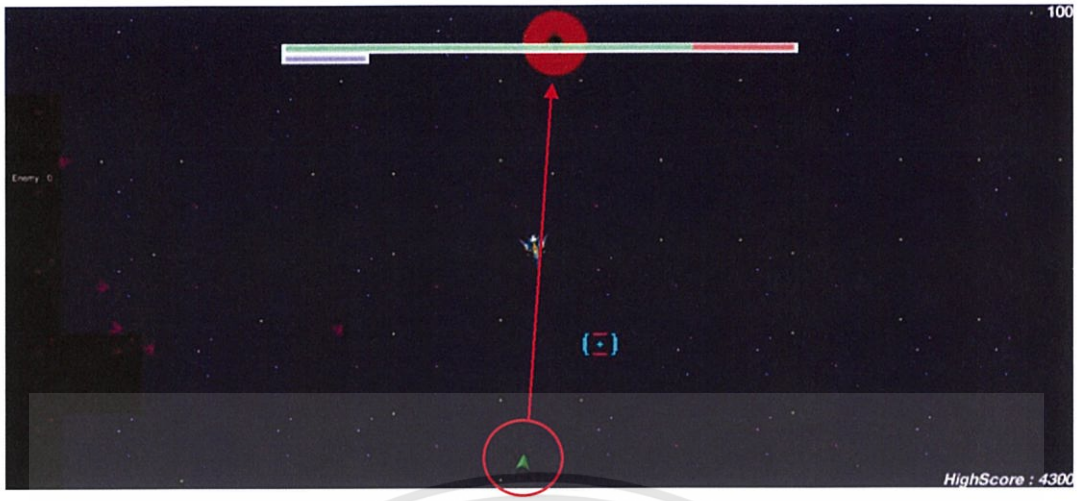
เป็นส่วนที่จัดการทุกอย่างในเกม คอยจัดการกับค่าตัวแปรต่าง ๆ ที่เกิดขึ้นและเปลี่ยนแปลงตลอดทั้งเกม เช่น เมื่อเริ่มเกม เป็นตัวกำหนดค่าเริ่มต้นแล้วเริ่มเกมขึ้นมา หลังจากกด space bar เมื่อเริ่มเกมแล้ว จะเริ่มดึงค่าต่าง ๆ มาเก็บไว้ เมื่อผู้เล่นยิงศัตรู manager ก็ปรับค่า HP ของศัตรู เป็นต้น เพื่อสนับสนุนฟังก์ชันอื่น ๆ ให้ทำงานร่วมกันได้อย่างมีประสิทธิภาพมากที่สุด เปรียบเสมือนเป็นกระดูกสันหลังของเกม

```
void InitGame() {
    doingSetup = true;
    levelImage = GameObject.Find ("LevelImage");
    levelText = GameObject.Find ("LevelText").GetComponent<Text> ();
    levelText.text = "Dive " + level;
    levelImage.SetActive (true);
    Invoke ("HideLevelImage", levelStartDelay);
    boardScript.SetupScene (level);
}
```

รูปที่ 4.1 ฟังก์ชั่น InitGame ซึ่งจะถูกเรียกทุกครั้งที่เริ่มด่าน

4.1.2 Compass (Objective Pointer)

เข็มทิศจะเริ่มชี้ไปยังทิศทางของศัตรู และ ประตูไปสู่เลเวลต่อไป เข็มทิศมีสคริปชี่งรับค่า Vector3 (x,y,z) ของจุดที่ศัตรู หรือ ประตูผ่านด่านปรากฏขึ้นมา นำมาหักลบกับ Vector3 ของผู้เล่น เพื่อหาองศาระหว่างผู้เล่นและเป้าหมาย จากนั้นนำค่าไปคำนวณหาค่าแกน z ของ Sprite เพื่อหมุนตาม



รูปที่ 4.2 เข็มทิศชี้ไปจุดเปลี่ยนด่านหลังจากชนะศัตรูหมดแล้ว

```
public GameObject player;
public GameObject target;
public GameObject enemy;

void Update () {
    enemy = GameObject.FindGameObjectWithTag("Enemy");
    target = GameObject.FindGameObjectWithTag("Exit");
    player = GameObject.FindGameObjectWithTag("Player");


    if (enemy != null) {
        Vector3 targetScreenPosition = Camera.main.WorldToScreenPoint (enemy.transform.position);
        Vector3 pointerScreenPosition = Camera.main.WorldToScreenPoint (player.transform.position);
        Vector3 distance = targetScreenPosition - pointerScreenPosition;
        float angle = Mathf.Atan2 (distance.y, distance.x) * Mathf.Rad2Deg;
        transform.localEulerAngles = new Vector3 (0f, 0f, angle - 90);
    }
    if (target != null) {
        Vector3 targetScreenPosition = Camera.main.WorldToScreenPoint (target.transform.position);
        Vector3 pointerScreenPosition = Camera.main.WorldToScreenPoint (player.transform.position);
        Vector3 distance = targetScreenPosition - pointerScreenPosition;
        float angle = Mathf.Atan2 (distance.y, distance.x) * Mathf.Rad2Deg;
        transform.localEulerAngles = new Vector3 (0f, 0f, angle - 90);
    }
```

รูปที่ 4.3 สคริปควบคุมเข็มทิศ

4.1.3 Scene Generation

การสร้างฉากของตัวเกมขึ้นมาแต่ละเลเวลจะเกิดการสร้าง 3 ขั้นตอนด้วยกัน โดยการสร้างภาพพื้นหลังจะถูกกระทำโดยสคริป MapGenerator โดยสคริปนี้จะใช้ Cellular Automata ในการสุ่มสร้างฉากขึ้นมา ส่วน ไอเทม, ศัตรู, และ อีเวนท์ จะถูกสร้างขึ้นโดย BoardManager โดยตำแหน่งของทั้งสามอย่างจะถูกสุ่มสร้างขึ้น จากพื้นที่ทั้งหมดใน Scene นั้นๆ โดย Object ที่สามารถถูกสร้างขึ้นมาได้จะถูกสุ่มขึ้นมาจาก Array ของ GameObject ที่ผูกอยู่กับ Manager โดยจำนวนของ Object ที่สร้างขึ้นจะถูกกำหนดโดย Function Mathf.Log(); ซึ่งในโปรเจคนี้ จะมีการสุ่มเลขจาก range ใกล้เคียงเข้ามาเพื่อให้การสร้างไอเทม ศัตรู และ อีเวนท์ แตกต่างกันไปในแต่ละเลเวลที่เล่น

```
public void SetupScene(int level) {

    InitialiseList ();
    minEnemyCount = 1+(int)Mathf.Log (level, 2f) * 2;
    maxEnemyCount = 1 + (int)Mathf.Log (level, 2f) * 5;
    minItemCount = (int)Mathf.Log (level, 2f) * 2;
    maxItemCount = (int)Mathf.Log (level, 2f) * 5;
    minEnvironmentCount = (int)Mathf.Log (level, 2f) * 5;
    maxEnvironmentCount = (int)Mathf.Log (level, 2f) * 5;



    LayoutObjectAtRandom (enemyTiles, minEnemyCount, maxEnemyCount);
    LayoutObjectAtRandom (environmentTiles, minEnvironmentCount, maxEnvironmentCount);
    LayoutObjectAtRandom (itemTiles, minItemCount, maxItemCount);


}
```

รูปที่ 4.4 ฟังก์ชั่น SetupScene ถูกเรียกขึ้นเพื่อจะสร้างด่านขึ้นมา

4.1.4 Cellular Automata

Cellular Automata เป็นทฤษฎีของ John Horton Conway โดยในโปรเจคนี้ ได้นำมาประยุกต์ใช้ในส่วนของการสุ่มสร้าง Background ของ Scene ซึ่งสามารถสุ่มจาก Tile ต่างๆใน Tileset ที่กำหนดไว้โดยแบ่งเป็นสองส่วนใหญ่ 1 และ 0 โดยในขั้นต้น ตัวสคริป MapGenerator จะสร้าง Array 2 มิติที่มีแต่เลข 1 และ 0 ขึ้นมาโดยเงื่อนไขของการสร้าง Array นี้ขึ้นมาคือ Random Fill Percent ซึ่งเป็น Range ตั้งแต่ 0-100 ทำให้เมื่อ Pseudorandom ตัวเลขขึ้นมาโดยเช็คกับ Fill Percent หากมากกว่าเป็น 0 หากน้อยกว่าเป็น 1

```
void SmoothMap() {
    for (int x = 0; x < width; x ++) {
        for (int y = 0; y < height; y ++) {
            int neighbourWallTiles = GetSurroundingWallCount(x,y);

            if (neighbourWallTiles > 4)
                map[x,y] = 1;
            else if (neighbourWallTiles < 4)
                map[x,y] = 0;

        }
    }
}
```

รูปที่ 4.5 ฟังก์ชั่น SmoothMap ช่วยในการแรนด้อมทำให้เกิด Pattern ขึ้นเมื่อสร้างเลเวลขึ้นมา

โดย Seed ของสคริปนี้สามารถรับ String หรือ ใช้เวลาในขณะนั้นที่แปลงเป็น String ได้ โดยหากใช้ String แล้วสุ่มซ้ำผลลัพธ์ที่ออกมาก็จะเหมือนเดิม แต่หากใช้ Time.time.tostring ผลลัพธ์ที่ได้ก็จะต่างไปเรื่อยๆเมื่อเวลาเปลี่ยนไป

```
void RandomFillMap() {
    if (useRandomSeed) {
        seed = Time.time.ToString();
    }

    System.Random pseudoRandom = new System.Random(seed.GetHashCode());

    for (int x = 0; x < width; x ++) {
        for (int y = 0; y < height; y ++) {
            if (x == 0 || x == width-1 || y == 0 || y == height -1) {
                map[x,y] = 1;
            }
            else {
                map[x,y] = (pseudoRandom.Next(0,100) < randomFillPercent)? 1: 0;
            }
        }
    }
}
```

รูปที่ 4.6 ฟังก์ชั่น RandomFillMap ช่วยในการสร้าง Array 2 มิติที่เป็น [0,1] ขั้นต้น

เมื่อได้ Array สองมิติที่มีเลข 0 กับ 1 แล้ว ตัว Scene จะสุ่มตัว prefab ที่เตรียมไว้ขึ้นมา โดยสคริปที่ใช้นี้ จะส่งผลต่อ Level Design โดยรวมของเกม โดยจะทำให้ตัวเกม สามารถที่จะสุ่มด่านที่แตกต่างกัน แต่ไม่มั่วจนเกินไป ทำให้ผู้เล่นสามารถที่จะผ่านด่านไปได้ โดยปัจจัยอื่นๆที่สามารถที่จะส่งผลต่อ Level Design ของตัวเกมขั้นพื้นฐานก็คือ จำนวน และ HP ของศัตรูที่ ผู้เล่นจะเผชิญ โดยทั้งสองอย่างนี้จะถูกเพิ่มขึ้นเรื่อยๆ โดยถูกควบคุมโดยสคริป Scene Generation อีกต่อนึง

4.1.5 Player Control

1) PlayerMobility

PlayerMobility เป็นสคริปส่วนการควบคุมตัวละครของตัวผู้เล่น โดยที่การควบคุมนั้นเป็นการควบคุมโดยรับ input จากคีย์บอร์ด และเม้าส์ เพื่อส่งค่าไปยังตัวละครโดยใช้คำสั่งการรับค่าจากปุ่มลูกศร ขึ้น ลง ซ้าย ขวา หรือ ปุ่ม w a s d ในการบังคับตัวละครให้เคลื่อนที่ไปในทิศทางต่าง ๆ ตามตำแหน่งของลูกศร และรับค่าจากเม้าส์ เพื่อทำการหมุนหน้าตัวละคร โดยที่หน้าของตัวละครจะหันหน้าหาลูกศรของเม้าส์เสมอ และรับคำสั่งจากเม้าส์ ในการยิงกระสุนเพื่อทำลายศัตรู ซึ่งการรับค่าจากคีย์บอร์ด นั้น unity engine มีฟังก์ชันการรับค่าจากลูกศรของคีย์บอร์ดโดยตรงเพื่อควบคุมตัวละครดังภาพต่อไปนี้

```
float verticalInput = Input.GetAxis ("Vertical");
float horizontalInput = Input.GetAxis ("Horizontal");
Vector2 movement = new Vector2 (horizontalInput, verticalInput);
GetComponent<Rigidbody2D>().velocity = (movement * speed)*(Time.deltaTime);
```

รูปที่ 4.7 ฟังก์ชันรับค่าจากลูกศรของคีย์บอร์ดสำหรับควบคุมตัวละคร

จากรูปที่ 4.8 จะเห็นการประกาศตัวแปรซึ่งใช้สำหรับรับค่าจากคีย์บอร์ด ซึ่งเป็นการรับจากปุ่มลูกศร โดยจะแบ่งเป็นรับค่าจากลูกศรแนวตั้ง และแนวนอน เพื่อใช้ควบคุมการเคลื่อนที่ของตัวละคร และการควบคุมการหมุนหน้าตัวละคร ซึ่งจะมีการหมุนตามตำแหน่งของ cursor นั้น ผู้พัฒนาได้ใช้ฟังก์ชัน quaternion เป็นตัวรับค่าตำแหน่งของ cursor เพื่อส่งค่าให้กับคำสั่งการหมุนหน้าตัวละคร คือ transform.rotation เพื่อเป็นการสั่งหมุน และควบคุมองศาการหมุนด้วย transform.eulerAngles ดังภาพต่อไปนี้

```
var mousePosition = Camera.main.ScreenToWorldPoint (Input.mousePosition);
Quaternion roit = Quaternion.LookRotation (transform.position - mousePosition, Vector3.forward);
transform.rotation = roit;
transform.eulerAngles = new Vector3 (0, 0, transform.eulerAngles.z);
GetComponent<Rigidbody2D>().angularVelocity = 0.5f;
```

รูปที่ 4.8 ฟังก์ชันที่ใช้ควบคุมองศาการหมุน

นอกจากนี้แล้ว PlayerMobility ยังเป็นตัวเก็บตัวแปรต่างในเกม ทั้งในส่วนของ ค่าพลังชีวิต ค่าพลังพิเศษ แล้วส่งค่าที่มีให้กับ script ที่ชื่อว่า BarController เพื่อทำการปรับขนาดของหลอดพลังชีวิตและพลังพิเศษให้ตรงกับจำนวนจริง ดังรูปที่ 4.9 จากรูปจะเห็นว่าเมื่อมีการชนกันของวัตถุ และมีการตรวจสอบชนิดของวัตถุที่ชนแล้วนั้น จะมีการเรียกใช้ตัวแปรจาก script อื่นเพื่อส่งคำสั่งเรียกการใช้งานจากนอก script

```
if (other.gameObject.CompareTag("Enemy")) {
    playerHP -= 2;
    FindObjectOfType<BarController>().decresebar2();
    //Instantiate(Explosion, transform.position, transform.rotation);
    //AudioSource.PlayClipAtPoint(explosion, transform.position);
    //Destroy(gameObject);
}

if (other.gameObject.CompareTag("enemyBullet"))
{
    FindObjectOfType<BarController>().decresebar();
    playerHP -= 1;
    Destroy(other.gameObject);
}
```

รูปที่ 4.9 ฟังก์ชันที่ทำหน้าที่ปรับขนาดของหลอดพลังชีวิต และ พลังพิเศษ

4.1.6 Hit Detection

Hit Detection เป็นการเช็คการชนกันของวัตถุภายในเกมส์ โดย unity engine นั้นมีเครื่องมือสำหรับเช็คการชนกันของวัตถุทั้งในรูปแบบสองมิติ และสามมิติ เรียกว่า Physic2d และ Physic3d ซึ่งจะแบ่งย่อยไปเป็นรูปทรงต่าง ๆ ให้ผู้พัฒนาได้เลือกใช้ อาทิ เช่น box collider, circle collider เป็นต้น โดยการเช็คการชนกันของวัตถุนั้น จะเช็คด้วยการใส่ collider ให้กับวัตถุ แล้วใส่ tag เพื่อเป็นตัวแยกประเภทของวัตถุต่าง ๆ ที่ชนกัน โดยมีการเขียนสคริปเพื่อเช็คการชนและตรวจสอบด้วยการตรวจจาก tag ของวัตถุที่ชนว่าคือวัตถุขนิดใด

```
void OnCollisionEnter2D(Collision2D other)
{

    if (other.gameObject.CompareTag("Exit")) {
        Invoke ("Restart", restartLevelDelay);
        Manager.instance.HP = playerHP;
        Manager.instance.score = score;
    }

    if (other.gameObject.CompareTag("Enemy")) {
        playerHP -= 2;
        FindObjectOfType<BarController>().decresebar2();
        //Instantiate(Explosion, transform.position, transfo
        //AudioSource.PlayClipAtPoint(explosion, transform.p
        //Destroy(gameObject);
    }
```

รูปที่ 4.10 ใช้ if ในการเช็คการชนโดยมีการตรวจสอบว่าวัตถุที่ชนนั้นเป็นชนิดใด ผ่านคำสั่ง compare.tag(); เพื่อเป็นการตรวจสอบชนิดของวัตถุที่ชน แล้วจึงดำเนินการต่อ

4.1.7 Scoring

Scoring เป็นการเก็บค่าคะแนนที่ได้ในการเล่นเกมส์ในแต่ละครั้ง โดยคะแนนที่ได้จะได้จากการทำลายศัตรูในเกมส์ เมื่อทำลายศัตรูได้ คะแนนจะเพิ่มขึ้นไปเรื่อยๆ จนกว่าผู้เล่นจะถูกทำลาย และเมื่อผู้เล่นถูกทำลาย จะมีการนำคะแนนที่ได้จากการเล่นมาเปรียบเทียบกับคะแนนสูงสุดในการเล่น ซึ่งถ้าหากได้มากกว่า ก็จะทำการเขียนทับคะแนนเดิม แต่ถ้าน้อยกว่าก็จะไม่ทำการบันทึก

```
if (highScore < score) {
    highScore = score;
}
```

รูปที่ 4.11 เปรียบเทียบค่า high score กับ score ที่เล่นได้
บันทึก score ทับค่าเดิม ถ้า score มีค่ามากกว่า high score

ค่า high score นั้นจะถูกเก็บไว้ในฟังชันพิเศษของ unity คือ playerpref ซึ่งเป็นฟังก์ชันพิเศษที่ทำการเก็บค่า และรับค่าต่าง ๆ

```
public void Save() {
    PlayerPrefs.SetInt (highScoreKey, highScore);
    PlayerPrefs.Save ();
    Initialise ();
}
```

รูปที่ 4.12 การใช้ฟังก์ชัน playerpref ภายในเกม

4.1.8 Passing Variable

Passing Variable เป็นการส่งหรือดึงค่าต่าง ๆ ผ่านสคริปอื่น ๆ นอกเหนือจากที่มีประกาศไว้ในสคริปที่มีอยู่ โดยการเรียกใช้ค่าต่าง ๆ จากสคริปอื่นนั้น ตัวแปรที่ต้องการจะดึงค่าไปใช้งานจะต้องเป็นตัวแปรแบบ public จึงจะสามารถดึงค่าออกไปใช้ได้ โดยแต่ละครั้งที่เริ่มเลเวลใหม่ Start() จะเรียกตัวแปรที่ถูกบันทึกไว้กลับขึ้นมาเพื่ออัพเดทตัวแปรของสคริปในเลเวลให้ค่าส่งต่อกันมาจากเลเวลก่อนหน้า โดยเมื่อผู้เล่นเข้าประตูสู่เลเวลถัดไป ทาง PlayerMobility จะส่งค่าไปเซฟไว้ที่ Manager

```
void Start(){
    playerHP = Manager.instance.HP;
    score = Manager.instance.score;
}
```

```
void setOnstart()
{
   curHp = FindObjectOfType<PlayerMobility2>().playerHP;

    float calHpBar = curHp / Maxhp;

    sethealtbar(calHpBar);
}
```

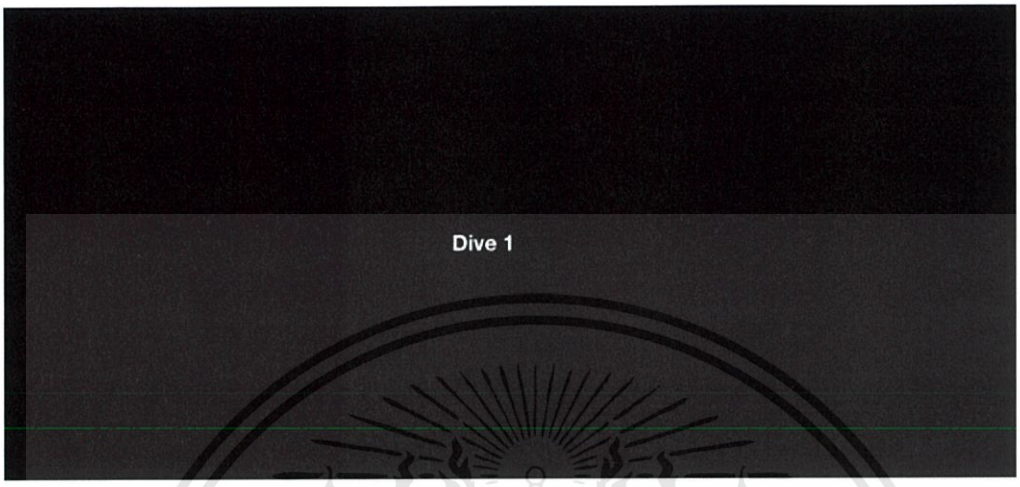
รูปที่ 4.13 ฟังก์ชันที่ทำหน้าที่ส่งผ่านค่าที่เก็บไว้

4.2 ส่วนติดต่อกับผู้ใช้ (User Interface: UI)

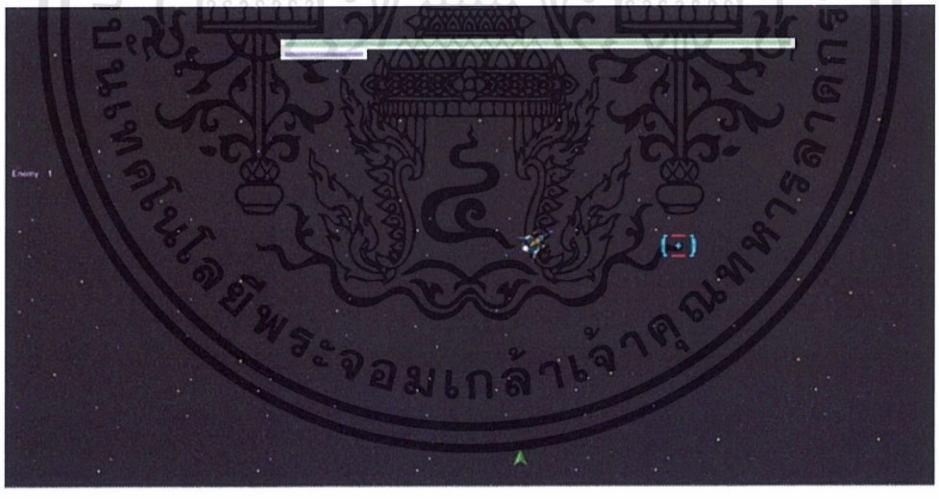
เมื่อเข้าเกมจะพบกับหน้าต่างแบบเรียบง่ายแสดงคะแนนสูงสุดที่เคยทำไว้ พร้อมอธิบายว่าเมื่อต้องการเล่นให้กด space bar

รูปที่ 4.14 หน้าต่างเริ่มต้น

หลังจากกด space bar สีพื้นหลังจะเปลี่ยนจากสีขาวเป็นสีดำ เพื่อให้ผู้เล่นตื่นตัว ตัวหนังสือแสดงเลเวลอย่างชัดเจนกลางหน้าจอ ผู้เล่นจะพบกับหน้านี้ทุกครั้งที่เปลี่ยนด่าน เมื่อรอสักครู่จะเข้าสู่เกม



รูปที่ 4.15 หน้าจอแสดงเลเวลที่กำลังจะเล่น

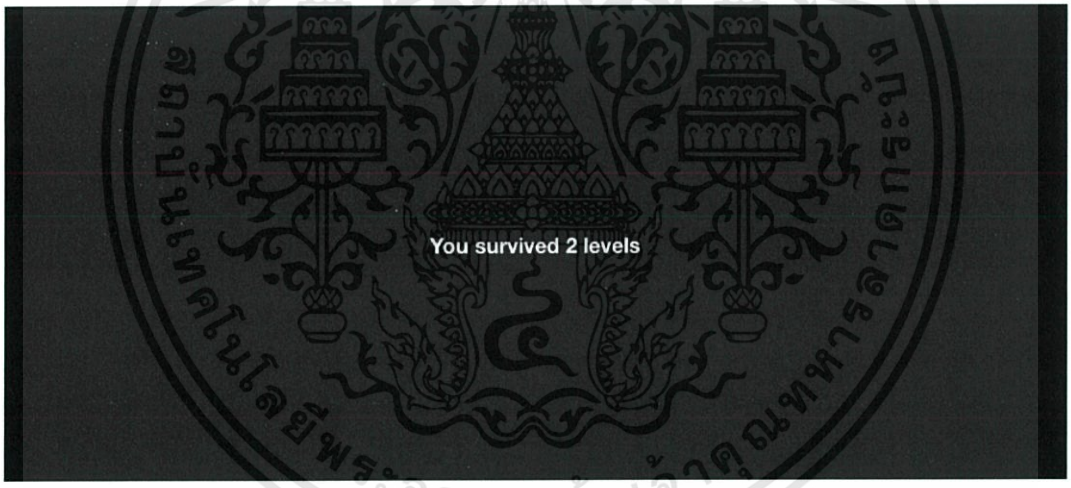


รูปที่ 4.16 ส่วนของ gameplay

จากรูปที่ 4.15 เมื่อเวลาผ่านไปจะปรากฏ gameplay ดังรูปที่ 4.16 โดยฉากจะถูกสุ่มขึ้นปรากฏยานที่ควบคุมตรงกลางจอ cursor เปลี่ยนเป็นเป้ายิง แสดงแถบ HP และ MP ด้านบน และเข็มทิศด้านล่าง ศัตรูจะถูกสุ่มเกิด ผู้เล่นต้องทำลายศัตรูให้หมด

รูปที่ 4.17 เมื่อทำลายศัตรูหมด ประตูเปลี่ยนด่านปรากฏขึ้น

จากรูปที่ 4.17 ประตูเปลี่ยนด่านปรากฏขึ้น เข็มทิศจะชี้ไปทางที่ประตูเปลี่ยนด่านเกิดขึ้น ผู้เล่นต้องเคลื่อนที่เข้าไปยังประตูเพื่อไปยังด่านถัดไป หลังจากนั้นจะปรากฏหน้าต่างดังรูป 4.15 อีกครั้ง



รูปที่ 4.18 หน้าจอเมื่อผู้เล่นแพ้

จากรูปที่ 4.18 เป็นหน้าจอเมื่อผู้เล่นแพ้ แสดงเลเวลที่แพ้ล่าสุด ถ้าต้องการเริ่มเกมใหม่ให้กด r เพื่อ restart โดยกลับไปที่หน้าจอเริ่มแรกอีกครั้ง

บทที่ 5

สรุปผลโครงงานและข้อเสนอแนะ

ในบทนี้กล่าวสรุปการทำงานโดยการใช้ Unity รวมถึงเทคนิคในการสร้างเกมต่าง ๆ อัลกอริทึมในการเดิน การสุ่ม การทำลายศัตรู การสร้างภาพเคลื่อนไหว และการทำงานร่วมกันของทุกอย่างที่กล่าวมา การทำงานร่วมกันหลาย ๆ ระบบอาจก่อให้เกิดปัญหาที่ตามมา ทั้งที่แก้ได้และแก้ไม่ได้ อย่างไรก็ตามระบบที่ซับซ้อนทำให้สามารถพัฒนาเกมออกไปได้ในหลายทิศทาง ในส่วนนี้จะกล่าวถึงปัญหาและข้อจำกัดที่เกิดขึ้นระหว่างพัฒนา รวมถึงข้อเสนอแนะแนวทางการศึกษาต่อไปในอนาคต ที่อาจมีประโยชน์ต่อผู้ที่สนใจศึกษา หรือนำไปพัฒนาให้เกมมีการทำงานและรูปแบบการเล่นที่ดียิ่งขึ้นในเวอร์ชันถัดไป

5.1 สรุปผลโครงงาน

ตัวเกมถูกพัฒนาโดยมีพื้นฐานจากเกม Shooting 2 มิติ โดยเกมมีศัตรูที่คอยทำลายผู้เล่น ศัตรูเหล่านี้จะสร้างความท้าทายให้กับผู้เล่น ผู้เล่นจะต้องทำลายศัตรูให้หมด ก่อนที่ศัตรูจะทำลายผู้เล่นได้ ผู้เล่นอาจต้องใช้กลยุทธ์ในการกำจัดศัตรูที่ต่างๆไป รวมไปถึงต้องใช้ประสาทสัมผัสที่ว่องไว การตอบสนองที่เฉียบคมและความชำนาญในการผ่านไปแต่ละด่าน รูปแบบของการแสดงผลจะใช้ภาพสองมิติ แต่ใช้แอนิเมชั่นที่ลื่นไหล เพื่อดึงดูดความสนใจของผู้เล่น

ตลอดการเล่นเกมศัตรูจะพยายามทำลายผู้เล่นอยู่ตลอดเวลา โดยการสาดกระสุนจำนวนมากใส่ผู้เล่น ในแต่ละด่านจำนวนของศัตรูจะเพิ่มขึ้น ศัตรูทุกตัวจะพยายามก่อความวุ่นวาย ทำให้ผู้เล่นผ่านด่านได้ยากขึ้น อย่างไรก็ตามทุกครั้งที่ผู้เล่นสามารถกำจัดศัตรูได้ ผู้เล่นจะได้คะแนนเพิ่มขึ้นเรื่อยๆ หากผู้เล่นสามารถทำลายสถิติการเล่นเดิมได้ คะแนนสูงสุดที่ทำได้จะนำไปเก็บไว้และแสดงในหน้าแรก เพื่อสร้างความภูมิใจและสร้างแรงจูงใจให้ผู้เล่นคนอื่นอยากทำลายสถิติ

โดยในฝั่งของตัวเกม เมื่อผู้เล่นได้ผ่านด่านขึ้นไปเรื่อยๆ ตัวเกมจะสร้างด่านที่ต่างออกไปขึ้นมาเรื่อยๆ และ เพิ่มจำนวนศัตรูที่ผู้เล่นต้องทำลายขึ้นเรื่อยๆ อย่างไรก็ตาม ตัวเกมจะสร้างไอเทมเพิ่มความสามารถต่างๆของผู้เล่นขึ้นมาเพื่อช่วยผู้เล่น ไม่ให้ตัวเกมยากจนไร้เหตุผลเกินไป

5.2 ปัญหาและข้อจำกัดในการทำงานของระบบ

5.2.1 คอนโทรลของตัวเกมส์ทำให้ผู้เล่นสับสน

ปัญหานี้เกิดขึ้นในส่วนดีไซน์ขั้นต้น โดยทางผู้จัดทำได้ดีไซน์ให้ตัว Avatar ที่ผู้เล่นบังคับเดินหน้าตามทิศที่เมาส์ของผู้เล่นชี้ไป แต่หลังจากได้รับความคิดเห็นจากผู้เล่นจำนวนไม่น้อยว่า การทำเช่นนั้น ทำให้การควบคุม Avatar ยากเกินไป ทางผู้จัดทำจึงเปลี่ยนการควบคุมให้ตัว avatar relative กับหน้าจอ และ เมาส์ไว้ใช้เล็งและยิงอย่างเดียวแทน

5.2.2 Randomization ไม่สามารถควบคุมได้ และ ไม่ก่อให้เกิดประโยชน์ ในด้าน Game Design

ปัญหานี้เกิดขึ้นใน Implementation แรกๆของตัวโปรเจคที่ทางผู้จัดทำได้ลองเอา Randomization ใส่เข้ามาในโปรเจค โดยที่ไม่มี Algorithm ใดๆ ช่วยควบคุมในการแรนด้อม ทำให้ด่านที่แรนด้อมออกมา มั่วจนเกินไปทำให้ผู้เล่นไม่สามารถผ่านได้ โดยทางผู้จัดทำได้ทำการแก้ปัญหาที่เกิดขึ้นโดย ใช้ Cellular Automata เข้ามาช่วยในการแรนด้อมเพื่อทำให้การแรนด้อมที่เกิดขึ้น มีความเป็นด่านที่ผู้เล่นสามารถผ่านได้มากขึ้น

5.2.3 Scene Generation ใช้ทรัพยากรเครื่องเกินความจำเป็น

ปัญหานี้เกิดในช่วงท้ายของโปรเจค โดยผู้จัดทำได้ใช้ Algorithm มาช่วยในการแรนด้อมแล้ว แต่ตัวเอนจิ้นยังไม่สามารถสร้างฉากออกมา โดยใช้ทรัพยากร Hardware ที่ผู้เล่นมีอยู่ได้อย่างเต็มประสิทธิภาพ โดยทางผู้จัดทำได้ทำการตัดการสร้าง Object ที่เป็น Background ออกให้มากที่สุดเท่าที่จะเป็นไปได้

5.3 ข้อเสนอแนะในการพัฒนาระบบต่อไป

5.3.1 พัฒนาให้ตัวเกมรองรับ Online Multiplayer

การที่จะทำให้ตัวเกมส์สามารถที่จะรองรับการเล่นออนไลน์ผ่านอินเตอร์เน็ตได้ สามารถทำได้โดย ใช้ Framework แยกอย่าง Photon เข้ามาช่วย หรือ ใช้ Feature Online ของตัว Unity เอง (ต้องเสียค่าใช้จ่ายรายเดือนตามจำนวนผู้เล่น) โดยทางตัวเอนจิ้นนั้นได้เอื้ออำนวยต่อการทำให้ เกมส์สามารถที่จะเล่นออนไลน์ได้อย่างง่ายอยู่แล้ว

5.3.2 ต่อยอด Algorithm ที่ใช้ในการแรนด้อมด่าน

Algorithm ที่ใช้ในปัจจุบัน ยังไม่ถือว่าเป็นที่สุดในการที่จะสร้างฉากที่สามารถที่จะเล่นไปเรื่อยๆ เนื่องจากมีข้อจำกัดว่า Structure ของตัวเกมจะต้องเป็น Tile Based ดั้งนั้นเพื่อที่จะแก้ไขปัญหานี้ เราสามารถที่จะ ใช้ Algorithm อย่าง Marching Square และ Z-Order Curve เข้ามาช่วยในการแรนด้อมได้

5.3.3 พอร์ทไปยัง Platform อื่น

เนื่องจาก Project ถูกสร้างใน Unity ดังนั้นตัวเอนจิ้นนั้นช่วยให้ง่ายในการพอร์ทไปยัง platform อื่นๆอยู่แล้ว โดยสิ่งที่ทางผู้จัดทำต้องทำเพิ่มก็คือ เขียนสคริป Control สำหรับการควบคุมของ User ใหม่ นอกจากนั้นทางตัวเอนจิ้นสามารถที่จะนำโค้ดเก่าที่มีอยู่มา build เฉพาะสำหรับ platform ใหม่ได้เลย

5.3.4 เพิ่มความหลากหลายให้กับ Prefab ที่สามารถแรนด้อมออกมาได้

ระบบ Randomization จะไม่จำเจก็ต่อเมื่อ Content ที่สามารถแรนด้อมได้มีความน่าสนใจและหลากหลายขึ้นกว่าเดิม โดยนอกจากความหลากหลายจะทำให้ตัวเกมส์น่าสนใจขึ้นแล้ว ยังเป็นหนึ่งในทางที่จะเพิ่มสไตล์ของศัตรูทำให้ ผู้เล่นต้องปรับตัวเพื่อที่จะจัดการกับศัตรูที่มีความสามารถเฉพาะทางที่ต่างๆกันไป

5.3.5 เพิ่มระบบ Pathfinding และ พฤติกรรมต่างๆ ให้กับศัตรู

ทำให้ศัตรูมีแพทเทิร์นการโจมตี และ พฤติกรรมต่างๆกันไป เมื่อเผชิญหน้ากับ ผู้เล่น และอุปสรรค์ต่างๆ ที่ระบบได้สร้างมาในฉากนั้นๆ

อ้างอิง

[1] F. Muliawan, " Enemy Speed Control on Shoot em' Up Game with Fuzzy Takagi Sugeno Method," in The 2014 Third ICT International Student Project Conference, 2014, pp. 87-90.

[2] J.T., Velikovsky, "Flow Theory, Evolution & Creativity: or, 'Fun & Games'", Proceedings of the 2014 Conference on Interactive Entertainment, pp. 1-10. NSW, Australia.

[3] The Art of Screenshake [online]:
https://www.youtube.com/watch?v=AJdEqssNZ-U

[4] Suwan Juntiwasarakij, Ph.D., "Control-display relationships", Human Computer Interaction

[5] Game Design Document [online]:
https://www.assembla.com/wiki/show/2b2n_gameGroup/Game_Design_Document

[6] เพิ่มพูน พลสุวรรณา, สรวิชญ์ โมกขาว ", การพัฒนาเกมกระดานสามมิติด้วยยูนิตี้เอนจิ้น ", ปพ. พ938ก 2554, ISBN 137561, 2554