รายงานผลโครงการและงานวิจัย

เรื่อง

เกม The Everlasting

นายณัฐกร พวงทอง

นายวชิรวิทย์ สุระนันท์

โครงการนี้เป็นส่วนหนึ่งของวิชาโครงการ รหัส 30128-8501

หลักสูตร ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

สาขาวิชาเทคโนโลยีคอมพิวเตอร์

ภาคเรียนที่ 2 ปีการศึกษา 2566

วิทยาลัยเทคนิคจันทบุรี สถาบันการอาชีวศึกษาภาคตะวันออก

ใบรับรองอนุปริญญานิพนธ์
สาขาเทคโนโลยีคอมพิวเตอร์ วิทยาลัยเทคนิคจันทบุรี

หัวข้อวิจัย	:	เกม The Everlasting
ผู้ดำเนินการวิจัย	:	นายณัฐกร พวงทอง
		นายวชิรวิทย์ สุระนันท์
ที่ปรึกษา	:	นายฉัตรพฤกษ์ สีทิม
		นางสาวประภาพร บันเทา
สาขาวิชา	:	เทคโนโลยีคอมพิวเตอร์
สาขางาน	:	คอมพิวเตอร์ระบบเครือข่าย
ปี พ.ศ.	:	2566

สาขาวิชาเทคโนโลยีคอมพิวเตอร์ให้นับอนุปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรประกาศนียบัตรชีพขั้นสูง สาขาวิชาเทคโนโลยีคอมพิวเตอร์

.....หัวหน้าสาขาวิชาเทคโนโลยีคอมพิวเตอร์
(นายศัตราวุธ ศรีชาติ)

คณะกรรมการสอบโครงการ

.....ประธานกรรมการ
(นายฉัตรพฤกษ์ สีทิม)

.....กรรมการ
(นายปิยพงษ์ เกตุภูงา)

.....กรรมการ
(นางสาวประภาพร บันเทา)

หัวข้อวิจัย เกม The Everlasting

ผู้จัดทำ นายณัฐกร พวงทอง

นายวชิรวิทย์ สุระนันท์

ที่ปรึกษา นายฉัตรพฤกษ์ สีทิม

นางสาวประภาพร บันเทา

สาขาวิชา เทคโนโลยีคอมพิวเตอร์

สาขางาน คอมพิวเตอร์ระบบเครือข่าย

ปีการศึกษา 2566

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์ เพื่อศึกษาการใช้โปรแกรม Unity , ภาษา C# , Photoshop และ Aseprite ผ่านทาง เกมEverlasting เพื่อสามารถให้ผู้เข้าใจหลักการสร้างเกม การใช้ภาษา C# ในการสร้างเกม การออกแบบต่างๆ และทำให้ผู้เล่นได้ฝึกการใช้ไหวพริบ และความรอบครอบในเกม Everlasting

การพัฒนาเกม Everlasting กำเนินการวิเคราะห์และออกแบบเกมโดยมีอนิเมะและนิยายในโลก MMORPG เป็นต้นแบบ ที่ผู้คนสามารถพบเห็นได้บ่อยๆ ทั้งในเกมอื่นๆ ภาพยนตร์ การ์ตูน หรือสื่ออื่นๆโดยใช้โปรแกรม Unity ในการสร้างเกม ออกแบบแมพ จัดวางตัวละคร จัดว่างมอนสเตอร์ต่างๆ โดยโปรแกรม Unity ใช้ ภาษา C# ที่เป็นคำสั่งในโปรแกรมกำหนดมาให้

จากการพัฒนาเกม Everlasting พบว่า เกมนี้เหมาะสำหรับผู้เล่นกลุ่มวัยรุ่นที่มีความ สนใจทางนิยายและชอบเล่นเกม โดยผู้เล่นได้เพลิดเพลินพร้อมกับการฝึกทักษะการใช้ไหวพริบ ไปพร้อมกัน และทำให้ผู้เล่นได้รู้จักการใช้ความรอบครอบในการสู้กับเหล่ามอนสเตอร์และบอส

Research Title	The Everlasting
Research	Mr.Nattagorn poungtong
	Mr.Wachirawit Suranan
Research Consultants	Mr.Chattapluek Seethim
	Ms.Prapaporn Banthao
Organization	Technology computer departmeny
	Chanthaburi technical collage
Academic Year	2023

Abstract

This project has the objective To study the use of Unity, C# language, Photoshop and Aseprite via Everlasting Games To be able to make people understand the principles of creating games, using the C# language to create.
Various design games and allows players to practice using their wits and comprehensiveness in the game Everlasting

Everlasting game development analyzes and designs games based on anime and novels in the MMORPG world as models that people can often see. Both in other games, movies, cartoons, or other media using the Unity program to create games, design maps, and place characters. Organize various monsters by using the Unity program using C# language which is a command in the program given to you.

From the development of the game Everlasting, it was found that this game is suitable for young players who have Interested in novels and likes playing games. The players will have fun while practicing their cunning skills at the same time and make the players learn how to use thoroughness in fighting monsters and bosses.

กิตติกรรมประกาศ

โครงการเรื่องเกม The Everlasting ฉบับนี้สามารถดำเนินจนประสบความสำเร็จลุล่วงไปด้วยดี เนื่องจากได้รับความอนุเคราะห์และสนับสนุนช่วยเหลือดูแลเป็นอย่างดียิ่ง จาก คณะครูอาจารย์แผนกเทคโนโลยีคอมพิวเตอร์ ที่ได้กรุณาให้คำปรึกษา ข้อคิด ข้อแนะนำ และปรับปรุงแก้ไขข้อบกพร่องต่างๆ จนกระทั่งโครงการครั้งนี้สำเร็จเรียบร้อยด้วยดี ผู้จัดทำโครงการขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้

ขอขอบคุณคณะครูอาจารย์ แผนกเทคโนโลยีคอมพิวเตอร์ ที่กรุณาอบรมสั่งสอนให้ความรู้ในการเขียนโปรแกรมและการออกแบบเพื่อให้คณะผู้จัดทำสามารถนำมาประยุกต์ใช้ในการสร้างผลงานสำหรับโครงงานเรื่องเกม The Everlasting นี้ให้สำเร็จลุล่วงด้วยดี

ท้ายที่สุด คณะผู้จัดทำโครงการหวังว่า โครงการฉบับนี้ จะเป็นประโยชน์ต่อผู้ใช้งาน ที่สนใจไม่มากก็น้อย

คณะผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ(ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงงาน

ในยุคปัจจุบันนี้ถือได้ว่าเกมมีบทบาทสำคัญสำหรับเยาวชนเป็นส่วนมาก เกมมีหลากหลายประเภทและแต่ละประเภทก็แตกต่าง ซึ่งแต่ละประเภทก็จะมีการเล่นที่ไม่เหมือนกัน เช่น เกมแนว RPG (Role-playing game คือเกมประเภทหนึ่งที่ผู้เล่นรับบทเป็นตัวละครหนึ่งในเกม โดยเล่นตามกฎกติกาของเกมผ่านการป้อนคำสั่งและเลือกเงื่อนไขที่เกมกำหนดมา โดยผลลัพธ์ที่เกิดจะแตกต่างกันตามเงื่อนไขที่เลือก โดยเกมอาจจะเป็นทั้งลักษณะ การเล่นโดยเขียนในกระดาษ วิดีโอเกมหรือคอมพิวเตอร์เกมก็ได้) , MMORPG (ย่อมาจากคตำว่า Massive Multiplayer Online ซึ่งจะแปลได้ตรงๆตัวว่า ออนไลน์รวมกันเป็นจำนวนมาก มันก็คือเกมประเภทเกมออนไลน์ต่างๆนั้นเอง ที่เอาทุกคนมารวมตัวกัน เหมือนกับการสร้างโลกสเหมือนจริงขึ้นมา โดยมีผู้เล่นสวมบทเป็นหนึ่งในโลกสเหมือนจริงนั้นๆ ขึ้นอยู่กับรูปแบบเกมที่ผู้พัฒนาเกมต้องการ) เกมแนว FPS (FPS ย่อมาจาก First-person shooter หรือที่เรียกกันว่าเกมยิงมุมมองบุคคลที่หนึ่งเป็นเกมยิง สามมิติรูปแบบหนึ่ง ที่เสนอมุมมองบุคคลที่หนึ่ง โดยผู้เล่นจะเห็นการกระทำผ่านสายตาของตัวละครผู้เล่น เกมยิงมุมมองบุคคลที่หนึ่งมักถูกจัดให้เป็นคนละประเภทกับเกมยิงด้วยปืนเบา เกมประเภทหนึ่งที่เป็น มุมมองบุคคลที่หนึ่งแต่จะบังคับโดยใช้ปืนเบา) เป็นต้น แต่ในปัจจุบันเกมมีเกมอีกหลายประเภททั้งดีละไม่ดี ดังนั้นหากเล่นเกมที่ไม่ได้ส่วนช่วยในการพัฒนาสมองก็อาจเป็นการเล่นเกมที่ส่งผลเสียได้ ทั้งนี้ที่กล่าวมาถือเป็นบันดาลใจให้จัดทำพัฒนาเกม The Everlasting ขึ้นมา

เนื่องจากในปัจจุบันเกมนั้นเข้าถึงได้ง่ายจึงมีเกมที่เนื้อหาเกมไม่ดีเนื้อหารุนแรงและไม่เหมาะสมกับเด็กและเยาวชนอาจทำให้เกิดการใช้ความรุนแรงได้ผู้จัดทำโครงงานจึงมีความคิดที่จะสร้างเกมที่สร้างสรรค์และช่วยพัฒนาทักษะการเล่นเกมของผู้เล่นเนื่องจากผู้จัดทำโครงงานมีความสนใจและชื่นชอบในการเล่นเกมและได้มีการศึกษาโค้ดและการใช้งานโปรแกรมในการสร้างเกมผู้จัดทำจึงมีจุดมุ่งหมายในการสร้างเกมนี้ขึ้นมา

จากเหตุผลข้างต้นจึงเป็นสาเหตุให้ผู้จัดทำโครงงานมีความต้องการที่จะสร้างเกม Everlasting ที่ให้ความสนุกสนานตื่นเต้นแก่ผู้เล่นโดยใช้โปรแกรม Unity3D และภาษาC#ขึ้นมาเพื่อต้องการให้ผู้เล่นที่ได้พัฒนาทักษะการเล่นเกมของผู้เล่นและได้ประโยชน์จากการเล่นเกม มากกว่าการใช้ความรุนแรง

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อสร้างเกม The Everlasting

1.2.2 เพื่อศึกษาการใช้โปรแกรม Unity , ภาษา C# , Photoshop และ Aseprite

1.2.3 เพื่อหาประสิทธิภาพของเกม Everlasting

1.3 ขอบเขตของโครงงาน

1.3.1 ประชากรและกลุ่มตัวอย่าง

1.3.1.1 ประชากร คือ นักศึกษาแผนกเทคโนโลยีคอมพิวเตอร์

1.3.1.2 กลุ่มตัวอย่าง คือ นักศึกษาแผนกเทคโนโลยีคอมพิวเตอร์ ระดับ ปวส.2/1

1.3.2 ตัวแปรที่ศึกษา

1.3.2.1 ตัวแปรต้น เกม The Everlasting

1.3.2.2 ตัวแปรตาม ประสิทธิภาพของเกม ด้านความสวยงาม ด้านความเสถียร และความน่าสนใจของเกม The Everlasting

1.3.3 เครื่องมือที่ใช้ในโครงการ

1.3.3.1 แบบสอบถามความพึงพอใจของผู้เล่นเกม Everlasting และ ประสิทธิภาพของเกม

1.3.4 วิธีเก็บรวมข้อมูล

1.3.4.1 กำหนดกลุ่มประชากรและกลุ่มตัวอย่าง

1.3.4.2 ศึกษาข้อมูลเบื้องต้น

1.3.4.3 ออกแบบและสร้างเกม Everlasting

1.3.4.4 นำไปทดลองเล่นจริง

1.3.4.5 นำแบบสอบถามความพึงพอใจให้กลุ่มผู้ทดลองได้ความคิดเห็น

1.3.4.6 เก็บรวบรวมแบบสอบถามทั้งหมดและนำมาประมวลผลให้สมบูรณ์ต่อไป

1.3.5 สถิติที่ใช้ในการวิเคราะห์

1.3.5.1 ค่าเฉลี่ย โดยใช้สูตร

$\vec{x} = \frac{\sum x}{n}$ ค่าเฉลี่ย

$\overline{x}$ = ค่าเฉลี่ยของคะแนน

$\sum x$ = ผลรวมของคะแนน

n = จำนวน

1.3.5.2 ส่วนเบี่ยงเบนมาตรฐานโดยใช้สูตร

$$S.D. = \sqrt{\frac{n\sum X^2 - (\sum X)^2}{n(n-1)}}$$

S.D. แทน ส่วนเบี่ยงเบนมาตรฐาน

X แทน คะแนนแต่ละคน

N แทน จำนวน

1.4 ประโยชน์ที่ได้รับ

1.4.1 ได้เกม Everlasting

1.4.2 ได้ใช้โปรแกรม Unity , ภาษา C# , Photoshop และ Aseprite

1.4.3 ประสิทธิภาพของเกม Everlasting มีความแม่นยำร้อยละ 80

1.5 สมมติฐานของโครงการและวิจัย

ได้เกมที่มีคุณภาพ ที่สามารถมอบความบันเทิง ความสนุกสนาน และความตื่นเต้นได้

1.6 นิยามศัพท์

1.6.1 เกม คือ เป็นลักษณะของกิจกรรมของมนุษย์ เพื่อประโยชน์อย่างใดอย่างหนึ่ง เช่น เพื่อความสนุกสนานบันเทิง เพื่อฝึกทักษะ และเพื่อการเรียนรู้ เป็นต้น และในบางครั้งอาจใช้เพื่อประโยชน์ทางการศึกษาได้

1.6.2 เกมแนวผจญภัย คือ ผู้เล่นจะสวมบทบาทเป็นตัวละครตัวหนึ่งและต้องกระทำเป้าหมายในเกมให้สำเร็จลุล่วงไปได้ โดยเน้นไปที่การแก้ปริศนา

1.6.3 เกมแนวธีมแฟนตาซี Fantasy คือ ในความคิด ของเราทุกคน มันคือภาพที่นำเราย้อนเวลากลับไปเป็นเด็กอีกครั้ง มันเริ่มมาจากที่ไหน เริ่มมาจากใคร การเล่านิทาน หรือเรื่องความเชื่อ มันคือความเพ้อฝัน หรือภาพหลอน มันคือจินตานาการ มันคือเรื่องโกหกหลอกลวงครั้งที่เรายังเป็นเด็กหรือเปล่า และทำไมเมื่อเราเติบโต มันถึงยังคงอยู่กับเรา เพราะอะไร? และอารยะธรรมเหล่านี้ มันฝังรากลึกในจิตใจคนเราได้มากน้อยขนาดไหน อะไรยังไง

1.6.4 เกมมุมด้านข้าง(Side Scrolling) คือ มุมมองสุดคลาสสิคที่นักเล่นเกมรุ่นเก่าต้องเคยเห็นหรือเล่นมาบ้าง เพราะศักยภาพในเครื่องเกมยุคก่อน ทำให้มุมมองนี้เป็นมุมมองยอดนิยมสุด ๆ เพราะไม่ต้องแสดงผลกราฟิกให้ละเอียดมากก็ทำให้เกมสนุกได้ แต่ในปัจจุบัน ด้วยเทคโนโลยีและขุมพลังของเอนจิ้นต่าง ๆ ในการทำเกม เกมแนว Side View ที่ทำออกมาแล้วภาพสวยมาก ๆ ก็มีให้เห็นเยอะเช่นกัน

1.6.5 เกมแนว Rogue-like คือ เกมที่มีการ "สุ่ม" เป็นหัวใจหลัก ไม่ว่าจะเป็นสุ่มด่าน, สุ่มศัตรูที่เจอ , สุ่มไอเทมที่ดรอปได้ หรือกระทั่งสุ่มไอเทมที่เราจะสามารถซื้อได้จากร้านค้าภายในเกม ซึ่งในปัจจุบัน

รูปแบบของ Rogue like ถูกเอาไปใส่เป็นส่วนประกอบหนึ่ง หรือแมคคานิคในเกมมากว่าใช้เป็นหัวใจหลักของระบบการเล่นเลย

1.6.6 แนว Single Player(เล่นคนเดียว) คือ วีดีโอเกมที่มีการนำเข้าคำสั่งและการบังคับต่างๆจากผู้เล่นเพียงคนเดียวตลอดตั้งแต่ต้นจนจบเกม ในบางเกมที่เล่นได้หลายคนอาจมี โหมดผู้เล่นเดี่ยว (single-player mode) ติดตั้งเข้ามาด้วยสำหรับการเล่นเพียงคนเดียว เกมคอมพิวเตอร์ส่วนใหญ่ในยุคเริ่มต้นมักจะเป็นเกมผู้เล่นคน

1.7 ระยะเวลาที่ใช้ในการจัดทำโครงงาน

ระยะเวลาในการวิจัย เป็นเวลาทั้งสิ้น 18 สัปดาห์ตั้งแต่วันที่ 16 ตุลาคม พ.ศ. 2566 ถึง วันที่ 17 กุมภาพันธ์ พ.ศ.2567

ตารางที่ 1.1 ระยะเวลาที่ใช้ในการตัดทำโครงงาน

ขั้นตอนการดำเนินงาน	สัปดาห์																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1. กำหนดโครงการ	◄	►																
2. เสนอโครงการ		◄	►															
3. ศึกษารายละเอียด			◄	—	—	—	►											
4. ออกแบบระบบ							◄	—	—	—	►							
5. ดำเนินการทำระบบ								◄	—	—	—	—	—	—	—	►		
6. นำเสนอระบบ															◄	—	►	
7. ประเมินผลและสรุป																◄	—	►
8. จัดทำเอกสารโครงการ																	◄	— ►

1.8 งบประมาณและทรัพยากร

ตารางที่ 1.2 งบประมาณที่ใช้ในการจัดทำโครงงาน

ที่	รายการพัสดุ	จำนวน	หน่วย	ราคา บาท
1	ปริ้นเอกสาร	1	1,000	1,000
2	โปรแกรม Aseprite	1	289	289
รวม				1,289.-

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

โครงการ Everlasting เป็นเกมทที่มีวัตถุประสงค์เพื่อที่สร้างเกม Everlasting เพื่อเสริมสร้างทักษะการใช้ การใช้โปรแกรม C# และ ช่วยเสริมสร้างการออกแบบตัวละคร และเพื่อเสริมสร้างทักษะในการตัดสินใจในสถานการณ์ต่างๆ เพื่อให้โครงการบรรลุวัตถุประสงค์ และเป็นไปตามขอบเขตที่กำหนดไว้ คณะจัดได้ทำการศึกษาทฤษฎีและเนื้อหาที่เกี่ยวข้อง ดังนี้

2.1 Unity

2.2 Adobe Photoshop

2.3 ภาษา C#

2.4 Aseprite

2.5 เอกสารและงานวิจัยที่เกี่ยวข้อง

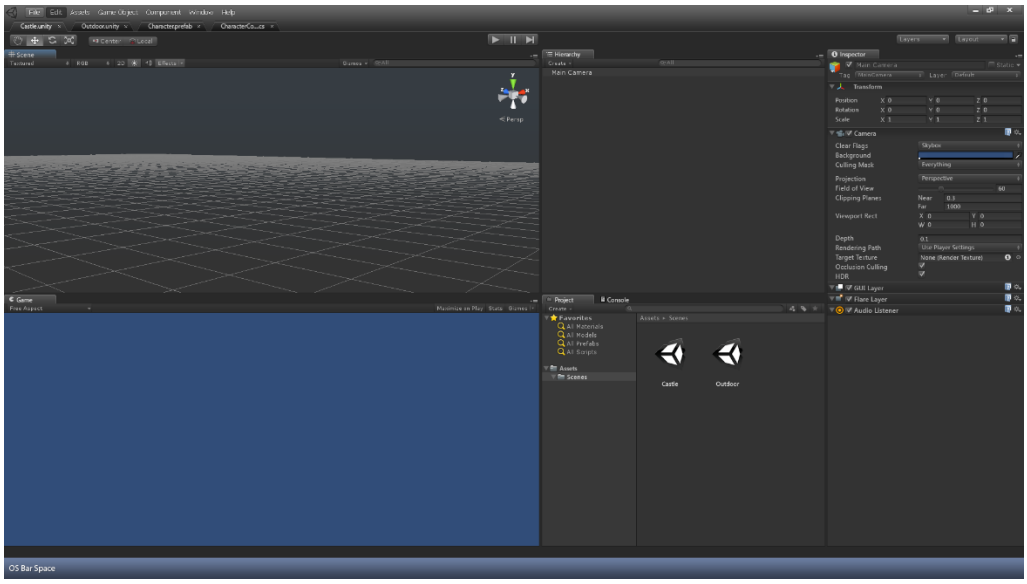
2.6 Story bord

2.1 Unity (https://th.wikipedia.org/wiki/ยูนิตี_(เกมเอนจิน))

Unity แพลตฟอร์มที่ใช้สร้างวิดีโอเกม ครึ่งหนึ่งของโลก /โดย ลงทุนแมนแม้ว่าโลกของเราจะมีเกมชื่อดังมากมาย เช่น Pokémon GO, League of Legends, Genshin Impact, Among Us, Garena Free Fire หรือ Overcooked แต่รู้หรือไม่ว่าเกมที่ยกตัวอย่างมานี้ ถูกพัฒนาขึ้นจาก เกมเอนจิน หรือซอฟต์แวร์ที่ใช้สร้างวิดีโอเกมที่ชื่อว่า “Unity” ซึ่งปัจจุบันครอบครองส่วนแบ่งเกินกว่าครึ่งหนึ่งของมูลค่าตลาดซอฟต์แวร์พัฒนาเกมบนโลก ที่สำคัญก็คือ Unity เป็นแพลตฟอร์มที่ใช้งานง่ายแต่ประสิทธิภาพสูง จนใครก็ตามที่อยากสร้างเกมเองสามารถใช้งานได้ ไม่จำเป็นต้องเขียนโคดเป็น ไม่ต้องเป็นนักพัฒนามืออาชีพ ไม่จำเป็นต้องมีเงินทุนก้อนใหญ่ นักพัฒนาเกมอินดี้หรือคนที่อยากทำเกมเป็นงานอดิเรกก็สามารถทำได้

รูปภาพที่ 2.1 Unity Engine

2.1.1) ส่วนประกอบหน้าต่าง Interface ของโปรแกรม Unity แบ่งออกดังนี้



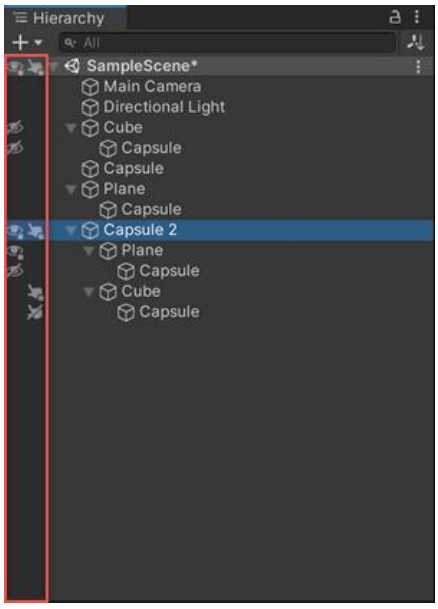
รูปภาพที่ 2.2 Interface ของ Unity

1) จากรูปภาพที่ 2.3 Toolbar เป็นแถบเครื่องมือที่มีไว้เข้าถึงคุณลักษณะการทำงานที่สำคัญด้านซ้าย มีเครื่องมือพื้นฐานสำหรับการจัดการมุมมองภาพและวัตถุภายในภาพ อยู่ตรงกลางคือการควบคุมการเล่นหยุดชั่วคราวและขั้นตอน ปุ่มทางด้านขวาจะทำให้คุณสามารถเข้าถึง Unity Cloud Service และบัญชี Unity ของคุณตามด้วยเมนูการมองเห็นของเลเยอร์และในที่สุดเมนูเค้าโครงของโปรแกรม

รูปภาพที่ 2.3 Toolbar

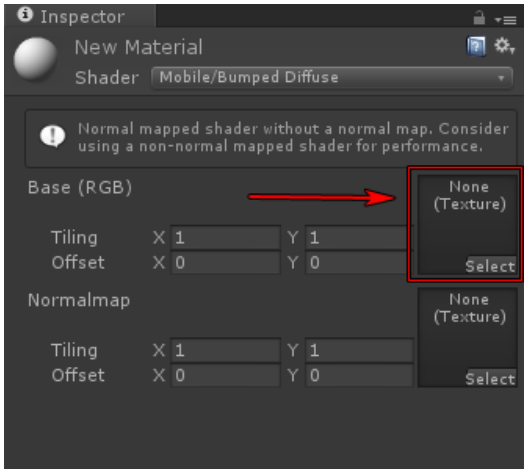
2) จากรูปภาพที่ 2.4 Hierarchy คือส่วนที่บอกลำดับชั้น ของ Object ต่างๆ ที่อยู่ใน Scene นั้นๆ ซึ่งมีทั้ง Object แบบเดี่ยว และ Object ที่เป็นแม่ลูกกัน ซึ่ง เมื่อมีการจัดการอะไรบางอย่างกับ Object แม่ Object ที่เป็นลูกนั้นก็จะมีการเปลี่ยนแปลงตามไปด้วย การสร้าง Object มีวิธีการคือ ลาก

Object ต่างๆ ที่อยู่ใน Project มาใส่ไว้ในส่วนของ Scene ซึ่ง Object ต่างๆ เหล่านี้สามารถเพิ่มแก้ไข/ลบ ได้โดยไม่ กระทบ Object ที่อยู่ใน Project



รูปภาพที่ 2.4 Hierarchy

3) จากรูปภาพที่ 2.5 Inspector เป็นส่วนบ่งบอกถึงคุณสมบัติต่างๆ ของ Object ซึ่งสามารถจาดการคุณสมบัติต่างๆ ของ Object ได้ในกรอบของ Inspector

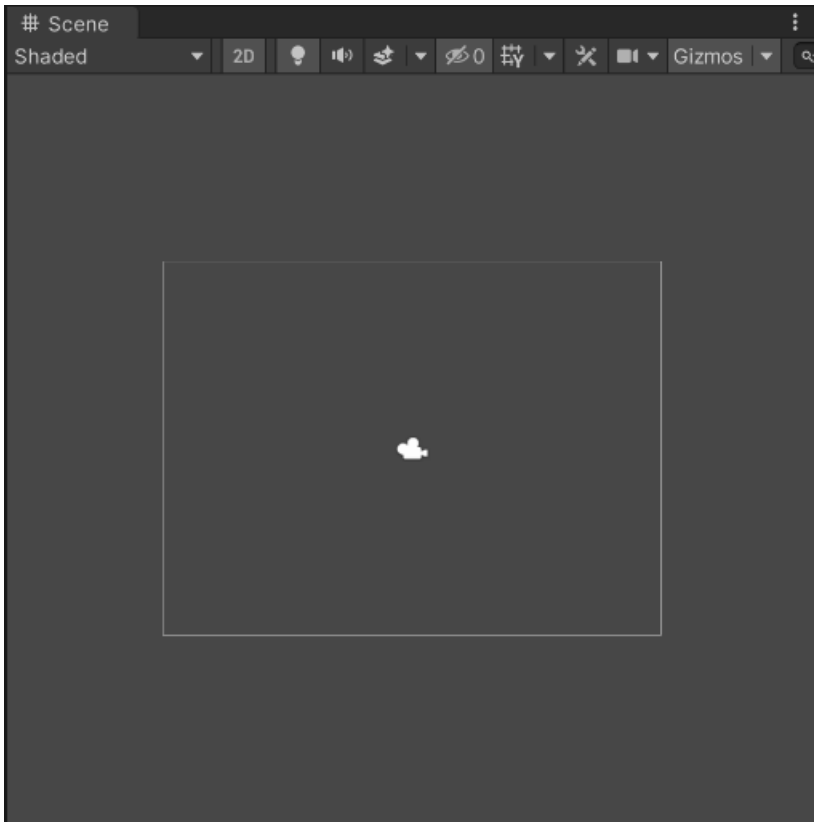


รูปภาพที่ 2.5 Inspector

4) จากรูปภาพที่ 2.6 Project เป็นส่วนที่ใช้ในการเก็บทรัพยากรต่างๆ ก่อนนำไปสร้างเกม เช่น - สคิปท์ต่างๆ ที่ใช้กำหนดควบคุมตัวเกม – 3D โมเดล ใช้เป็นตัวละครหรือวัตถุต่างๆในเกม – Textures หรือ พื้นผิวต่างๆ – ไฟล์เสียง หรือวิดีโอ – Prefabs – อื่นๆ

รูปภาพที่ 2.6 Project

5) จากรูปภาพที่ 2.7 Scene เป็นส่วนที่บ่งบอกว่าในฉากที่กำลังทำงาน มี Object อะไรบ้าง สามารถจัดการ Object ต่างๆ เช่น กล้อง แสง เอฟเฟค หรือโมเดล 3 มิติ ได้จากส่วนนี้



รูปภาพที่ 2.7 Scene

เครื่องมือของโปรแกรม Unity มีดังนี้

ตารางที่ 2.1 ตารางเครื่องมือของโปรแกรม Unity

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Pan	เป็นเครื่องมือสำหรับจัดการมุมมองของโปรแกรม Unity3D ถ้ากดปุ่ม middle mouse ค้างไว้จะเป็น การเคลื่อนย้าย viewport ทั้งหมด ถ้าปุ่ม right mouse ค้างไว้จะเป็นการเคลื่อนย้าย viewport โดยที่ยังรักษาจุกศูนย์กลางอยู่ที่มุมมองกล้อง
	Move	เป็นเครื่องมือสำหรับปรับตำแหน่งของ Object ทั้งหมดใน Viewport สามารถใช้เครื่องมือนี้ในการเคลื่อนย้ายตำแหน่ง x,y,z ได้
	Rotate	เป็นเครื่องมือสำหรับปรับองศาของ Object ทั้งในแนวแกน x,y และ z
	Scale	เป็นเครื่องมือสำหรับปรับขนาดของ Object แบบค่า scale โดยยังคงค่า width และ height ไว้

2.2 Adobe Photoshop (http://techerjaray.blogspot.com/2014/05/blog-post.html)

Adobe Photoshop มักเรียกสั้นๆ ว่า โฟโตชอป เป็นโปรแกรมประยุกต์ที่มีความสามารถในการจัดการแก้ไขและตกแต่งรูปภาพ (photo editing and retouching) แบบแรสเตอร์ ผลิตโดยบริษัทอะโดบีซิสเต็มส์ ซึ่งผลิตโปรแกรมด้านการพิมพ์อีกหลายตัวที่ได้รับความนิยม

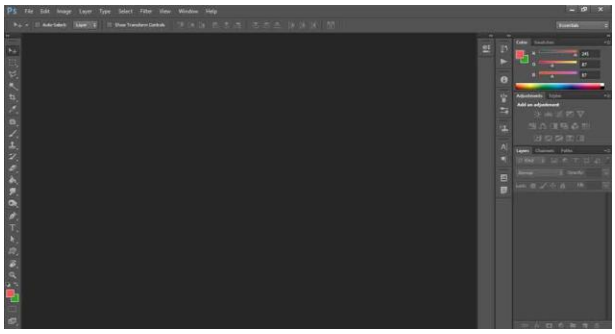
เช่น Illustrator และ InDesign ปัจจุบันโปรแกรมโฟโตชอปได้พัฒนามาถึงรุ่น CC (Creative Cloud)

โปรแกรม Photoshop เป็นโปรแกรมในตระกูล Adobe ที่ใช้สำหรับตกแต่งภาพถ่ายและภาพกราฟิก ได้อย่างมีประสิทธิภาพ ไม่ว่าจะเป็นงานด้านสิ่งพิมพ์ นิตยสาร และงานด้านมัลติมีเดีย อีกทั้งยังสามารถ retouching ตกแต่งภาพและการสร้างภาพ ซึ่งกำลังเป็นที่นิยมสูงมากในขณะนี้ เราสามารถใช้โปรแกรม Photoshop ในการตกแต่งภาพ การใส่ Effect ต่าง ๆให้กับภาพ และตัวหนังสือ การทำภาพขาวดำ การทำภาพถ่ายเป็นภาพเขียน การนำภาพมารวมกัน การ Retouch ตกแต่งภาพต่างสามารถเรียนรู้วิธีการใช้โปรแกรม Adobe Photoshop นี้ได้ด้วยตัวเอง สามารถที่จะทำการแก้ไขภาพ ตกแต่งภาพ ซ้อนภาพในรูปแบบต่างๆ ได้อย่างง่ายดาย และสิ่งที่ขาดไม่ได้ก็คือ การใส่ข้อความประกอบลงในภาพด้วย และเนื่องด้วย Adobe Photoshop มีการพัฒนาโปรแกรมมาอย่างต่อเนื่อง แต่ที่สำคัญ เมื่อเรียนรู้การใช้คำสั่งในเวอร์ชันเก่า ก็ยังคงสามารถนำไปประยุกต์ใช้กับเวอร์ชันใหม่ๆ ได้ด้วย

Photoshopสามารถทำงานกับระบบสี RGB, CMYK, Lab และ Grayscale และสามารถจัดการกับไฟล์รูปภาพที่สำคัญได้ เช่น ไฟล์นามสกุล JPG, GIF, PNG, TIF, TGA โดยไฟล์ที่โฟโตชอปจัดเก็บในรูปแบบเฉพาะของตัวโปรแกรมเองจะใช้นามสกุลของไฟล์ว่า PSD จะสามารถจัดเก็บคุณลักษณะพิเศษของไฟล์ที่เป็นของโฟโตชอป เช่น เลเยอร์, ชันแนล, โหมดสี รวมทั้งสไลส์ ได้ครบถ้วนโปรแกรมโฟโตชอปเป็นโปรแกรมที่มีความสามารถใช้ได้หลายรอบในการจัดการไฟล์ข้อมูลรูปภาพที่มีประสิทธิภาพ การทำงานกับไฟล์ข้อมูลรูปภาพของโฟโตชอปนั้น ส่วนใหญ่จะทำงานกับไฟล์ข้อมูลรูปภาพที่จัดเก็บข้อมูลรูปภาพแบบ Raster โฟโตชอปสามารถใช้ในการตกแต่งภาพได้หลากหลาย เช่น ลบตาแดง, ลบรอยแตกของภาพ, ปรับแก้สี, เพิ่มสีและแสง หรือการใส่เอฟเฟกต์ให้กับรูป เช่น ทำภาพสีซีเปีย, การทำภาพโมเซค, การสร้างภาพพาโนรามาจากภาพหลายภาพต่อกัน นอกจากนี้ยังใช้ได้ในการตัดต่อภาพ และการซ้อนฉากหลังเข้ากับภาพจัดเก็บในรูปแบบเฉพาะของตัวโปรแกรม จะใช้นามสกุลของไฟล์ว่า PSD จะสามารถจัดเก็บคุณลักษณะพิเศษ ของไฟล์เป็นของ Photoshop เช่น เลเยอร์,ชันแนล,โหมดสี รวมทั้งไสลด์ เป็นต้น

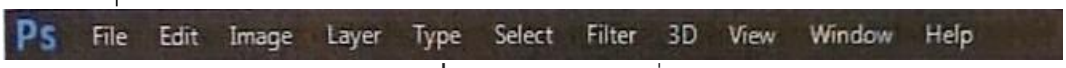
รูปภาพที่ 2.8 Adobe Photoshop

2.2.1 ส่วนประกอบหน้าต่าง Interface ของโปรแกรม Adobe Photoshop แบ่งออก ดังนี้



รูปภาพที่ 2.9 Interface ของ Adobe Photoshop

1) จากรูปภาพที่ 2.10 แถบเมนูคำสั่ง(Menu Bar) เป็นจุบรวบรวมชุดคำสั่งที่ใช้สำหรับเรียกคำสั่งต่างๆ เพื่อใช้จัดการไฟล์ภาพหรือตกแต่งภาพ

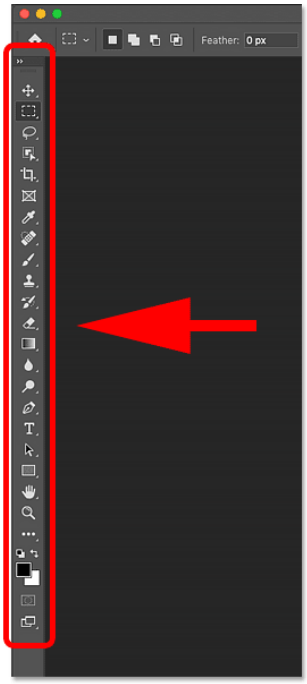


รูปภาพที่ 2.10 แถบเมนูคำสั่ง(Menu Bar)

2) จากรูปภาพที่ 2.11 แถบตัวเลือก (Options Bar)เป็นส่วนที่ใช้ในการปรับแต่งค่าการทำงานของเครื่องมือ ต่างๆ การกำหนดค่านแถบตัวเลือกจะเปลี่ยนไปตามเครื่องมือที่ใช้งานอยู่

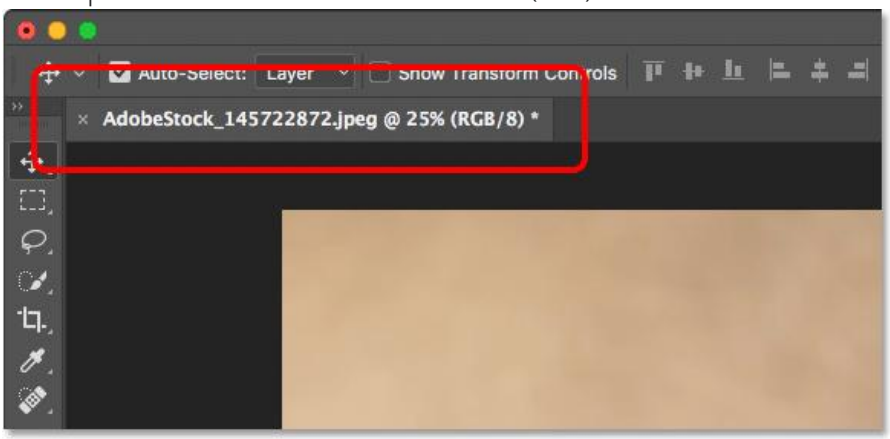
รูปภาพที่ 2.11 แถบตัวเลือก (Options Bar)

3) กล่องเครื่องมือ (Toolbox) เป็นส่วนที่ใช้เก็ยเครื่องมือพื้นฐานในการทำงานในนโปรแกรม สามารถเรียกใช้ชุดเครื่องมือย่อยโดยการคลิกรูปสามเหลี่ยมที่มุมด้านล่าง



รูปภาพที่ 2.12 กล่องเครื่องมือ (Toolbox)

4) แถบชื่อเรื่อง (Title Bar) เป็นส่วนที่แสดงชื่อไฟล์ภาพที่เปิดใช้งานอยู่ สำหรับโปรแกรม Adobe Photoshop CS6 แถบชื่อเรื่องจะเรียงกันเป็นแท็บ (Tab)



รูปภาพที่ 2.13 แถบชื่อเรื่อง (Title Bar)

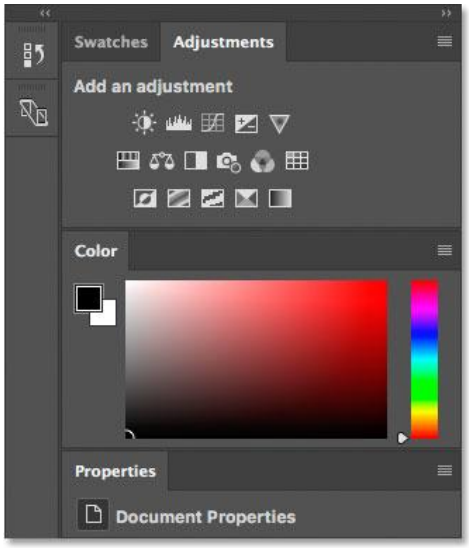
5) จากรูปภาพที่ 2.14 แถบสถานะ (Status Bar) เป็นส่วนที่แสดงคุณสมบัติเกี่ยวกับภาพ เช่น เปอร์เซ็นต์ในการย่อยขยายไฟล์ภาพ ขนาดไฟล์ภาพ เป็นต้น

รูปภาพที่ 2.14 แถบสถานะ(Status Bar)

6) จากรูปภาพที่ 2.15 พื้นที่ใช้งาน (Working Area) เป็นส่วนที่ใช้ในการสร้างงานการฟิก โดยการเปิดไฟล์ภาพเพื่อแก้ไขบนพื้นที่ใช้งาน หรือวาดภาพใหม่ลงไปบนพื้นที่ใช้งาน

รูปภาพที่ 2.15 พื้นที่ใช้งาน (Working Area)

7) จากรูปภาพที่ 2.16 พาเนล (Panel) ใช้สำหรับจัดภาพกับภาพ โดยแยกออกเป็น หมวดหมู่ เช่น พาเนลสำหรับเลือกสี พาเนลสำหรับปรับแต่งความสว่าง เป็นต้น



รูปภาพที่ 2.16 พาเนล (Panel)

เครื่องมือของโปรแกรม Adobe Photoshop มีดังนี้

ตารางที่ 2.3 ตารางเครื่องมือของโปรแกรม Adobe Photoshop

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Move	ใช้สำหรับย้ายพื้นที่เลือกไว้ของภาพ หรือย้ายภาพในเลเยอร์หรือย้ายเส้นไกด์
	Marquee	ใช้สำหรับเลือกพื้นที่บนภาพเป็นรูปสี่เหลี่ยม วงกลม วงรี หรือ เลือกเป็นแถวคอลัมน์ ขนาด 1 pixel
	Lasso	ใช้เลือกที่บนภาพเป็นแนวเขตแบบอิสระ
	Magic Wand	ใช้สำหรับเลือกพื้นที่ด้วยวิธีระบายบนภาพ หรือเลือก จากสี ที่ใกล้เคียงกัน
	Crop	ใช้ตัดขอบภาพ
	Slice	ใช้ตัดแบ่งภาพเรื่องเพื่อบันทึกไฟล์ภาพย่อย ที่เรียกว่า สไลซ์(Slice) สำหรับนำไปสร้างเว็บเพจ
	Eyedropper	ใช้เลือกสีจากสีต่างๆ บนภาพ
	Healing Brush	ใช้ตกแต่งลบรอยตำหนิในภาพ

ตารางที่ 2.3 ตารางเครื่องมือของโปรแกรม Adobe Photoshop (ต่อ)

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Clone Stamp	ใช้ทำสำนำภาย โดยก๊อปปี้ภาพจากบริเวณอื่นมาระบาย หรือระบายตัวลวดลาย
	History Bush	ใช้ระบายภาพด้วยภาพของขั้นตอนเดิมที่ผ่านมา หรือภาพของสถานะเดิมที่บันทึกไว้
	Eraser	ใชลบภาพบางส่วนที่ไม่ต้องการ
	Gradient	ใช้เติมสีแบบไล่ระดับโทนสีหรือความทึบ
	Blur	ใช้ระบายภาพให้เบลอ
	Brem	ใช้ระบายเพื่อให้ภาพมืดลง
	Dodge	ใช้ระบายเพื่อให้ภาพสว่างขึ้น
	Brush	ใช้ระบายลงบนภาพ

ตารางที่ 2.3 ตารางเครื่องมือของโปรแกรม Adobe Photoshop (ต่อ)

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Pen	ใช้วาดเส้นพาธ(Path)
	Horizontal Type	ใชพิมพ์ตัวอักษรหรือข้อความลงบนภาพ
	Path Selection	ใช้เลือกและปรับแต่งรูปทรงของเส้นพาธ
	Rectangle	ใช้วาดรูปทรงเลขาคณิตหรือรูปทรงสำเร็จรูป
	Hand	ใช้เลื่อนดูส่วนต่าง ๆ ของภาพ
	Zoom	ใช้ย่อหรือขยายมุมของภาพ
	Set Foreground Color, Set Background Color	ใช้สำหรับกำหนดสีที่ใช้ย่อหรือขยายมุมมอง Foreground Color และ Background Color

2.3 ภาษา C# (http://www.binaryprogramming.net/1-1)

ภาษา C# (C# Programming Language) เป็นภาษาโปรแกรมระดับสูง ที่ใช้ระบบชนิดข้อมูลแบบรัดกุม (strongtyping)และสนับสนุนการเขียนโปรแกรมเชิงคำสั่ง การเขียนโปรแกรมเชิงประกาศ การ

เขียนโปรแกรมเชิงฟังก์ชัน การเขียนโปรแกรมเชิงกระบวนการ การเขียนโปรแกรมเชิงวัตถุ (แบบคลาส) และการเขียนโปรแกรมเชิงส่วนประกอบ

ภาษานี้พัฒนาเริ่มแรกโดยโดยมีอนัส ไฮลส์เบิร์ก (Anders Hejlsberg) จากบริษัทไมโครซอฟท์ ในปีพ.ศ. 2543 ต่อมามีการรับรองให้เป็นมาตรฐานโดยเอ็กมาอินเตอร์เนชันแนล (ECMA-334) ในปีพ.ศ. 2545 และองค์การระหว่างประเทศว่าด้วยการมาตรฐาน (ISO/IEC 23270) ในปีพ.ศ. 2546 ไมโครซอฟท์ได้เปิดตัวภาษาซีชาร์ปพร้อมกับดอตเน็ตเฟรมเวิร์ก และVisual Studio ซึ่งเป็นผลิตภัณฑ์โคลสซอร์ส (closed-source) ทั้งหมด เนื่องจากตอนนั้นไมโครซอฟต์ไม่มีผลิตภัณฑ์ที่เป็นโอเพ่นซอร์ส. ต่อมาไมโครซอฟท์ได้เปิดตัว Visual Studio Code, Roslyn และ ดอตเน็ตคอร์ ซึ่งทั้งหมดนั้นรองรับภาษาซีชาร์ป เป็นซอฟต์แวร์ฟรีและโอเพ่นซอร์ส และทำงานแบบครอสแพลตฟอร์มปัจจุบันภาษาซีชาร์ปมีรุ่นล่าสุดคือ C# 11.0 ที่ออกมาพร้อมกับ .NET 7.0 ในปี พ.ศ. 2565

รูปภาพที่ 2.17 C# Programming Language

ตาราง 1-6 เครื่องหมายดำเนินการด้านการเปรียบเทียบ

เครื่องหมาย	ความหมาย
==	ค่าทางซ้ายและขวาเท่ากัน
!=	ค่าทางซ้ายไม่เท่ากับทางขวา
<	ค่าทางซ้ายน้อยกว่าทางขวา
>	ค่าทางซ้ายมากกว่าทางขวา
<=	ค่าทางซ้ายน้อยกว่าหรือเท่ากับทางขวา
>=	ค่าทางซ้ายมากกว่าหรือเท่ากับทางขวา

รูปภาพที่ 2.18 โครงสร้างของภาษา C#

2.3.1 ชนิดข้อมูล (Data Types) ใน C#

C# กำหนดชนิดของข้อมูลไว้หลากหลายชนิดเพื่อรองรับการจัดเก็บข้อมูลหลายๆ ประเภท ดังนี้

ตารางที่ 2.4 ตารางชนิดข้อมูล C#

ชนิดข้อมูล	คำอธิบาย
Char	ใช้ประกาศตัวแปร ให้เก็บค่าที่เป็นตัวอักขระ
Bool	ค่าความจรง เป็นไปได้สองค่าคือ True หรือ False
Byte	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 - 255
Int	ใช้ประกาศตัวแปร ให้เก็บค่าที่เป็นเลขจำนวนเต็ม
Uint	จำนวนเต็มไม่มีเครื่องหมายตั้งแต่ 0 – 4,294,967,295
long	ใช้ประกาศตัวแปร ให้เก็บค่าที่เป็นเลขจำนวนเต็มหรือ จำนวนจริง ที่มีจำนวนบิตมากเป็น 2 เท่า
ulong	
float	ใช้ประกาศตัวแปร ให้เก็บค่าที่เป็นเลขจำนวนจริง
double	ใช้ประกาศตัวแปร ให้เก็บค่าที่เป็นเลขจำนวนจริง ที่มีจำนวนบิตมากเป็น 2 เท่าของ float
string	ข้อความ (สายอักขระ) เช่น Hello

2.4 Aseprite (https://www.aseprite.org)

Aseprite เป็นอีกหนึ่งโปรแกรมศิลปะภาพที่ค่อนข้างง่ายที่จะรับ เมนูและอินเทอร์เฟซลีนุกซ์ออกมาวางไว้การควบคุมและเครื่องมือส่วนใหญ่จะง่ายต่อการค้นหาและส่วนใหญ่มีแป้นพิมพ์ลัด โปรแกรมนี้เป็นศูนย์กลางในการออกแบบซึ่งทำให้การสร้างทำได้ง่ายเพียงใดโดยไม่ทำให้คุณมีข้อมูลและตัวเลือกมากเกินไป

รูปภาพที่ 2.19 Aseprite

รูปภาพที่ 2.20 Interface Aseprite

1) จากรูปภาพที่ 2.21 แถบเมนูคำสั่ง(Menu Bar) เป็นจุบรวบรวมชุดคำสั่งที่ใช้สำหรับเรียกไฟล์ต่าง ๆ ออกมาทำ

รูปภาพที่ 2.21 แถบเมนูคำสั่ง(Menu Bar)

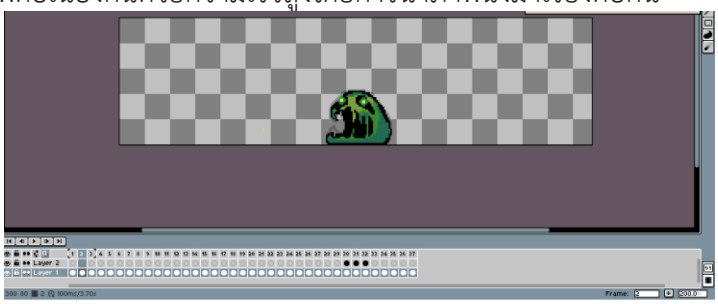
2) จากรูปภาพที่ 2.22 แถบเครื่องมือ(Tool Bar) เป็นจุบรวบรวมเครื่องมือที่ใช้วาด ลบ ลงสี เบลอภาพ ลากเส้น

รูปภาพที่ 2.22 แถบเครื่องมือ(Tool Bar)

3) จากรูปภาพที่ 2.23 แถบสี(Color Bar) เป็นจุดรวมสีที่จะมีหลายๆสีให้เลือกใช้

รูปภาพที่ 2.23 แถบสี(Color Bar)

4) จากรูปภาพที่ 2.24 แถบอนิเมชั่น(animation Bar) การสร้างภาพเคลื่อนไหวโดยการฉายภาพนิ่งหลาย ๆ ภาพต่อเนื่องกันด้วยความเร็วสูงโดยการนำภาพนิ่งมาเรียงต่อกัน



รูปภาพที่ 2.24 แถบอนิเมชั่น(animation Bar)

5) จากรูปภาพที่ 2.25 หน้าต่าง Preview เป็นหน้าต่างการแสดงผลและส่วนควบคุม เป็นส่วนที่ใช้เปิดดูตัวอย่าง ของวิดีโอที่เรากำลังทำ

รูปภาพที่ 2.25 Preview

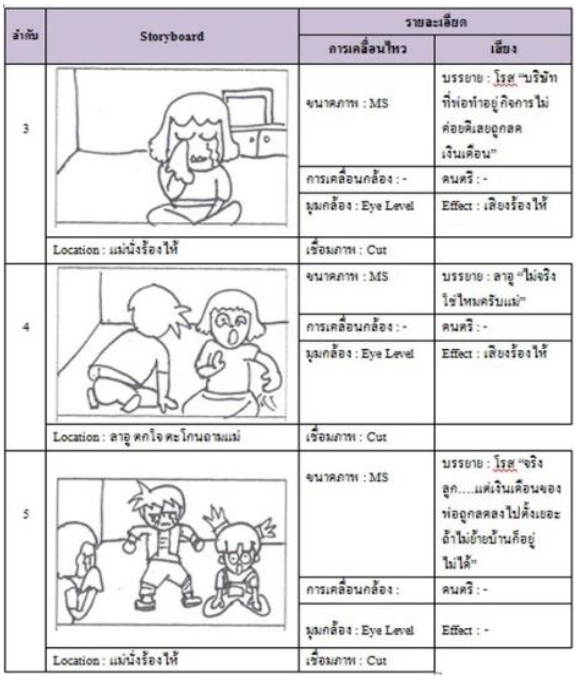
ตารางที่ 2.5 ตารางเครื่องมือของโปรแกรม Aseprite

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Pencil Tool	ใช้วาดรูปภาพต่างๆ
	Eraser	ใช้ลบภาพส่วนที่ทำผิดพลาด
	Eyedropper	ใช้เลือกสีจากสีต่างๆ บนภาพ
	Zoom	ใช้ย่อหรือขยายมุมของภาพ
	Move	ใช้สำหรับย้ายพื้นที่เลือกไว้ของภาพ หรือย้ายภาพในเลเยอร์ หรือย้ายเส้นไกด์
	Paint Bucket	ใช้ลงสี
	Line	ใช้วาดเส้นต่างๆ
	Rectangle	ใช้วาดรูปสี่เหลี่ยม
	Contour	ใช้วาดเส้นเค้าโครง
	Blur	ใช้ปรับให้ภาพเหมือนลงเฉพาะบางส่วน

2.5 Story Board (https://www.cotactic.com/blog/5-tricks-to-create-storyboard/)

การเขียนสตอรี่บอร์ด (Storyboard) คือ การนำเรื่องราวมาถ่ายทอดออกมาในรูปแบบภาพ โดยใช้ภาพวาด ภาพถ่าย หรือภาพกราฟิก เรียงต่อกันเป็นลำดับเพื่อบอกเล่าเรื่องราว มักใช้ในการผลิตภาพยนตร์ แอนิเมชั่น โฆษณา และวิดีโออื่นๆ สตอรี่บอร์ดช่วยให้ผู้สร้างสื่อสามารถวางแผนและสื่อสารเรื่องราวได้อย่างมีประสิทธิภาพ ช่วยให้เห็นภาพรวมของเรื่องราว ลำดับเหตุการณ์ มุมกล้อง และองค์ประกอบอื่นๆ ของสื่อได้อย่างชัดเจน นอกจากนี้ยังช่วยให้ผู้สร้างสื่อสามารถทดสอบไอเดียใหม่ๆ และปรับปรุงเรื่องราวได้ก่อนที่จะนำไปผลิตจริง

การเขียน Storyboard คือการเรียงลำดับภาพในความคิดออกมา ก่อนที่จะถ่ายจริง ซึ่งเขียนออกมาเป็นฉากเรียงลำดับ 1,2,3,.... ซึ่งทำให้เห็นภาพของเรื่องราวที่จะเล่า ปัญหาส่วนใหญ่ของคนที่จะเริ่มทำคือ ไม่รู้ว่าจะเริ่มจากตรงไหนมีองค์ประกอบอะไรที่ต้องใส่เข้าไปบ้าง



ลำดับ	Storyboard	รายละเอียด	
		การเคลื่อนไหว	เสียง
3		ขนาดภาพ : MS	บรรยาย : โรส "บริษัทที่พ่อทำอยู่กิจการไม่ค่อยดีเลยถูกลดเงินเดือน"
		การเคลื่อนกล้อง : -	คนตรี : -
		มุมกล้อง : Eye Level	Effect : เสียงร้องไห้
	Location : แม่นั่งร้องไห้	เชื่อมภาพ : Cut	
4		ขนาดภาพ : MS	บรรยาย : ลาอู "ไม่จริงใช่ไหมครับแม่"
		การเคลื่อนกล้อง : -	คนตรี : -
		มุมกล้อง : Eye Level	Effect : เสียงร้องไห้
	Location : ลาอู หดใจ ตะโกนถามแม่	เชื่อมภาพ : Cut	
5		ขนาดภาพ : MS	บรรยาย : โรส "จริงลูก.....แต่เงินเดือนของพ่อถูกลดลงไปตั้งเยอะถ้าไม่ย้ายบ้านก็อยู่ไม่ได้"
		การเคลื่อนกล้อง :	คนตรี : -
		มุมกล้อง : Eye Level	Effect : -
	Location : แม่นั่งร้องไห้	เชื่อมภาพ : Cut	

รูปภาพที่ 2.26 ตัวอย่าง Storyboard

2.5.1 เทคนิค วิธีการทำ Storyboard

1) เทคนิคหา KEY MESSAGE ให้โดนใจคนดู

- ทำ RESEARCH กับกลุ่มคนดู ว่าเขาเป็นใคร อายุเท่าไหร่ มีความชอบหรือสนใจ ซึ่ง Insight การทำ Storyboard เหล่านี้จะเป็นจุดที่คอยดึงคนเข้ามาดู Video ของเรามากขึ้น เช่น โฆษณาของ K PLUS ชื่อ FACE OFF นั้นหยิบ Insight ที่ว่าคนไทยส่วนใหญ่ไม่ชอบการเปลี่ยนแปลงจากสิ่งที่คุ้นเคย ซึ่ง

แอป KPLUS ในตอนนั้นกำลังจะ Update ครั้งใหญ่ โฆษณา FACE OFF จึงสื่อสารว่าถึงหน้าตาจะเปลี่ยนไปแต่คุณภาพในการใช้งานยังเหมือนเดิม

- ปัญหา ของคนดู บางคนอาจจะไม่รู้ตัวว่ากำลังประสบปัญหาเรื่องเล็กน้อยในชีวิต เราต้องสังเกตพฤติกรรม เพื่อนำจุดนั้นมาทำ Key Message และตอบ Pain Point ของคนดูได้ เช่น โฆษณา AirPay ที่ได้จับ Pain Point “คนข้างหน้าเรื่องเยอะเสมอ” ทั้งๆที่เราอยากจะแค่จองตั๋วหนัง แต่ต้องมารอคนจ่ายบิลค่าน้ำค่าไฟ ซึ่งเป็น Pain Point ที่เจอได้ทุกวัน และ AirPay ก็นำเสนอตัวเองเป็น Solution ในการแก้ปัญหานี้

- ความแตกต่าง การบอกว่าเราแตกต่าง แปลกใหม่กว่าคนอื่นอย่างไร ก็เป็นอีกหนึ่ง Key Message ที่ใช้กันเยอะ เพราะถ้ายิ่งทำออกมาได้ชัดมากเท่าไหร่ ก็ยิ่งสร้างความน่าสนใจได้มากขึ้นเท่านั้น เช่น โฆษณาของ Sunsilk ชื่อ The Hair Talk ที่บอกเล่าเรื่องผ่าน LGBT กับครอบครัว ซึ่ง Sunsilk ใช้ความแตกต่างในเรื่องของยาสระผมที่คนส่วนใหญ่คิดว่าเป็นแบรนด์สำหรับผู้หญิงเท่านั้น

2) STORYTELLING

- เล่าตามเหตุการณ์ปกติ เป็นการเล่าเรื่อง จาก 1 ไป 2 ไป 3 ปกติ ซึ่งการเล่าเรื่องแบบนี้มีข้อดีคือ ทำให้คนเข้าใจเนื้อหาได้ง่าย การทำ Storyboard ไม่ต้องคิดเยอะ แต่ถ้าเนื้อหาของเรานั้นไม่ได้มีความน่าสนใจพอ ก็จะทำให้คนดูอาจจะเบื่อ ดูไม่จบ แต่เทคนิคนี้ถือว่าเป็นพื้นฐานที่ดี คือการเล่าเรื่องให้คนเข้าใจ ตัวอย่างโฆษณาของ Kleenex Thailand ชื่อ Tiny Doll ที่ใช้เทคนิคการเล่าตามเหตุการณ์ปกติ แต่เพิ่มความน่าสนใน Video ด้วยการถ่ายแบบ Long Take

- การเรียงลำดับเรื่องใหม่ การเล่าเรื่องแบบนี้ สามารถกระโดดข้ามไปมาได้ จะเอาตอนท้ายเรื่องมาไว้ต้นเรื่อง หรือ ต้นเรื่องไปไว้กลางเรื่องก็ได้ เทคนิคนี้จะมีความยืดหยุ่นในการเล่าเรื่อง เพราะมีความอิสระในการตีความใหม่ทั้งหมด แต่จะมีข้อเสียคือถ้าลำดับเรื่องไม่ดี ก็จะทำให้คนดูไม่เข้าใจเนื้อหาที่อยากจะสื่อไป โฆษณา My Beautiful Woman ของ Wacoal ใช้เทคนิคการเล่าเรื่องแบบตัดสลับกันในตอนท้าย

3) MOOD AND TONE

- MOOD คือ อารมรณ์ของ Video เช่น สนุกสนาน เศร้า ร่าเริง ความสงบ ประหลาดใจ โกรธ

- TONE คือ สีที่ใช้ในวีดีโอ โดยสีจะบอกความรู้สึกที่เรารู้กัน สีโทนเย็นหรือสีโทนร้อน ซึ่งขึ้นกับว่าต้องการจะสื่อออกมาในทิศทางไหน เช่น ถ้าอยากให้ภาพใน Video ดูสนุก สดใส อาจจะต้องใช้สีโทนร้อนเพื่อทำให้คนดูรู้สึกถึงความมีชีวิตชีวา ไม่หยุดนิ่ง แต่กลับกันเราอยากจะให้ Video รู้สึกน่ากลัว ลึกลับ อาจจะต้องใช้สีโทนเย็นเพื่อให้รู้สึกสุขุมขึ้น นิ่งขึ้น

4) SHOT REFERENCES

- เทคนิคการหา SHOT REFERENCES นั้นคือการเลือกภาพที่สื่อความหมายที่เราอยากสื่อมากที่สุด เพราะต่อให้เป็นท่าทางเดียวกัน แต่มุมกล้องต่างกันบริบทไม่เหมือนกัน ก็ทำให้อารมณ์ของภาพเปลี่ยนไป ซึ่งในส่วนนี้รวมถึงแสงเงาของภาพเข้าไปด้วย จากตัวอย่างจะเป็นฉากการเขียนสมุดบันทึก ซึ่งเป็นท่าทางเดียวกัน แต่ด้วยมุมกล้องและแสง ทำให้มีอารมณ์ที่ต่างกันสิ้นเชิง

- งานที่ควรนำมาเป็น REFERENCES การหา Shot Reference นั้นควรหาจากงานโฆษณา / Music Video / หนัง / Filmmaker เพราะงานประเภททาง Production จะมีการคิดเรื่องแสงเงา ท่าทาง เพื่อให้ฉากสวยงามอยู่แล้ว เหมาะแก่การนำมาเป็น Reference ได้เลยโดยไม่ต้องไปคิดเอาเอง

5) TRANSITION ที่ใช้เพิ่มสีสันในงาน VIDEO

- CUT คือการเปลี่ยนจากภาพหนึ่งไปยังอีกภาพหนึ่งแบบทันทีทันใด โดยไม่ใช้เทคนิคใดๆ ในการเชื่อมต่อภาพ ลักษณะที่ปรากฏออกมาจึงเป็นภาพที่ต่อด้วยภาพ จะถ่ายทอดความรู้สึกฉับไว นอกจากนี้เรายังรับรู้สิ่งต่างๆ อย่างตรงไปตรงมา

- FADE คือการเปลี่ยนภาพ โดยภาพจะแทนที่ด้วยฉาก โดยการ Fade นี้ สามารถแบ่งออกเป็น Fade In และ Fade Out

Fade In ใช้สำหรับการเริ่มต้นของเรื่องราว หรือเหตุการณ์

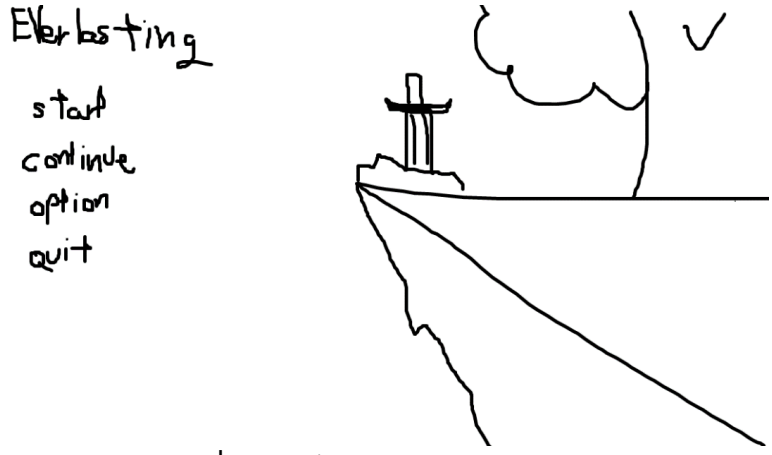
Fade Out คือภาพหลักจะค่อยๆ จางลงกลายเป็นภาพสีดำ

- DISSOLVE คือการเปลี่ยนภาพ โดยภาพแรกค่อยๆจางไปสู่อีกภาพ โดยที่การเปลี่ยนแปลงนั้นจะไม่ได้เกิดขึ้นแบบทันทีทันใดเหมือนการ Cut เทคนิคจะช่วยสร้างความรู้สึกที่ราบรื่นและนุ่มนวล โดยจะใช้เพื่อเชื่อมโยงเรื่องราว

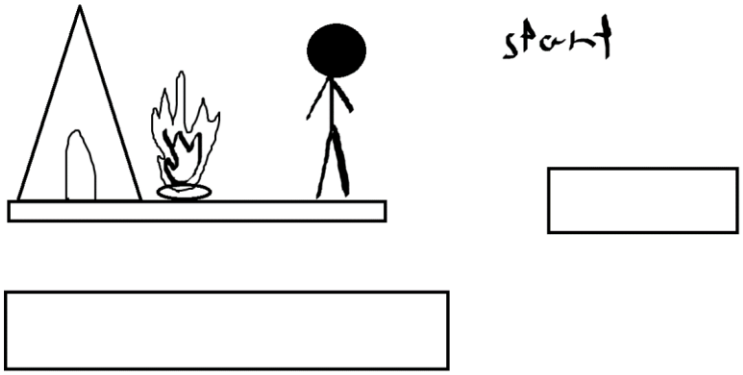
- WIPE การปาด คือหน้าจอจะมีลักษณะของการปาดภาพที่ต้องการจะเปลี่ยน ทับลงบนภาพที่ปรากฏอยู่ การใช้ Wipe อาจสร้างความรู้สึกไม่สมจริงของเรื่องราว ซึ่งส่วนใหญ่จะถูกใช้เพื่อแบ่งเรื่องราวทั้งหมดออกเป็นส่วนๆ อย่างชัดเจน

- DIGITAL EFFECT เป็นเทคนิค Transition แบบใหม่ ที่นำมาใช้กับงานที่เน้นความน่าสนใจของภาพ เช่น งานโฆษณา หรือ Motion Graphics เทคนิคช่วยเพิ่มความน่าสนใจให้กับงาน หากเป็นงานประเภทที่ต้องการดึงดูดความสนใจสูง

2.5.2 Storyboard ของ Project



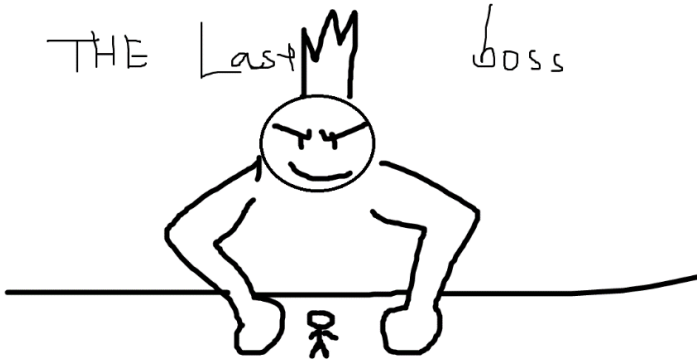
รูปภาพที่ 2.27 เป็นหน้า Menu ของเกม



รูปภาพที่ 2.28 ตัวละครต้องออกไปกำจัดมอนสเตอร์และบอส

รูปภาพที่ 2.29 ออกกำจัด มินิบอส

รูปภาพที่ 2.30 เมื่อเราโดนจัดการก็จะกลับมาเริ่มต้นใหม่



รูปภาพที่ 2.31 เมื่อเราจัดการบอสได้แล้วก็จะถือว่าจบเกม

2.6 เอกสารและงานวิจัยที่เกี่ยวข้อง

อภินันท์ ภิรมย์ ภาคย์ สธนเสาวภาคย์ และภัทราวัลย์ คำปลิว งานวิจัยนี้มีวัตถุประสงค์เพื่อ 1) พัฒนาเกมสามมิติเรื่อง เกมทางออกอยู่ไหน 2) ศึกษาความพึงพอใจของนักศึกษา ที่ มีต่อเกมสามมิติ เรื่อง เกมทางออกอยู่ไหน กลุ่มเป้าหมาย นักศึกษาสาขาวิศวกรรมคอมพิวเตอร์และสาขาเครือข่าย คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยราชภัฏมหาสารคาม จำนวน 80 คน เครื่องมือที่ใช้ในการวิจัย ได้แก่ แบบสอบถามความพึงพอใจของนักศึกษา สถิติที่ใช้ในการวิจัย คือ ค่าเฉลี่ย และส่วนเบี่ยงเบนมาตรฐาน ผลการวิจัยสรุปได้ดังนี้ 1) ได้เกมสามมิติเรื่อง เกมทางออกอยู่ไหน ประกอบด้วย เกาะในเกมจำนวน 3 เกาะ โดยแต่ละเกาะจะมีมอนสเตอร์และความยากแตกต่างกันออกไป โดยเกาะที่ 1 จะง่ายที่สุด และมีมอนสเตอร์คือ ซอมบี้สำหรับ มุมมองในเกมจะเป็นมุมมองของบุคคลที่หนึ่ง ผู้วิจัยได้นำมอนสเตอร์จำนวน 4 ชนิดที่มาจากใน Unity 3D คือ มอนสเตอร์ เอเลี่ยน มอนสเตอร์กบกลายพันธุ์ ซอมบี้ และก๊อบบิน 2)

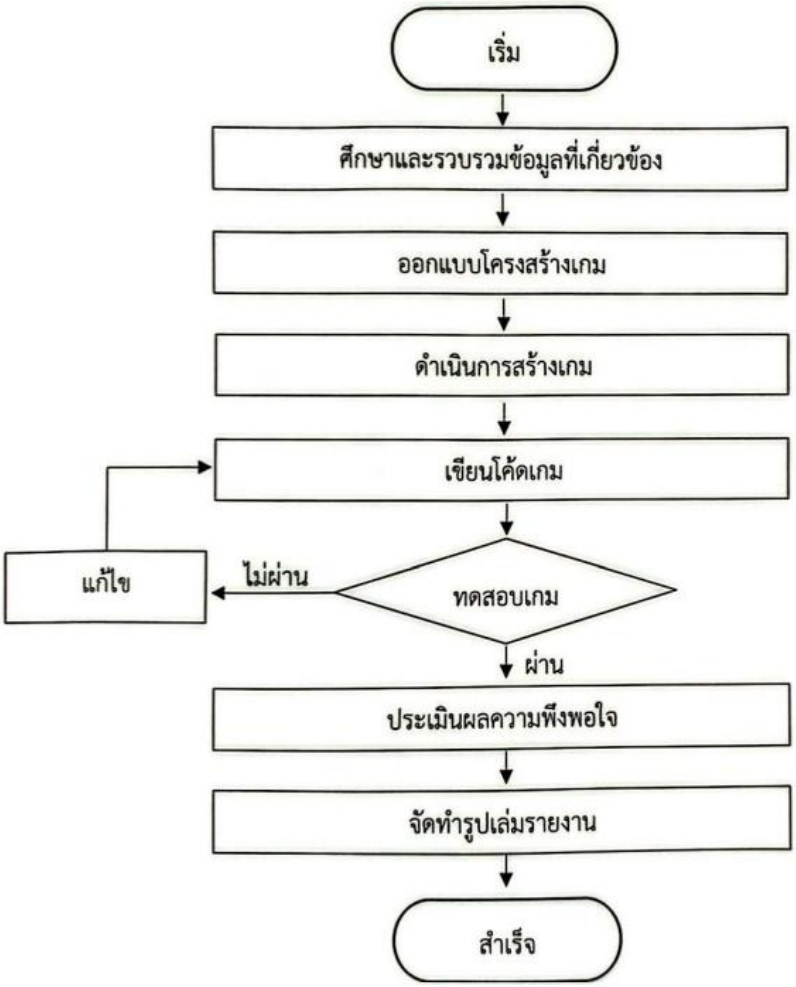
นักศึกษาสาขาวิศวกรรมคอมพิวเตอร์และสาขาเครือข่าย คอมพิวเตอร์คณะวิศวกรรมศาสตร์ มหาวิทยาลัยราชภัฏมหาสารคาม มีความพึงพอใจต่อเกมสามมิติเรื่องเกมทางออกอยู่ไหนโดยรวมอยู่ในระดับมาก

นายนัฐวุฒิ หวลละห้อย นายภูวนาท แสงฤทธิ์ และนายรัฐพล สุขสว่างการจัดทำโครงการครั้งนี้มีวัตถุประสงค์เพื่อสร้างเกมสามมิติผีในฝัน ด้วยโปรแกรม Unity (Ghost in Dream 3D Game)ศึกษาความพึงพอใจ ของนักเรียนประกาศนียบัตรวิชาชีพ สาขาวิชาคอมพิวเตอร์ธุรกิจวิทยาลัยเทคนิคระยองที่มีต่อเกมสามมิติ ในฝันด้วยโปรแกรม Unity (Ghost in Dream 3D Game) และเผยแพร่เกมสามมิติผีในฝัน ด้วยโปรแกรม Unity (Ghost in Dream 3D Game) ผ่านโครงการประกวดโครงการวิชาชีพ ขมรมวิชาชีพคอมพิวเตอร์ธุรกิจ กลุ่มตัวอย่างที่ใช้คือนักเรียนประกาศนียบัตรวิชาชีพ สาขาคอมพิวเตอร์ธุรกิจวิทยาลัยเทคนิคระยอง จำนวน 59 คนเครื่องมือที่ใช้ในการวิเคราะห์ข้อมูล ได้แก่ เกมสามมิติผีในฝันด้วยโปรแกรม Unity (Ghost in Dream 3D Game) และแบบสอบถาม ความพึงพอใจของนักเรียนประกาศนียบัตรวิชาชีพ สาขาคอมพิวเตอรธ์รุกิจ วิทยาลัยเทคนิคระยอง ที่มีต่อเกมสามมิติในฝันด้วยโปรแกรม Unity (Ghost in Dream 3D Game) สถิติที่ใช้ในการวิเคราะห์ข้อมูล คือ ค่าสถิติร้อยละ ค่าเฉลี่ย และส่วนเบี่ยงเบนมาตรฐาน

รัชพล วรรณวิจิตร โครงงานนี้ทำขึ้นมาโดยมีจุดมุ่งหมายเพื่อโปรโมทสถาบันเทคโนโลยีไทย-ญี่ปุ่น และเพื่อ เป็นแนวทางสำหรับผู้ที่มีความสนใจ ต้องการศึกษาเกี่ยวกับการสร้างเกม 3D โดยแบ่งเป็นขั้นตอนการเลือกอาคารที่จะนำมาใช้ การจัดวางองค์ประกอบให้ดูน่ากลัว การทำ NavMesh ให้ NPC เดิน การจัดการกับแสงเงา และความรู้เบื่องต้นเกี่ยวกับการ optimize ตัวเกม และสามารถนำเทคนิคจากโครงงานนี้ไปใช้ต่อยอดเพื่อพัฒนาผลงานตัวเองได้

บทที่ 3
วิธีการดำเนินการโครงงาน

ในการจัดทำโครงงาน คณะผู้จัดทำได้ดำเนินการตามขั้นตอนดังนี้



รูปภาพที่ 3.1 แสดงแผนผังการดำเนินโครงงาน

3.1 ศึกษาและรวบรวมข้อมูลที่เกี่ยวข้อง

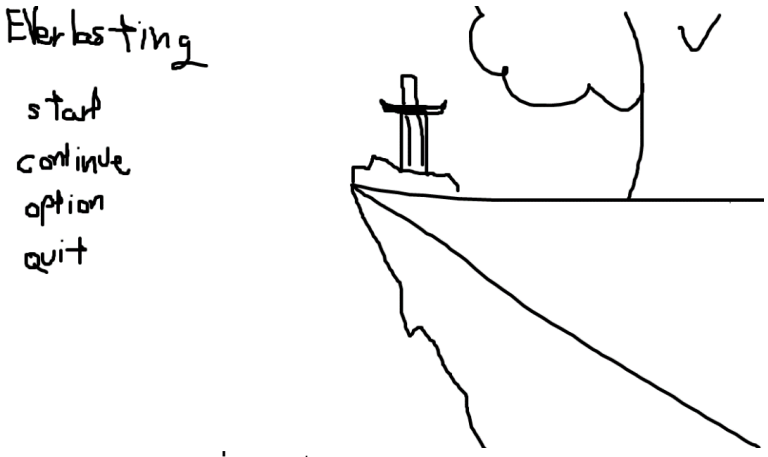
Unity คือ แพลตฟอร์มที่ใช้สร้างวิดีโอเกม ครึ่งหนึ่งของโลก โดย ลงทุนแมนแม้ว่าโลกของเราจะมี เกมชื่อดังมากมาย เช่น Pokémon GO, League of Legends, Genshin Impact, Among Us, Garena Free Fire หรือ Overcooked แต่รู้หรือไม่ว่าเกมที่ยกตัวอย่างมานี้ ถูกพัฒนาขึ้นจาก เกมเอนจิน หรือซอฟต์แวร์ที่ใช้สร้างวิดีโอเกมที่ชื่อว่า “Unity”

Photoshop คือ โปรแรกมที่ใช้ในการตกแต่งภาพได้หลากหลาย เช่น ลบตาแดง, ลบรอยแตกของภาพ, ปรับแก้สี, เพิ่มสีและแสง หรือการใส่เอฟเฟกต์ให้กับรูป เช่น ทำภาพสีซีเปีย, การทำภาพโมเซค, การสร้างภาพพาโนรามาจากภาพหลายภาพต่อกัน นอกจากนี้ยังใช้ได้ในการตัดต่อภาพ และการซ้อนฉากหลังเข้ากับภาพจัดเก็บในรูปแบบเฉพาะของตัวโปรแกรม

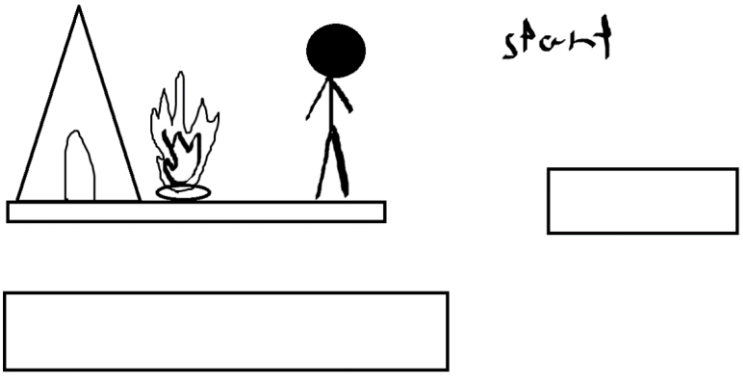
ภาษา C# (C# Programming Language) เป็นภาษาโปรแกรมระดับสูง ที่ใช้ระบบชนิดข้อมูลแบบรัดกุม (strongtyping)และสนับสนุนการเขียนโปรแกรมเชิงคำสั่ง การเขียนโปรแกรมเชิงประกาศ การเขียนโปรแกรมเชิงฟังก์ชัน การเขียนโปรแกรมเชิงกระบวนการ การเขียนโปรแกรมเชิงวัตถุ (แบบคลาส) และการเขียนโปรแกรมเชิงส่วนประกอบ

Aseprite คือ เป็นอีกหนึ่งโปรแกรมศิลปะภาพที่ค่อนข้างที่จะได้รับความนิยมใช้งานง่าย หน้าเมนู,หน้าอินเทอร์เฟสออกแบบมาวางการควบคุมและเครื่องมือส่วนใหญ่จะง่ายต่อการค้นหาและส่วนใหญ่มีแป้นพิมพ์ลัด โปรแกรมนี้เป็นศูนย์กลางในการออกแบบซึ่งทำให้การสร้างทำได้ง่ายเพียงใดโดยไม่ทำให้คุณมีข้อมูลและตัวเลือกมากเกินไป

3.2 Storyboard



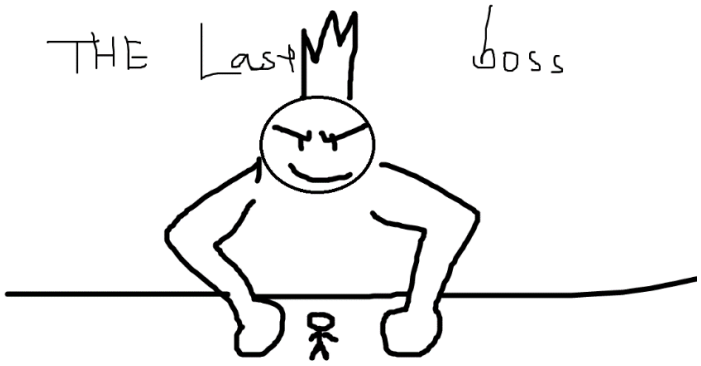
รูปภาพที่ 3.2 เป็นหน้า Menu ของเกม



รูปภาพที่ 3.3 ตัวละครต้องออกไปกำจัดมอนสเตอร์และบอส

รูปภาพที่ 3.4 ออกกำจัด มินิบอส

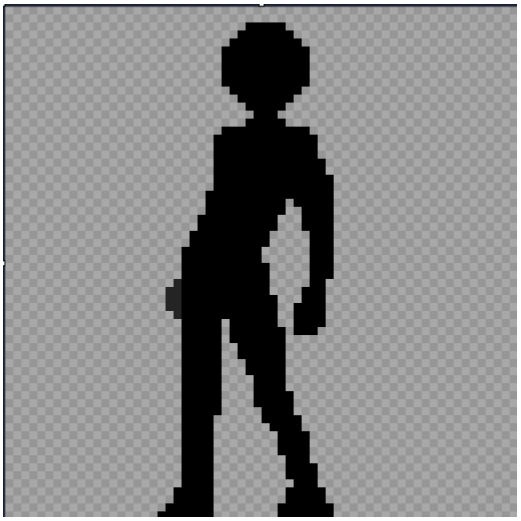
รูปภาพที่ 3.5 เมื่อเราโดนจัดการก็จะกลับมาเริ่มต้นใหม่



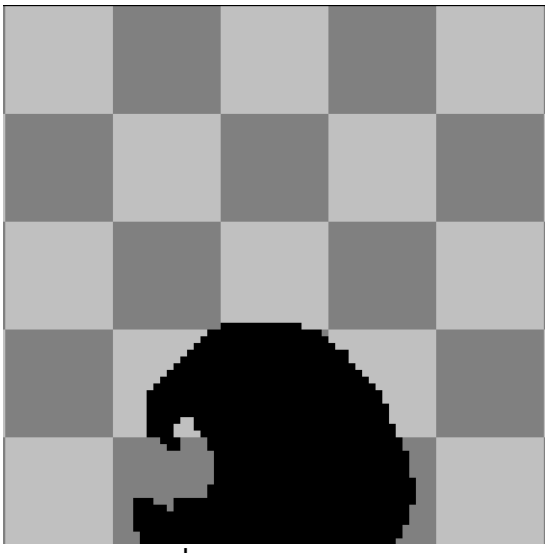
รูปภาพที่ 3.6 เมื่อเราจัดการบอสได้แล้วก็จะถือว่าจบเกม

3.3 ออกแบบโครงสร้างเกม

3.3.1 ออกแบบตัวละครหลัก และ ออกแบบ Monster

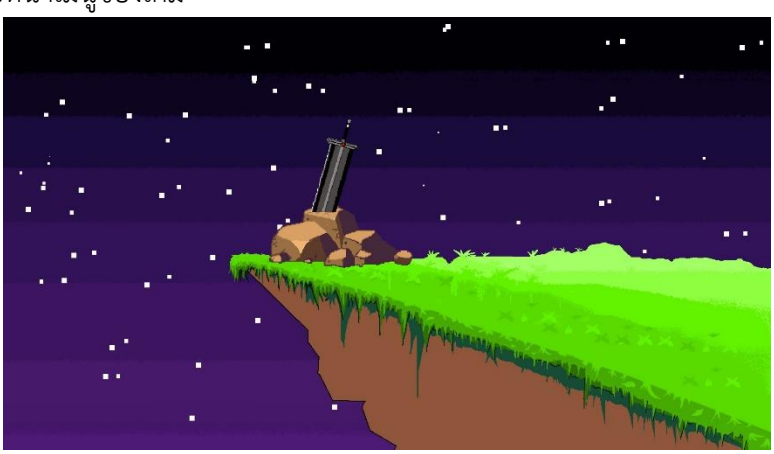


รูปภาพที่ 3.7 ออกแบบตัวละครหลัก



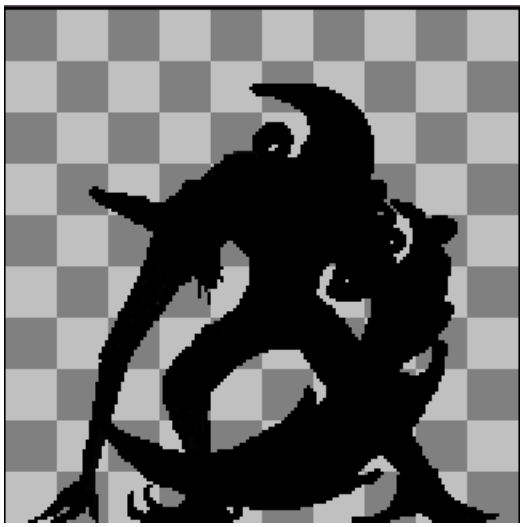
รูปภาพที่ 3.8 ออกแบบ Monster

3.3.2 ออกแบบหน้าเมนูของเกม



รูปภาพที่ 3.9 ออกแบบหน้าเมนูของเกม

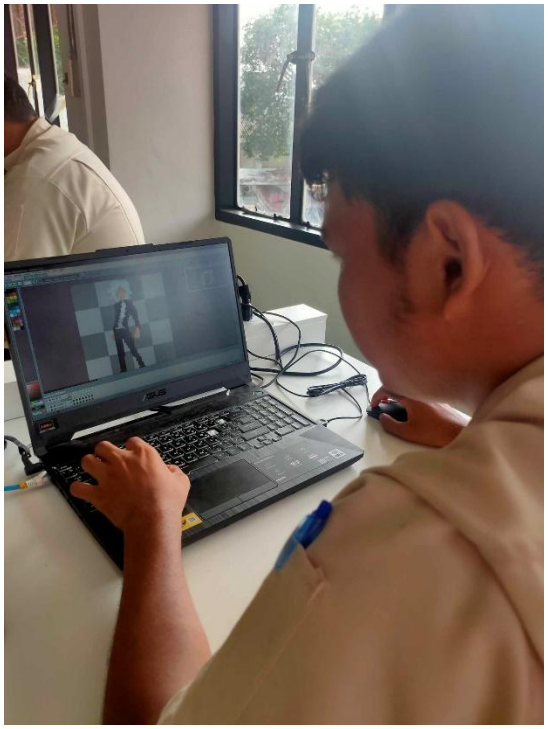
3.3.3 ออกแบบ Boss ของเกม



รูปภาพที่ 3.10 ออกแบบ Boss ของเกม

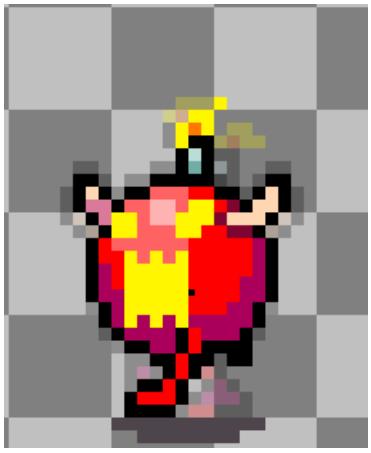
3.4 ดำเนินการสร้างเกม

3.4.1 สร้างตัวละคร



รูปภาพที่ 3.11 สร้างตัวละคร

รูปภาพที่ 3.12 สร้างตัวละคร ที่ 1



รูปภาพที่ 3.13 สร้าง Monster ที่ 2

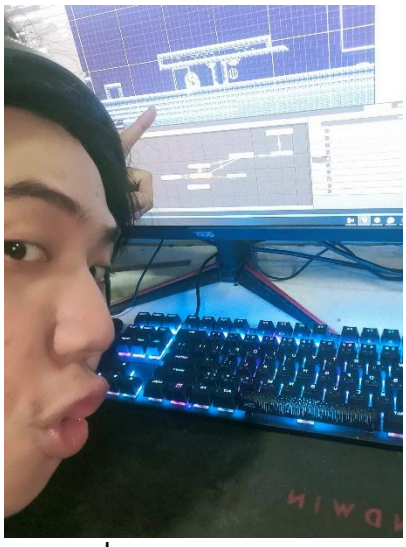
รูปภาพที่ 3.14 สร้าง Monster ที่ 3

รูปภาพที่ 3.15 สร้าง Monster ที่ 4

3.4.2 สร้างหน้าเมนูเกม The Everlasting

รูปภาพที่ 3.16 หน้าเมนูเกม The Everlasting

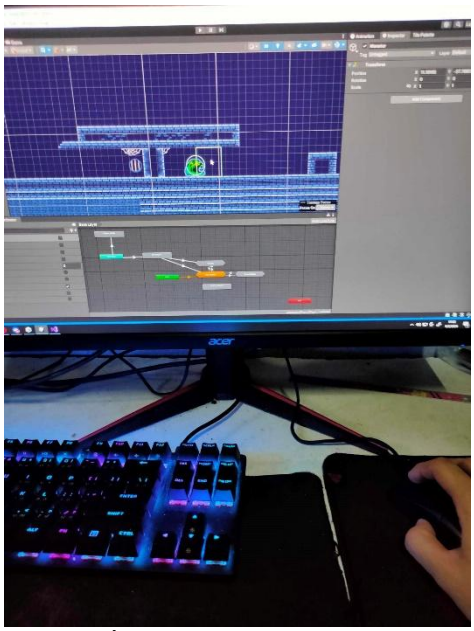
3.4.3 ทำระบบการเล่นเกม



รูปภาพที่ 3.17 ทำระบบภายในเกม

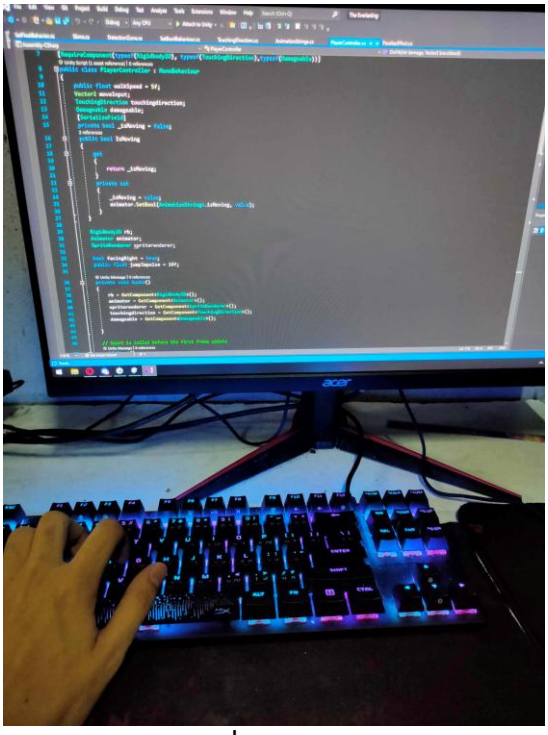
3.5 การเขียนโค้ด

3.5.1 การสร้างไฟล์ C# Script



รูปภาพที่ 3.18 การสร้างไฟล์ C# Script

3.5.2 เขียนโค้ดเกม



รูปภาพที่ 3.19 เขียนโค้ด

3.6 ทดสอบการทำงาน

3.5.1 หลังจากเข้าสู่เกมเป็นที่เรียบร้อยแล้ว ให้กดปุ่ม New game เพื่อเริ่มเกม

3.5.2 จากนั้นเกมก็จะให้เริ่มต้นในดันเจี้ยน

3.5.3 ถ้าเราเล่น game over ก็จะเริ่มใหม่ที่จุดเริ่มต้น

3.5.4 ถ้าเราล้มบอสได้สำเร็จก็จะชนะเกมนี้

3.7 การประเมินความพึงพอใจ

แบบประเมินความพึงพอใจของผู้ใช้งาน Everlasting เป็นแบบมาตราส่วนประมาณค่า 5 ระดับของลิเคิร์ท (Likert) มีคำถามจำนวน 5 ข้อ โดยกำหนดระดับความคิดเห็นแต่ละช่วงคะแนนและความหมายดังนี้

ระดับ 1 หมายถึง ปรับปรุง

ระดับ 2 หมายถึง พอใช้

ระดับ 3 หมายถึง ปานกลาง

ระดับ 4 หมายถึง ดี

ระดับ 5 หมายถึง ดีมาก

สำหรับการให้ความหมายของค่าที่วัดได้ ผู้วิจัยได้กำหนดเกณฑ์ที่ใช้ในการให้ความหมายโดยได้จากแนวคิดของเบสท์(Best 1986) การให้ความหมายโดยการให้ค่าเฉลี่ย ดังนี้

1.00 – 1.49 หมายถึง ปรับปรุง

1.50 – 2.49 หมายถึง พอใช้

2.50 – 3.49 หมายถึง ปานกลาง

3.50 – 4.49 หมายถึง ดี

4.50 – 5.00 หมายถึง ดีมาก

ผู้ทดลองใช้งานจะได้ประเมินคุณภาพของโปรแกรม โดยใช้แบบสอบถามเพื่อประเมินความคิดเห็น

ผลที่ได้จากการประเมินของผู้ทดลองงานหลักการทางสถิติเข้ามาช่วยในการสรุปการประเมินเกมที่ได้สร้างขึ้น โดยคำนวณหาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของระดับความติดในแต่ละด้าน

สูตรหาค่าเฉลี่ย (ล้วนและอังคณา,2538: 73)

$$\bar{X} = \frac{\sum x}{N}$$

เมื่อ $\bar{X}$ แทน ค่าเฉลี่ยของคะแนน

$\sum x$ แทน ผลรวมของคะแนน

N แทน จำนวน

$$S.D. = \sqrt{S^2}$$

$$S^2 = \frac{N\sum x^2 - (\sum x)^2}{N(N-1)}$$

เมื่อ S.D. แทน ส่วนเบี่ยงเบนมาตรฐาน

X แทน คะแนนแต่ละคน

n แทน จำนวนคน

บทที่ 4
ผลการทดลอง

โครงงานครั้งนี้ คณะผู้จัดทำได้ทำการสร้างเกม The Everlasting มีการเก็บรวบรวมข้อมูลผลการทดลองดังนี้

4.1 ผลจากการทดลองเล่นเกม The Everlasting

4.2 ผลการประเมินหาคุณภาพของเกม The Everlasting

4.1 ผลจากการทดลองเล่นเกม The Everlasting

ทดสอบระบบเกม The Everlasting ครั้งที่ 1

ปัญหา : ตัวละครที่ออกแบบ ท่าวิ่งไม่สมูท

แก้ไข : วาดอนิเมชั่นวิ่งให้เฟมในการวิ่งมากขึ้น

ทดสอบระบบเกม The Everlasting ครั้งที่ 2

ปัญหา : มอนสเตอร์ไม่มีท่าตี

แก้ไข : เพิ่มอนิเมชั่นท่าตีให้มอนนเตอร์

ทดสอบระบบเกม The Everlasting ครั้งที่ 3

ปัญหา : แถบเลือดตัวละคร ไม่มี

แก้ไข : สร้างแถบเลือด โดยการใช้โค้ด

```
healthSlider.value = CalculateSliderPercentage
(playerDamageable.Health , playerDamageable.maxHealth);
```

ทดสอบระบบเกม The Everlasting ครั้งที่ 4

ปัญหา : ตัวละครท่าตีไม่ต่อเนื่อง

แก้ไข : วาดอนิเมชั่นท่าตีใหม่ และตรวจสอบท่าตีอีกครั้ง

ทดสอบระบบเกม The Everlasting ครั้งที่ 5

ปัญหา : มอนสเตอร์น้อยเกินไป

แก้ไข : วาดมอนสเตอร์เพิ่ม และวาดอนิเมชั่นมอนสเตอร์

ทดสอบระบบเกม The Everlasting ครั้งที่ 6

ปัญหา : ไม่สามารถเข้าประตูมอนสเตอร์

แก้ไข : เพิ่มโค้ด if (other.gamObject.tag == “Player”){

StarCoroutine(Teleport());

}

IEnumeator Teleport()

{

yield return new WaitForSeconds(tpTime);

Player.transform.position = new

Vector2(Portal.tranform.osition.x,PP\ortal.transform.position.y);

}

ทดสอบระบบเกม The Everlasting ครั้งที่ 7

ปัญหา : บอสไม่มีท่าตี

แก้ไข : ตรวจสอบอนิเมชั่นของบอส และแก้ไข

ทดสอบระบบเกม The Everlasting ครั้งที่ 8

ปัญหา : ตัวละครไม่สามารถกระโดดได้

แก้ไข : เพิ่มโค้ด playerPos = GameObject.FindGameObjectWithTag

("Player").GetComponent<Transform>();

timer = Random.Range(minTime, maxTime);

ทดสอบระบบเกม The Everlasting ครั้งที่ 9

ปัญหา : ตัวละครอนิเมชั่นตอนกระโดดไม่สมูท

แก้ไข : ตรวจสอบและเพิ่มลายละเอียดตอนกระโดด เพิ่มอนิเมชั่น

ทดสอบระบบเกม The Everlasting ครั้งที่ 10

ปัญหา : ตัวละครไม่สามารถเพิ่มลือดได้

แก้ไข : เพิ่มไอเท็มเพิ่มเลือดลงไป

ทดสอบระบบเกม The Everlasting ครั้งที่ 11

ไม่เกิดปัญหาใดๆทั้งสิ้นสามารถจบเกมได้โดยไม่เกิดบัค

4.2 ผลการประเมินหาคุณภาพของเกม The Everlasting

จากการทดสอบเกมโดยมีผู้ใช้งาน จำนวน 10 ท่าน เป็นผู้ประเมินคุณภาพของเกม ผลการทดสอบแสดงให้เห็นว่าเมื่อนำคะแนนเฉลี่ยของแต่ละหัวข้อมาผ่านระเบียบวิธีการทางสถิติเพื่อหาค่าเฉลี่ย (Mean) จะพบว่าค่าเฉลี่ยที่ได้อยู่ที่ 4.2 ดังนั้นเกมที่ได้ทำการสร้างขึ้นมาอยู่ในระดับ ดี

ตารางที่ 4.1 ผลการประเมินความพึงพอใจจากผู้ใช้งาน

ผลการประเมินความพึงพอใจ	**ค่าเฉลี่ย** $(\bar{x})$	**คำเบี่ยงเบนมาตรฐาน (S.D.)**	**ระดับความคิดเห็น**
1. ออกแบบสวยงาม	4.6	0.51	ดีมาก
2. ใช้งานง่าย ไม่ซับซ้อน	4.6	0.51	ดีมาก
3. มีความน่าสนใจ	4.1	0.56	ดี
4. มีความน่าใช้งาน	3.6	0.69	ดี
5. ความสนุก	4.1	0.31	ดี
ค่าเฉลี่ยรวม	4.2	0.52	ดี

จากตารางที่ 4.1 เป็นการประเมินความพึงพอใจของ Everlasting การออกแบบสวยงามมีค่าเฉลี่ยสูงสุดอยู่ที่ x = 4.6 S.D. = 0.51 อยู่ในเกณฑ์ระดับดีมาก ต่อมาความน่าสนใจมีค่าเฉลี่ยสูงสุดอยู่ที่ x = 3.6 S.D. = 0.56 อยู่ในเกณฑ์ระดับดี ต่อมาความสนุกมีค่าเฉลี่ยสูงสุดอยู่ที่ x = 4.1 S.D. = 0.31 และ สุดท้ายการใช้งานง่ายไม่ซับซ้อนมีค่าเฉลี่ยสูงสุดอยู่ที่ x = 4.6 S.D. = 0.51 อยู่ในเกณฑ์ระดับดีมาก

บทที่ 5
สรุปผล ปัญหาอุปสรรค และข้อเสนอแนะ

5.1 สรุปผล

จากการดำเนินการออกแบบและพัฒนาระบบเกม Everlasting โดยได้พัฒนามาเป็นระบบที่สมบูรณ์ได้เอาแนวคิดจาก เกมดังอื่นๆ นิยาย การ์ตูน ลงไปผสม แล้วให้ความรู้เกี่ยวกับการสร้างเกม ออกแบบตัวละคร การจัดวางตัวละคร เขียนโค้ด ให้เข้ากับระบบเกม Everlasting เพื่อตอบสนองต่อผู้ใช้งานในการสนับสนุนในการดำเนินงานได้ตามเป้าหมายของโครงงาน

ระบบเกม Everlasting สามารถจำลองมากจาก การ์ตูน นิยาย แนวMMORPG แล้วให้สนุกสนานเพลิดเพลินในการเล่น แล้วช่วยเสริมทักษะไหวพริบ ความรอบครอบ และการสังเกต ในการเล่นเพื่อที่จะได้จัดการบอส เพื่อที่จะชนะเกมนี้

จากการประเมินความพึงพอใจโดยมีผู้ประเมิน จำนวน 10 ท่านสรุปได้ว่าเกมThe Everlasting มีคุณภาพอยู่ในระดับดี สามารถใช้งานได้ตามขอบเขตที่กำหนด

5.2 ปัญหาและอุปสรรค

5.2.1 มอนส์ตีเราแต่มอนสเตอร์ตาย

5.2.2 อนิเมชั่นขาดๆหาย

5.3 แนวทางแก้ไข

5.3.1 ปรับค่าการตีสะท้อน

5.3.2 ตรวจสอบอนิเมชั่น และ เติมอนิเมชั่น

5.4 ข้อเสนอแนะ

5.4.1 ในด้านระบบการเล่นควรมีอาวุธที่มากขึ้น มีมอนสเตอร์มากขึ้น เพื่อตอบสนองความต้องการของผู้ใช้งานมากขึ้น

5.4.2 เกม The Everlasting ควรนำไปพัฒนาระบบอื่นๆ อีกเช่น ระบบสุ่มอาวุธ ระบบอัพเลเวล ระบบร้านค้า

บรรณานุกรม

ปุจฉา วิสัชนา. (2561). **Unity.** สืบค้น 10 มกราคม 2564
จาก https://th.wikipedia.org/wiki/ยูนิตี_(เกมเอนจิน)

Rapetcherman. (2557). **Adobe Photoshop.** สืบค้น 10 มกราคม 2564
จาก http://teacherjaray.blogspot.com/2014/05/blog-post.html

นางสาวกาญจนา โนหลักหมื่น. (2559). **ภาษา C#.** สืบค้น 11 มกราคม 2564
จาก https://www.binaryprogramming.net/1-1

David Capello. (2559). **Aseprite.** สืบค้น 1 กันยา 2016
จาก https://en.wikipedia.org/wiki/Aseprite

ภาคผนวก

ภาคผนวก ก.

แบบขออนุมัติโครงการ

ภาคผนวก ข.

แบบสอบถามความคิดเห็นผู้ใช้งาน

ภาคผนวก ค.

คู่มือการใช้งาน

คู่มือการใช้งาน

1.ขั้นตอนการเข้าใช้งาน

รูปภาพที่ ค.1 หน้าต่างเมนูของเกม The Everlasting

1.1 Start Game กดเพื่อเริ่มเกม

1.2 Quit Game กดเพื่อออกเกม

2.ขั้นตอนการใช้งานโปรแกรม

รูปภาพที่ ค.2 วิธีการเล่น

2.1 กดคลิ๊กซ้ายในการโจมตีมอนสเตอร์ กดSpacebar เพื่อกระโดด A,D เคลื่อที่ซ้าย และ ขวา

ภาคผนวก ง.

Source Code

AnimationStrings.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
class AnimationStrings
{
    internal static string isMoving = "isMoving";
    internal static string isGrounded = "isGrounded";
    internal static string yVelocity = "yVelocity";
    internal static string jump = "jump";
    internal static string isOnWall ="isOnWall";
    internal static string isOnCeiling ="isOnCeiling";
    internal static string attack ="attack";
    internal static string canMove = "canMove";
    internal static string hasTarget = "hasTarget";
    internal static string isAlive = "isAlive";
    public static string IsHit = "isHit";
    internal static string hit = "hit";
    internal static string lockVelocity = "lockVelocity";
}
```

Attack.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Attack : MonoBehaviour
{
    public int attackDamage = 10;
```

```
    public Vector2 knockback = Vector2.zero;
    Collider2D attackCollider;
    // Start is called before the first frame update
    void Start()
    {
        attackCollider = GetComponent<Collider2D>();
    }

    // Update is called once per frame
    void Update()
    {

    }
    private void OnTriggerEnter2D(Collider2D collision)
    {
        //ดูว่าตีได้มั้ย
        Damageable damageable = collision.GetComponent<Damageable>();
        if(damageable != null)
        {
            //ตีเป้าหมาย
            bool gotHit = damageable.Hit(attackDamage, knockback);

            if(gotHit)
            Debug.Log(collision.name + " hit for " + attackDamage);
        }
    }
}
```

Boss.cs

```
using System;
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(Rigidbody2D), typeof(TouchingDirection),
typeof(Damageable))]
public class Boss : MonoBehaviour
{
    public float walkSpeed = 3f;
    public float walkStopRate = 0.05f;
    public DetectionZone attackZone;
    Rigidbody2D rb;
    TouchingDirection touchingDirection;
    Animator animator;
    Damageable damageable;
    public bool _hasTarget = false;
    public bool hasTarget
    {
        get { return _hasTarget; }
        private set
        {
            _hasTarget = value;
            animator.SetBool(AnimationStrings.hasTarget, value);
        }
    }

    public bool CanMove
    {
        get
        {
            return animator.GetBool(AnimationStrings.canMove);
```

```
        }
    }
    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        touchingDirection = GetComponent<TouchingDirection>();
        animator = GetComponent<Animator>();
        damageable = GetComponent<Damageable>();
    }
    void Update()
    {
        hasTarget = attackZone.detectedColliders.Count > 0;
    }
    private void FixedUpdate()
    {

    }
    public void OnHit(int damage, Vector2 knockback)
    {
        rb.velocity = new Vector2(knockback.x, rb.velocity.y + knockback.y);
    }
}
```

Damageable.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.UI;

public class Damageable : MonoBehaviour
```

```
{
    public UnityEvent<int, Vector2> damageableHit;
    public UnityEvent<int, int> healthChange;
    public Slider healthBar;
    Animator animator;
    [SerializeField]
    private int _maxHealth;
    public GameObject lootDrop;
    public GameManagerScript gameManager;
    public int maxHealth
    {
        get
        {
            return _maxHealth;
        }
        set
        {
            _maxHealth = value;
        }
    }

    public int _health = 100;
    public int Health
    {
        get
        {
            return _health;
        }
        set
        {
```

```
            _health = value;
            healthChange?.Invoke(_health, maxHealth);
            //ถ้าเลือดน้อยกว่าหรือเป็น 0 ตัวละครจะตาย
            if (_health <= 0)
            {
                IsAlive = false;
                Instantiate(lootDrop, transform.position, Quaternion.identity);
            }
        }
    }
    [SerializeField]
    private bool _isAlive = true;
    [SerializeField]
    private bool isInvincible = false;

    public bool IsHit { get
        {
            return animator.GetBool(AnimationStrings.IsHit);
        }
        private set
        {
            animator.SetBool(AnimationStrings.IsHit, value);
        }
    }

    private float timeSinceHit = 0;
    public float invincibilityTimer = 0.25f;

    public bool IsAlive
    {
```

```
    get
    {
        return _isAlive;
    }
    set
    {
        _isAlive = value;
        animator.SetBool(AnimationStrings.isAlive, value);
        Debug.Log("IsAlive set " + value);
    }
}
private void Awake()
{
    animator = GetComponent<Animator>();
}
public void Update()
{
    if(_health <= 0 && IsAlive == false){
        Cursor.visible = true;
        gameManager.gameOver();
        Debug.Log("dead");
    }
    if (isInvincible)
    {
        if(timeSinceHit > invincibilityTimer)
        {
            //ลบ I-Frame
            isInvincible = false;
            timeSinceHit = 0;
        }
```

```
            timeSinceHit += Time.deltaTime;
        }
        healthBar.value = Health;
    }
    public bool LockVelocity
    {
        get
        {
            return animator.GetBool(AnimationStrings.lockVelocity);

        }
        set
        {
            animator.SetBool(AnimationStrings.lockVelocity, value);
        }

    }


    public bool Hit(int damage, Vector2 knockback)
    {
        if(IsAlive && !isInvincible)
        {
            Health -= damage;
            isInvincible = true;
            animator.SetTrigger(AnimationStrings.hit);
            LockVelocity = true;
            damageableHit?.Invoke(damage, knockback);
            CharacterEvent.characterDamaged.Invoke(gameObject, damage);
            return true;
```

```
        }
        //ตีไม่ได้
        return false;
    }
    public void Heal(int healthRestore)
    {
        if (IsAlive) {
            Health += healthRestore;
            CharacterEvent.characterHealed(gameObject, healthRestore);
        }

    }
}
```

FadeRemove.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//ลดสี Sprite ทำให้ค่อยๆหายไป
public class FadeRemove : StateMachineBehaviour
{
    public float fadetime = 0.5f;
    private float timeElapsed = 0f;
    SpriteRenderer spriteRenderer;
    GameObject objToRemove;
    Color startColor;

    // OnStateEnter is called when a transition starts and the state machine
starts to evaluate this state
```

```
    override public void OnStateEnter(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    {
        timeElapsed = 0f;
        spriteRenderer = animator.GetComponent<SpriteRenderer>();
        startColor = spriteRenderer.color;
        objToRemove = animator.gameObject;
    }

    // OnStateUpdate is called on each Update frame between
OnStateEnter and OnStateExit callbacks
    override public void OnStateUpdate(Animator animator,
AnimatorStateInfo stateInfo, int layerIndex)
    {
        timeElapsed += Time.deltaTime;
        float newAlpha = startColor.a * (1 - (timeElapsed / fadetime));
        spriteRenderer.color = new Color(startColor.r, startColor.g,
startColor.b, newAlpha);

        if(timeElapsed > fadetime)
        {
            Destroy(objToRemove);
        }
    }

    // OnStateExit is called when a transition ends and the state machine
finishes evaluating this state
    //override public void OnStateExit(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    //{
```

```
    //
    //}

    // OnStateMove is called right after Animator.OnAnimatorMove()
    //override public void OnStateMove(Animator animator,
AnimatorStateInfo stateInfo, int layerIndex)
    //{
    //    // Implement code that processes and affects root motion
    //}

    // OnStateIK is called right after Animator.OnAnimatorIK()
    //override public void OnStateIK(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    //{
    //    // Implement code that sets up animation IK (inverse kinematics)
    //}
}
```

FlyingBook.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FlyingBook : MonoBehaviour
{
    public float flightSpeed = 2f;
    public float waypointReachedDistance = 0.1f;
    public DetectionZone biteDetectionZone;
    public List<Transform> waypoints;
    Animator animator;
```

```
Rigidbody2D rb;
Damageable damageable;
Transform nextWayPoint;
int waypointNum = 0;

public bool _hasTarget = false;

public bool HasTarget
{
    get
    {
        return _hasTarget;
    }
    private set
    {
        _hasTarget = value;
        animator.SetBool(AnimationStrings.hasTarget, value);
    }
}
public bool CanMove
{
    get
    {
        return animator.GetBool(AnimationStrings.canMove);
    }
}
private void Awake()
{
    animator = GetComponent<Animator>();
    rb = GetComponent<Rigidbody2D>();
```

```
        damageable = GetComponent<Damageable>();
    }
    private void Start()
    {
        nextWayPoint = waypoints[waypointNum];
    }
    void Update()
    {
        HasTarget = biteDetectionZone.detectedColliders.Count > 0;
    }
    private void FixedUpdate()
    {
        if (damageable.IsAlive)
        {
            if (CanMove)
            {
                Flight();
            }
            else
            {
                rb.velocity = Vector3.zero;
            }
        }
    }

    private void Flight()
    {
        // บินไปจุดถัดไป
        Vector2 directionToWaypoint = (nextWayPoint.position -
transform.position).normalized;
```

```
        //เช็คว่าถึงจุดหรือยัง
        float distance = Vector2.Distance(nextWayPoint.position,
transform.position);
        rb.velocity = directionToWaypoint * flightSpeed;
        UpdateDirection();
        //ดูว่าต้องเปลี่ยนจุดมั้ย
        if(distance <= waypointReachedDistance)
        {
            //เปลี่ยนจุด
            waypointNum++;
            if(waypointNum >= waypoints.Count)
            {
                //วนลูปกลับไปที่จุดแรก
                waypointNum = 0;
            }
            nextWayPoint = waypoints[waypointNum];
        }
    }

    private void UpdateDirection()
    {
        Vector3 locScale = transform.localScale;
        if(transform.localScale.x > 0)
        {
            //หันขวา
            if(rb.velocity.x > 0)
            {
                //พลิก
                transform.localScale = new Vector3(-1 * locScale.x, locScale.y,
locScale.z);
```

```
            }
        }
        else
        {
            //หันซ้าย
            if(rb.velocity.x < 0)
            {
            transform.localScale = new Vector3(-1 * locScale.x, locScale.y,
locScale.z);
            }
        }
    }
}
```

GameManagerScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameManagerScript : MonoBehaviour
{
    public GameObject gameOverUI;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
```

```
    }
    public void gameOver()
    {
        gameOverUI.SetActive(true);
    }
}
```

HealthBar.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class HealthBar : MonoBehaviour
{
    public Slider healthSlider;
    Damageable playerDamageable;
    public TMP_Text healthBarText;

    private void Awake()
    {
        GameObject player = GameObject.FindGameObjectWithTag("Player");

        if(player == null)
        {
            Debug.Log("ไม่พบผู้เล่น ดูว่ามีแท็ก Player หรือไม่");
        }
```

```
        playerDamageable = player.GetComponent<Damageable>();
    }
    void Start()
    {
        healthSlider.value =
CalculateSliderPercentage(playerDamageable.Health,
playerDamageable.maxHealth);
        healthBarText.text = "HP " + playerDamageable.Health;
    }

    private void OnEnable()
    {
        playerDamageable.healthChange.AddListener(OnPlayerHealthChange);
    }

    private void OnDisable()
    {

playerDamageable.healthChange.RemoveListener(OnPlayerHealthChange);
    }

    private float CalculateSliderPercentage(float currentHealth, float
maxHealth)
    {
         return currentHealth / maxHealth;
    }

    private void OnPlayerHealthChange(int newHealth, int maxHealth)
    {
```

```
        healthSlider.value = CalculateSliderPercentage(newHealth,
maxHealth);
        healthBarText.text = "HP " + newHealth;
    }
}
```

HealthPickup.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HealthPickup : MonoBehaviour
{
    public int healthRestore = 400;

    AudioSource pickupSource;

    private void Awake()
    {
        pickupSource = GetComponent<AudioSource>();
    }

    void Start()
    {

    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        Damageable damageable = collision.GetComponent<Damageable>();
        if (damageable)
```

```
        {
            damageable.Heal(healthRestore);
            if (pickupSource)
                AudioSource.PlayClipAtPoint(pickupSource.clip,
gameObject.transform.position, pickupSource.volume);
            Destroy(gameObject);
        }
    }
}
```

HealthText.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class HealthText : MonoBehaviour
{
    public Vector3 moveSpeed = new Vector3(0, 75 ,0);
    public float timeToFade = 1f;
    RectTransform textTransform;
    TextMeshProUGUI textMeshPro;
    private float timeElapsed = 0f;
    private Color startColor;
    public void Awake()
    {
        textTransform = GetComponent<RectTransform>();
        textMeshPro = GetComponent<TextMeshProUGUI>();
        startColor = textMeshPro.color;
    }
```

```
    // Update is called once per frame
    private void Update()
    {
        textTransform.position += moveSpeed * Time.deltaTime;
        timeElapsed += Time.deltaTime;
        if(timeElapsed < timeToFade)
        {
            float fadeAlpha = startColor.a * (1 - (timeElapsed / timeToFade));
            textMeshPro.color = new Color(startColor.r,startColor.g,startColor.b,
fadeAlpha);
        }
        else
        {
            Destroy(gameObject);
        }
    }
}
```

JumpBehavior.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class JumpBehavior : StateMachineBehaviour
{



    private float timer;
    public float minTime;
    public float maxTime;
```

```
    private Transform playerPos;
    public float speed;

    override public void OnStateEnter(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    {
        playerPos =
GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>()
;
        timer = Random.Range(minTime, maxTime);
    }

    override public void OnStateUpdate(Animator animator,
AnimatorStateInfo stateInfo, int layerIndex)
    {
        if (timer <= 0)
        {
            animator.SetTrigger("idle");
        }
        else
        {
            timer -= Time.deltaTime;
        }

        Vector2 target = new Vector2(playerPos.position.x,
animator.transform.position.y);
        animator.transform.position =
Vector2.MoveTowards(animator.transform.position, target, speed *
Time.deltaTime);
    }
```

```
    override public void OnStateExit(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    {

    }

}
```

PlayerController.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;

[RequireComponent(typeof(Rigidbody2D),
typeof(TouchingDirection),typeof(Damageable))]
public class PlayerController : MonoBehaviour
{
    public float walkSpeed = 5f;
    AudioSource audioSource;
    Vector2 moveInput;
    TouchingDirection touchingdirection;
    Damageable damageable;
    [SerializeField]
    private bool _isMoving = false;
    public bool IsMoving
    {
        get
        {
```

```
        return _isMoving;
    }
    private set
    {
        _isMoving = value;
        animator.SetBool(AnimationStrings.isMoving, value);
    }
}

Rigidbody2D rb;
Animator animator;
SpriteRenderer spriterenderer;

bool facingRight = true;
public float jumpImpulse = 10f;

private void Awake()
{
    rb = GetComponent<Rigidbody2D>();
    animator = GetComponent<Animator>();
    spriterenderer = GetComponent<SpriteRenderer>();
    touchingdirection = GetComponent<TouchingDirection>();
    damageable = GetComponent<Damageable>();
    audioSource = GetComponent<AudioSource>();
}

// Start is called before the first frame update
void Start()
{
```

```
}

// Update is called once per frame
void Update()
{

}
private void FixedUpdate()
{
    if (!damageable.LockVelocity)
    rb.velocity = new Vector2(moveInput.x * walkSpeed, rb.velocity.y);
    float move = Input.GetAxisRaw("Horizontal");
    animator.SetFloat(AnimationStrings.yVelocity, rb.velocity.y);
    if (move < 0 && facingRight)
    {
        flip();
    }
    else if (move > 0 && !facingRight)
    {
        flip();
    }

}
public void onMove(InputAction.CallbackContext context)
{
    moveInput = context.ReadValue<Vector2>();
    if (IsAlive)
    {
        IsMoving = moveInput != Vector2.zero;
    }
```

```
    else
    {
        IsMoving = false;
    }

}
void flip()
{
    facingRight = !facingRight;
    transform.Rotate(0, 180f, 0);
}

public void OnJump(InputAction.CallbackContext context)
{
    if (context.started && touchingdirection.isGrounded)
    {
        animator.SetTrigger(AnimationStrings.jump);
        rb.velocity = new Vector2(rb.velocity.x, jumpImpulse);
    }
}
public void OnAttack(InputAction.CallbackContext context)
{
    if (context.started)
    {
        animator.SetTrigger(AnimationStrings.attack);
    }
}
public bool IsAlive
{
    get
```

```
        {
            return animator.GetBool(AnimationStrings.isAlive);
        }
    }


    public void OnHit(int damage, Vector2 knockback)
    {
        rb.velocity = new Vector2(knockback.x, rb.velocity.y + knockback.y);
    }
}
```

PlayOneShotBehavior.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayOneShotBehavior : StateMachineBehaviour
{
    public AudioClip sountToPlay;
    public float volume = 1f;
    public bool playOnEnter = true, playOnExit = false, playAfterDelay =
false;

    public float playDelay = 0.25f;
    private float timeSinceEnter = 0;
    private bool hasDelayedSoundPlay = false;
    // OnStateEnter is called when a transition starts and the state machine
starts to evaluate this state
    override public void OnStateEnter(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
```

```
    {
        if (playOnEnter)
        {
            AudioSource.PlayClipAtPoint(sountToPlay,
animator.gameObject.transform.position, volume);
        }
        timeSinceEnter = 0f;
        hasDelayedSoundPlay = false;
    }

    // OnStateUpdate is called on each Update frame between
OnStateEnter and OnStateExit callbacks
    override public void OnStateUpdate(Animator animator,
AnimatorStateInfo stateInfo, int layerIndex)
    {
        if (playAfterDelay && !hasDelayedSoundPlay)
        {
            timeSinceEnter += Time.deltaTime;
            if (timeSinceEnter > playDelay)
            {
                AudioSource.PlayClipAtPoint(sountToPlay,
animator.gameObject.transform.position, volume);
                hasDelayedSoundPlay = true;
            }
        }
    }

    // OnStateExit is called when a transition ends and the state machine
finishes evaluating this state
```

```
    override public void OnStateExit(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    {
        if (playOnExit)
        {
            AudioSource.PlayClipAtPoint(sountToPlay,
animator.gameObject.transform.position, volume);
        }
    }
}
```

SceneController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneController : MonoBehaviour
{
    public static SceneController instance;

    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
```

```
        }
    }
    public void NextLevel()
    {

SceneManager.LoadSceneAsync(SceneManager.GetActiveScene().buildIndex
+ 1);
    }
    public void LoadScene(string sceneName)
    {
        SceneManager.LoadSceneAsync(sceneName);
    }
}
```

SetBoolBehaviour.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SetBoolBehaviour : StateMachineBehaviour
{
    public string boolname;
    public bool updateOnState;
    public bool updateOnStateMachine;
    public bool valueOnEnter, valueOnExit;
    // OnStateEnter is called before OnStateEnter is called on any state
inside this state machine
    override public void OnStateEnter(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    {
        if (updateOnState)
```

```
        {
            animator.SetBool(boolname, valueOnEnter);
        }
    }

    // OnStateUpdate is called before OnStateUpdate is called on any state inside this state machine
    //override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
    //{
    //
    //}

    // OnStateExit is called before OnStateExit is called on any state inside this state machine
    //override public void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
    //{
    //
    //}

    // OnStateMove is called before OnStateMove is called on any state inside this state machine
    override public void OnStateMove(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
    {
        if (updateOnState)
        {
            animator.SetBool(boolname, valueOnExit);
        }
```

```
    }

    // OnStateIK is called before OnStateIK is called on any state inside this state machine
    //override public void OnStateIK(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
    //{
    //
    //}

    // OnStateMachineEnter is called when entering a state machine via its Entry Node
    override public void OnStateMachineEnter(Animator animator, int stateMachinePathHash)
    {
        if(updateOnStateMachine)
        animator.SetBool(boolname, valueOnEnter);
    }

    // OnStateMachineExit is called when exiting a state machine via its Exit Node
    override public void OnStateMachineExit(Animator animator, int stateMachinePathHash)
    {
        if(updateOnStateMachine)
        animator.SetBool(boolname, valueOnExit);
    }
}
```

Teleporter.cs

```
using System;
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Teleporter : MonoBehaviour
{
    [SerializeField] GameObject Portal, Player;
    [SerializeField] float tpTime;

    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.gameObject.tag == "Player")
        {
            StartCoroutine(Teleport());
        }
    }
    IEnumerator Teleport()
    {
        yield return new WaitForSeconds(tpTime);
        Player.transform.position = new Vector2(Portal.transform.position.x,
Portal.transform.position.y);
    }
}
```

TouchingDirection.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TouchingDirection : MonoBehaviour
{
```

```
public ContactFilter2D castFilter;
public float groundDistance = 0.05f;
public float wallDistance = 0.2f;
public float ceilingDistance = 0.05f;
CapsuleCollider2D touchingCol;
Animator animator;
RaycastHit2D[] groundHits = new RaycastHit2D[5];
RaycastHit2D[] wallHits = new RaycastHit2D[5];
RaycastHit2D[] ceilingHits = new RaycastHit2D[5];
[SerializeField]
private bool _isGrounded;
public bool isGrounded { get
    {
        return _isGrounded;
    }
    private set
    {
        _isGrounded = value;
        animator.SetBool(AnimationStrings.isGrounded, value);
    }
}
[SerializeField]
private bool _isOnWall;
public bool isOnWall
{
    get
    {
        return _isOnWall;
    }
    private set
```

```
        {
            _isOnWall = value;
            animator.SetBool(AnimationStrings.isOnWall, value);
        }
    }
    [SerializeField]
    private bool _isOnCeiling;
    private Vector2 wallCheckDirection => gameObject.transform.localScale.x
> 0 ? Vector2.right : Vector2.left;

    public bool isOnCeiling
    {
        get
        {
            return _isOnCeiling;
        }
        private set
        {
            _isOnCeiling = value;
            animator.SetBool(AnimationStrings.isOnCeiling, value);
        }
    }



    private void Awake()
    {
        touchingCol = GetComponent<CapsuleCollider2D>();
        animator = GetComponent<Animator>();
    }
    void Start()
```

```
    {

    }

    // Update is called once per frame
    void FixedUpdate()
    {
        isGrounded = touchingCol.Cast(Vector2.down, castFilter, groundHits,
groundDistance) > 0;
        isOnWall = touchingCol.Cast(wallCheckDirection, castFilter, wallHits,
wallDistance) > 0;
        isOnCeiling = touchingCol.Cast(Vector2.up, castFilter, ceilingHits,
ceilingDistance) > 0;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TouchingDirection : MonoBehaviour
{
    public ContactFilter2D castFilter;
    public float groundDistance = 0.05f;
    public float wallDistance = 0.2f;
    public float ceilingDistance = 0.05f;
    CapsuleCollider2D touchingCol;
    Animator animator;
    RaycastHit2D[] groundHits = new RaycastHit2D[5];
    RaycastHit2D[] wallHits = new RaycastHit2D[5];
    RaycastHit2D[] ceilingHits = new RaycastHit2D[5];
```

```
    [SerializeField]
    private bool _isGrounded;
    public bool isGrounded { get
        {
            return _isGrounded;
        }
        private set
        {
            _isGrounded = value;
            animator.SetBool(AnimationStrings.isGrounded, value);
        }
    }
    [SerializeField]
    private bool _isOnWall;
    public bool isOnWall
    {
        get
        {
            return _isOnWall;
        }
        private set
        {
            _isOnWall = value;
            animator.SetBool(AnimationStrings.isOnWall, value);
        }
    }
    [SerializeField]
    private bool _isOnCeiling;
    private Vector2 wallCheckDirection => gameObject.transform.localScale.x
> 0 ? Vector2.right : Vector2.left;
```

```
    public bool isOnCeiling
    {
        get
        {
            return _isOnCeiling;
        }
        private set
        {
            _isOnCeiling = value;
            animator.SetBool(AnimationStrings.isOnCeiling, value);
        }
    }



    private void Awake()
    {
        touchingCol = GetComponent<CapsuleCollider2D>();
        animator = GetComponent<Animator>();
    }
    void Start()
    {

    }

    // Update is called once per frame
    void FixedUpdate()
    {
        isGrounded = touchingCol.Cast(Vector2.down, castFilter, groundHits,
groundDistance) > 0;
```

```
        isOnWall = touchingCol.Cast(wallCheckDirection, castFilter, wallHits, wallDistance) > 0;
        isOnCeiling = touchingCol.Cast(Vector2.up, castFilter, ceilingHits, ceilingDistance) > 0;
    }
}
```

UIManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.InputSystem;

public class UIManager : MonoBehaviour
{
    public GameObject damageTextPrefab;
    public GameObject HealthTextPrefab;
    public Canvas gameCanvas;

    private void Awake()
    {
        gameCanvas = FindObjectOfType<Canvas>();

    }

    private void OnEnable()
    {
        CharacterEvent.characterDamaged += (CharacterTookDamage);
        CharacterEvent.characterHealed += (CharacterHealed);
    }
```

```
    private void OnDisable()
    {
        CharacterEvent.characterDamaged -= (CharacterTookDamage);
        CharacterEvent.characterHealed -= (CharacterHealed);
    }

    public void CharacterTookDamage(GameObject character, int
damageReceived)
    {
        Vector3 spawnPosition =
Camera.main.WorldToScreenPoint(character.transform.position);

        TMP_Text tmpText = Instantiate(damageTextPrefab, spawnPosition,
Quaternion.identity, gameCanvas.transform).GetComponent<TMP_Text>();
        tmpText.text = damageReceived.ToString();
  }

    public void CharacterHealed(GameObject character, int healthRestored)
    {
        Vector3 spawnPosition =
Camera.main.WorldToScreenPoint(character.transform.position);

        TMP_Text tmpText = Instantiate(damageTextPrefab, spawnPosition,
Quaternion.identity, gameCanvas.transform).GetComponent<TMP_Text>();
        tmpText.text = healthRestored.ToString();
    }

    public void OnExitGame(InputAction.CallbackContext context)
    {
```

```
        if (context.started)
        {
#if (UNITY_EDITOR || DEVELOPMENT_BUILD)
            Debug.Log(this.name + " : " + this.GetType() + " : " +
System.Reflection.MethodBase.GetCurrentMethod().Name);
#endif

#if (UNITY_EDITOR)
            UnityEditor.EditorApplication.isPlaying = false;
#elif (UNITY_STANDALONE)
            Application.Quit();
#elif (UNITY_WEBGL)
            SceneManager.LoadScene("QuitScene");
#endif
        }
    }

}
```

ประวัติผู้จัดทำ

ประวัติผู้จัดทำโครงการ

ชื่อ นายณัฐกรพวงทอง

เกิดเมื่อวันที่ 12 เดือน มิถุนายน พ.ศ. 2546

ที่อยู่ 31/39 หมู่ 7 ต.พลิ้ว อ.แหลมสิงห์ จ.จันทบุรี

หมายเลขโทรศัพท์ 097-306-2544

E-Mail : nattagorn789@gmail.com

การศึกษา

ชั้นประถมศึกษา จากโรงเรียนวัดหนองบัว(ปทุมาภิพัฒน์)

ชั้นมัธยมศึกษาตอนต้น จากโรงเรียนวัดหนองบัว(ปทุมาภิพัฒน์)

ชั้นประกาศนียบัตรวิชาชีพ จากวิทยาลัยเทคนิคจันทบุรี

ขณะนี้กำลังศึกษาอยู่ในชั้นประกาศนียบัตรวิชาชีพชั้นสูงที่วิทยาลัยเทคนิคจันทบุรี

ประวัติผู้จัดทำโครงการ

ชื่อ นายวชิรวิทย์ สุระนันท์

เกิดเมื่อวันที่ 14 เดือน มกราคม พ.ศ. 2547

อยู่ที่ 102 ต.วัดใหม่ อ.เมือง จ.จันทบุรี

หมายเลขโทรศัพท์ 084-514-9455

E-Mail : wachirawitsuranan@gmail.com

การศึกษา

ชั้นประถมศึกษา จากโรงเรียนอำนวยวิทย์

ชั้นมัธยมศึกษาตอนต้น จากโรงเรียนอำนวยวิทย์พัฒน์

ชั้นประกาศนียบัตรวิชาชีพ จากวิทยาลัยเทคนิคจันทบุรี

ขณะนี้กำลังศึกษาอยู่ในชั้นประกาศนียบัตรวิชาชีพชั้นสูงที่วิทยาลัยเทคนิคจันทบุรี