รายงานผลโครงการและงานวิจัย

เรื่อง

เกม Eerie Woods

นายนิรุทธ์ อินวัฒนา

นายพงษ์พัฒน์ สีแก

โครงการนี้เป็นส่วนหนึ่งของวิชาโครงการ รหัส 30128-8501

หลักสูตร ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

สาขาวิชาเทคโนโลยีคอมพิวเตอร์

ภาคเรียนที่ 2 ปีการศึกษา 2566

วิทยาลัยเทคนิคจันทบุรี สถาบันการอาชีวศึกษาภาคตะวันออก

หัวข้อวิจัย Eerie Woods
ผู้ดำเนินการวิจัย นายพงพัฒน์ สีแก
นายนิรุทธ์ อินวัฒนา
ที่ปรึกษา นายฉัตรพฤกษ์ สีทิม
นางสาวประภาพร บันเทา
สาขาวิชา เทคโนโลยีคอมพิวเตอร์
สาขางาน คอมพิวเตอร์ระบบเครือข่าย
ปีการศึกษา 2566

บทคัดย่อ

โครงการนี้มีวัตถุประสงค์เพื่อสร้างเกม Eerie Woods ให้สำเร็จลุล่วง เพื่อศึกษาการใช้งานโปรแกรม Unity 3D และ ภาษา C# เพื่อหาประสิทธิภาพของเกม Eerie Woods ด้วยแบบสอบถามความพึงพอใจ

ในการจัดทำการศึกษาเกี่ยวกับการสร้างเกม Eerie Woods ด้วยโปรแกรม Unity 3D หลังจากนั้นได้ทำการวางแผนการทำงานโดยใช้ความรู้ที่ได้ศึกษามาประยุกต์ใช้ในการสร้าง ปริศนา และจัดทำแบบสอบถามเพื่อประเมินความพึงพอใจของการจัดทำโครงการ การสร้างเกม Eerie Woods

ผลการสำรวจความพึงพอใจจากแบบสอบถามความพึงพอใจ จำนวน 10 ท่าน สรุปผลการประเมินที่ได้จากการสอบถามมีโดยรวมแล้ว มีความพึงพอใจเฉลี่ยอยู่ที่ ($\bar{X}$ = 4.22) และค่าเบี่ยงเบนมาตรฐาน **(S.D.)** เท่ากับ 0.56 ซึ่งค่าเฉลี่ยโดยร่วมอยู่ในระดับ ดี

Research Title Eerie Woods

Research Ms. Pongpat Sikae

Ms. Nirut inwattana

Research Consultants Mr. Chattapluek Seethim

Ms. Prapaporn Banthao

Organization Technology computer departmeny

Chanthaburi technical collage

Academic Year 2023

Abstract

The objective of this project is to create a successful Eerie Woods game. To study the use of the Unity 3D program and C# language to determine the efficiency of the Eerie Woods game using a satisfaction questionnaire.

In preparing a study about creating the game Eerie Woods with the Unity 3D program, after that a work plan was made using the knowledge learned to apply it in the creation. puzzles and create a questionnaire to assess satisfaction with the project, creating the Eerie Woods game.

Results of the satisfaction survey from the satisfaction questionnaire of 10 people. Summary of the evaluation results obtained from the questionnaire are overall. The average satisfaction is ($\bar{X}$ = 4.22) and the standard deviation (S.D.) is 0.56, which the overall average is at a good level.

กิตติกรรมประกาศ

โครงการ Eerie Woods สำเร็จลุล่วงได้ด้วยความกรุณาจากอาจารย์ฉัตรพฤกษ์ สีทิม อาจารย์ประภาพร บันเทา ครูที่ปรึกษาที่ได้คำเสนอแนะ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่างๆ มาโดยตลอด จนโครงการเล่มนี้เสร็จสมบูรณ์ ผู้จัดทำจึงขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณอาจารย์ นายฉัตรพฤกษ์ สีทิม นางสาวประภาพร บันเทา กรรมการสอบโครงการวิจัยนี้ที่ได้กรุณาให้ข้อเสนอแนะ แก้ไข และให้แนวคิดต่าง ๆ ที่เป็นประโยชน์

สุดท้ายนี้คณะผู้จัดทำหวังเป็นอย่างยิ่งว่าโครงการนี้ จะเป็นประโยชน์สำหรับผู้ที่สนใจศึกษาไม่มากก็น้อย หากผิดพลาดและเกิดข้อบกพร่องใดๆ คณะผู้จัดทำต้องขออภัยเป็นอย่างสูงไว้ ณ ทีนี้

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงการ

ในปัจจุบันเทคโนโลยีและเกมเป็นสิ่งที่ทุกคนสามารถเข้าถึงได้ง่ายไม่ว่าจะเป็นเด็ก เยาวชน หรือผู้ใหญ่เนื่องจากเกมนั้นหาเล่นได้ง่ายและให้ความสนุกเพลินซึ่งเกมในปัจจุบันนั้นก็มีมากมายหลายรูปแนว เช่น เกมแนว RPG (role-playing game) โดยที่ผู้เล่นสมมุติรับบทเป็นตัวละครหนึ่งในเกม เกมแนว FPS (first-person shooter) เป็นวิดีโอเกมประเภทหนึ่งที่มีจุดศูนย์กลางอยู่ที่อาวุธปืนและการโจมตีโดยใช้อาวุธผ่านมุมมองบุคคลที่หนึ่ง เกมแนว เกมวางแผน (Strategy) วิดีโอเกมหลักที่เน้นการคิดและการวางดูแลทรัพยากร และดูแลหมู่บ้าน เกมแนวปริศนา (Puzzle Game) เป็นเกมที่ได้รับความนิยมสำหรับทุกเพศทุกวัย เพราะแนวเกมนี้เป็นการแก้ไขปริศนา ใช้ทักษะการคิดแก้ไขปัญหาต่างๆ เกมแนวกีฬา (Sport Game) เป็นกึ่งๆ เกมจำลองการเล่นกีฬาแต่ละชนิด โดยส่วนมากเกมกีฬามักจะมีความถูกต้องและเที่ยงตรงในกฎกติกาค่อนข้างมาก เกมต่อสู้ (Fighting) เป็นเกมที่ผู้เล่นควบคุมตัวละครในจอภาพและเข้าร่วมในการต่อสู้ระยะประชิดกับคู่ต่อสู้ ตัวละครเหล่านี้จะมีระดับพลังและจำนวนรอบแข่งขันเท่ากัน และต่อสู้กันในสนามประลอง เกมแนว (Horror) คือเกมสยองขวัญที่มีจุดมุ่งหมาย เน้นการนำเสนอให้ผู้เล่นรู้สึกหวาดกลัวด้วยสภาวะทางอารมณ์ ความเครียด และจิตใจที่ผิดปกติ แต่ไม่ค่อยมีเกมแนว (Survival Horror) ผู้จัดทำโครงงานได้เห็นถึงความสำคัญในจุดนี้จึงมีความคิดที่จะสร้างเกมแนวนี้ขึ้นมาเพื่อให้ความสนุกและความตื่นเต้นกับผู้เล่นเกมกับผู้เล่นด้วย

เนื่องจากในปัจจุบันเกมนั้นเข้าถึงได้ง่ายจึงมีเกมที่เนื่อหาเกมไม่ดี เนื้อหารุนแรงและไม่เหมาะสมกับเด็กและเยาวชนและอาจทำให้เกิดการใช้ความรุนแรงได้ผู้จัดทำโครงงานจึงมีความคิดที่จะสร้างเกมที่สร้างสรรค์และช่วยพัฒนาทักษะการเล่นเกมของผู้เล่นเนื่องจากผู้จัดทำโครงงานมีความสนใจและชื่นชอบในการเล่นเกมและได้มีการศึกษาโค้ดและการใช้งานโปรแกรมในการสร้างเกมผู้จัดทำจึงมีจุดมุ่งหมายในการสร้างเกมนี้ขึ้นมา

จากเหตุผลข้างต้นจึงเป็นสาเหตุให้ผู้จัดทำโครงงานมีความต้องการที่จะสร้างเกม Eerie Woods ที่ให้ความสนุกสนานตื่นเต้นแก่ผู้เล่นโดยใช้โปรแกรม Unity 3D และภาษา C# ขึ้นมาเพื่อต้องการให้ ผู้เล่นที่ได้พัฒนาทักษะการเล่นเกมของผู้เล่นและได้ประโยชน์จากการเล่นเกมมากกว่าการใช้ความรุนแรง

1.2 วัตถุประสงค์ของโครงการและการวิจัย

1.2.1 เพื่อสร้างเกม Eerie Woods

1.2.2 เพื่อศึกษาการใช้โปรแกรม Unity 3D และ ภาษา C#

1.2.3 เพื่อหาประสิทธิภาพของเกม Eerie Woods

1.3 ขอบเขตขอบเขตของโครงการและงานวิจัย

1.3.1 ประชากรและกลุ่มตัวอย่าง

1.3.1.1 ประชากร คือ นักศึกษาแผนกเทคโนโลยีคอมพิวเตอร์

กลุ่มตัวอย่าง คือ นักศึกษาแผนกเทคโนโลยีคอมพิวเตอร์ระดับ ปวส.2/1

1.3.2 ตัวแปรที่ศึกษา

1.3.2.1 ตัวแปรต้น เกม Eerie Woods

1.3.2.2 ตัวแปรตาม ความพึงพอใจของผู้ทดลองเล่นเกม Eerie Woods

ความเสถียรของเกมและประสิทธิภาพของเกม

1.3.3 เครื่องมือที่ใช้ในโครงการ

1.3.3.1 แบบสอบถามความพึงพอใจของผู้เล่นเกม Eerie Woods และ

ประสิทธิภาพของเกม Eerie Woods

1.3.4 วิธีการเก็บรวบรวมข้อมูล

1.3.4.1 กำหนดกลุ่มประชากรและกลุ่มตัวอย่าง

1.3.4.2 ศึกษาข้อมูลเบื้องต้น

1.3.4.3 ออกแบบและสร้างเกม Eerie Woods

1.3.4.4 นำไปทดลองเล่นจริง

1.3.4.5 นำแบบสอบถามความพึงพอใจให้กลุ่มผู้ทดลองได้ลงความคิดเห็น

1.3.4.6 เก็บรวบรวมแบบสอบถามทั้งหมดและนำมาประมวลผลให้สมบูรณ์ต่อไป

1.3.5 สถิติที่ใช้ในการวิเคราะห์ข้อมูล

1.3.5.1 ค่าเฉลี่ย โดยใช้สูตร

ค่าเฉลี่ย

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

$\vec{x}$ = ค่าเฉลี่ยของคะแนน

$\sum x$ = ผลรวมของคะแนน

N = จำนวน

1.3.5.2 ส่วนเบี่ยงเบนมาตรฐานโดยใช้สูตร

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

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

S.D แทน ส่วนเบี่ยงเบนมาตรฐาน

X แทน คะแนนแต่ละคน

N แทน จำนวน

1.4 ประโยชน์ของโครงการ

1.4.1 ได้เกม Eerie Woods

1.4.2 ได้ทักษะการใช้โปรแกรม Unity 3D และ ภาษา C#

1.4.3 ประสิทธิภาพของเกม Eerie Woods มีความแม่นยำร้อยละ 80

1.5 สมมติฐานของงานวิจัย

ได้เกมที่มีคุณภาพ ที่สามารถมอบความบันเทิง ความสนุกสนาน และความตื่นเต้นได้

1.6 นิยามศัพท์

1.6.1 เกม หมายถึง เพื่อประโยชน์อย่างใดอย่างหนึ่งเพื่อความ สนุกสนานบันเทิง เพื่อฝึกทักษะเพื่อการเรียนรู้ และบางครั้งอาจใช้เพื่อประโยชน์ทางการศึกษา

1.6.2 Game engine หมายถึง เป็น ซอฟต์แวร์ (Software) ที่ถูกสร้างขึ้นมาเพื่อใช้เป็น โครงร่างซอฟต์แวร์ (Framework) สำหรับพัฒนาเกมโดยมันช่วยให้นักพัฒนาเกมไม่จำเป็นต้องสร้างระบบ ต่าง ๆ

1.6.3 survival horror หมายถึง แนวเกมย่อยของวิดีโอ เกมแนวแอ็กชันผจญภัย ที่มีเนื้อเรื่องมานิยายแนวสยองขวัญตัวละครที่ผู้เล่นบังคับจะอ่อนแอและมีอาวุธน้อยโดย เน้นการหลบหนีมากกว่าต่อสู้

1.6.4 developer หมายถึง นักพัฒนาซอฟต์แวร์ ผู้ที่ทำการพัฒนา Digital Product ต่าง ๆด้วยการ Coding หรือการเขียนโปรแกรม ไม่ว่าจะเป็นการพัฒนาเว็บไซต์ แอปพลิเคชัน

1.7 ระยะเวลาในการทำโครงการ

ระยะเวลาในการวิจัย เป็นเวลาทั้งสิ้น 18 สัปดาห์ตั้งแต่วันที่ 16 ตุลาคม 2566 ถึงวันที่ 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. จัดทำเอกสารโครงงาน																	◄	►

1.8 งบประมาณและทรัพยากร

ตารางที่ 1.2 งบประมาณที่ใช้ในการจัดทำโครงงาน

ที่	รายการพัสดุ	จำนวน	หน่วย	ราคาบาท
1	ค่าจัดทำเอกสาร	1	1500	1500
รวม				1500 .-

บทที่ 2
แนวคิด ทฤษฎี เอกสารและงานวิจัยที่เกี่ยวข้อง

โครงการ Eerie Woods มีวัตถุประสงค์เพื่อสร้างเกม Eerie Woods เพื่อเสริมสร้างทักษะการการสร้างเกม เพื่อเสริมสร้างทักษะการเขียนโปรแกรมภาษา C# เพื่อให้โครงการบรรลุวัตถุประสงค์ และเป็นไปตามขอบเขตที่กำหนดไว้ คณะผู้จัดทำได้ทำการศึกษาทฤษฎีและเนื้อหาที่เกี่ยวข้อง ดังนี้

2.1 Unity

2.2 Adobe Photoshop

2.3 Blender 3D

2.4 ภาษา C#

2.5 storyboard

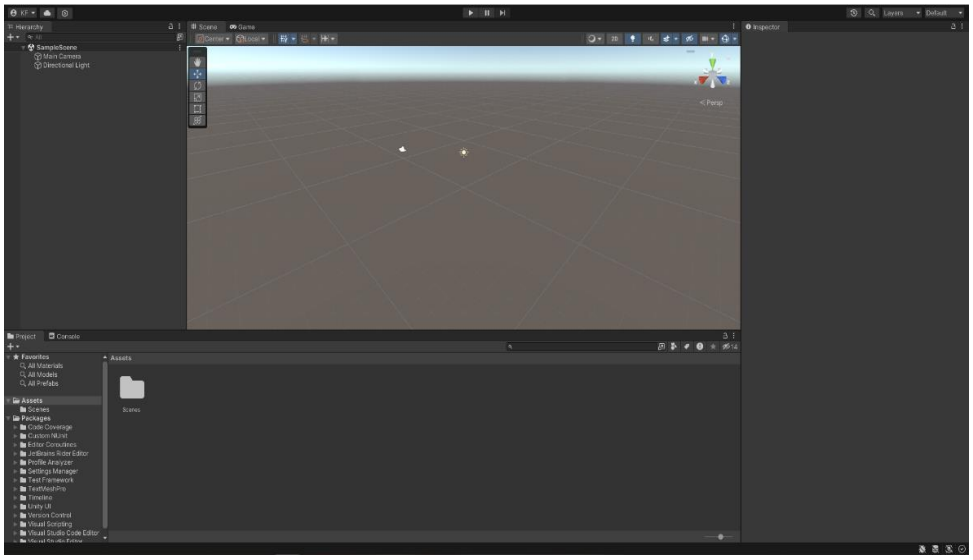
2.6 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 Unity (ที่มา https://www.wiki3.th-th.nina.az/Unity_(game_engine).html)

ยูนิตี (อังกฤษ: Unity) เป็นเกมเอนจินแบบข้ามแพลตฟอร์มสำหรับใช้ในการพัฒนาวีดีโอเกมทั้งแบบ 2 มิติและ 3 มิติรวมทั้งการสร้างซิมูเลชั่นต่างๆลงบนเครื่องคอมพิวเตอร์(ทั้งแบบตั้งโต๊ะและแล็ปท็อป), คอนโซล, สมาร์ททีวี, เว็บไซต์ และอุปกรณ์พกพาต่างๆ ยูนิตีถูกพัฒนาโดย Unity Technologies และเปิดตัวครั้งแรกในเดือนมิถุนายน ค.ศ.2005 ในงานWorldwide Developers Conference ที่ Apple Inc. โดยตัวเอนจินในขณะนั้นรองรับการใช้งานบนแพลตฟอร์ม MacOSX โดยเฉพาะเท่านั้นแต่ในปัจจุบัน (ค.ศ. 2021) ยูนิตีได้ขยายการรองรับไปยังแพลตฟอร์มอื่นๆเพิ่มเติมรวมกว่า 18 แพลตฟอร์ม

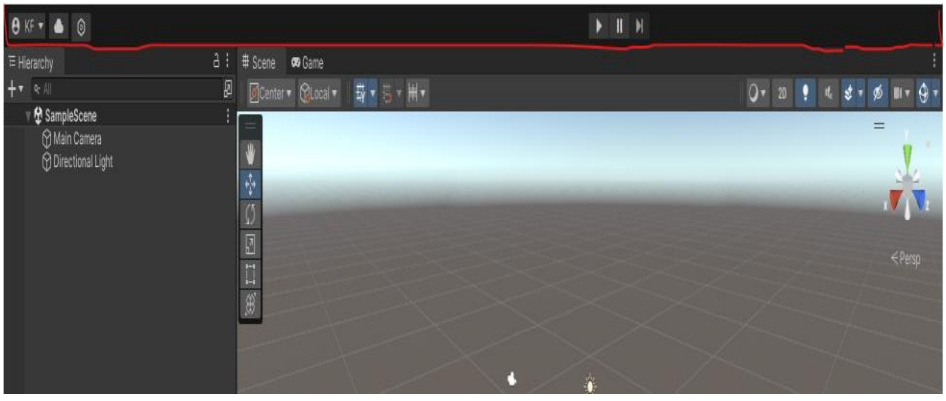
รูปภาพที่ 2.1 Unity Engine

2.1.1 ส่วนประกอบหน้าต่าง Interface ของโปรแกรม Unity แบ่งออก ดังนี้



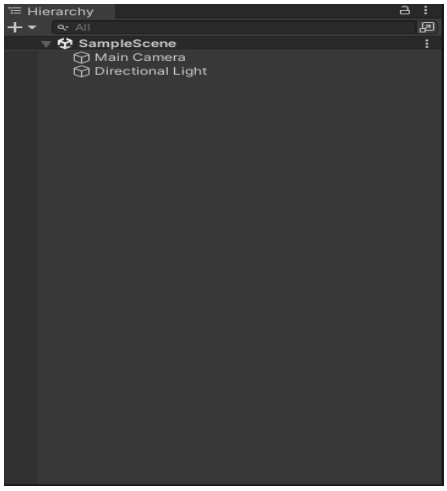
รูปภาพที่ 2.2 interface ของ Unity

2.1.2 จากรูปภาพที่ 2.3 Toolbar เป็นแถบเครื่องมือที่มีไว้เข้าถึงคุณลักษณะการทำงานที่สำคัญที่สุด ด้านซ้าย มีเครื่องมือพื้นฐานสำหรับการจัดการมุมมองภาพและวัตถุภายในภาพ อยู่ตรงกลางคือการควบคุมการเล่นหยุดชั่วคราวและขั้นตอน ปุ่มทางด้านขวาจะทำให้คุณสามารถเข้าถึง Unity Cloud Services และบัญชี Unity ของคุณด้วยเมนูการมองเห็นเลเยอร์และในที่สุดเมนูเค้าโครงของโปรแกรมการแก้ไข



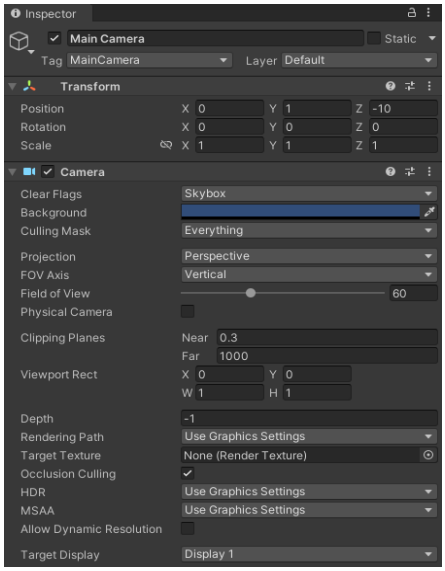
รูปภาพที่ 2.3 Toolbar

2.1.3 จากรูปภาพที่ 2.4 Hierarchy คือส่วนที่บอกลำดับชั้น ของ Object ต่างๆ ที่อยู่ใน Scene นั้นๆ ซึ่งมีทั้ง Object แบบเดี่ยว และ Object ที่เป็นแม่ลูกกันซึ่ง เมื่อมีการจัดการอะไรบางอย่างกับ Object แม่ Object ที่เป็นลูกนั้นก็จะมีการเปลี่ยนแปลงตามไปด้วย การสร้าง Object มีวิธีการคือ ลาก Object ต่างๆที่อยู่ใน Project มาใส่ไว้ในส่วนของ Hierarchy หลังจาก นั้นจะปรากฏวัตถุที่ลากจาก Project มาวางบน Hierarchy ปรากฏขึ้นบน Scene ซึ่ง Object ต่างๆ เหล่านี้สามารถเพิ่ม/แก้ไข/ลบ ได้โดยไม่ กระทบกับ Object ที่อยู่ใน Project



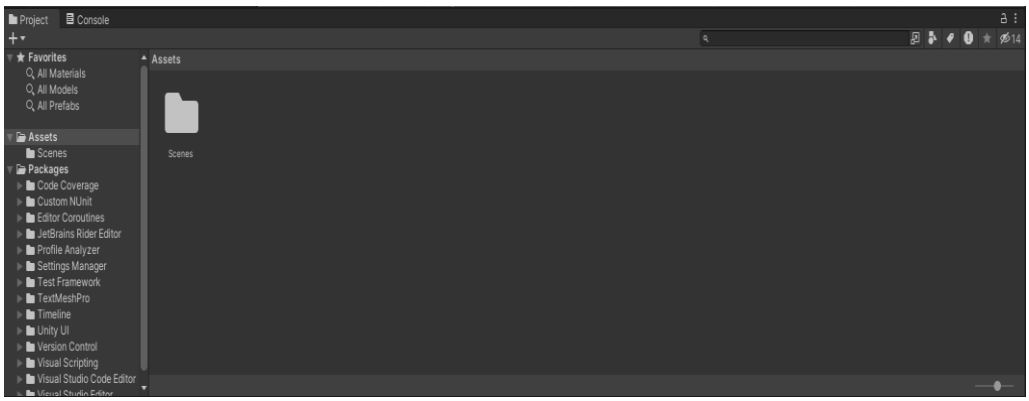
รูปภาพที่ 2.4 Hierarchy

2.1.4 จากรูปภาพที่ 2.5 Inspector เป็นส่วนที่บ่งบอกถึงคุณสมบัติต่างๆ ของ Object ซึ่งสามารถจัดการคุณสมบัติต่างๆ ของ Object ได้ในกรอบของ Inspector



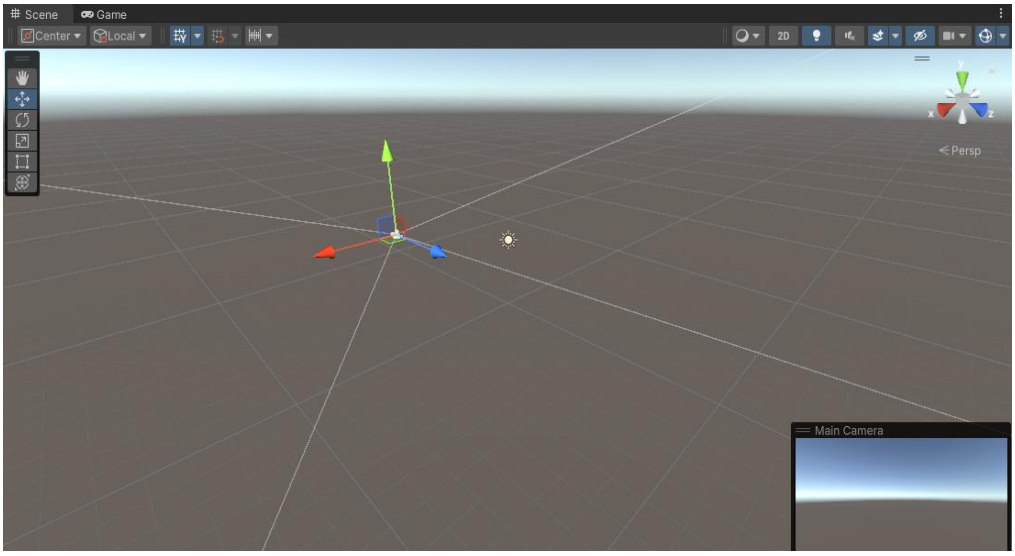
รูปภาพที่ 2.5 Inspector

2.1.5 จากรูปภาพที่ 2.6 Project เป็นส่วนที่ใช้ในการเก็บทรัพยากรต่างๆ ก่อนนำไปสร้างเกม เช่น สคริปท์ต่างๆ ที่ใช่ในการควบคุมตัวเกม 3D โมเดล ใช้เป็นตัวละครหรือวัตถุต่างๆ ในเกม Textures หรือ พื้นผิวต่างๆ ไฟล์เสียง หรือวิดีโอ Prefabs อื่นๆ



รูปภาพที่ 2.6 Project

2.1.6 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

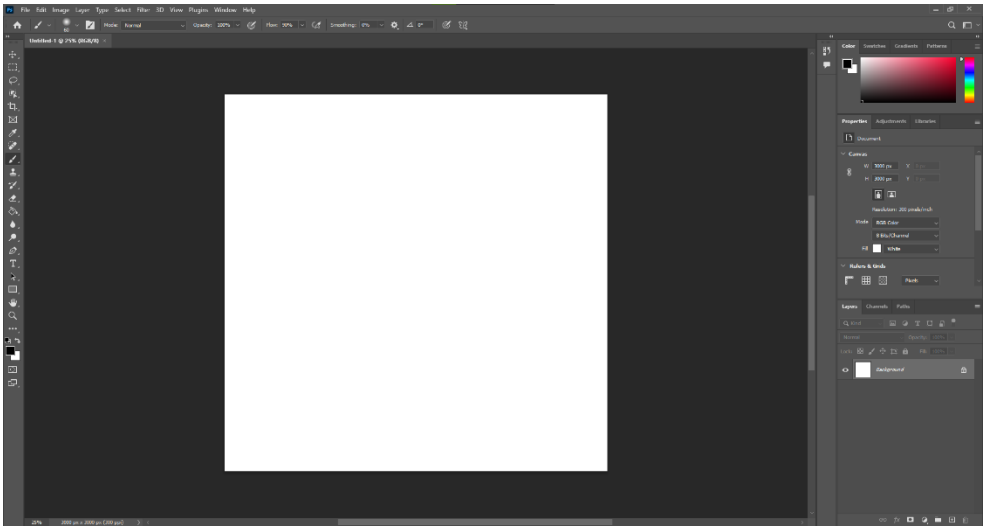
รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Pan	เป็นเครื่องมือสำหรับจัดการมุมมองโปรแกรม Unity3D ถ้ากดปุ่ม middle mouse ค้างไว้จะเป็นการเคลื่อนย้าย Viewport ทั้งหมด ถ้ากดปุ่ม right mouse ค้างไว้จะเป็นการเคลื่อนย้าย Viewport โดย ที่ยังรักษาจุดศูนย์กลางอยู่ที่มุมของกล้อง
	Move	เป็นเครื่องมือสำหรับปรับตำแหน่งของ Object 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
	Rect Tool	เป็นเครื่องมือสำหรับปรับขนาดของ Object กับ Scale แต่จะเปลี่ยนแปลงเฉพาะค่า Width และ Height ค่า Scale จะยังคงเป็น 1 เสมอ

2.2 Adobe Photoshop (ที่มา https://teacherjaray.blogspot.com/2014/05/blog- post)

โปรแกรม Photo shop เป็นโปรแกรมในตระกูล Adobe ที่ใช้สำหรับตกแต่งภาพถ่ายและภาพกราฟฟิก ได้อย่างมีประสิทธิภาพ ไม่ว่าจะเป็นงานด้านสิ่งพิมพ์ นิตยสาร และงานด้านมัลติมีเดีย อีกทั้งยังสามารถ retouching ตกแต่งภาพและการสร้างภาพ ซึ่งกำลังเป็นที่มนิยมสูงมากในขณะนี้ เราสามารถใช้โปรแกรม Photoshop ในการตกแต่งภาพ การใส่ Effect ต่าง ๆให้กับภาพ และตัวหนังสือ การทำภาพขาวดำ การทำภาพถ่ายเป็นภาพเขียน การนำภาพมารวมกัน การ Retouch ตกแต่งภาพต่าง คุณสามารถที่จะทำการแก้ไขภาพ ตกแต่งภาพ ซ้อนภาพในรูปแบบต่างๆ ได้อย่างง่ายดาย และสิ่งที่ขาดไม่ได้ก็คือ การใส่ข้อความประกอบลงในภาพด้วย

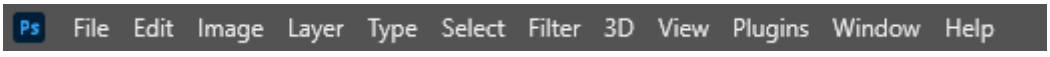
รูปภาพที่ 2.8 Adobe Photoshop

2.2.1 ส่วนประกอบหน้าต่าง Interface ของโปรแกรม Adobe Photoshop แบ่งออก ดังนี้



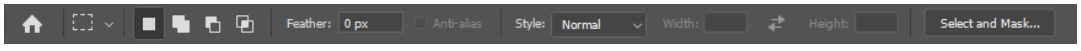
รูปภาพที่ 2.9 Interface ของ Adobe Photoshop

2.2.2 จากรูปภาพที่ 2.10 แถบเมนูคำสั้ง (Menu Bar) เป็นจุดรวบรวมชุดคำสั่งที่ใช้สำหรับเรียกคำสั่งต่างๆ เพื่อใช้จัดการไฟล์หรือตกแต่งภาพ



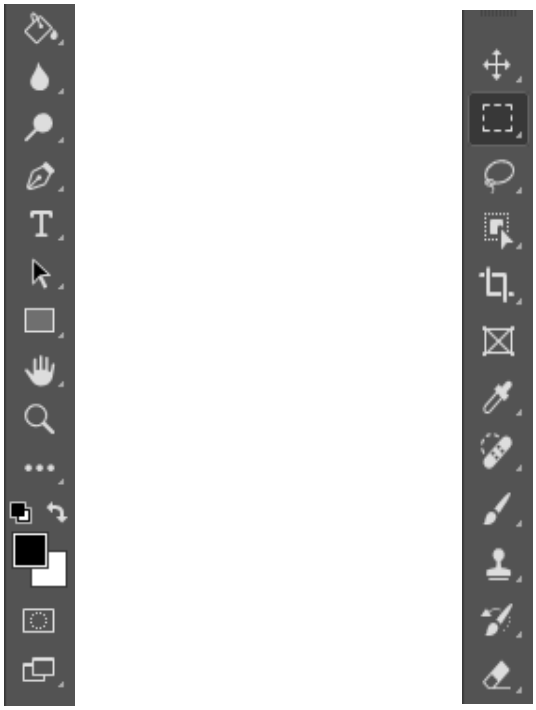
รูปภาพที่ 2.10 แถบเมนูคำสั้ง (Menu Bar)

2.2.3 จากรูปภาพที่ 2.11 แถบตัวเลือก (Options Bar) เป็นส่วนที่ใช้ในการปรับแต่งค่าการทำงานของเครื่องมือ ต่างๆ การกำหนดค่าในแถบตัวเลือกจะเปลี่ยนไปตามเครื่องมือที่ใช้งานอยู่



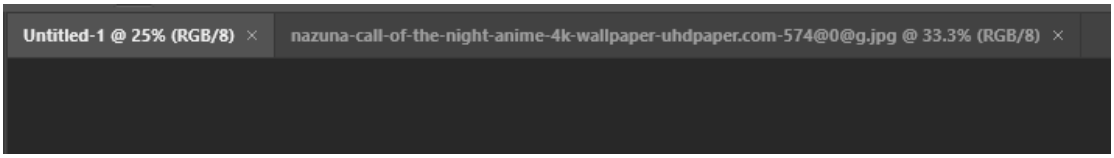
รูปภาพที่ 2.11 แถบตัวเลือก (Options Bar)

2.2.4 จากรูปภาพที่ 2.12 กล่องเครื่องมือ (Toolbox) เป็นส่วนที่ใช้เก็บเครื่องมือพื้นฐานในการทำงานในโปรแกรมสามารถเรียกใช้ชุดเครื่องมือย่อยโดยการคลิกรูปสามเหลี่ยมที่มุมด้านล่าง



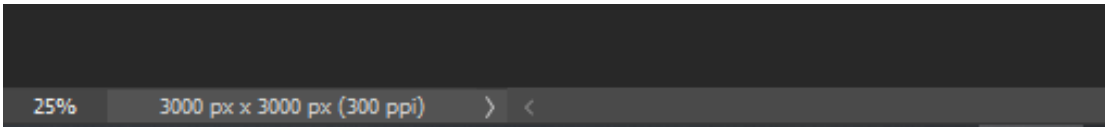
รูปภาพที่ 2.12 กล่องเครื่องมือ (Toolbox)

2.2.5 จากรูปภาพที่ 2.13 แถบชื่อเรื่อง (Title Bar) เป็นส่วนที่แสดงชื่อไฟล์ภาพที่เปิดใช้งานอยู่ สำหรับโปรแกรม Adobe Photoshop แถบชื่อเรื่องจะเรียงกันเป็นแท็ป (Tab)



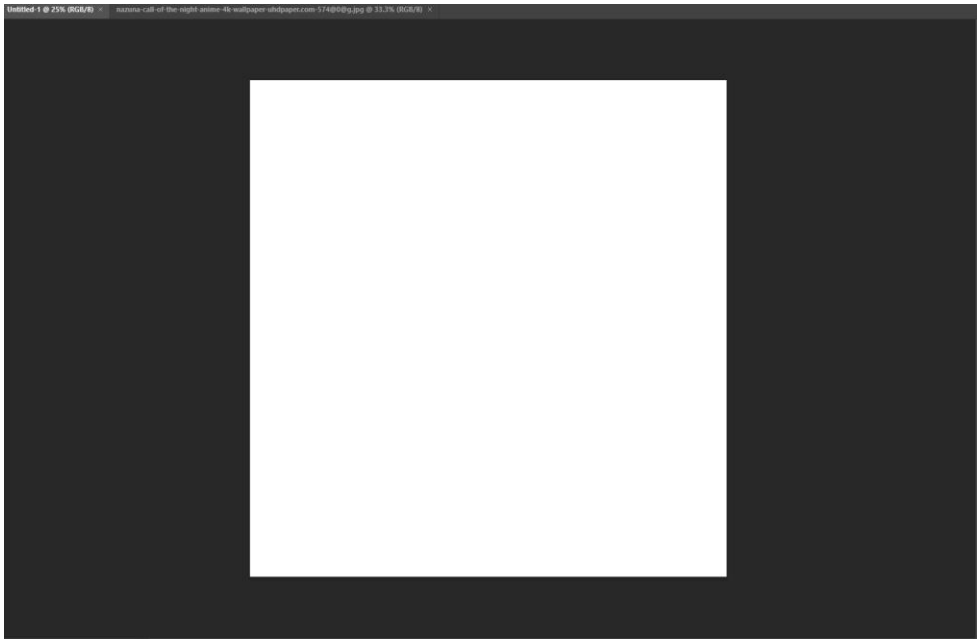
รูปภาพที่ 2.13 แถบชื่อเรื่อง (Title Bar)

2.2.6 จากรูปภาพที่ 2.14 แถบสถานะ (Status Bar) เป็นส่วนที่แสดงคุณสมบัติเกี่ยวกับภาพ เช่น เปอร์เซ็นต์การย่อขยายไฟล์ภาพ ขนาดไฟล์ภาพ เป็นต้น



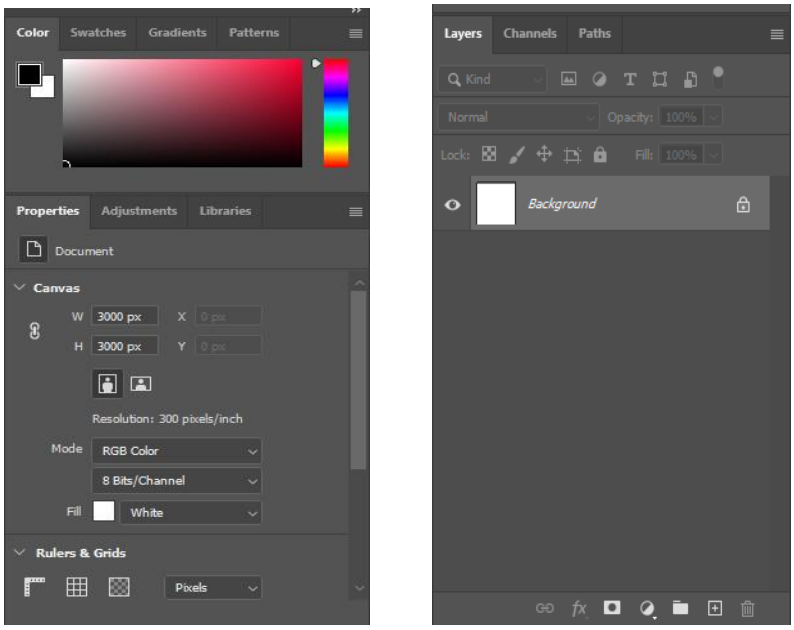
รูปภาพที่ 2.14 แถบสถานะ (Status Bar)

2.2.7 จากรูปภาพที่ 2.15 พื้นที่ใช้งาน (Working Area) เป็นส่วนที่ใช้ในการสร้างงานกราฟิก โดยการเปิดไฟล์ภาพเพื่อแก้ไขบนพื้นที่ใช้งาน หรือวาดภาพใหม่ลงไปบนพื้นที่ใช้งาน



รูปภาพที่ 2.15 พื้นที่ใช้งาน (Working Area)

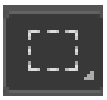
2.2.8 จากรูปภาพที่ 2.16 พาเนล (Panel) ใช้สำหรับจัดการกับภาพ โดยแยกออกเป็นหมวดหมู่ เช่น พาเนลสำหรับเลือกสี พาเนลสำหรับปรับแต่งความสว่าง เป็นต้น



รูปภาพที่ 2.16 พาเนล (Panel)

เครื่องมือของโปรแกรม Adobe Photoshop มีดังนี้

ตารางที่ 2.2 ตารางเครื่องมือของโปรแกรม Adobe Photoshop

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Move	ใช้สำหรับเลือกพื้นที่บนภาพเป็นรูปสี่เหลี่ยมวงกลม วงรี หรือเลือกเป็นแถวคอลัมน์ ขนาด 1 พิกเซล
	Marquee	ใช้สำหรับย้ายพื้นที่ที่เลือกไว้ของภาพหรือย้าย ภาพในเลเยอร์หรือย้ายเส้นไกด์
	Lasso	ใช้เลือกพื้นที่บนภาพเป็นแนวเขตแบบอิสระ
	Magic Wand	ใช้เลือกพื้นที่ด้วยวิธีระบายบนภาพหรือเลือกจากสี ที่ใกล้เคียงกัน
	Crop	ใช้ตัดขอบภาพ
	Slice	ใช้ตัดแบ่งภาพเพื่อบันทึกไฟล์ภาพย่อย ที่เรียกว่า สไลซ์ (Slice) สำหรับนำไปสร้างเว็บเพจ
	Eyedropper	ใช้เลือกสีจากสีต่างๆ บนภาพ

ตารางที่ 2.3 ตารางเครื่องมือของโปรแกรม Adobe Photoshop (ต่อ)

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Healing Brush	ใช้ตกแต่งลบรอยตำหนิในภาพ
	Brush	ใช้ระบายลงบนภาพ
	Clone Stamp	ใช้ทำสำเนาภาพ โดยก๊อปปี้ภาพจากบริเวณอื่น มาระบาย หรือระบายด้วยลวดลาย
	History Brush	ใช้ระบายภาพด้วยภาพของขั้นตอนเดิมที่ผ่านมา หรือภาพของสถานะเดิมที่บันทึกไว้
	Eraser	ใช้ลบภาพบางส่วนที่ไม่ต้องการ
	Gradient	ใช้เติมสีแบบไล่ระดับโทนสีหรือความทึบ
	Blur	ใช้ระบายภาพให้เบลอ
	Bern	ใช้ระบายเพื่อให้ภาพมืดลง

ตารางที่ 2.4 ตารางเครื่องมือของโปรแกรม Adobe Photoshop (ต่อ)

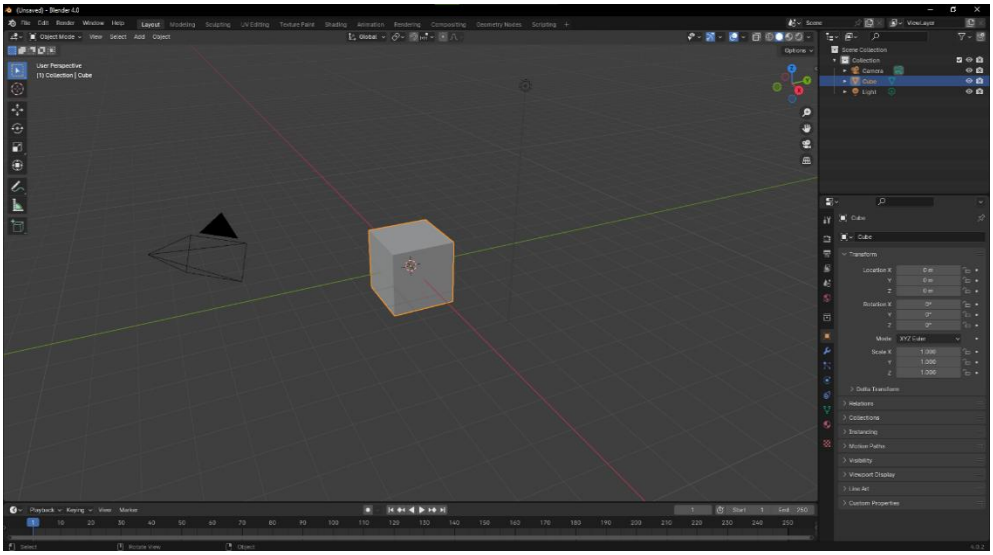
รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Dodge	ใช้ระบายเพื่อให้ภาพสว่างขึ้น
	Pan	ใช้วาดเส้นพาธ (Path)
	Horizontal Type	ใช้พิมพ์ตัวอักษรหรือข้อความลงบนภาพ
	Path Selection	ใช้เลือกและปรับแต่งรูปทรงของเส้นพาธ
	Rectangle	ใช้วาดรูปทรงเรขาคณิตหรือรูปทรงสำเร็จรูป
	Hand	ใช้เลื่อนดูส่วนต่าง ๆ ของภาพ
	Zoom	ใช้ย่อหรือขยายมุมมองภาค
	Set Foreground color Set Background color	ใช้สำหรับกำาหนดสีใช้ย่อหรือขยายมุมมอง Foreground Color was Background Color

2.3 blender 3d (ที่มา http://www.comscicafe.com/article/110/what-is-blender-3d)

โปรแกรม Blender เป็นโปรแกรม Open Source เปิดให้ดาวน์โหลดไปใช้งานฟรี และร่วมเขียนโปรแกรมหรือส่วนเสริมเพิ่มเติมได้ มีความสามารถมากพอๆกับโปรแกรมสร้างงาน 3 มิติ ตัวอื่นๆ หรือเหนือกว่าโปรแกรมที่เสียค่าลิขสิทธิ์บางตัวอีกด้วย นอกจากนี้โปรแกรม Blender นั้นยังทำงานได้หลายแพลตฟอร์มระบบปฏิบัติการ Windows 8 7 Vista, Mac OSX, GNU/Linux และ FreeBSD นอกจากคุณสมบัติที่กล่าวมาโปรแกรม Blender ยังมีจุดเด่นอื่นๆอีกดังนี้ เป็นโปรแกรมที่กินทรัพยากรระบบน้อย พื้นที่ในการติดตั้งก็น้อยโดยตัวติดตั้งมีขนาด ไม่ถึง 100 เมกกะไบต์ ติดตั้งง่ายใช้เวลาน้อย สร้างงานได้หลากหลายรูปแบบ และมีเว็บไซต์ที่ให้ความรู้อยู่มากมาย

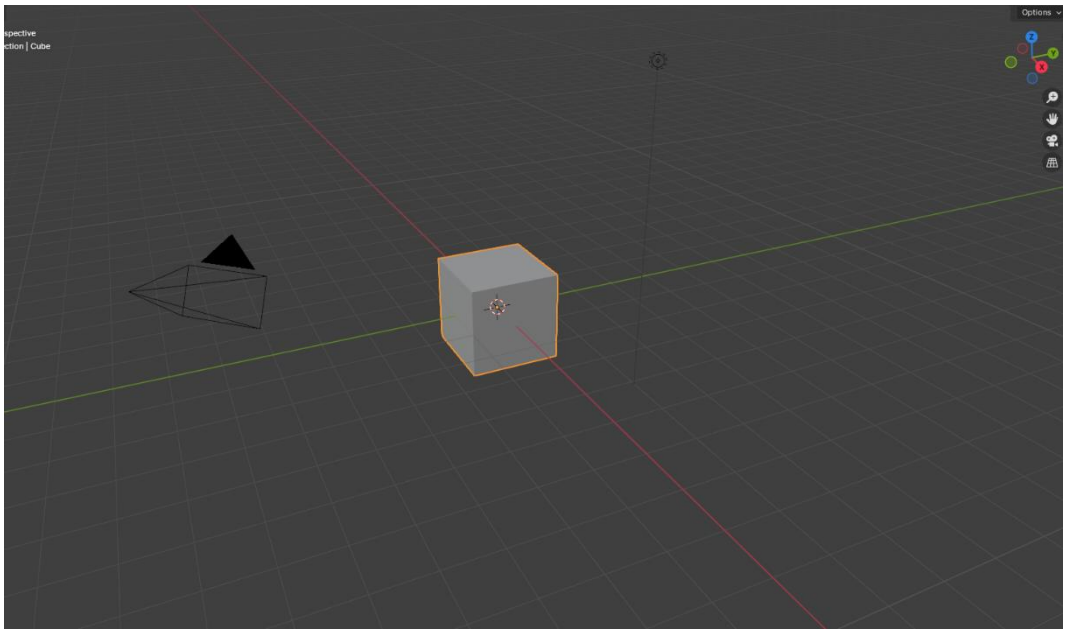
รูปภาพที่ 2.17 blender 3d

2.3.1 ส่วนประกอบหน้าต่าง Interface ของโปรแกรม Blender 3d แบ่งออก ดังนี้



รูปภาพที่ 2.18 Interface ของ Blender 3d

2.3.2 จากรูปภาพที่ 2.19 View port เป็นหน้าต่างการทำงาน 3มิติ เป็นพื้นที่การทำงานในการร้าง

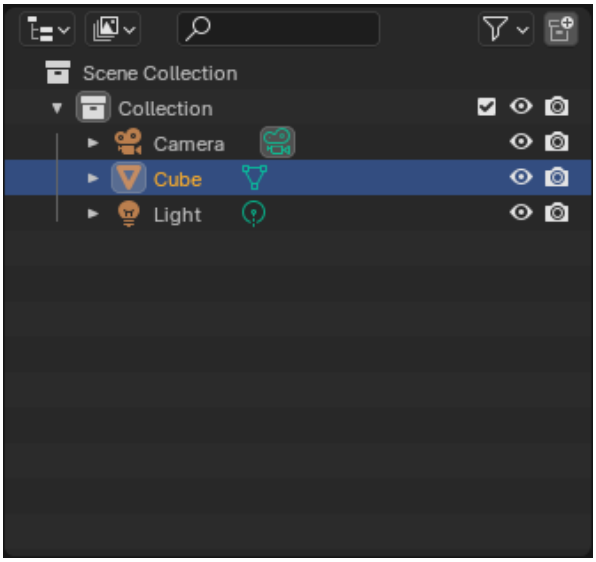


รูปภาพที่ 2.19 View port

2.3.3 จากรูปภาพที่ 2.20 Tool Box เป็นแถบเครื่องมือต่างๆที่จะใช้ในการสร้างโมเดล

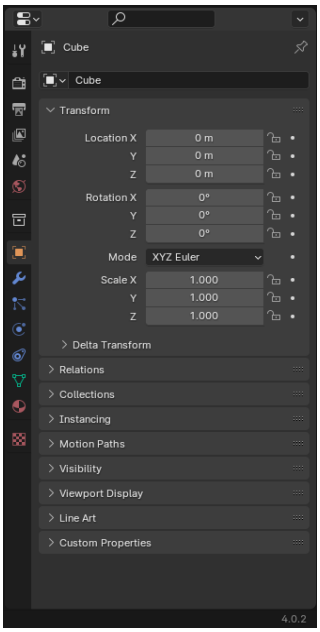
รูปภาพที่ 2.20 Tool Box

2.3.4 จากรูปภาพที่ 2.21 Out liner ส่วนช่วยในการจัดการวัตถุ (object) ต่างๆ ที่อยู่ในงานของเรา



รูปภาพที่ 2.21 Out liner

2.3.5 จากรูปภาพที่ 2.22 Properties เป็นแถบคุณสมบัติต่างๆ ของวัตถุ และคุณสมบัติของงานที่เราทำขึ้น ซึ่งจะเป็นส่วนที่ใช้ในการจัดการเรื่องต่างๆ ไม่ว่าจะเป็นเรื่องของการ ประมวลผลภาพงานของเรา การสร้างฉากพื้นหลัง คุณสมบัติของวัตถุที่เลือกไว้ การปรับแต่ง สี พื้นผิว การสร้างฟิสิกส์เสมือนจริง ล้วนแล้วอยู่ในแถบนี้ทั้งหมด

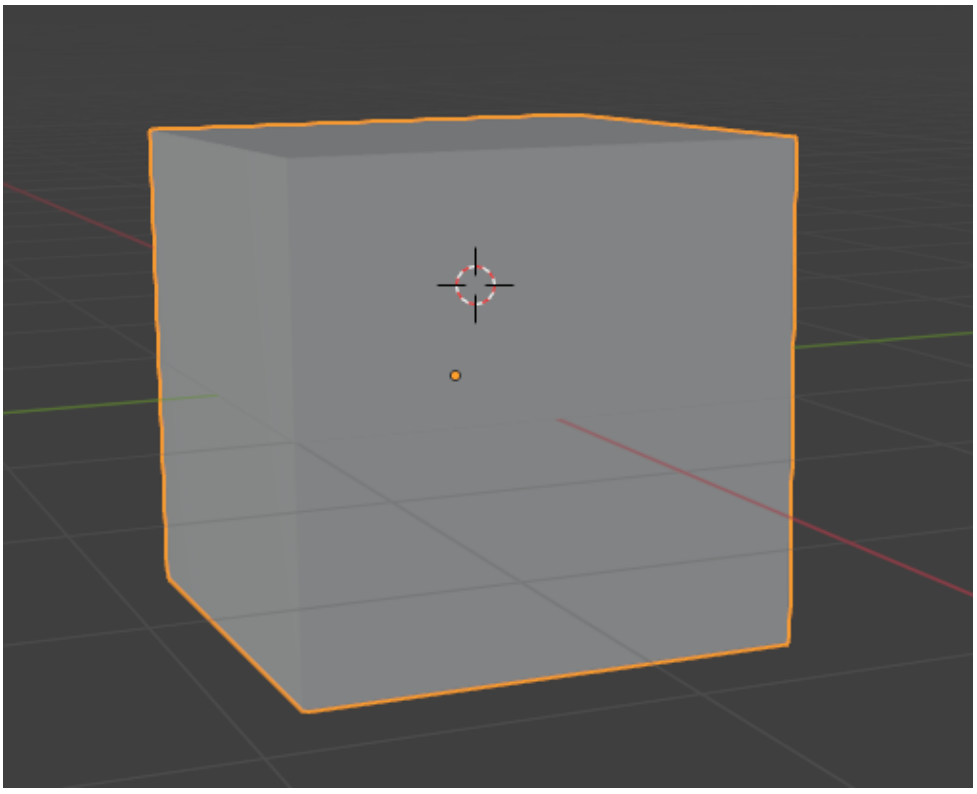


รูปภาพที่ 2.22 Properties

2.3.6 จากรูปภาพที่ 2.23 Time line เป็นส่วนในการสร้าง animation

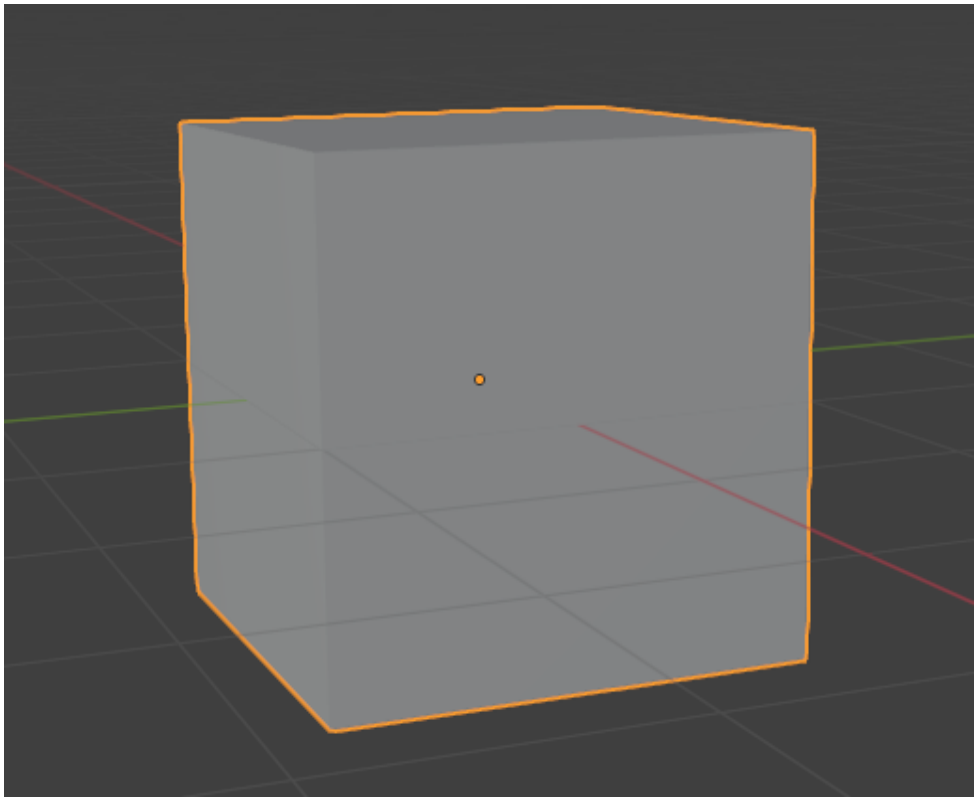
รูปภาพที่ 2.23 Time line

2.3.7 จากรูปภาพที่ 2.24 Cursor เป็นจุดอ้างอิงตำแหน่งในการสร้างชิ้นงานใหม่



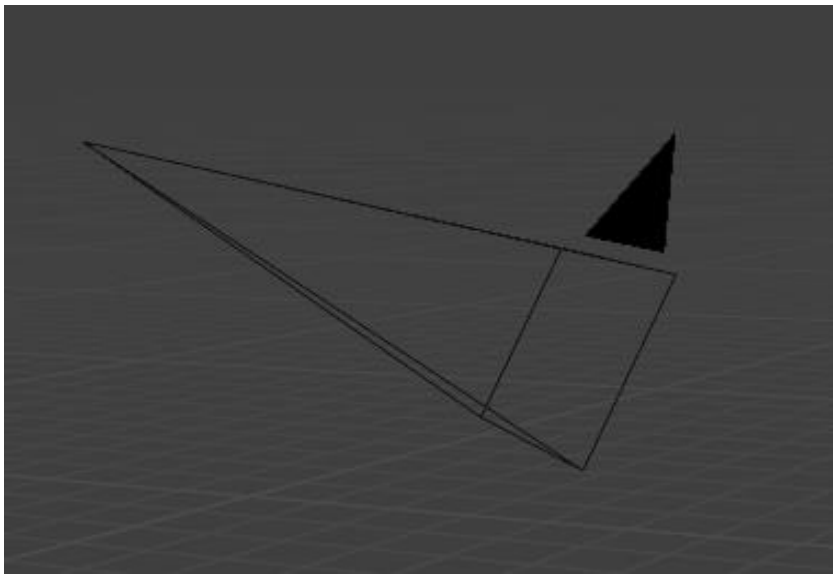
รูปภาพที่ 2.24 Cursor

2.3.8 จากรูปภาพที่ 2.25 Cube เป็นวัตถุที่โปรแกรมสร้างให้เองเมื่อเริ่มต้นใช้งานโปรแกรม



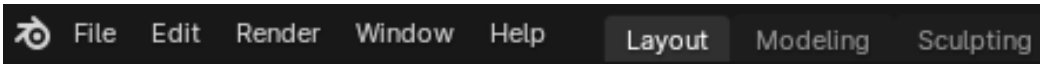
รูปภาพที่ 2.25 Cube

2.3.9 จากรูปภาพที่ 2.26 Camera กล้องที่ใช้ในการถายภาพวัตถุ



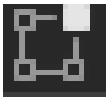
รูปภาพที่ 2.26 Camera

2.3.10 จากรูปภาพที่ 2.27 แถบ information เป็นส่วนเมนูคำสั่งต่างๆ เช่น Open ,Save , Import ,Export เมนูปรับหน้าต่างการทำงานและเมนูในการเลือก Engine ที่ใช้ในการ Render

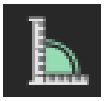
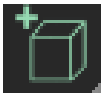


รูปภาพที่ 2.27 แถบ information

ตารางที่ 2.5 ตารางโหมดการทำงานของโปรแกรม Blender 3d

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Object Mode	เป็นโหมดที่ใช้ในการจัดการวัตถุในงานของเราเช่น การเพิ่มวัตถุ ลบวัตถุ เพิ่มแสงเพิ่มฉาดจัดตำแหน่งวัตถุต่างๆ ซึ่งจะเป็นโหมดเริ่มต้นเมื่อเปิดโปรแกรมขึ้นมา
	Edit Mode	เป็นโหมดที่ใช้ไรการแก้ไขวัตถุที่เลือกไว้
	Sculpt Mode	เป็นโหมดการขึ้นโมเดลดวยวิธีการปั้นเหมือนการปั้นรูปสามมิติ
	Vertex Paint	การลงสีตามจุด
	Weight paint	เป็นการลงสีเพื่อกำหนดน้ำหนักของชิ้นส่วนใช้ในการจับคู่กันของ ชิ้นส่วนต่างๆของโมเดลกับกระดูกแอนิเมชัน
	Texture Paint	เป็นโหมดใช้ในการ ลงสีพื้นผิว Texture

ตารางที่ 2.6 ตารางเครื่องมือของโปรแกรม Blender 3d

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	select box	เป็นคำสั่งเลือกวัตถุ
	Cursor	เป็นคำสั่งกำหนดจุดสร้างวัตถุใหม่
	Move	เป็นคำสั่งขยับวัตถุ ย้ายวัตถุ
	Rotate	เป็นคำสั่งหมุนวัตถุ หมุนชิ้นงาน
	Scale	เป็นคำสั่งเพิ่มย่อ/ขยายวัตถุหรือชิ้นงาน
	Transform	เป็นคำสั่งที่ใช้แบ่งส่วนของวัตถุ
	Annotate	เป็นคำสั่งเขียนคำอธิบายกับข้อมูลที่ใช้งาน
	Measure	เป็นคำสั่งวัดระยะทางและมุม
	Add cube	เป็นคำสั่งเพิ่มลูกบาศก์
	Extrude Region	เป็นคำสั่งสร้าง item เหมือน item เดิม

ตารางที่ 2.7 ตารางเครื่องมือของโปรแกรม Blender 3d (ต่อ)

รูปภาพ	ชื่อเครื่องมือ	คำอธิบาย
	Inset faces	เป็นคำสั่งใช้สร้างเส้นขอบข้างในหน้า face
	Bevel	เป็นคำสั่งใช้สำหรับการลบเหลี่ยมหรือมุมของวัตถุให้มีความโค้งมนมากขึ้นนั้นเอง
	Loop cut	เป็นคำสั่งใช้แบ่งส่วนของวัตถุ
	Knife	เป็นคำสั่งตัดเส้นเพิ่มเติมในวัตถุ
	Smooth	เป็นคำสั่งปรับมุมของยอดที่เลือกให้เรียบ
	Shrink/fatten	เป็นคำสั่งย่อ/ทำให้จุดยอดที่เลือกหดตัว
	Shear	เป็นคำสั่งเฉือนรายการที่เลือกตามแกนที่กำหนด
	Spin	เป็นคำสั่งขับไล่จุดยอดที่เลือกเป็นวงหลมรอบๆ เคอร์เซอร์ที่ระบุ
	Fill	เป็นคำสั่งเติมสี
	Blur	เป็นคำสั่งทำให้วัตถุที่เลือกไม่ชัด

2.4 ภาษา C# (ที่มา https://www.binaryprogramming.net/1-1)

ภาษา C# เป็นภาษาโปรแกรมเชิงวัตถุทำงานบน Netframework พัฒนาโดยบริษัทไมโครซอฟท์ และมี Anders Hejlsberg เป็นหัวหน้าโครงการ โดยมีรากฐานมาจากภาษา C++ และภาษาอื่นๆ (โดยเฉพาะภาษา Delphi และ Java) โดยปัจจุบันภาษา C# เป็นภาษามาตรฐานรองรับโดย ECMA และ ISO ซึ่งในปัจจุบันได้พัฒนาและปรับรูปแบบของ ภาษา C# อยู่ตลอดเวลาโดยทาง Microsoft ได้นำภาษา C# ไปอยู่ในชุดพัฒนา software อย่าง visual studio ซึ่งทำให้เป็นที่นิยมเพิ่มมากขึ้น

รูปภาพที่ 2.28 C# Programming Language

Assets > Script > Enemy > EnemyFollow.cs

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.AI;
5
6   public class EnemyFollow : MonoBehaviour
7   {
8
9       public UnityEngine.AI.NavMeshAgent enemy;
10      public Transform player;
11
12      void Update()
13      {
14          enemy.SetDestination(player.position);
15      }
16  }
17
```

รูปภาพที่ 2.29 ภาษา c#

2.4.1 ชนิดข้อมูล (Data Types) ใน C#

C# กำหนดชนิดของข้อมูลไว้หลากหลายชนิดเพื่อรองรับการจัดเก็บข้อมูลหลายๆ ประเภท ดั้งนี้

ตาราง 2.8 ตารางชนิดข้อมูลใน ภาษา C#

ชนิดข้อมูล	คำอธิบาย
Char	อักขระเดี่ยว เช่น a
Bool	ค่าความจริง เป็นไปได้สองค่าคือ true หรือ false
Byte	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 ถึง 255
int	จำนวนเต็มมีเครื่องหมาย ตั้งแต่ 2,147,483,648 ถึง 2,147,483,647
uint	จำนวนเต็มไม่มีเครื่องหมายตั้งแต่ 0 ถึง 4,294,967,295
long	จำนวนเต็มมีเครื่องหมายตั้งแต่ 9,223,372,036,854,775,808 ถึง 9,223 , 372,036,854,775,807
ulong	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 ถึง 18,446,744,073,709,551,615
float	จำนวนจริง (มีทศนิยมได้) เช่น 3.14159
double	จำนวนจริงที่เก็บความละเอียดมากเป็นสองเท่า
string	ข้อความ (สายอักขระ) เช่น Hello

2.4.2 ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการสำหรับคำนวณทางคณิตศาสตร์เป็นตัวดำเนินการที่ใช้มากที่สุดในการเขียนโปรแกรมคอมพิวเตอร์ ตัวดำเนินการทางคณิตศาสตร์มีทั้งหมด 7 ตัว ดังตารางต่อไปนี้

ตาราง 2.9 ตารางชนิดข้อมูลใน ภาษา C#

สัญลักษณ์	**ตัวดำเนินการ**	**ตัวอย่างและความหมาย**
+	บวก (add)	x = a + b; นำค่า a บวกกับ b แล้วเก็บไว้ใน X
-	ลบ (subtract)	x = a - b; นำค่า a ลบด้วย 5 แล้วเก็บไว้ใน x
*	คูณ (multiply)	x = a * b; นำค่า a คูณกับค่า b แล้วเก็บไว้ใน X
/	หาร (divide)	x = a / b นำค่า a หารด้วย 6 ผลลัพธ์เก็บไว้ใน X
%	หารเอาเศษ (modulo)	x = a % b นำเศษจากการหาร a ด้วย 5 แล้วเก็บไว้ใน X
++	เพิ่มค่า	x++; เพิ่มค่า x ขึ้นหนึ่งค่า
--	ลดค่า	X-; ลดค่า x ลงหนึ่งค่า x

นิพจน์ทางคณิตศาสตร์นั้นสามารถเกิดจากตัวดำเนินการหลายๆ ตัวมากระทำกับตัวแปรหรือค่าคงที่ได้ และสามารถใส่ตัวดำเนินการไปพร้อมกับการประกาศตัวแปรได้ แต่ประเภทของข้อมูลที่ประกาศตัวแปรนั้นจะต้องสอดคล้องกับผลลัพธ์ที่ได้จากการประมวลผลของนิพจน์นั้น

2.4.3 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการเปรียบเทียบใช้สำหรับนำค่าของ ตัวแปรหรือค่าคงที่จำนวนสองตัวมา เปรียบเทียบกัน ผลลัพธ์ที่ได้จะเป็นค่าทางตรรกะ (bool) คือมีค่าเป็นจริง (true) หรือเท็จ (false)ตัวดำเนินการประเภทนี้มี 6 ตัว ดังตารางต่อไปนี้

ตาราง 2.10 ตัวดำเนินการเปรียบเทียบ

สัญลักษณ์	ตัวดำเนินการ	ตัวอย่างและความหมาย
==	เท่ากับ	a == b; เป็นจริง ถ้าค่าในตัวแปร a เท่ากับตัวแปร b
>	มากกว่า	a > b; เป็นจริง ถ้าค่าในตัวแปร a มากกว่าตัวแปร b
>=	มากกว่าหรือเท่ากับ	a >= b; เป็นจริง ถ้าค่าในตัวแปร a มากกว่าหรือ เท่ากับตัวแปร b
<	น้อยกว่า	a < b; เป็นจริง ถ้าค่าในตัวแปร a น้อยกว่าตัว แปร b
<=	น้อยกว่าหรือเท่ากับ	a < = b; เป็นจริง ถ้าค่าในตัวแปร a น้อยกว่าหรือเท่ากับตัวแปร b
!=	ไม่เท่ากับ	a != b; เป็นจริง ถ้าค่าในตัวแปร a ไม่เท่ากับค่าใน ตัวแปร b

ในภาษา C# ตัวดำเนินการเปรียบเทียบสามารถใช้กับข้อมูลประเภทตัวเลขได้ทุกประเภท รวมไปถึงข้อมูลแบบ float, double, decimal และ char โดยการเปรียบเทียบข้อมูลแบบ char นั้นจะเป็นการนำค่ารหัส Unicode มาเปรียบเทียบกัน

2.4.4 ตัวดำเนินการตรรกศาสตร์

ตัวดำเนินการทางตรรกศาสตร์ (Logical Operators) เป็นการนำตัวแปรหรือค่าคงที่ สองค่ามาประมวลผลทางตรรกศาสตร์ต่อกันผลลัพธ์ที่ได้จะเป็นจริง (true) หรือเท็จ (false) เท่านั้นตัวดำเนินการประเภทนี้บางตัวยังสามารถประมวลผลในระดับบิตได้อีกด้วยสัญลักษณ์ของตัว ดำเนินการแสดงได้ ดังตารางต่อไปนี้

ตาราง 2.11 ตัวดำเนินการตรรกศาสตร์

สัญลักษณ์	ตัวดำเนินการ	ตัวอย่างและความหมาย
!	NOT	!a กลับค่าลอจิกของ a ถ้าหาก a เป็นจริง ผลลัพธ์จะเป็นเท็จ
&	AND	ab ดำเนินการ OR ระหว่าง a กับ b
|	OR	a ^ b ดำเนินการ exclusive ระหว่าง a กับ b
^	XOR	a && b ดำเนินการ AND แบบ short-circuit AND
&&	Short-circuit AND	a && b ดำเนินการ AND แบบ short-circuit AND
||	Short-circuit OR	a || b ดำเนินการ OR แบบ short-circuit OR

2.4.5 โครงสร้างของภาษา C# ขั้นพื้นฐานมีรายละเอียด ดังนี้

- หมายเลข (1) เป็นการระบุชื่อของ namespaceใช้ในการกำหนดขอบเขตให้กับคลาสต่างๆ รวมถึงใช้ในการจัดโครงสร้างของโปรแกรมขนาดใหญ่ให้เป็นสัดส่วนอีกด้วย โดยเฉพาะอย่างยิ่งในการเขียนโปรแกรมคอมพิวเตอร์ที่ซับซ้อนโดยมีผู้เขียนโปรแกรมหลายคน นอกจากนี้การกำหนด namespace ยังช่วยป้องกันปัญหาการตั้งชื่อคลาสหรือค่าคงที่อื่นๆ ซ้ำกันได้

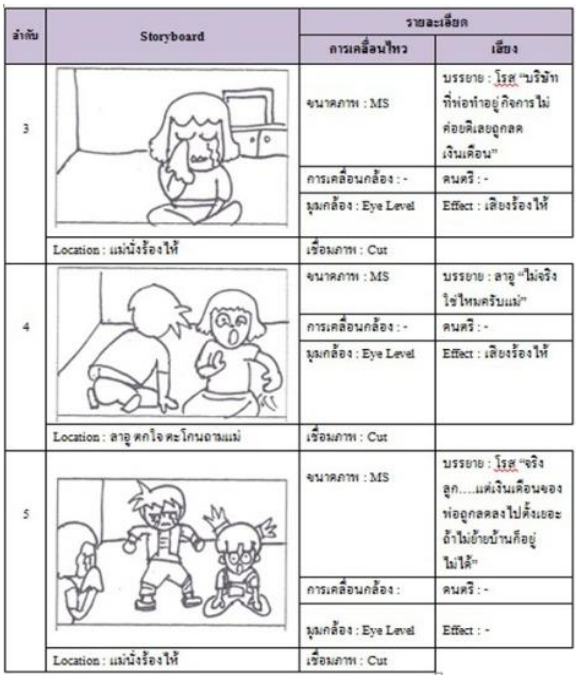
- หมายเลข (2) เป็นการระบุชื่อของ class ใช้ในการกำหนดส่วนประกอบต่างๆ ที่จะนำไปสร้าง Object

- หมายเลข (3) เป็นการะบุพื้นที่สาหรับคำสั่งต่างๆ ที่ผู้เขียนโปรแกรมต้องการให้คอมพิวเตอร์ปฏิบัติตาม

2.5 Storyboard (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 : -	ดนตรี : -
		มุมกล้อง : Eye Level	Effect : เสียงร้องไห้
	Location : แม่นั่งร้องไห้	เชื่อมภาพ : Cut	
4		ขนาดภาพ : MS	บรรยาย : ลาอู "ไม่จริงใช่ไหมครับแม่"
		การเคลื่อนกล้อง : -	ดนตรี : -
		มุมกล้อง : Eye Level	Effect : เสียงร้องไห้
	Location : ลาอู หดใจ คะโกนถามแม่	เชื่อมภาพ : Cut	
5		ขนาดภาพ : MS	บรรยาย : โรส "จริงลูก.....แต่เงินเดือนของพ่อถูกลดลงไปตั้งเยอะถ้าไม่ย้ายบ้านก็อยู่ไม่ได้"
		การเคลื่อนกล้อง :	ดนตรี : -
		มุมกล้อง : Eye Level	Effect : -
	Location : แม่นั่งร้องไห้	เชื่อมภาพ : Cut	

รูปภาพที่ 2.30 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 เทคนิคนี้จะมีความยืดหยุ่นในการเล่าเรื่องเพราะมีความอิสระในการตีความใหม่ทั้งหมด 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 จะมีการคิดเรื่องแสง เงา 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

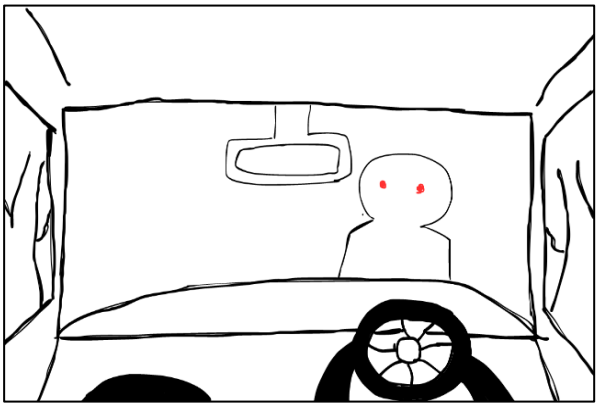
- DISSOLVE คือการเปลี่ยนภาพ โดยภาพแรกค่อยๆจางไปสู่อีกภาพ โดยที่การเปลี่ยนแปลงนั้นจะไม่ได้เกิดขึ้นแบบทันทีทันใดเหมือนการ Cut เทคนิคจะช่วยสร้างความรู้สึกที่ราบรื่นและนุ่มนวล โดยจะใช้เพื่อเชื่อมโยงเรื่องราว

- WIPE การปาด คือหน้าจอจะมีลักษณะของการปาดภาพที่ต้องการจะเปลี่ยน ทับลงบนภาพที่ปรากฏอยู่ การใช้ Wipe อาจสร้างความรู้สึกไม่สมจริงของเรื่องราว ซึ่งส่วนใหญ่จะถูกใช้เพื่อแบ่งเรื่องราวทั้งหมดออกเป็นส่วนๆ อย่างชัดเจน

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

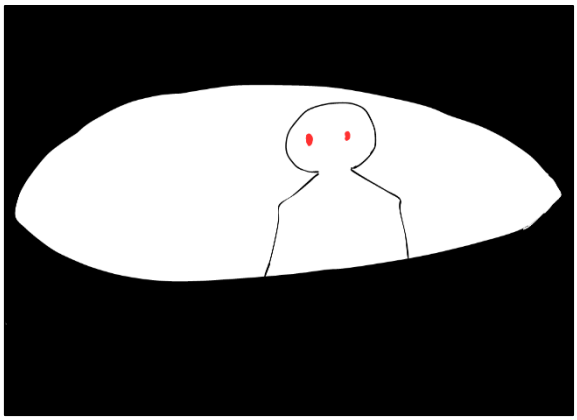
2.4.2 Storyboard ของ Project

รูปภาพที่ 2.31 มีเด็กหนุ่ม2 คนไปเที่ยวในป่าช่วงวันหยุดยาว เพื่อจะไปตั้งแคมป์

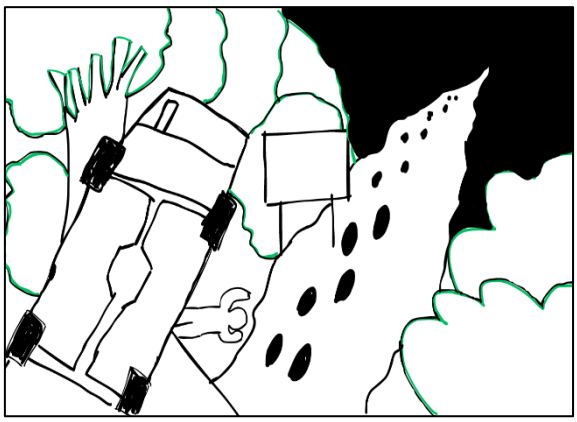


รูปภาพที่ 2.32 แต่มีคนตัดหน้ารถของทั้ง 2 คน ตัวเอกจึงตัดสินใจหักหลบคนที่ตัดหน้ารถทำให้รถเสียการทรงตัว

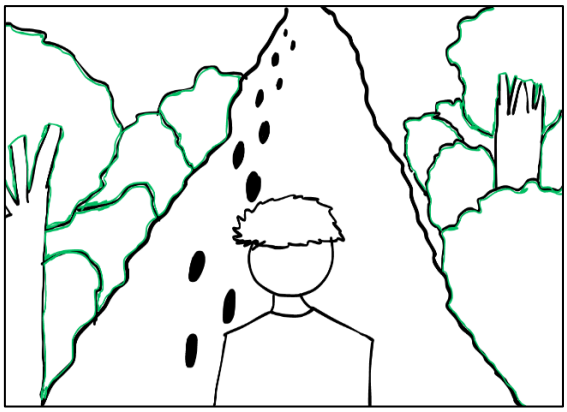
รูปภาพที่ 2.33 รถพุ่งไปชนกลับต้นไม้ข้างทาง ทำให้รถพลิกคว่ำ



รูปภาพที่ 2.34 ตัวเอกจึงเห็นคน 1 คนที่กำลังอุ้มเพื่อนของตัวเองอยู่



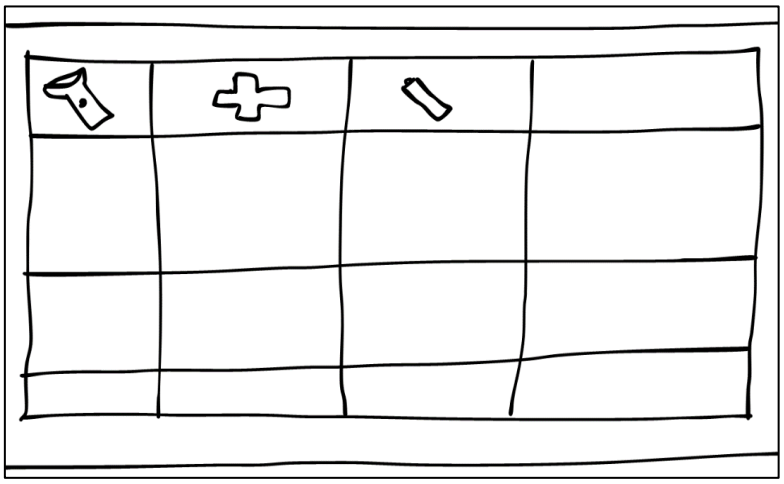
รูปภาพที่ 2.35 ตัวเอกตื่นขึ้นแล้วพบกับรอยเท้าของคนเดินไปตามทางลึกเข้าไปในป่า



รูปภาพที่ 2.36 ตัวเอกจึงเดินตามรอยเท้าเข้าไปในป่าเพื่อตามหาเพื่อนของตัวเองและหาทางออกจากป่า

รูปภาพที่ 2.37 หน้า Main Menu

รูปภาพที่ 2.38 หน้า GAME OVER



รูปภาพที่ 2.39 ช่องเก็บของ

2.6 เอกสารและงานวิจัยที่เกี่ยวข้อง

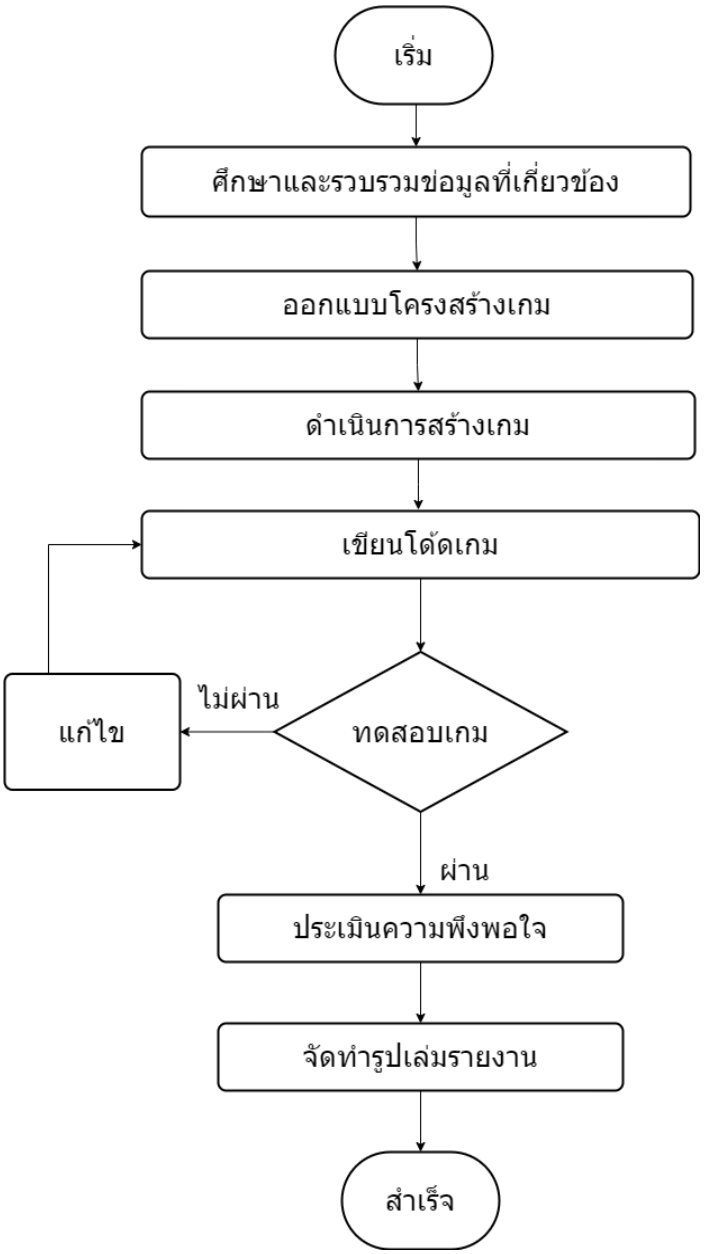
รัชพล วรรณวิจิตร โครงงานนี้ทำขึ้นมาโดยมีจุดมุ่งหมายเพื่อโปรโมทสถาบันเทคโนโลยีไทย-ญี่ปุ่น และเพื่อ เป็นแนวทางสำหรับผู้ที่มีความสนใจ ต้องการศึกษาเกี่ยวกับการสร้างเกม 3D โดยแบ่งเป็นขั้นตอนการเลือกอาคารที่จะนำมาใช้ การจัดวางองค์ประกอบให้ดูน่ากลัว การทำ NavMesh ให้ NPC เดิน การจัดการกับแสงเงา และความรู้เบื่องต้นเกี่ยวกับการ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

นายนัฐวุฒิ หวลละห้อย นายภูวนาท แสงฤทธิ์ และนายรัฐพล สุขสว่างการจัดทำโครงการครั้งนี้มีวัตถุประสงค์เพื่อสร้างเกมสามมิติผีในฝัน ด้วยโปรแกรม Unity (Ghost in Dream 3D Game) ศึกษาความพึงพอใจของนักเรียนประกาศนียบัตรวิชาชีพ สาขาวิชาคอมพิวเตอร์ธุรกิจวิทยาลัยเทคนิคระยองที่มีต่อเกมสามมิติ ในฝันด้วยโปรแกรม Unity (Ghost in Dream 3D Game) และเผยแพร่เกมสามมิติผีในฝัน ด้วยโปรแกรม Unity (Ghost in Dream 3D Game) ผ่านโครงการประกวดโครงการวิชาชีพ ขมรมวิชาชีพคอมพิวเตอร์ธุรกิจ กลุ่มตัวอย่างที่ใช้คือนักเรียนประกาศนียบัตรวิชาชีพ สาขาคอมพิวเตอร์ธุรกิจ วิทยาลัยเทคนิคระยอง 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

อภินันท์ ภิรมย์ ภาคย์ สธนเสาวภาคย์ และภัทราวัลย์ คำปลิว งานวิจัยนี้มีวัตถุประสงค์เพื่อ 1) พัฒนาเกมสามมิติเรื่อง เกมทางออกอยู่ไหน 2) ศึกษาความพึงพอใจของนักศึกษา ที่ มีต่อเกมสามมิติ เรื่อง เกมทางออกอยู่ไหน กลุ่มเป้าหมาย นักศึกษาสาขาวิศวกรรมคอมพิวเตอร์และสาขาเครือข่ายคอมพิวเตอร์คณะวิศวกรรมศาสตร์ มหาวิทยาลัยราชภัฏมหาสารคาม จ านวน 80 คน เครื่องมือที่ใช้ในการวิจัย ได้แก่ แบบสอบถามความพึงพอใจของนักศึกษา สถิติที่ใช้ในการวิจัย คือ ค่าเฉลี่ย และส่วนเบี่ยงเบนมาตรฐานผลการวิจัยสรุปได้ดังนี้ 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

บทที่ 3
วิธีการดำเนินโครงงาน

ในการจัดทำโครงการ Eerie Woods คณะผู้จัดทำได้ดำเนินการตามขั้นตอนดังนี้



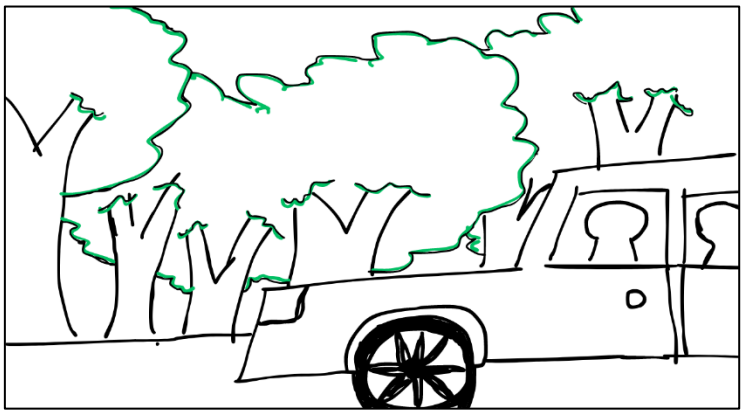
รูปภาพที่ 3.1 แสดงแผนผังการดำเนินการ

3.1 ศึกษาและรวบรวมข้อมูลที่เกี่ยวข้อง

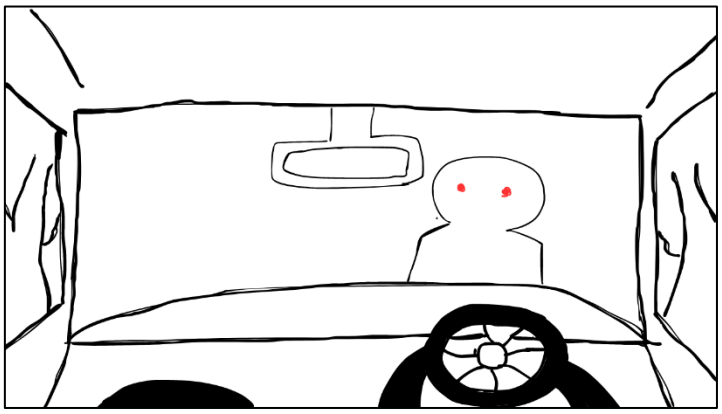
ผู้จัดทำได้ศึกษาการใช้งาน Unity จากอินเทอร์เน็ต การใช้งานโปรแกรม เครื่องมือต่างๆ ของโปรแกรม Unity ศึกษาหน้า interface ของ Unity ศึกษาการสร้าง Object การสร้าง Project เพื่อเก็บทรัพยากรต่างๆ ก่อนนำไปสร้างเกม เช่น สคริปท์ต่างๆ ศึกษาการสร้าง Scene เป็นส่วนที่บ่งบอกว่าในฉากที่กำลังทำงาน มี Object อะไรบ้าง สามารถจัดการ Object ต่างๆ เช่น กล้อง แสง เอฟเฟค หรือโมเดล 3 มิติ ได้จากส่วนนี้ ศึกษาการใช้งานโปรแกรม Adobe Photoshop เครื่องมือต่างๆ ของโปรแกรม Adobe Photoshop ศึกษาการใช้งาน Panel การใช้งาน Working Area แถบสถานะ ศึกษาการใช้งานโปรแกรม blender 3d หน้า interface ของโปรแกรม การใช้ View port การใช้เครื่องมือในโปรแกรม blender 3d การใช้งาน Properties ในการปรับแต่งตัวชิ้นงาน ศึกษาการจัดแสงฉากในเกม ศึกษาการเขียน ภาษา C# ศึกษาของชนิดข้อมูลในภาษา C# ตัวดำเนินการเปรียบเทียบ ตัวดำเนินการตรรกศาสตร์ ศึกษาโครงสร้างของภาษา C# ใช้ในการเขียนไฟล์ script ในโปรแกรม Unity ศึกษาการเขียน Storyboard เทคนิคการทำ Storyboard ในการสร้างฉาก cutscene ในตัวเกม

3.2 เขียน Storyboard

3.2.1 storyboard



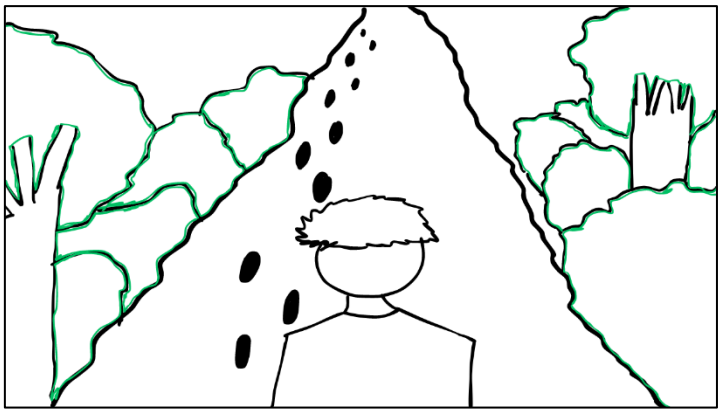
รูปภาพที่ 3.2 storyboard ขับรถเข้าป่า



รูปภาพที่ 3.3 storyboard เจอคนตัดหน้ารถ

รูปภาพที่ 3.4 storyboard รถชนกับต้นไม้

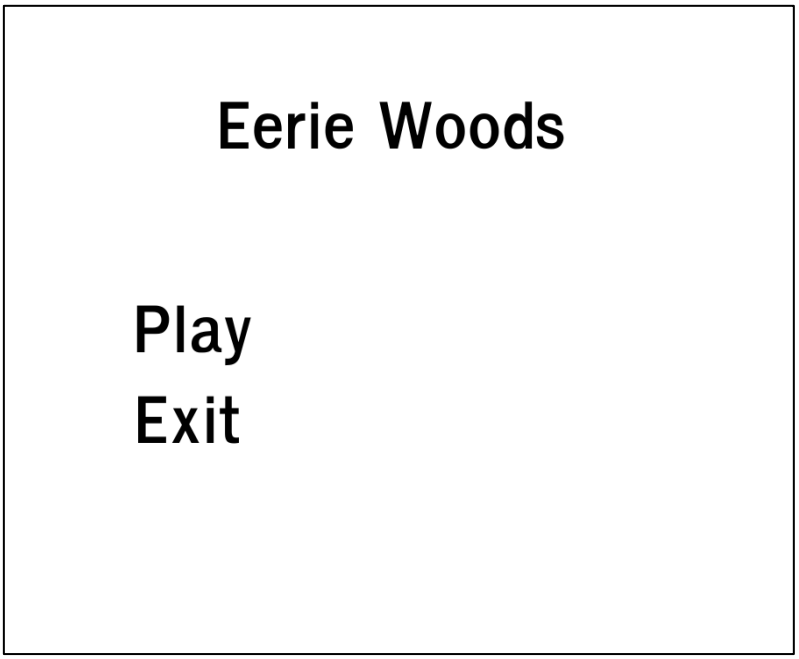
รูปภาพที่ 3.5 storyboard ตัวเอกสลบ



รูปภาพที่ 3.6 storyboard เดินเข้าป่า

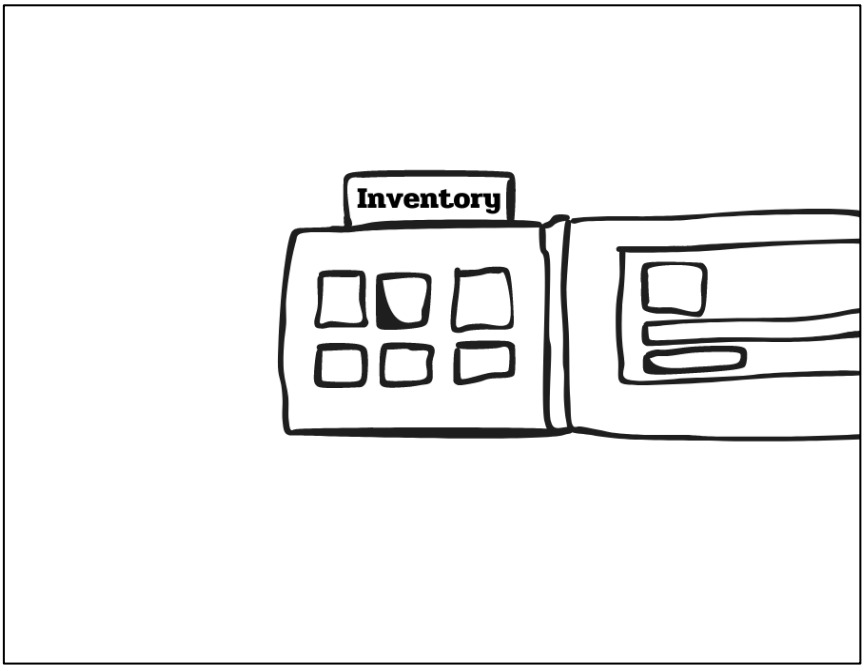
3.3 ออกแบบโครงสร้างเกม

3.3.1 ออกแบบหน้าเมนูของเกม



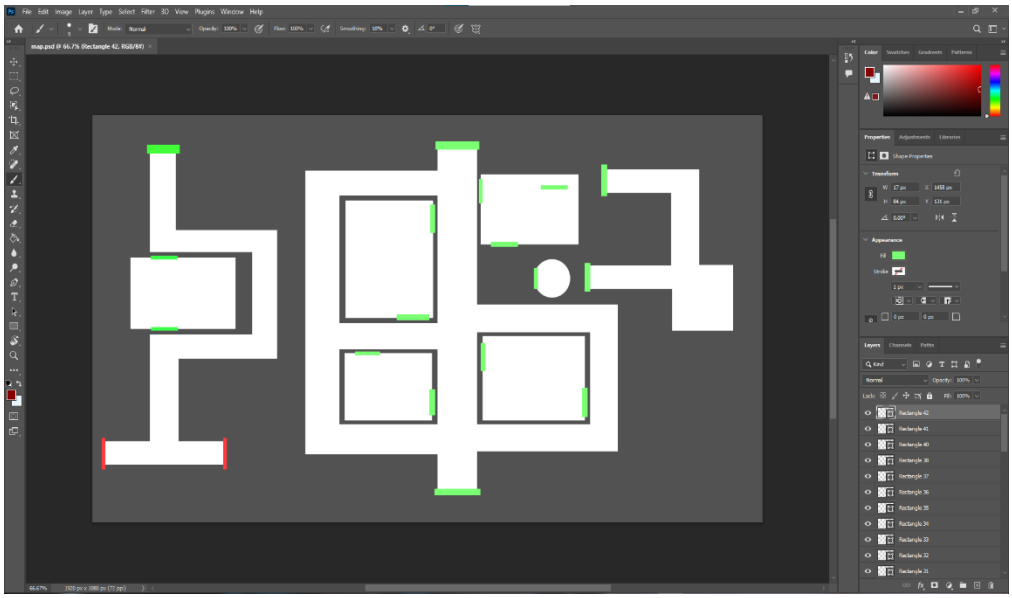
รูปภาพที่ 3.7 ออกแบบหน้าเมนู

รูปภาพที่ 3.8 ออกแบบหน้า Game over



รูปภาพที่ 3.9 ออกแบบหน้า inventory

3.3.2 ออกแบบแผนที่

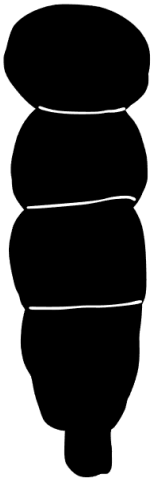


รูปภาพที่ 3.10 ออกแบบแผนที่

3.3.3 ออกแบบตัวละคร

รูปภาพที่ 3.11 ออกแบบตัวละครหลัก

รูปภาพที่ 3.12 ออกแบบตัวละครศัตรู 1

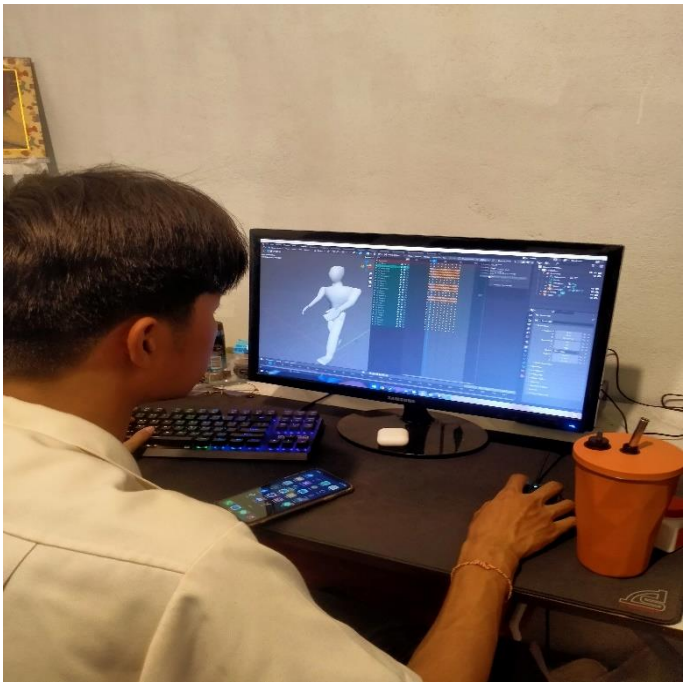


รูปภาพที่ 3.13 ออกแบบตัวละครศัตรู 2

3.4 ดำเนินการสร้างเกม

3.4.1 สร้างตัวละคร

รูปภาพที่ 3.14 สร้างตัวละครหลัก



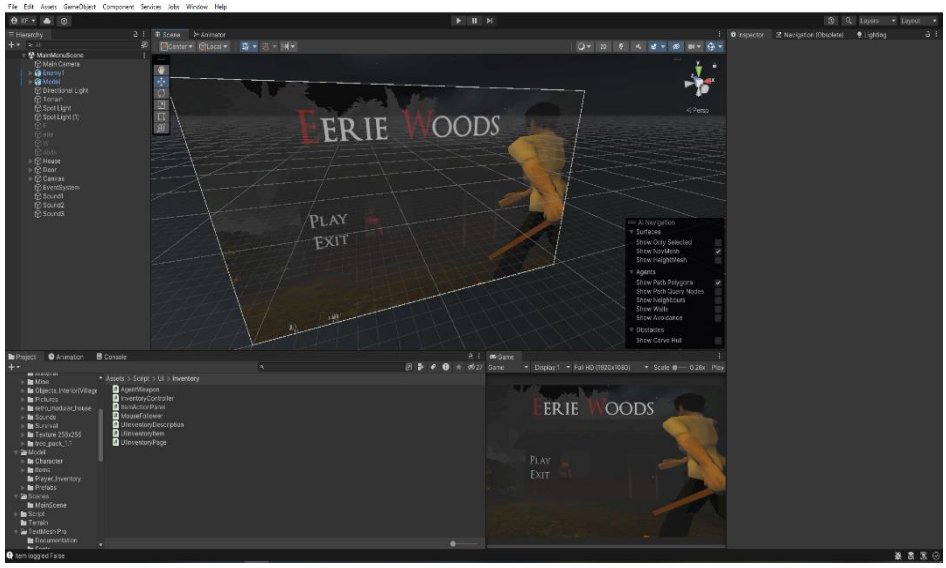
รูปภาพที่ 3.15 สร้างตัวละครศัตรู

รูปภาพที่ 3.16 ตัวละครหลัก

รูปภาพที่ 3.17 ตัวละครศัตรู 1

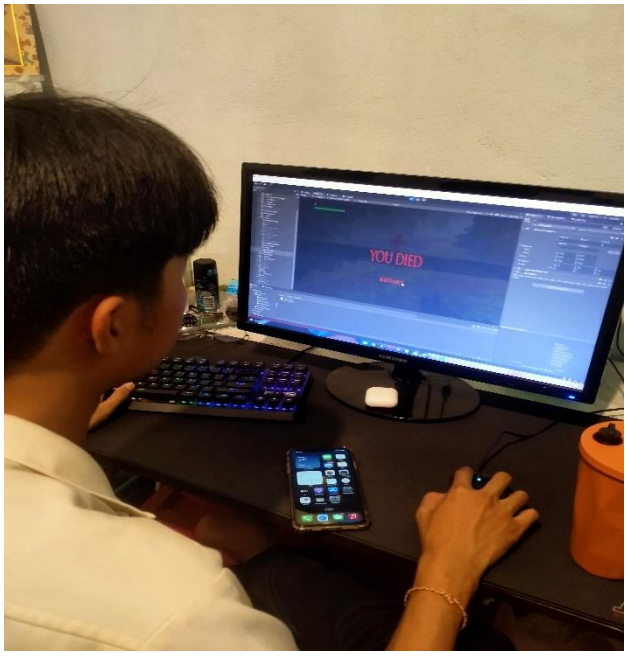
รูปภาพที่ 3.18 ตัวละครศัตรู 2

3.4.2 สร้างหน้าเมนูของเกม



รูปภาพที่ 3.19 สร้างหน้าเมนูของเกม

รูปภาพที่ 3.20 หน้าเมนูของเกม Eerie Woods

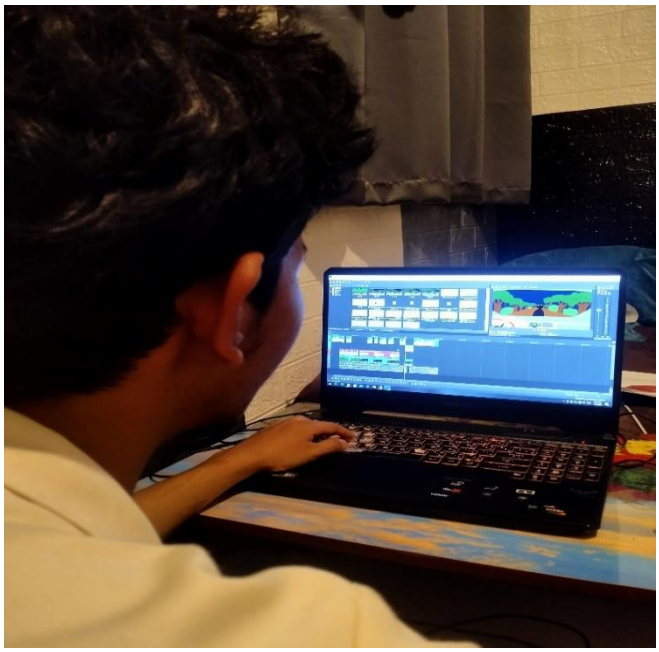


รูปภาพที่ 3.21 สร้างหน้า Game over

รูปภาพที่ 3.22 หน้า Game over

3.4.3 สร้างฉากในเกม

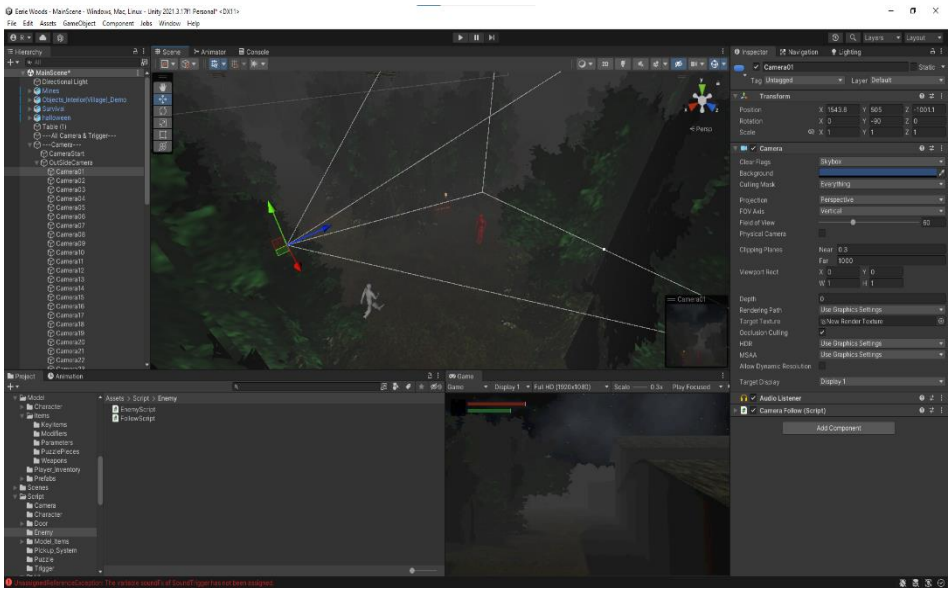
รูปภาพที่ 3.23 วาดฉากในเกม



รูปภาพที่ 3.24 ตัดต่อฉากในเกม

รูปภาพที่ 3.25 ฉากในเกม

3.4.4 ทำระบบการเล่นเกม



รูปภาพที่ 3.26 ระบบมุมกล้อง

รูปภาพที่ 3.27 ฉากมุมกล้องภายในเกม

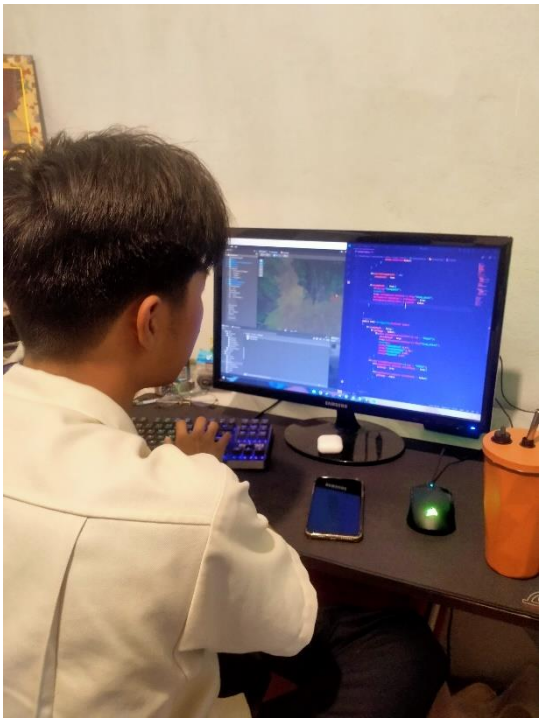
รูปภาพที่ 3.28 ระบบเก็บของ

รูปภาพที่ 3.29 ระบบเปิดช่องเก็บของ

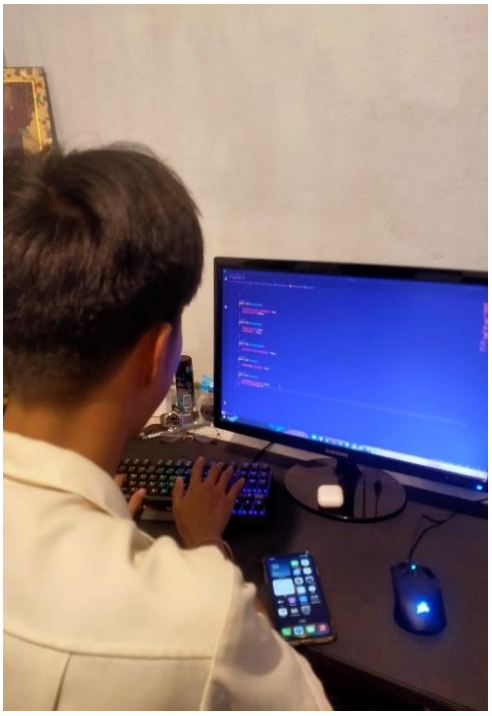
รูปภาพที่ 3.30 ระบบต่อสู้

3.5 การเขียนโค้ด

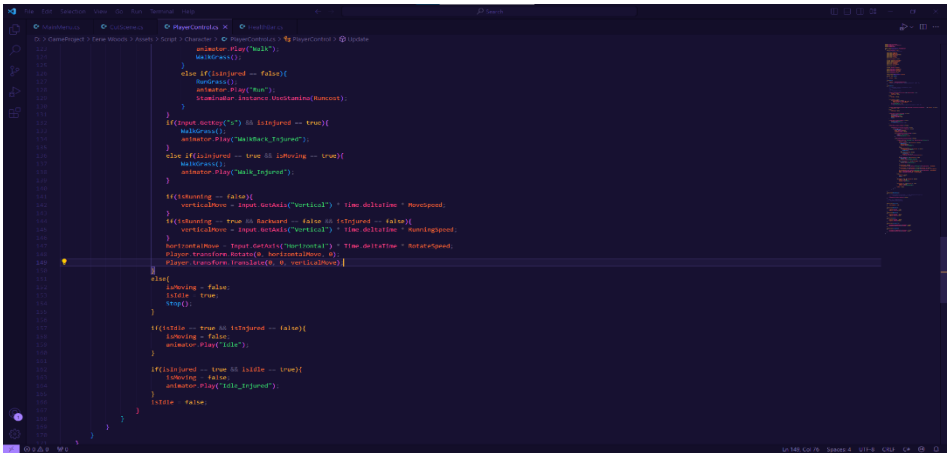
3.5.1 สร้างไฟล์ C# Script ขึ้นมา เพื่อป้อนโค้ดเข้าสู่เกม



รูปภาพที่ 3.31 การสร้างไฟล์ C# Script



รูปภาพที่ 3.32 เขียนโค้ดเกม



รูปภาพที่ 3.33 หน้าต่างเขียนโค้ดเกม

3.6 ทดสอบการทำงาน

3.5.1 หลังจากเข้าสู่เกมเป็นที่เรียบร้อยแล้ว ให้กดปุ่ม Play เพื่อเริ่มเล่นเกม

3.5.2 จากนั้นตัวเกมจะปรากฏ Cutscene หลังจาก Cutscene จบลงจะเข้าสู่การเล่นเกม

3.5.3 หาของ key quest เพื่อเปิดประตูไปหาห้อง Boss จัดการสู้กับ boss จนชนะจากนั้นตัวเกมจะปรากฏ Cutscene จบเกม

3.5.4 หากโดนศัตรูโจมตีจนหลอดเลือดหมดจะ Game over

3.7 การประเมินความพึงพอใจ

แบบประเมินความพึงพอใจของผู้ใช้งานจากการใช้งาน Eerie Woods เป็นแบบมาตราส่วนประมาณค่า 5 ระดับ ของลิเคิร์ท(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

ผลที่ได้จากการประเมินของผู้ทดลองใช้งานหลักการทางสถิติเข้ามาช่วยในการสรุปการประเมินเกมที่ได้สร้างขึ้น โดยคำนวณหาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของระดับความติดในแต่ละด้าน

สูตรหาค่าเฉลี่ย (ล้วนและอังคณา, 2538: 73)

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

เมื่อ $\vec{\mathrm{X}}$ แทน ค่าเฉลี่ยของคะแนน

$\sum \mathrm{X}$ แทน ผลรวมของคะแนน

N แทน จำนวน

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

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

เมื่อ S.D แทน ส่วนเบี่ยงเบนมาตรฐาน

X แทน คะแนนแต่ละคน

N แทน จำนวนคน

บทที่ 4

ผลการทดลอง

โครงงานครั้งนี้ คณะผู้จัดทำได้ทำการสร้างเกม Eerie Woods มีการเก็บรวบรวมข้อมูลผลการทดลองดังนี้

4.1 ผลจากการทดลองเล่นเกม Eerie Woods

4.2 ผลการประเมินหาคุณภาพของเกม Eerie Woods

4.1 ผลจากการทดลองเล่นเกม Eerie Woods

ผลจากการทดลองเล่นครั้งที่ 1

ปัญหา Animation ตัวละครเคลื่อนไหวบัคเวลาตัวละครโจมตี

วิธีแก้ ตั้ง Delay ให้กับ Animation เพื่อให้ Animation เล่นไม่ซ้ำกัน ตัวละครจะได้ไม่เคลื่อนไหวบัค

ผลจากการทดลองเล่นครั้งที่ 2

ปัญหา เปิดประตูบ้านหลังที่ 1 แล้วมุมกล้องไม่ตาม กล้องติดผิดตัว

วิธีแก้ ทำให้กล้องมันเปิดถูกตัว กำหนดกล้องให้เมื่อเปิดประตูบ้านหลังที่ 1 จะแสดงมุมกล้องภายในบ้านหลังที่ 1

ผลจากการทดลองเล่นครั้งที่ 3

ปัญหา ตัวศัตรูไม่เดินเข้าหาผู้เล่น

วิธีแก้ กำหนดพื้นที่ให้ตัวศัตรูสามารถเดินได้

ผลจากการทดลองเล่นครั้งที่ 4

ปัญหา เปิดประตูหลังบ้านของบ้านหลังที่ 2 แล้วมุมกล้องแสดงบ้านผิดหลัง

วิธีแก้ ตั้งกล้องเมื่อเปิดประตูหลังบ้านของบ้านหลังที่ 2 ให้แสดงมุมกล้องหลังบ้าน

ผลจากการทดลองเล่นครั้งที่ 5

ปัญหา ลงไปในบ่อน้ำแล้วขึ้นไม่ได้

วิธีแก้ ตัว trigger มันไม่ได้กำหนดว่าให้ตัวละครไปที่ไหน เลยขึ้นไม่ได้กำหนด trigger ให้เมื่อกดขึ้นบ่อน้ำจะแสดงมุมกล้องข้างนอกบ่อน้ำ

ผลจากการทดลองเล่นครั้งที่ 6

ปัญหา กดปุ่ม Restart แล้วพอเริ่มใหม่จะเปิดช่องเก็บของไม่ได้

วิธีแก้ ให้ตัว Code Inventory ตรวจสอบว่าตัว Item เป็นค่าว่างหรือไม่ ถ้าค่าว่างจะให้ออกจากฟังก์ชันทันทีโดยคำสั่ง (return)

ผลจากการทดลองเล่นครั้งที่ 7

ปัญหา เปิดประตูเข้าบ้านสองชั้น แล้วมุมกล้องไม่แสดงมุมกล้องภายในบ้าน

วิธีแก้ ตั้งกล้องเมื่อเปิดประตูเข้าบ้านสองชั้น ให้แสดงมุมกล้องภายในบ้าน

ผลจากการทดลองเล่นครั้งที่ 8

ไม่เกิดปัญหา เล่นได้จนจบ

ผลจากการทดลองเล่นครั้งที่ 9

ไม่เกิดปัญหา เล่นได้จนจบ

4.2 ผลการประเมินหาคุณภาพของเกม Eerie Woods

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. ความสมบูรณ์ของตัวเกม	4.6	0.54	ดีมาก
2. ความละเอียดในแต่ละส่วนของพื้นที่ในเกม	4.4	0.54	ดี
3. ความเหมาะสมของอุปสรรคในเกม	4.6	0.54	ดีมาก
4. ระยะเวลาในการเล่นเกม	3.8	0.44	ดี
5. ความเหมาะสมและการใช้งาน	4.2	0.83	ดี
ค่าเฉลี่ยรวม	4.22	0.56	ดี

จากตารางที่ 4.1 เป็นการประเมินความพึงพอใจของ Eerie Woods โดยรวมแล้ว มีความพึงพอใจเฉลี่ยอยู่ที่ ($\bar{X}$ = 4.22) และค่าเบี่ยงเบนมาตรฐาน**(S.D.)** เท่ากับ 0.56 ซึ่งค่าเฉลี่ยโดยร่วมอยู่ในระดับ ดี

บทที่ 5
สรุปผล ปัญหาอุปสรรค และข้อเสนอแนะ

5.1 สรุปผล

โครงการนี้มีวัตถุประสงค์เพื่อสร้างเกม Eerie Woods ให้สำเร็จลุล่วง เพื่อให้เสริมสร้างทักษะในการเล่นเกมของผู้เล่น และผู้เล่นต้องเอาชีวิตรอดจากศัตรูในหมู่บ้านและหาทางออกจากหมู่บ้าน โดยให้ได้ทดสอบการทำงานของเกม

เกม Eerie Woods สามารถให้ผู้เล่นต่อสู้กับศัตรูในหมู่บ้าน เพื่อหา key items แล้วหาทางออกจากป่าให้ได้

จากการประเมินความพึงพอใจโดยมีผู้ประเมิน จำนวน 10 ท่าน สรุปได้ว่าเกม Eerie Woods มีคุณภาพอยู่ในระดับดี สามารถใช้งานได้ตามขอบเขตที่กำหนดไว้

5.2 ปัญหาและอุปสรรค

5.2.1 กดปุ่ม Restart แล้วเกมบัค

5.2.2 Animation ตัวละครเคลื่อนไหวบัค

5.3 แนวทางแก้ไข

5.3.1 ให้ตัว Code Inventory ตรวจสอบว่าตัว Item เป็นค่าว่างหรือไม่ ถ้าค่าว่างจะให้ออกจากฟังก์ชันทันทีโดยคำสั่ง (return)

5.3.2 ตั้ง Delay ให้กับ Animation เพื่อให้ Animation เล่นไม่ซ้ำกัน ตัวละครจะได้ไม่เคลื่อนไหวบัค

5.4 ข้อเสนอแนะ

5.4.1 ควรเพิ่มระบบการเล่นให้มากขึ้น

5.4.2 ควรศึกษาการออกแบบเกมให้ดียิ่งขึ้น

ภาคผนวก

ภาคผนวก ก.

แบบขออนุมัติโครงการ

ภาคผนวก ข.

แบบสอบถามความคิดเห็นผู้ใช้งาน

ภาคผนวก ค.

คู่มือการใช้งาน

คู่มือการใช้งาน

1. ขั้นตอนการเข้าใช้งาน

รูปภาพที่ ค.1 หน้าเมนูของเกม Eerie woods

1) กด Play เพื่อเริ่มเกม

2) กด Exit เพื่อออกจากเกม

2. ขั้นตอนการใช้งานโปรแกรม

รูปภาพที่ ค.2 ฉากในเกม

1) หลอดเลือดของตัวละคร จะลดเมื่อโดนโจมตี

2) หลอด stamina ของตัวละคร จะลดเมื่อวิ่ง

3) ตัวละครหลัก กด W,A,S,D เพื่อเดิน กด Shift เพื่อวิ่ง

4) ประตู กด F เพื่อเปิดประตู

5) Item กด F เพื่อเก็บ

รูปภาพที่ ค.3 หน้าช่องเก็บของ

1) หน้าช่องเก็บของ กด TAB เพื่อเปิด

2) ของเพิ่มเลือด กดคลิกเมาส์ขวาแล้วกด use เพื่อใช้

3) อาวุธ กดคลิกเมาส์ขวาแล้วกด Equip เพื่อถือ กดคลิกเมาส์ขวาเพื่อเล็ง คลิกเมาส์ซ้ายเพื่อโจมตี

4) Key item ใช้แก้ปริศนา ต้องเก็บให้ได้ 5 ชิ้น

5) กุญแจ ใช้เพื่อเปิดประตูที่ล็อกอยู่

ภาคผนวก ง.

Source code

โค้ดที่ใช้ใน Eerie woods

ChangePosition

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ChangePosition : MonoBehaviour
{
    public GameObject Player;
    public Transform targetPosition;
    public static ChangePosition instance;
    private void Awake()
    {
        instance = this;
    }
    public void OnTriggerStay(Collider other)
    {
        // textF.SetActive(true);
        if(HealthBar.instance.isDeath == false){
            if(other.gameObject.tag == "Player"){
                Player.transform.position = targetPosition.position;
            }
        }
    }
}
```

CameraChange

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraChange : MonoBehaviour
{
    public GameObject FromCamera;
    public GameObject ToCamera;

    public int cameraNumber;
    // Start is called before the first frame update
    void Start()
    {
        cameraNumber = 1;
    }
    // Update is called once per frame
    void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Player"){
            Debug.Log("change");
            FromCamera.SetActive(false);
            ToCamera.SetActive(true);
        }
    }
}
```

CameraFollow

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraFollow : MonoBehaviour
{
```

```
    public Transform target;
    // public float rotationSpeed;
    void Update()
    {
        if(target != null){
            // transform.position = target.position;
            transform.LookAt(target);
            // float horizontal = Input.GetAxis("Horizontal");
            // float vertical   = Input.GetAxis("Vertical");
            // if(horizontal != 0 || vertical != 0){
            //     float targetAngle = Mathf.Atan2(horizontal, vertical) * Mathf.Rad2Deg;
            //     float angle = Mathf.LerpAngle(transform.eulerAngles.y, targetAngle,
rotationSpeed * Time.deltaTime);
            //     transform.rotation = Quaternion.Euler(0f, angle, 0f);
            // }
        }
    }
}
```

HealthBar

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Inventory.Model;
public class HealthBar : MonoBehaviour
{
    public Slider Health;
    public Slider EaseHealth;
    public GameObject HealEff;
```

```
public float MaxHealth;
public float CurrentHealth;
public float lerpSpeed;
public float Damage;
public float ItemHeal;
public float ItemHeal1;
public bool GetDamage;
public bool Heal;
public bool Heal1;
public bool isDeath = false;
public static HealthBar instance;
private void Awake()
{
    instance = this;
}
void Start()
{
    CurrentHealth = MaxHealth;
    Health.GetComponent<Slider>().maxValue = MaxHealth;
    Health.GetComponent<Slider>().value = CurrentHealth;
    EaseHealth.GetComponent<Slider>().maxValue = MaxHealth;
    // Health.maxValue = MaxHealth;
    // Health.value = MaxHealth;
    Damage = EnemyScript.instance.FromDamage;
    // ItemHeal = ModifierData.instance.value;
    ItemHeal = 7;
    ItemHeal1 = 12;
}
```

```
void Update()
{
    if(CurrentHealth/MaxHealth*100<=30){
        PlayerControl.instance.isInjured = true;
    }
    else{
        PlayerControl.instance.isInjured = false;
    }

    if(Health.value != CurrentHealth){
        Health.value = CurrentHealth;
    }
    if(CurrentHealth >= MaxHealth){
        CurrentHealth = MaxHealth;
    }
    if(CurrentHealth <= 0){
        isDeath = true;
        CurrentHealth = 0;
    }
    else{
        isDeath = false;
    }
    // if(Input.GetKey(KeyCode.Space)){
    //     takeDamage(5);
    // }
    if(Health.value != EaseHealth.value){
        EaseHealth.value = Mathf.Lerp(EaseHealth.value, CurrentHealth, lerpSpeed);
    }
    if(GetDamage == true){
```

```
            CurrentHealth -= Damage;
        }
        if(Heal == true){
            CurrentHealth += ItemHeal;
            HealEff.SetActive(true);
            // AddHealth(ItemHeal);
            Invoke("Healoff",3.0f);
            Invoke("HealFalse",0.25f);
        }
        if(Heal1 == true){
            CurrentHealth += ItemHeal1;
            HealEff.SetActive(true);
            // AddHealth(ItemHeal);
            Invoke("Healoff",3.0f);
            Invoke("HealFalse",0.25f);
        }
    }
    // public void takeDamage(float damage)
    // {
    //     GetDamage = true;
    //     Invoke("GetDamagefalse",0.7f);
    // }
    // private void GetDamagefalse()
    // {
    //     GetDamage = false;
    // }
    // public void AddHealth(float HealHealth)
    // {
    //     CurrentHealth += HealHealth;
```

```
    // }
    public void HealFalse()
    {
        Heal = false;
        Heal1 = false;
    }
    public void Healoff()
    {
        HealEff.SetActive(false);
    }
}
```

PlayerControl

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class PlayerControl : MonoBehaviour
{
    public GameObject Player;
    // public GameObject DeathFade;
    public bool isMoving;
    public bool isRunning;
    public bool Backward = false;
    public bool isInjured;
    public bool isIdle;
    public bool turnBack;
    bool isFacingRight = true;
    private float horizontalMove;
    private float verticalMove;
```

```
public int RotateSpeed;
public int MoveSpeed;
public int RunningSpeed;
public int Runcost;
private Animator Run cost;
public AudioSource Stepgrass;
public AudioSource Rungrass;
public AudioSource getDamge;
public GameObject Blood;
public static PlayerControl instance;
private void Awake()
{
    instance = this;
}
void Start()
{
    animator = GetComponent<Animator>();
    // Quaternion targetRotation = Quaternion.Euler(0f, +180f, 0f);
}
void Update()
{
    // if(HealthBar.instance.CurrentHealth <= 30){
    //     isInjured = true;
    // }
    // else{
    //     isInjured = false;
    // }
    if(Input.GetKey(KeyCode.LeftShift) && Input.GetKey("s")){
        turnBack = true;
```

```
            isRunning = false;
        }
        else{
            turnBack = false;
        }
        if(turnBack == true){
            if(isFacingRight == true){
                Player.transform.Rotate (0f, 180f, 0f);
                isFacingRight = !isFacingRight;
            }
            Invoke("ResetFlip",1.0f);
            // Player.transform.rotation = Quaternion.Euler(0f, +180f, 0f);
        }
        if(Input.GetKey(KeyCode.LeftShift) && StaminaBar.instance.CurrentStamina > 0
&& Backward == false){
            isRunning = true;
        }
        else{
            isRunning = false;
        }
        if(HealthBar.instance.GetDamage == true){
            animator.Play("GetDamage");
            colliderOff();
            Blood.SetActive(true);
            getDamge.enabled = true;
            Stop();
        }
        if(HealthBar.instance.isDeath == true){
            animator.Play("Death");
```

```
    colliderOff();
    // Invoke("fadeDeathScene",5.0f);
}
if(HealthBar.instance.isDeath == false){
    if(WeaponControl.instance.isAiming == false){
        if(HealthBar.instance.GetDamage == false){
            colliderOn();
            Blood.SetActive(false);
            getDamge.enabled = false;

            if(PickupSystem.instance.IsPickup == true){
                animator.Play("Pickup");
                Stop();
                Invoke("DelayPickup",2.0f);
            }
            if(PickupSystem.instance.IsPickup == false){
                if(Input.GetButton("Horizontal") || Input.GetButton("Vertical")){
                    isMoving = true;

                    if(Input.GetKey("s") && isInjured == false){
                        WalkGrass();
                        Backward = true;
                        animator.Play("WalkBack");
                    }
                    else{
                        Backward = false;
                        if(isRunning == false && isInjured == false){
                            animator.Play("Walk");
                            WalkGrass();
```

```
                }
                else if(isInjured == false){
                    RunGrass();
                    animator.Play("Run");
                    StaminaBar.instance.UseStamina(Runcost);
                }
            }
            if(Input.GetKey("s") && isInjured == true){
                WalkGrass();
                animator.Play("WalkBack_Injured");
            }
            else if(isInjured == true && isMoving == true){
                WalkGrass();
                animator.Play("Walk_Injured");
            }
            if(isRunning == false){
                verticalMove = Input.GetAxis("Vertical") * Time.deltaTime *
MoveSpeed;
            }
            if(isRunning == true && Backward == false && isInjured == false){
                verticalMove = Input.GetAxis("Vertical") * Time.deltaTime *
RunningSpeed;
            }
            horizontalMove = Input.GetAxis("Horizontal") * Time.deltaTime *
RotateSpeed;
            Player.transform.Rotate(0, horizontalMove, 0);
            Player.transform.Translate(0, 0, verticalMove);
        }
        else{
```

```
                    isMoving = false;
                    isIdle = true;
                    Stop();
                }
                if(isIdle == true && isInjured == false){
                    isMoving = false;
                    animator.Play("Idle");
                }
                if(isInjured == true && isIdle == true){
                    isMoving = false;
                    animator.Play("Idle_Injured");
                }
                isIdle = false;
            }
        }
    }
}
public void DelayPickup()
{
    // if(Input.GetButton("Horizontal") || Input.GetButton("Vertical")){
    //     PickupSystem.instance.IsPickup = false;
    // }
    PickupSystem.instance.IsPickup = false;
}
// public void fadeDeathScene(){
//     DeathFade.SetActive(true);
// }
public void ResetFlip()
```

```
    {
        isFacingRight = true;
    }
    public void WalkGrass()
    {
        Stepgrass.enabled = true;
        Rungrass.enabled = false;
    }
    public void RunGrass()
    {
        Stepgrass.enabled = false;
        Rungrass.enabled = true;
    }
    public void Stop()
    {
        Stepgrass.enabled = false;
        Rungrass.enabled = false;
    }
    public void colliderOff()
    {
        GetComponent<Rigidbody>().isKinematic = true;
        GetComponent<BoxCollider>().enabled = false;
    }
    public void colliderOn()
    {
        GetComponent<Rigidbody>().isKinematic = false;
        GetComponent<BoxCollider>().enabled = true;
    }
}
```

PlayerSound

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class PlayerSound : MonoBehaviour
{
    public AudioSource stepGrass;
    public AudioSource runGrass;
    public AudioSource getDamage;
    public static PlayerSound instance;
    private void Awake()
    {
        instance = this;
    }
    void update()
    {
        if(HealthBar.instance.GetDamage == true){
            getDamage.enabled = true;
            Stop();
        }
        if(HealthBar.instance.GetDamage == false){
            getDamage.enabled = false;
        }
        if(PickupSystem.instance.IsPickup == true){
            Stop();
        }
        if(PlayerControl.instance.isMoving == true || PlayerControl.instance.Backward ==
true){
```

```
            WalkGrass();
        }
        if(PlayerControl.instance.isRunning == true){
            RunGrass();
        }
        if(PlayerControl.instance.isMoving == false){
            Stop();
        }
    }
    public void WalkGrass()
    {
        stepGrass.enabled = true;
        runGrass.enabled = false;
    }
    public void RunGrass()
    {
        stepGrass.enabled = false;
        runGrass.enabled = true;
    }
    public void Stop()
    {
        stepGrass.enabled = false;
        runGrass.enabled = false;
    }
}
```

StaminaBar

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class StaminaBar : MonoBehaviour
{
    public Slider Stamina;
    public Slider EaseStamina;
    public int MaxStamina;
    public int CurrentStamina;
    public float lerpSpeed;
    private WaitForSeconds regenTick = new WaitForSeconds(0.1f);
    private Coroutine regen;
    public static StaminaBar instance;
    // Start is called before the first frame update
    private void Awake()
    {
        instance = this;
    }
    void Start()
    {
        CurrentStamina = MaxStamina;
        Stamina.GetComponent<Slider>().maxValue = MaxStamina;
        Stamina.GetComponent<Slider>().value = CurrentStamina;
        EaseStamina.GetComponent<Slider>().maxValue = MaxStamina;
        // Stamina.maxValue = MaxStamina;
        // Stamina.value = MaxStamina;
    }
```

```
    void Update()
    {
        Stamina.GetComponent<Slider>().value = CurrentStamina;
        if(Stamina.value != EaseStamina.value){
            EaseStamina.value = Mathf.Lerp(EaseStamina.value, CurrentStamina,
lerpSpeed);
        }
        if(CurrentStamina >= MaxStamina){
            CurrentStamina = MaxStamina;
        }
    }
    public void UseStamina(int amount)
    {
        if(CurrentStamina - amount >= 0){
            CurrentStamina -= amount;
            Stamina.value = CurrentStamina;
            if(regen != null){
                StopCoroutine(regen);
            }
            regen = StartCoroutine(RegenStamina());
        }
    }
    private IEnumerator RegenStamina()
    {
        yield return new WaitForSeconds(1.5f);
        while(CurrentStamina < MaxStamina){
            CurrentStamina += MaxStamina/50;
            Stamina.value = CurrentStamina;
            yield return regenTick;
```

```
        }
        regen = null;
    }
}
```

StartP

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class StartP : MonoBehaviour
{
    private Animator animator;
    public GameObject Player;
    public GameObject PlayerAnim;
    public GameObject CamStart;
    public GameObject ToCam;
    public GameObject uii;
    // public GameObject edge;
    // public GameObject edgeAnim;
    void Start()
    {
        animator = GetComponent<Animator>();
        Player.SetActive(false);
        ToCam.SetActive(false);
        uii.SetActive(false);
        // edgeAnim.SetActive(false);
        // edge.SetActive(true);
    }
    void Update()
    {
```

```
        animator.Play("Start");
        // Invoke("edgefade",6.4f);
        Invoke("gameStart",9.5f);
        Invoke("UIfade",10.0f);
    }
    // public void edgefade()
    // {
    //     edge.SetActive(false);
    //     edgeAnim.SetActive(true);
    // }
    public void gameStart()
    {
        Player.SetActive(true);
        CamStart.SetActive(false);
        PlayerAnim.SetActive(false);
        ToCam.SetActive(true);
    }
    public void UIfade()
    {
        uii.SetActive(true);
    }
}
```

WeaponControl

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class WeaponControl : MonoBehaviour
{
    public GameObject Player;
```

```
public GameObject inventory;
public GameObject woodStick;
public GameObject axe;
public GameObject pickaxe;
public GameObject sword;
public AudioSource attackS;
// public List<GameObject> weaponObjectList;
public float attackDamage;
public bool isAiming = false;
public bool isAttack;
public bool weaponReady;
public bool AttackEnemy;
private float horizontalMove;
public int attackCost;
private Animator animator;
public static WeaponControl instance;
private void Awake()
{
    instance = this;
}
void Start()
{
    animator = GetComponent<Animator>();
    // weaponObjectList = new List<GameObject>();
    // weaponObjectList.Add(new GameObject("woodStick"));
    // weaponObjectList.Add(new GameObject("axe"));
    attackS.enabled = false;
    woodStick.SetActive(false);
    axe.SetActive(false);
```

```
    }
    void Update()
    {
        // DeactivateObject();
        if( !woodStick.activeSelf || !axe.activeSelf || !pickaxe.activeSelf || !sword.activeSelf
){
            if(inventory.activeSelf){
               weaponReady = false;
            }
        }
        if(woodStick.activeSelf || axe.activeSelf || pickaxe.activeSelf || sword.activeSelf ){
            if(!inventory.activeSelf){
                weaponReady = true;
            }
        }
        if(woodStick.activeSelf){
            attackDamage = 2;
        }
        if(axe.activeSelf){
            attackDamage = 4;
        }
        if(pickaxe.activeSelf){
            attackDamage = 6;
        }
        if(sword.activeSelf){
            attackDamage = 8;
        }
        if(HealthBar.instance.isDeath == false){
            if(weaponReady == true){
```

```
            if(Input.GetMouseButtonDown(1)){
                if(axe.transform.rotation.eulerAngles.z != -112f){
                    axe.transform.Rotate(0f, 0f, -112f);
                    Debug.Log("rotate");
                }
                if(pickaxe.transform.rotation.eulerAngles.z != 90f){
                    pickaxe.transform.Rotate(0f, 0f, 90f);
                    Debug.Log("rotate");
                }
                if(sword.transform.rotation.eulerAngles.z != 0f){
                    sword.transform.Rotate(0f, 0f, 0f);
                    Debug.Log("rotate");
                }
                PlayerControl.instance.Stop();
                isAiming = true;
                animator.Play("Prefigure");
            }
            if(Input.GetMouseButtonUp(1)){
                // isAiming = false;
                animator.Play("UnPrefigure");
                Invoke("resetRotate",0.1f);
                Invoke("unAim",1.0f);
            }
            if(isAiming == true){
                if(Input.GetButton("Horizontal")){
                    horizontalMove = Input.GetAxis("Horizontal") * Time.deltaTime *
PlayerControl.instance.RotateSpeed;
                    Player.transform.Rotate(0, horizontalMove, 0);
                }
```

```
            }
            if(Input.GetMouseButtonDown(0) && isAiming == true){
                isAttack = true;
                Debug.Log("attack");
                animator.Play("Attack");
            }
            if(Input.GetMouseButtonUp(0) && isAiming == true){
                Invoke("ResetAttack",1.0f);
            }
        }
    }
}
public void OnTriggerStay(Collider other)
{
    if(isAttack == true){
        StaminaBar.instance.UseStamina(attackCost);
        if(other.GetComponent<Collider>().tag == "Enemy"){
            Invoke("enemyGetdamage",0.5f);
            Invoke("resetGetdamage",1.0f);
        }
    }
}
public void unAim()
{
    isAiming = false;
}
public void ResetAttack()
{
    isAttack = false;
```

```
        animator.Play("UnPrefigure");
        // Invoke("resetRotate",0.1f);
        Invoke("unAim",0.5f);
    }
    public void resetRotate()
    {
        axe.transform.Rotate(0f, 0f, 112f);
        pickaxe.transform.Rotate(0f, 0f, 0f);
        sword.transform.Rotate(0f, 0f, 90f);
        Debug.Log("rotatee");
    }
    public void enemyGetdamage()
    {
        // EnemyScript.instance.getDamge = true;
        AttackEnemy = true;
        attackS.enabled = true;
    }
    public void resetGetdamage()
    {
        // EnemyScript.instance.getDamge = false;
        AttackEnemy = false;
        attackS.enabled = false;
    }
}
```

LadderDown

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

```
public class LadderDown : MonoBehaviour
{
    public AudioSource ladderSound1;
    public AudioSource ladderSound2;
    void Start()
    {
        StartCoroutine(ladderup());
        soundOFF();
    }
    IEnumerator ladderup()
    {
        yield return new WaitForSeconds(1.5f);
        soundON();
        yield return new WaitForSeconds(0.5f);
        soundOFF();
        yield return new WaitForSeconds(2.0f);
        soundON();
        yield return new WaitForSeconds(0.5f);
        soundOFF();
        yield return new WaitForSeconds(2.0f);
        soundON();
        yield return new WaitForSeconds(0.5f);
        soundOFF();
        yield return new WaitForSeconds(2.5f);
        ladderSound1.enabled = true;
        SceneManager.UnloadScene("DownLadderScene");
        Debug.Log("return");
    }
    public void soundOFF()
```

```
    {
        ladderSound1.enabled = false;
        ladderSound2.enabled = false;
    }
    public void soundON()
    {
        ladderSound1.enabled = true;
        ladderSound2.enabled = true;
    }
}
```

LadderUp

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class LadderUp : MonoBehaviour
{
    public AudioSource ladderSound1;
    public AudioSource ladderSound2;
    void Start()
    {
        StartCoroutine(ladderup());
        soundOFF();
    }
    IEnumerator ladderup()
    {
        yield return new WaitForSeconds(1.5f);
        soundON();
        yield return new WaitForSeconds(0.5f);
```

```
        soundOFF();
        yield return new WaitForSeconds(2.0f);
        soundON();
        yield return new WaitForSeconds(0.5f);
        soundOFF();
        yield return new WaitForSeconds(2.0f);
        soundON();
        yield return new WaitForSeconds(0.5f);
        soundOFF();
        yield return new WaitForSeconds(2.5f);
        ladderSound1.enabled = true;
        SceneManager.UnloadScene("UpLadderScene");
        Debug.Log("return");
    }
    public void soundOFF()
    {
        ladderSound1.enabled = false;
        ladderSound2.enabled = false;
    }
    public void soundON()
    {
        ladderSound1.enabled = true;
        ladderSound2.enabled = true;
    }
}
```

MoveObjectLadder

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MoveObjectLadder : MonoBehaviour
{
    public Transform safeTarget;
    public Transform Target;
    public GameObject Player;
    public string Scenename;
    // public GameObject textF;
    public GameObject CanvaOff;
    public static MoveObjectLadder instance;
    private void Awake()
    {
        instance = this;
    }
    // void Start(){
    //     textF.SetActive(false);
    // }
    public void OnTriggerStay(Collider other)
    {
        // textF.SetActive(true);
        if(HealthBar.instance.isDeath == false){
            if(other.gameObject.tag == "Player"){
                if(Input.GetKey("f")){
                    CanvaOff.SetActive(false);
                    SceneManager.LoadScene(Scenename,LoadSceneMode.Additive);
```

```
                    MoveObjectToSafe();
                    Invoke("MoveObjectToTaget",10.0f);
                }
            }
        }
    }
    // public void OnTriggerExit(Collider other)
    // {
    //     textF.SetActive(false);
    // }
    // void Update()
    // {
    //     if(Input.GetKey("x")){
    //         MoveObjectsToTaget();
    //     }
    // }
    public void MoveObjectToSafe()
    {
        Player.transform.position = safeTarget.position;
    }
    public void MoveObjectToTaget()
    {
        // Vector3 currentPosition = Character.transform.position;
        Player.transform.position = Target.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
    }
}
```

KeySystem

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class KeySystem : MonoBehaviour
{
    public GameObject KeyBedroom;
    public bool keyItemBedroom = false;
    public GameObject KeySecondFloor;
    public bool keyItemSecondFloor = false;
    public static KeySystem instance;
    private void Awake()
    {
        instance = this;
    }
    void Start()
    {
    }
    void Update()
    {
        if(KeyBedroom == null){
            Debug.Log("destroy");
            keyItemBedroom = true;
        }
        else{
            keyItemBedroom = false;
        }
        if(KeySecondFloor == null){
            Debug.Log("destroy");
```

```
            keyItemSecondFloor = true;
        }
        else{
            keyItemSecondFloor = false;
        }
    }
}
```

LockDoor

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class LockDoor : MonoBehaviour
{
    public GameObject door;
    public AudioSource doorSound;
    void Start()
    {
        StartCoroutine(cannotOpenDoor());
    }
    IEnumerator cannotOpenDoor()
    {
        yield return new WaitForSeconds(1.0f);
        door.GetComponent<Animator>().Play("DoorLock");
        yield return new WaitForSeconds(0.4f);
        doorSound.Play();
        yield return new WaitForSeconds(0.9f);
        doorSound.Play();
        yield return new WaitForSeconds(2.5f);
```

```
        SceneManager.UnloadScene("LockDoorScene");
    }
}
```

MoveObjectLockDoor

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MoveObjectLockDoor : MonoBehaviour
{
    public Transform safeTarget;
    public Transform Target;
    public Transform returnTarget;
    public GameObject Player;
    // public GameObject textF;
    public GameObject ColLock;
    public GameObject ColOpen;
    public GameObject CanvaOff;
    public static MoveObjectLockDoor instance;
    private void Awake()
    {
        instance = this;
    }
    void Start(){
        ColOpen.SetActive(false);
    }
    public void OnTriggerStay(Collider other)
    {
        // textF.SetActive(true);
```

```
        if(HealthBar.instance.isDeath == false){
            if(other.gameObject.tag == "Player"){
                if(Input.GetKey("f")){
                    if(KeySystem.instance.keyItemBedroom == true){
                        CanvaOff.SetActive(false);
SceneManager.LoadScene("OpenDoorScene",LoadSceneMode.Additive);
                        MoveObjectToSafe();
                        Invoke("MoveObjectToTaget",8.5f);
                    }
                    if(KeySystem.instance.keyItemBedroom == false){
                        CanvaOff.SetActive(false);
SceneManager.LoadScene("LockDoorScene",LoadSceneMode.Additive);
                        MoveObjectToSafe();
                        Invoke("MoveObjectToreturnTaget",6.5f);
                    }
                }
            }
        }
    }
    public void MoveObjectToSafe()
    {
        Player.transform.position = safeTarget.position;
    }
    public void MoveObjectToTaget()
    {
        Player.transform.position = Target.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
        ColLock.SetActive(false);
```

```
        ColOpen.SetActive(true);
    }
    public void MoveObjectToreturnTaget()
    {
        Player.transform.position = returnTarget.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
    }
}
```

MoveObjectLockDoor1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MoveObjectLockDoor1 : MonoBehaviour
{
    public Transform safeTarget;
    public Transform Target;
    public Transform returnTarget;
    public GameObject Player;
    // public GameObject textF;
    public GameObject ColLock;
    public GameObject ColOpen;
    public GameObject CanvaOff;
    public static MoveObjectLockDoor1 instance;
    private void Awake()
    {
        instance = this;
    }
```

```
    void Start(){
        ColOpen.SetActive(false);
    }
    public void OnTriggerStay(Collider other)
    {
        // textF.SetActive(true);
        if(HealthBar.instance.isDeath == false){
            if(other.gameObject.tag == "Player"){
                if(Input.GetKey("f")){
                    if(KeySystem.instance.keyItemSecondFloor == true){
                        CanvaOff.SetActive(false);
SceneManager.LoadScene("OpenDoorScene",LoadSceneMode.Additive);
                        MoveObjectToSafe();
                        Invoke("MoveObjectToTaget",8.5f);
                    }
                    if(KeySystem.instance.keyItemSecondFloor == false){
                        CanvaOff.SetActive(false);
SceneManager.LoadScene("LockDoorScene",LoadSceneMode.Additive);
                        MoveObjectToSafe();
                        Invoke("MoveObjectToreturnTaget",6.5f);
                    }
                }
            }
        }
    }
    public void MoveObjectToSafe()
    {
        Player.transform.position = safeTarget.position;
    }
```

```
    public void MoveObjectToTaget()
    {
        Player.transform.position = Target.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
        ColLock.SetActive(false);
        ColOpen.SetActive(true);
    }
    public void MoveObjectToreturnTaget()
    {
        Player.transform.position = returnTarget.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
    }
}
```

MoveObjectLockDoorr

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MoveObjectLockDoorr : MonoBehaviour
{
    public Transform safeTarget;
    public Transform Target;
    public Transform returnTarget;
    public GameObject Player;
    // public GameObject textF;
    public GameObject ColLock;
    // public GameObject ColOpen;
```

```
    public GameObject CanvaOff;
    public static MoveObjectLockDoorr instance;
    private void Awake()
    {
        instance = this;
    }
    void Start(){
        // ColOpen.SetActive(false);
    }
    public void OnTriggerStay(Collider other)
    {
        // textF.SetActive(true);
        if(HealthBar.instance.isDeath == false){
            if(other.gameObject.tag == "Player"){
                if(Input.GetKey("f")){
                    // if(KeySystem.instance.keyItemBedroom == true){
                    //      CanvaOff.SetActive(false);
                    //
SceneManager.LoadScene("OpenDoorScene",LoadSceneMode.Additive);
                    //      MoveObjectToSafe();
                    //      Invoke("MoveObjectToTaget",8.5f);
                    // }
                    if(KeySystem.instance.keyItemBedroom == false){
                        CanvaOff.SetActive(false);
SceneManager.LoadScene("LockDoorScene",LoadSceneMode.Additive);
                        MoveObjectToSafe();
                        Invoke("MoveObjectToreturnTaget",6.5f);
                    }
                }
```

```
            }
        }
    }
    public void MoveObjectToSafe()
    {
        Player.transform.position = safeTarget.position;
    }
    public void MoveObjectToTaget()
    {
        Player.transform.position = Target.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
        ColLock.SetActive(false);
        // ColOpen.SetActive(true);
    }
    public void MoveObjectToreturnTaget()
    {
        Player.transform.position = returnTarget.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
    }
}
```

MoveObjects

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MoveObjects : MonoBehaviour
{
```

```
public Transform safeTarget;
public Transform Target;
public GameObject Player;
// public GameObject textF;
public GameObject CanvaOff;
public static MoveObjects instance;
private void Awake()
{
    instance = this;
}
// void Start(){
//     textF.SetActive(false);
// }
public void OnTriggerStay(Collider other)
{
    // textF.SetActive(true);
    if(HealthBar.instance.isDeath == false){
        if(other.gameObject.tag == "Player"){
            if(Input.GetKey("f")){
                CanvaOff.SetActive(false);
                SceneManager.LoadScene("DoorScene",LoadSceneMode.Additive);
                MoveObjectToSafe();
                Invoke("MoveObjectToTaget",7.0f);
            }
        }
    }
}
// public void OnTriggerExit(Collider other)
// {
```

```
    //      textF.SetActive(false);
    // }
    // void Update()
    // {
    //      if(Input.GetKey("x")){
    //          MoveObjectsToTaget();
    //      }
    // }
    public void MoveObjectToSafe()
    {
        Player.transform.position = safeTarget.position;
    }
    public void MoveObjectToTaget()
    {
        // Vector3 currentPosition = Character.transform.position;
        Player.transform.position = Target.position;
        CanvaOff.SetActive(true);
        StaminaBar.instance.CurrentStamina += 500;
    }
}
```

OpenDoor

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class OpenDoor : MonoBehaviour
{
    public GameObject door;
    public AudioSource doorOpen;
```

```
    public AudioSource doorClose;
    void Start()
    {
        StartCoroutine(DoorOpening());
    }
    IEnumerator DoorOpening()
    {
        yield return new WaitForSeconds(0.3f);
        doorOpen.Play();
        door.GetComponent<Animator>().Play("Door");
        yield return new WaitForSeconds(5.5f);
        doorClose.Play();
        yield return new WaitForSeconds(1.5f);
        SceneManager.UnloadScene("DoorScene");
    }
}
```

OpenLockDoor

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class OpenLockDoor : MonoBehaviour
{
    public GameObject door;
    public AudioSource doorOpen;
    public AudioSource doorClose;
    void Start()
    {
        StartCoroutine(DoorOpening());
```

```
    }
    IEnumerator DoorOpening()
    {
        yield return new WaitForSeconds(2.0f);
        doorOpen.Play();
        door.GetComponent<Animator>().Play("Door");
        yield return new WaitForSeconds(6.0f);
        doorClose.Play();
        yield return new WaitForSeconds(1.5f);
        SceneManager.UnloadScene("OpenDoorScene");
    }
}
```

BossScript

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
using UnityEngine.SceneManagement;
public class BossScript : MonoBehaviour
{
    public GameObject Enemy;
    public NavMeshAgent navmeshEnemy;
    public Transform player;
    public GameObject hitBox;
    public GameObject blood;
    public GameObject SlainUI;
    public AudioSource DeathS;
    public AudioSource SlainS;
    public bool getDamge;
```

```
public bool attackPlayer;
public bool enemyDeath = false;
public float MaxEnemyHealth;
public float CurrentEnemyHealth;
public float FromDamage;
public static BossScript instance;
private void Awake()
{
    instance = this;
}
void Start()
{
   CurrentEnemyHealth = MaxEnemyHealth;
   SlainUI.SetActive(false);
   DeathS.enabled = false;
   SlainS.enabled = false;
}
void Update()
{
    navmeshEnemy.SetDestination(player.position);
    if(enemyDeath == false){
        if(getDamge == true){
            PauseAi();
            Debug.Log("EnemyHurt");
            Enemy.GetComponent<Animator>().Play("Enemy_GetDamage");
            blood.SetActive(true);
            CurrentEnemyHealth -= WeaponControl.instance.attackDamage;
            Invoke("ResumeAi",1.5f);
        }
```

```
        if(attackPlayer == false){
            if(getDamge == false){
                Enemy.GetComponent<Animator>().Play("Enemy_Walk");
                blood.SetActive(false);
            }
        }
    }
    if(CurrentEnemyHealth <= 0){
        enemyDeath = true;
    }
    if(enemyDeath == true){
        DeathS.enabled = true;
        Invoke("beenSlain",2.0f);
        Invoke("DeathEnd",10.0f);
        Debug.Log("EnemyDeath");
        PauseAi();
        Enemy.GetComponent<Animator>().Play("Enemy_Death");
        GetComponent<Rigidbody>().isKinematic = true;
        GetComponent<BoxCollider>().enabled = false;
        // Destroy(gameObject,3.0f);
    }
}
public void OnTriggerStay(Collider other)
{
    if(enemyDeath == false){
        if(getDamge == false){
            if(other.GetComponent<Collider>().tag == "Player"){
                attackPlayer = true;
                Enemy.GetComponent<Animator>().Play("Enemy_Attack");
```

```
                PauseAi();
                Invoke("delayattack",0.5f);
                Invoke("delayGetDamage",1.0f);
                Invoke("delayhitbox",2.0f);
                Invoke("ResumeAi",2.5f);
            }
        }
        if(other.GetComponent<Collider>().tag == "HitBox"){
            if(WeaponControl.instance.AttackEnemy == true){
                getDamge = true;
            }
            if(WeaponControl.instance.AttackEnemy == false){
                getDamge = false;
            }
        }
    }
}
public void delayattack()
{
    HealthBar.instance.GetDamage = true;
    hitBox.SetActive(false);
}
public void delayhitbox()
{
    hitBox.SetActive(true);
    attackPlayer = false;
}
public void delayGetDamage()
{
```

```
        HealthBar.instance.GetDamage = false;
    }
    public void PauseAi()
    {
        navmeshEnemy.isStopped = true;
    }
    public void ResumeAi()
    {
        navmeshEnemy.isStopped = false;
    }
    public void DeathEnd()
    {
         SceneManager.LoadScene("CutSceneEnd");
    }
    public void beenSlain()
    {
        SlainUI.SetActive(true);
        SlainS.enabled = true;
    }
}
```

EnemyScript

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
public class EnemyScript : MonoBehaviour
{
    public GameObject Enemy;
    public NavMeshAgent navmeshEnemy;
```

```
public Transform player;
public GameObject hitBox;
public GameObject blood;
// public AudioSource EnemyS;
public AudioSource EnDeathS;
public bool getDamge;
public bool attackPlayer;
public bool enemyDeath = false;
public float MaxEnemyHealth;
public float CurrentEnemyHealth;
public float FromDamage;
public static EnemyScript instance;
private void Awake()
{
    instance = this;
}
void Start()
{
   CurrentEnemyHealth = MaxEnemyHealth;
   EnDeathS.enabled = false;
}
void Update()
{
    // Invoke("enemySound",7.0f);
    navmeshEnemy.SetDestination(player.position);
    if(enemyDeath == false){
        if(getDamge == true){
            PauseAi();
            Debug.Log("EnemyHurt");
```

```
            Enemy.GetComponent<Animator>().Play("Enemy_GetDamage");
            blood.SetActive(true);
            CurrentEnemyHealth -= WeaponControl.instance.attackDamage;
            Invoke("ResumeAi",1.5f);
        }
        if(attackPlayer == false){
            if(getDamge == false){
                Enemy.GetComponent<Animator>().Play("Enemy_Walk");
                blood.SetActive(false);
            }
        }
    }
    if(CurrentEnemyHealth <= 0){
        enemyDeath = true;
    }
    if(enemyDeath == true){
        EnDeathS.enabled = true;
        Debug.Log("EnemyDeath");
        PauseAi();
        Enemy.GetComponent<Animator>().Play("Enemy_Death");
        GetComponent<Rigidbody>().isKinematic = true;
        GetComponent<BoxCollider>().enabled = false;
        // EnemyS.enabled = false;
        // Destroy(gameObject,3.0f);
    }
}
public void OnTriggerStay(Collider other)
{
    if(enemyDeath == false){
```

```
        if(getDamge == false){
            if(other.GetComponent<Collider>().tag == "Player"){
                attackPlayer = true;
                Enemy.GetComponent<Animator>().Play("Enemy_Attack");
                PauseAi();
                Invoke("delayattack",0.5f);
                Invoke("delayGetDamage",1.0f);
                Invoke("delayhitbox",2.0f);
                Invoke("ResumeAi",2.5f);
            }
        }
        if(other.GetComponent<Collider>().tag == "HitBox"){
            if(WeaponControl.instance.AttackEnemy == true){
                getDamge = true;
            }
            if(WeaponControl.instance.AttackEnemy == false){
                getDamge = false;
            }
        }
    }
}
public void delayattack()
{
    HealthBar.instance.GetDamage = true;
    hitBox.SetActive(false);
}
public void delayhitbox()
{
    hitBox.SetActive(true);
```

```
        attackPlayer = false;
    }
    public void delayGetDamage()
    {
        HealthBar.instance.GetDamage = false;
    }
    public void PauseAi()
    {
        navmeshEnemy.isStopped = true;
    }
    public void ResumeAi()
    {
        navmeshEnemy.isStopped = false;
    }
    // public void enemySound()
    // {
    //     EnemyS.enabled = true;
    // }
}
```

FollowScript

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class FollowScript : MonoBehaviour
{
    public Transform targetObj;
    public int moveSpeed;
    void Update()
    {
```

```
        transform.position = Vector3.MoveTowards(this.transform.position,
targetObj.position, moveSpeed * Time.deltaTime);
    }
}
```

CutScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class CutScene : MonoBehaviour
{
    public string SceneName;
    public float times;
    void Update()
    {
        Invoke("changeScene",times);
    }
    public void changeScene()
    {
        SceneManager.LoadScene(SceneName);
    }
}
```

MainMenu

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MainMenu : MonoBehaviour
{
```

```
    public void PlayGame()
    {
        SceneManager.LoadScene("CutScene1");
    }
    public void ExitGame()
    {
        Application.Quit();
    }
}
```

ItemParameterSo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
namespace Inventory.Model
{
[CreateAssetMenu]
public class ItemParameterSo : ScriptableObject
{
    [field: SerializeField]
    public string ParameterName {get; private set;}
}
}
```

CharacterStatHealthModifierSo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
[CreateAssetMenu]
public class CharacterStatHealthModifierSo : CharacterStatModifierSo
{
```

```
    public static CharacterStatHealthModifierSo instance;
    private void Awake()
    {
        instance = this;
    }
    public override void AffectCharacter(GameObject character, float val)
    {
        Debug.Log("heal");
        HealthBar.instance.Heal = true;
        // HealthBar health = character.GetComponent<HealthBar>();
        // if(health != null){
        //     health.AddHealth(val);
        // }
    }
}
```

CharacterStatHealthModifierSo1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
[CreateAssetMenu]
public class CharacterStatHealthModifierSo1 : CharacterStatModifierSo
{
    public static CharacterStatHealthModifierSo1 instance;
    private void Awake()
    {
        instance = this;
    }
    public override void AffectCharacter(GameObject character, float val)
    {
```

```
        Debug.Log("heal1");
        HealthBar.instance.Heal1 = true;
        // HealthBar health = character.GetComponent<HealthBar>();
        // if(health != null){
        //     health.AddHealth(val);
        // }
    }
}
```

CharacterStatModifierSo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public abstract class CharacterStatModifierSo : ScriptableObject
{
    public abstract void AffectCharacter(GameObject character, float val);
}
```

EdibleItemSo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
namespace Inventory.Model
{
[CreateAssetMenu]
public class EdibleItemSo : ItemSo, IDestroyableItem, IItemAction
{
    [SerializeField]
    private List<ModifierData> modifiersData = new List<ModifierData>();
    public string ActionName => "Use";
```

```
    [field: SerializeField]
    public AudioClip actionSFX {get; private set;}
    public bool PerformAction(GameObject character, List<ItemParameter> itemState
= null)
    {
        foreach (ModifierData data in modifiersData)
        {
            data.statModifier.AffectCharacter(character, data.value);
        }
        return true;
    }
}
public interface IDestroyableItem
{
}
public interface IItemAction
{
    public string ActionName {get;}
    public AudioClip actionSFX {get;}
    bool PerformAction(GameObject character, List<ItemParameter> itemState);
}
[Serializable]
public class ModifierData
{
    public CharacterStatModifierSo statModifier;
    public float value;
    public static ModifierData instance;
    private void Awake()
    {
```

```
        instance = this;
    }
}
}
```

EquippableItemSo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
namespace Inventory.Model
{
[CreateAssetMenu]
public class EquippableItemSo : ItemSo, IDestroyableItem, IItemAction
{
    public string ActionName => "Equip";
    [field: SerializeField]
    public AudioClip actionSFX {get; private set;}
    public bool PerformAction(GameObject character, List<ItemParameter> itemState
= null)
    {
        AgentWeapon weaponSystem = character.GetComponent<AgentWeapon>();
        if(weaponSystem != null){
            weaponSystem.SetWeapon(this, itemState == null ? DefaultParametersList :
itemState);
            return true;
        }
        return false;
    }
}
}
```

InventorySo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using System.Linq;
namespace Inventory.Model
{
[CreateAssetMenu]
public class InventorySo : ScriptableObject
{
   [SerializeField] private List<inventoryItem> inventoryItems;
   [field: SerializeField] public int Size {get; private set; } = 10;
    public event Action<Dictionary<int, inventoryItem>> OnInventoryUpdated;
    public void Initialize()
    {
        inventoryItems = new List<inventoryItem>();
        for(int i = 0; i < Size; i++){
            inventoryItems.Add(inventoryItem.GetEmptyItem());
        }
    }
    public int AddItem(ItemSo item, int quantity, List<ItemParameter> itemState =
null)
    {
        if(item.IsStackable == false){
            for(int i = 0; i < inventoryItems.Count; i++){
                while(quantity > 0 && IsinventoryFull() == false){
                    quantity -= AddItemToFirstFreeSlot(item, 1, itemState);
                }
```

```
                InformAboutChange();
                return quantity;
                // if(inventoryItems[i].IsEmpty){
                //     inventoryItems[i] = new inventoryItem
                //     {
                //         item = item,
                //         quantity = quantity
                //     };
                //     return;
                // }
            }
        }
        quantity = AddStackableItem(item, quantity);
        InformAboutChange();
        return quantity;
    }
    public int AddItemToFirstFreeSlot(ItemSo item, int quantity, List<ItemParameter>
itemState = null)
    {
        inventoryItem newItem = new inventoryItem
        {
            item = item,
            quantity = quantity,
            itemState = new List<ItemParameter>(itemState == null ?
item.DefaultParametersList : itemState)
        };
        for(int i = 0; i < inventoryItems.Count; i++){
            if(inventoryItems[i].IsEmpty){
                inventoryItems[i] = newItem;
```

```
                return quantity;
            }
        }
        return 0;
    }
    public bool IsinventoryFull()
        => inventoryItems.Where(item => item.IsEmpty).Any() == false;
    public int AddStackableItem(ItemSo item, int quantity)
    {
        for(int i = 0; i < inventoryItems.Count; i++){
            if(inventoryItems[i].IsEmpty){
                continue;
            }
            if(inventoryItems[i].item.ID == item.ID){
                int amountPossibleToTake = inventoryItems[i].item.MaxStackSize -
inventoryItems[i].quantity;
                if(quantity > amountPossibleToTake){
                    inventoryItems[i] =
inventoryItems[i].ChangeQuantity(inventoryItems[i].item.MaxStackSize);
                    quantity -= amountPossibleToTake;
                }
                else{
                    inventoryItems[i] =
inventoryItems[i].ChangeQuantity(inventoryItems[i].quantity + quantity);
                    InformAboutChange();
                    return 0;
                }
            }
        }
```

```
        while(quantity > 0 && IsinventoryFull() == false){
            int newQuantity = Mathf.Clamp(quantity, 0, item.MaxStackSize);
            quantity -= newQuantity;
            AddItemToFirstFreeSlot(item, newQuantity);
        }
        return quantity;
    }
    public Dictionary<int, inventoryItem> GetCurrentInventoryState()
    {
        Dictionary<int, inventoryItem> returnValue = new Dictionary<int,
inventoryItem>();
        for(int i = 0; i < inventoryItems.Count; i++){
            if(inventoryItems[i].IsEmpty){
                continue;
            }
            returnValue[i] = inventoryItems[i];
        }
        return returnValue;
    }
    public inventoryItem GetItemAt(int itemIndex)
    {
        return inventoryItems[itemIndex];
    }
    public void AddItem(inventoryItem item)
    {
        AddItem(item.item, item.quantity);
    }
    public void RemoveItem(int itemIndex, int amount)
    {
```

```
        if(inventoryItems.Count > itemIndex){
            // if(inventoryItems[itemIndex].IsEmpty){
            //     return;
            // }
            int reminder = inventoryItems[itemIndex].quantity - amount;
            if(reminder <= 0){
                inventoryItems[itemIndex] = inventoryItem.GetEmptyItem();
            }
            else{
                inventoryItems[itemIndex] =
inventoryItems[itemIndex].ChangeQuantity(reminder);
            }
            InformAboutChange();
        }
    }
    public void SwapItems(int itemIndex_1, int itemIndex_2)
    {
        inventoryItem item1 = inventoryItems[itemIndex_1];
        inventoryItems[itemIndex_1] = inventoryItems[itemIndex_2];
        inventoryItems[itemIndex_2] = item1;
        InformAboutChange();
    }
    public void InformAboutChange()
    {
        OnInventoryUpdated?.Invoke(GetCurrentInventoryState());
    }
}
[Serializable]
public struct inventoryItem
```

```
{
    public int quantity;
    public ItemSo item;
    public List<ItemParameter> itemState;
    public bool IsEmpty => item == null;
    public inventoryItem ChangeQuantity(int newQuantity)
    {
        return new inventoryItem
        {
            item = this.item,
            quantity = newQuantity,
            itemState = new List<ItemParameter>(this.itemState)
        };
    }
    public static inventoryItem GetEmptyItem()
        => new inventoryItem
        {
            item = null,
            quantity = 0,
            itemState = new List<ItemParameter>()
        };
}
}
```

ItemSo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
namespace Inventory.Model
```

```
{
public abstract class ItemSo : ScriptableObject
{
    public int MyProp {get; set;}
   [field: SerializeField] public bool IsStackable {get; set;}
   public int ID => GetInstanceID();
   [field: SerializeField] public int MaxStackSize {get; set;} = 1;
   [field: SerializeField] public string name {get; set;}
   [field: SerializeField]
   [field: TextArea] public string description {get; set;}
   [field: SerializeField] public Sprite ItemImage {get; set;}

   [field: SerializeField] public List<ItemParameter> DefaultParametersList {get; set;}

}
[Serializable]
public struct ItemParameter : IEquatable<ItemParameter>
{
    public ItemParameterSo itemParameter;
    public float value;
    public bool Equals(ItemParameter other)
    {
        return other.itemParameter == itemParameter;
    }
}
}
```

KeyItemSo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
namespace Inventory.Model
{
[CreateAssetMenu]
// public class KeyItemSo : ScriptableObject
// {
//     [field: SerializeField]
//     public bool IsStackable {get; set;}
//     public int ID => GetInstanceID();
//     [field: SerializeField]
//     public int MaxStackSize {get; set;} = 1;
//     [field: SerializeField]
//     public string Name {get; set;}
//     [field: SerializeField]
//     [field: TextArea]
//     public string description {get; set;}
//     [field: SerializeField]
//     public Sprite ItemImage {get; set;}
// }
public class KeyItemSo : ItemSo, IDestroyableItem, IItemAction
{
    public string ActionName => "KeyItem";
    [field: SerializeField]
    public AudioClip actionSFX {get; private set;}
    public bool PerformAction(GameObject character, List<ItemParameter> itemState
= null)
```

```
    {
        AgentWeapon weaponSystem = character.GetComponent<AgentWeapon>();
        if(weaponSystem != null){
            // weaponSystem.SetWeapon(this, itemState == null ? DefaultParametersList
: itemState);
            return true;
        }
        return false;
    }
}
}
```

Item

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Inventory.Model;
public class Item : MonoBehaviour
{
    [field: SerializeField] public ItemSo inventoryItem {get; private set;}
    [field: SerializeField] public int Quantity {get; set;} = 1;
    [SerializeField] public AudioSource PickupSound;
    [SerializeField] public float duration = 0.3f;
    // public GameObject Txtpickup;
    public static Item instance;
    internal void DestroyItem()
    {
        GetComponent<Collider>().enabled = false;
        StartCoroutine(AnimateItemPickup());
    }
```

```
    public void Awake()
    {
        instance = this;
    }
    public void OnTriggerEnter(Collider other)
    {
        // if(other.tag == "Player"){
            // Txtpickup.SetActive(true);
        // }
    }
    public void OnTriggerExit(Collider other)
    {
        // Txtpickup.SetActive(false);
    }
    public IEnumerator AnimateItemPickup()
    {
        yield return new WaitForSeconds(0.5f);
        PickupSound.Play();
        Vector3 startScale = transform.localScale;
        Vector3 endScale = Vector3.zero;
        float currentTime = 0;
        while (currentTime < duration){
            currentTime += Time.deltaTime;
            transform.localScale = Vector3.Lerp(startScale, endScale, currentTime /
duration);
            yield return null;
        }
        transform.localScale = endScale;
        Destroy(gameObject);
```

```
    }
}
```

PickupSystem

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Inventory.Model;
public class PickupSystem : MonoBehaviour
{
   [SerializeField] public InventorySo InventoryData;
//     public GameObject Txtpickup;
   [SerializeField] public bool IsPickup ;
     public GameObject inventoryFull;
     public static PickupSystem instance;
     public void Awake()
     {
         instance = this;
     }
     public void OnTriggerStay(Collider other)
     {
         // Item.instance.Txtpickup.SetActive(false);
         if(other.gameObject.tag == "Item"){
             if(Input.GetKey(KeyCode.F)){
                 IsPickup = true;
                 Item item = other.GetComponent<Item>();
                 if(item != null){
                     int reminder = InventoryData.AddItem(item.inventoryItem,
item.Quantity);
                     if(reminder == 0){
```

```
                    item.DestroyItem();
                }
                else{
                    item.Quantity = reminder;
                    inventoryFull.SetActive(true);
                    Invoke("DelayInvFull",3.0f);
                }
            }
        }
    }
    public void DelayInvFull()
    {
        inventoryFull.SetActive(false);
    }
}
```

Continue

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class Continue : MonoBehaviour
{
    public static Continue instance;
    private void Awake()
    {
        instance = this;
    }
    public GameObject notpass;
```

```
    void Start()
    {
        GetComponent<UnityEngine.UI.Button>().onClick.AddListener(ContinueBtn);
    }
    public void ContinueBtn()
    {
        if(PuzzleSys.instance.Pass == true){
            Debug.Log("Con");
            SceneManager.UnloadScene("PuzzleScene");
            PuzzleScreen.instance.CanvaOff.SetActive(true);
            PuzzleScreen.instance.MoveObjectToTarget1();
        }
        if(PuzzleSys.instance.Pass == false){
            Debug.Log("NOtCon");
            notpass.SetActive(true);
            Invoke("npdel",1.0f);
        }
    }
    public void npdel()
    {
        notpass.SetActive(false);
    }
}
```

DragDrop

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class DragDrop : MonoBehaviour
{
```

```
Vector3 offset;
public string destinationTag = "DropArea";
void OnMouseDown()
{
    offset = transform.position - MouseWorldPosition();
    transform.GetComponent<Collider>().enabled = false;
}
void OnMouseDrag()
{
    transform.position = MouseWorldPosition() + offset;
}
void OnMouseUp()
{
    var rayOrigin = Camera.main.transform.position;
    var rayDirection = MouseWorldPosition() - Camera.main.transform.position;
    RaycastHit hitInfo;
    if(Physics.Raycast(rayOrigin, rayDirection, out hitInfo)){
        if(hitInfo.transform.tag == destinationTag){
            transform.position = hitInfo.transform.position;
        }
    }
    transform.GetComponent<Collider>().enabled = true;
}
Vector3 MouseWorldPosition()
{
    var mouseScreenPos = Input.mousePosition;
    mouseScreenPos.z = Camera.main.WorldToScreenPoint(transform.position).z;
    return Camera.main.ScreenToWorldPoint(mouseScreenPos);
}
```

}

ExitPuzzleScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class ExitPuzzleScene : MonoBehaviour
{
    public static ExitPuzzleScene instance;
    private void Awake()
    {
        instance = this;
    }
    void Start()
    {
        GetComponent<UnityEngine.UI.Button>().onClick.AddListener(ExitBtn);
    }
    public void ExitBtn()
    {
        Debug.Log("dsa");
        SceneManager.UnloadScene("PuzzleScene");
        PuzzleScreen.instance.CanvaOff.SetActive(true);
        PuzzleScreen.instance.MoveObjectToTarget();
    }
    // void Update()
    // {
    //     if(!SceneManager.GetSceneByName("PuzzleScene").isLoaded){
    //         Debug.Log("Exit");
    //     }
```

```
    // }
}
```

KeyPuzzleSystem

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class KeyPuzzleSystem : MonoBehaviour
{
    public GameObject puzzleP1;
    public bool keyItemPuzzle1 = false;
    public GameObject puzzleP2;
    public bool keyItemPuzzle2 = false;
    public GameObject puzzleP4;
    public bool keyItemPuzzle4 = false;
    public GameObject puzzleP6;
    public bool keyItemPuzzle6 = false;
    public GameObject puzzleP9;
    public bool keyItemPuzzle9 = false;
    public static KeyPuzzleSystem instance;
    private void Awake()
    {
        instance = this;
    }
    void Update()
    {
        if(puzzleP1 == null){
            keyItemPuzzle1 = true;
        }
        if(puzzleP2 == null){
```

```
            keyItemPuzzle2 = true;
        }
        if(puzzleP4 == null){
            keyItemPuzzle4 = true;
        }
        if(puzzleP6 == null){
            keyItemPuzzle6 = true;
        }
        if(puzzleP9 == null){
            keyItemPuzzle9 = true;
        }
    }
}
```

PuzzleScreen

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class PuzzleScreen : MonoBehaviour
{
    public Transform safeTarget;
    public Transform Target;
    public Transform Target1;
    public GameObject Player;
    // public GameObject textF;
    public GameObject CanvaOff;
    public static PuzzleScreen instance;
    private void Awake()
    {
```

```
    instance = this;
}
public void OnTriggerStay(Collider other)
{
    // textF.SetActive(true);
    if(HealthBar.instance.isDeath == false){
        if(other.gameObject.tag == "Player"){
            if(Input.GetKey("f")){
                CanvaOff.SetActive(false);
                SceneManager.LoadScene("PuzzleScene",LoadSceneMode.Additive);
                MoveObjectToSafe();
                // Invoke("MoveObjectToTarget",7.5f);
            }
        }
    }
}
public void MoveObjectToSafe()
{
    Player.transform.position = safeTarget.position;
}
public void MoveObjectToTarget()
{
    Player.transform.position = Target.position;
    // CanvaOff.SetActive(true);
    // StaminaBar.instance.CurrentStamina += 500;
}
public void MoveObjectToTarget1()
{
    Player.transform.position = Target1.position;
```

```
        // CanvaOff.SetActive(true);
        // StaminaBar.instance.CurrentStamina += 500;
    }
}
```

PuzzleSys

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class PuzzleSys : MonoBehaviour
{
    public GameObject puzzle1;
    public GameObject puzzle2;
    public GameObject puzzle4;
    public GameObject puzzle6;
    public GameObject puzzle9;
    public Transform target1;
    public Transform target2;
    public Transform target4;
    public Transform target6;
    public Transform target9;
    public bool Pass = false;
    private bool exitPuzzle = false;
    public static PuzzleSys instance;
    private void Awake()
    {
        instance = this;
    }
    void Start(){
```

```
    if(KeyPuzzleSystem.instance.keyItemPuzzle1 == true){
        puzzle1.SetActive(true);
    }
    if(KeyPuzzleSystem.instance.keyItemPuzzle2 == true){
        puzzle2.SetActive(true);
    }
    if(KeyPuzzleSystem.instance.keyItemPuzzle4 == true){
        puzzle4.SetActive(true);
    }
    if(KeyPuzzleSystem.instance.keyItemPuzzle6 == true){
        puzzle6.SetActive(true);
    }
    if(KeyPuzzleSystem.instance.keyItemPuzzle9 == true){
        puzzle9.SetActive(true);
    }
}
void Update()
{
    if(puzzle1.transform.position == target1.position &&
        puzzle2.transform.position == target2.position &&
        puzzle4.transform.position == target4.position &&
        puzzle6.transform.position == target6.position &&
        puzzle9.transform.position == target9.position
        ){
        Pass = true;
    }
    else{
        Pass = false;
    }
```

```
    }
}
```

SoundTrigger

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class SoundTrigger : MonoBehaviour
{
    public GameObject trigger;
    public GameObject model;
    public AudioSource soundFx;
    void Start()
    {
        soundFx.enabled = false;
        model.SetActive(false);
    }
    public void OnTriggerStay(Collider other)
    {
        if(other.gameObject.tag == "Player"){
            model.SetActive(true);
            soundFx.enabled = true;
            trigger.SetActive(false);
            // Invoke("FadeModel",0.5f);
        }
    }
    public void FadeModel()
    {
        model.SetActive(false);
    }
```

```
}
```

Trigger

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Trigger : MonoBehaviour
{
    public GameObject trigger;
    public GameObject model;
    void Start()
    {
        model.SetActive(false);
    }
    public void OnTriggerStay(Collider other)
    {
        if(other.gameObject.tag == "Player"){
            model.SetActive(true);
            trigger.SetActive(false);
        }
    }
}
```

AgentWeapon

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using Inventory.Model;
public class AgentWeapon : MonoBehaviour
{
```

```
    public GameObject weaponInHand1;
    public GameObject weaponInHand2;
    public GameObject weaponInHand3;
    public GameObject weaponInHand4;
    [SerializeField]
    private EquippableItemSo weapon;
    [SerializeField]
    private InventorySo inventoryData;
    [SerializeField]
    private List<ItemParameter> parametersToModify, itemCurrentState;
    // void Update()
    // {
    //     EquippableItemSo.instance.weaponEquipable = weaponInHand;
    // }
    public void SetWeapon(EquippableItemSo weaponItemSo, List<ItemParameter>
itemState)
    {
        // if(weapon == baseweapon1){
        //     Debug.Log("Equiped");
        //     weaponInHand.gameObject.SetActive(true);
        // }
        // string weaponName = weapon.itemName;
        // Debug.Log("Weapon Name: " + weaponName);
        // if(weapon.Equals(baseweapon1)){
        // }
        if(weapon != null){
            inventoryData.AddItem(weapon, 1, itemCurrentState);
        }
        this.weapon = weaponItemSo;
```

```
this.itemCurrentState = new List<ItemParameter>(itemState);
ModifyParameters();
if(weapon.name == "StickWood"){
    Debug.Log("Equiped");
    weaponInHand1.gameObject.SetActive(true);
    weaponInHand2.gameObject.SetActive(false);
    weaponInHand3.gameObject.SetActive(false);
    weaponInHand4.gameObject.SetActive(false);
}
if(weapon.name == "Axe"){
    Debug.Log("Equiped");
    weaponInHand1.gameObject.SetActive(false);
    weaponInHand2.gameObject.SetActive(true);
    weaponInHand3.gameObject.SetActive(false);
    weaponInHand4.gameObject.SetActive(false);
}
if(weapon.name == "Pickaxe"){
    Debug.Log("Equiped");
    weaponInHand1.gameObject.SetActive(false);
    weaponInHand2.gameObject.SetActive(false);
    weaponInHand3.gameObject.SetActive(true);
    weaponInHand4.gameObject.SetActive(false);
}
if(weapon.name == "Sword"){
    Debug.Log("Equiped");
    weaponInHand1.gameObject.SetActive(false);
    weaponInHand2.gameObject.SetActive(false);
    weaponInHand3.gameObject.SetActive(false);
    weaponInHand4.gameObject.SetActive(true);
```

```
        }
    }
    private void ModifyParameters()
    {
        // Debug.Log("Equiped");
        foreach (var parameter in parametersToModify){
            if(itemCurrentState.Contains(parameter)){
                int index = itemCurrentState.IndexOf(parameter);
                float newValue = itemCurrentState[index].value + parameter.value;
                itemCurrentState[index] = new ItemParameter{
                    itemParameter = parameter.itemParameter,
                    value = newValue
                };
            }
      }
    }
}
```

InventoryController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Text;
using Inventory.UI;
using Inventory.Model;
namespace Inventory
{
public class InventoryController : MonoBehaviour
{
    public static InventoryController instance;
```

```
private void Awake()
{
    instance = this;
}
[SerializeField] private UIInventoryPage inventoryUI;
[SerializeField] private InventorySo inventoryData;
[SerializeField] private AudioClip dropClip;
[SerializeField] private AudioSource audioSource;
public GameObject inventoryFade;
public AudioSource inventoryOpen;
public AudioSource inventoryClose;
public bool isOpenInv = false;
public bool canClose = false;
public List<inventoryItem> initialItems = new List<inventoryItem>();
public void Start()
{
    PrepareUI();
    PrepareInventoryData();
}
public void PrepareInventoryData()
{
    inventoryData.Initialize();
    inventoryData.OnInventoryUpdated += UpdateInventoryUI;
    foreach(inventoryItem item in initialItems)
    {
        if(item.IsEmpty){
            continue;
        }
        inventoryData.AddItem(item);
```

```
        }
    }
    public void UpdateInventoryUI(Dictionary<int, inventoryItem> inventoryState)
    {
        inventoryUI.ResetAllItem();
        foreach(var item in inventoryState){
            inventoryUI.UpdateData(item.Key, item.Value.item.ItemImage,
item.Value.quantity);
        }
    }
    public void PrepareUI()
    {
        inventoryUI.InitializeInventoryUI(inventoryData.Size);
        this.inventoryUI.OnDescriptionRequested += HandleDescriptionRequest;
        this.inventoryUI.OnSwapItems += HandleSwapItems;
        this.inventoryUI.OnStartDragging += HandleDragging;
        this.inventoryUI.OnItemActionRequested += HandleItemActionRequest;
    }
    public void HandleDescriptionRequest(int itemIndex)
    {
        inventoryItem inventoryItem = inventoryData.GetItemAt(itemIndex);
        if(inventoryItem.IsEmpty){
            inventoryUI.ResetSelection();
            return;
        }
        ItemSo item = inventoryItem.item;
        string description = PrepareDescription(inventoryItem);
        inventoryUI.UpdateDescription(itemIndex, item.ItemImage, item.name,
description);
```

```
    }
    private string PrepareDescription(inventoryItem inventoryItem)
    {
        StringBuilder sb = new StringBuilder();
        sb.Append(inventoryItem.item.description);
        sb.AppendLine();
        for(int i = 0; i < inventoryItem.itemState.Count; i++){
            sb.Append($"{inventoryItem.itemState[i].itemParameter.ParameterName}" +
            $": {inventoryItem.itemState[i].value} / " +
            $"{inventoryItem.item.DefaultParametersList[i].value}");
        }
        return sb.ToString();
    }
    public void HandleSwapItems(int itemIndex_1, int itemIndex_2)
    {
        inventoryData.SwapItems(itemIndex_1, itemIndex_2);
    }
    public void HandleDragging(int itemIndex)
    {
        inventoryItem inventoryItem = inventoryData.GetItemAt(itemIndex);
        if(inventoryItem.IsEmpty){
            return;
        }
        inventoryUI.CreateDraggedItem(inventoryItem.item.ItemImage,
inventoryItem.quantity);
    }
    public void HandleItemActionRequest(int itemIndex)
    {
        inventoryItem inventoryItem = inventoryData.GetItemAt(itemIndex);
```

```
        if(inventoryItem.IsEmpty){
            return;
        }
        IItemAction itemAction = inventoryItem.item as IItemAction;
        if(itemAction != null){
            inventoryUI.ShowItemAction(itemIndex);
            inventoryUI.AddAction(itemAction.ActionName, () =>
PerformAction(itemIndex));
            Debug.Log("action");
        }
        IDestroyableItem destroyableItem = inventoryItem.item as IDestroyableItem;
        if(destroyableItem != null){
            inventoryUI.AddAction("Drop",()=> DropItem(itemIndex,
inventoryItem.quantity));
        }
    }
    public void DropItem(int itemIndex, int quantity)
    {
        inventoryData.RemoveItem(itemIndex, quantity);
        inventoryUI.ResetSelection();
        audioSource.PlayOneShot(dropClip);
    }
    public void PerformAction(int itemIndex)
    {
        inventoryItem inventoryItem = inventoryData.GetItemAt(itemIndex);
        if(inventoryItem.IsEmpty){
            return;
        }
        IDestroyableItem destroyableItem = inventoryItem.item as IDestroyableItem;
```

```
    if(destroyableItem != null){
        inventoryData.RemoveItem(itemIndex,1);
    }
    IItemAction itemAction = inventoryItem.item as IItemAction;
    if(itemAction != null){                         //ใช้Item
        itemAction.PerformAction(gameObject, inventoryItem.itemState);
        audioSource.PlayOneShot(itemAction.actionSFX);
        // Debug.Log("equip");
        if(inventoryData.GetItemAt(itemIndex).IsEmpty){
            inventoryUI.ResetSelection();
        }
    }
}
void Update()
{
    if(HealthBar.instance.isDeath == false){
        if(Input.GetKey(KeyCode.Tab) && isOpenInv == false && canClose == false){
            isOpenInv = true;
            inventoryOpen.Play();
            inventoryFade.SetActive(true);
            StartCoroutine(InvControl());
        }
        if(Input.GetKey(KeyCode.Tab) && isOpenInv == true && canClose == true){
            isOpenInv = false;
            inventoryClose.Play();
            inventoryFade.SetActive(true);
            StartCoroutine(InvControl());
        }
    }
```

```
    }
    IEnumerator InvControl(){
        yield return new WaitForSeconds(0.25f);
        if(isOpenInv == true){
            inventoryUI.Show();
            foreach(var item in inventoryData.GetCurrentInventoryState()){
                inventoryUI.UpdateData(item.Key,
                    item.Value.item.ItemImage,
                    item.Value.quantity
                    );
            }
            yield return new WaitForSeconds(0.25f);
            inventoryFade.SetActive(false);
        }
        else{
            inventoryUI.Hide();
        }
        yield return new WaitForSeconds(0.25f);
        inventoryFade.SetActive(false);
        if(isOpenInv == true){
            canClose = true;
        }
        else{
            canClose = false;
        }
    }
}
}
```

ItemActionPanel

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
namespace Inventory.UI
{
public class ItemActionPanel : MonoBehaviour
{
   [SerializeField]
   private GameObject buttonPrefab;


    public void AddButton(string name, Action onClickAction)
    {
        GameObject button = Instantiate(buttonPrefab, transform);
        button.GetComponent<Button>().onClick.AddListener(() => onClickAction());
        button.GetComponentInChildren<TMPro.TMP_Text>().text = name;
    }
    internal void Toggle(bool val)
    {
        if(val == true){
            RemoveOldButtons();
        }
        gameObject.SetActive(val);
    }
    public void RemoveOldButtons()
    {
        foreach(Transform transformChildObject in transform){
```

```
            Destroy(transformChildObject.gameObject);
        }
    }
}
}
```

MouseFollower

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Inventory.Model;
using Inventory.UI;
public class MouseFollower : MonoBehaviour
{
   [SerializeField] private Canvas canvas;
   [SerializeField] private UIInventoryItem item;
    public void Awake()
    {
        canvas = transform.root.GetComponent<Canvas>();
        item = GetComponentInChildren<UIInventoryItem>();
    }
    public void SetData(Sprite sprite, int quantity)
    {
        item.SetData(sprite, quantity);
    }
    void Update()
    {
        Vector2 position;
        RectTransformUtility.ScreenPointToLocalPointInRectangle(
```

```
            (RectTransform)canvas.transform,
            Input.mousePosition,
            canvas.worldCamera,
            out position
                );
        transform.position = canvas.transform.TransformPoint(position);
    }
    public void Toggle(bool val)
    {
        Debug.Log($"Item toggled {val}");
        gameObject.SetActive(val);
    }
}
```

UIInventoryDescription

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
namespace Inventory.UI
{
public class UIInventoryDescription : MonoBehaviour
{
    [SerializeField] private Image itemImage;
    [SerializeField] private TMP_Text title;
    [SerializeField] private TMP_Text description;
    [SerializeField] private TMP_Text titleDurability;
    [SerializeField] private TMP_Text durability;
    [SerializeField] private Slider durabilityBar;
```

```
    public void Awake()
    {
        ResetDescription();
    }
    public void ResetDescription()
    {
        this.itemImage.gameObject.SetActive(false);
        this.title.text = "";
        this.description.text = "";
        this.titleDurability.text = "";
        this.durability.text = "";
        this.durabilityBar.gameObject.SetActive(false);
    }
    public void SetDescription(Sprite sprite, string itemName, string itemDescription)
    {
        this.itemImage.gameObject.SetActive(true);
        this.itemImage.sprite = sprite;
        this.title.text = itemName;
        this.description.text = itemDescription;
    }
}
}
```

UIInventoryItem

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

```
using UnityEngine.EventSystems;
namespace Inventory.UI
{
    [DisallowMultipleComponent]
public class UIInventoryItem : MonoBehaviour,
    IPointerClickHandler, IBeginDragHandler, IEndDragHandler, IDropHandler,
IDragHandler
{
        [SerializeField]
        private Image itemImage;
        [SerializeField]
        private TMP_Text quantityTxt;
        [SerializeField]
        private Image borderImage;
        public event Action<UIInventoryItem> OnItemClicked,
            OnItemDroppedOn, OnItemBeginDrag, OnItemEndDrag,
            OnRightMouseBtnClick;
        private bool empty = true;
        public void Awake()
        {
            if(itemImage == null){
                return;
            }
            if(itemImage != null){
                ResetData();
                Deselect();
            }
        }
        public void ResetData()
```

```
{
    if(itemImage == null){
        return;
    }
    if(itemImage != null){
       itemImage.gameObject.SetActive(false);
        empty = true;
    }
}
public void Deselect()
{
    if(itemImage == null){
        return;
    }
    if(itemImage != null){
        borderImage.enabled = false;
    }
}
public void SetData(Sprite sprite, int quantity)
{
    if(itemImage == null){
        return;
    }
    if(itemImage != null){
        itemImage.gameObject.SetActive(true);
        itemImage.sprite = sprite;
        quantityTxt.text = quantity + "";
        empty = false;
    }
```

```
}
public void Select()
{
    if(itemImage == null){
        return;
    }
    if(itemImage != null){
       borderImage.enabled = true;
    }
}
public void OnPointerClick(PointerEventData pointerData)
{
    if (pointerData.button == PointerEventData.InputButton.Right)
    {
        OnRightMouseBtnClick?.Invoke(this);
    }
    else
    {
        OnItemClicked?.Invoke(this);
    }
}
public void OnEndDrag(PointerEventData eventData)
{
    OnItemEndDrag?.Invoke(this);
}
public void OnBeginDrag(PointerEventData eventData)
{
    if (empty)
        return;
```

```
            OnItemBeginDrag?.Invoke(this);
        }
        public void OnDrop(PointerEventData eventData)
        {
            OnItemDroppedOn?.Invoke(this);
        }
        public void OnDrag(PointerEventData eventData)
        {
        }
    }
}
```

UIInventoryPage

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
namespace Inventory.UI
{
public class UIInventoryPage : MonoBehaviour
{
    [SerializeField] private UIInventoryItem itemPrefab;
    [SerializeField] private RectTransform contentPanel;
    [SerializeField] private UIInventoryDescription itemDescription;
    [SerializeField] private MouseFollower mouseFollower;
    List<UIInventoryItem> listofUIItems = new List<UIInventoryItem>();
    // public Sprite image, image2;
    // public int quantity;
    // public string title, description;
```

```
    public event Action<int> OnDescriptionRequested,
        OnItemActionRequested,
        OnStartDragging;
    public event Action<int, int> OnSwapItems;
    private int currentlyDraggedItemIndex = -1;
    [SerializeField] private ItemActionPanel actionPanel;
    private void Awake()
    {
        // Hide();
        mouseFollower.Toggle(false);
        itemDescription.ResetDescription();
    }
    public void InitializeInventoryUI(int inventorysize)
    {
        for(int i = 0; i< inventorysize; i++){
            UIInventoryItem uiItem = Instantiate(itemPrefab, Vector3.zero,
Quaternion.identity);
            uiItem.transform.SetParent(contentPanel);
            listofUIItems.Add(uiItem);
            uiItem.OnItemClicked += HandleItemSelection;
            uiItem.OnItemBeginDrag += HandleBeginDrag;
            uiItem.OnItemDroppedOn += HandleSwap;
            uiItem.OnItemEndDrag += HandleEndDrag;
            uiItem.OnRightMouseBtnClick += HandleShowItemActions;
        }
    }
    public void ResetAllItem()
    {
        foreach(var item in listofUIItems){
```

```
            item.ResetData();
            item.Deselect();
        }
    }
    public void UpdateData(int itemIndex, Sprite itemImage, int itemQuantity)
    {
        if(listofUIItems.Count > itemIndex)
        {
            listofUIItems[itemIndex].SetData(itemImage, itemQuantity);
        }
    }
    private void HandleItemSelection(UIInventoryItem inventoryItemUI)
    {
        // itemDescription.SetDescription(image, title, description);
        int index = listofUIItems.IndexOf(inventoryItemUI);
        if(index == -1){
            return;
        }
        // listofUIItems[0].Select();
        OnDescriptionRequested?.Invoke(index);
    }
    private void HandleBeginDrag(UIInventoryItem inventoryItemUI)
    {
        int index = listofUIItems.IndexOf(inventoryItemUI);
        if(index == -1){
            return;
        }
        currentlyDraggedItemIndex = index;
        HandleItemSelection(inventoryItemUI);
```

```
    OnStartDragging?.Invoke(index);
    // mouseFollower.Toggle(true);
    // mouseFollower.SetData(index == 0? image : image2, quantity);
}
public void CreateDraggedItem(Sprite sprite, int quantity)
{
    mouseFollower.Toggle(true);
    mouseFollower.SetData(sprite, quantity);
}
private void HandleSwap(UIInventoryItem inventoryItemUI)
{
    int index = listofUIItems.IndexOf(inventoryItemUI);
    if(index == -1){
        // mouseFollower.Toggle(false);
        // currentlyDraggedItemIndex = -1;
        return;
    }
    // listofUIItems[currentlyDraggedItemIndex]
    //     .SetData(index == 0 ? image : image2, quantity);
    // listofUIItems[index]
    //     .SetData(currentlyDraggedItemIndex == 0 ? image : image2, quantity);
    // mouseFollower.Toggle(false);
    // currentlyDraggedItemIndex = -1;
    OnSwapItems?.Invoke(currentlyDraggedItemIndex, index);
    HandleItemSelection(inventoryItemUI);
}
private void HandleEndDrag(UIInventoryItem inventoryItemUI)
{
    ResetDraggedItem();
```

```
    // mouseFollower.Toggle(false);
}
private void HandleShowItemActions(UIInventoryItem inventoryItemUI)
{
    int index = listofUIItems.IndexOf(inventoryItemUI);
    if(index == -1){
        return;
    }
    OnItemActionRequested?.Invoke(index);
}
private void ResetDraggedItem()
{
    mouseFollower.Toggle(false);
    currentlyDraggedItemIndex = -1;
}
public void Show()
{
    gameObject.SetActive(true);
    // itemDescription.ResetDescription();
    ResetSelection();
    // listofUIItems[0].SetData(image, quantity);
    // listofUIItems[1].SetData(image2, quantity);
}
public void Hide()
{
    actionPanel.Toggle(false);
    gameObject.SetActive(false);
    ResetDraggedItem();
}
```

```
    public void ResetSelection()
    {
        itemDescription.ResetDescription();
        DeselectAllItem();
    }
    public void AddAction(string actionName, Action performAction)
    {
        actionPanel.AddButton(actionName, performAction);
    }
    public void ShowItemAction(int itemIndex)
    {
        actionPanel.Toggle(true);
        actionPanel.transform.position = listofUIItems[itemIndex].transform.position;
    }
    public void DeselectAllItem()
    {
        foreach(UIInventoryItem item in listofUIItems)
        {
            item.Deselect();
        }
        actionPanel.Toggle(false);
    }
    public void UpdateDescription(int itemIndex, Sprite itemImage, string name, string
description)
    {
        itemDescription.SetDescription(itemImage, name, description);
        DeselectAllItem();
        listofUIItems[itemIndex].Select();
    }
```

```
}
}
```

DeathScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class DeathScene : MonoBehaviour
{
    public static DeathScene instance;
    private void Awake()
    {
        instance = this;
    }
    public GameObject imageBG;
    public GameObject deathFade;
    public GameObject youdied;
    public GameObject btnRe;
    public AudioSource udied;
    void Start()
    {
        deathFade.SetActive(false);
        imageBG.SetActive(false);
        youdied.SetActive(false);
        btnRe.SetActive(false);
        udied.enabled = false;
    }
    void Update()
```

```
    {
        if(HealthBar.instance.isDeath == true){
            StartCoroutine(DeathScreen());
        }
    }
    IEnumerator DeathScreen()
    {
        yield return new WaitForSeconds(3.0f);
        udied.enabled = true;
        deathFade.SetActive(true);
        yield return new WaitForSeconds(0.5f);
        imageBG.SetActive(true);
        youdied.SetActive(true);
        yield return new WaitForSeconds(1.5f);
        btnRe.SetActive(true);
    }
    public void RestartBtn(){
        SceneManager.LoadScene("MainScene");
    }
}
```

restartBtn

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class restartBtn : MonoBehaviour
{
    void Start()
    {
```

```
        GetComponent<UnityEngine.UI.Button>().onClick.AddListener(RestartBtn);
    }
    public void RestartBtn(){
        // SceneManager.UnloadScene("MainScene");
        SceneManager.LoadScene("MainScene",LoadSceneMode.Single);
    }
}
```

EndScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class EndScene : MonoBehaviour
{
    public GameObject ContinueBtn;
    void Start()
    {
        ContinueBtn.SetActive(false);
    }
    void Update()
    {
        Invoke("conTrue",24.0f);
    }
    public void conTrue()
    {
        ContinueBtn.SetActive(true);
    }
    public void BtnCon()
    {
```

```
        SceneManager.LoadScene("MainMenuScene");
    }
}
```

PauseGame

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class PauseGame : MonoBehaviour
{
    public bool isPause = false;
    public GameObject pauseMenu;
    void Update()
    {
        if(Input.GetKeyDown(KeyCode.Escape)){
            if(isPause == true){
                Resumegame();
            }
            else{
                Pausegame();
            }
        }
    }
    public void Pausegame()
    {
        Time.timeScale = 0f;
        isPause = true;
        pauseMenu.SetActive(true);
```

```
    }
    public void Resumegame()
    {
        Time.timeScale = 1f;
        isPause = false;
        pauseMenu.SetActive(false);
    }
    public void returntoMainMenu()
    {
        Resumegame();
        SceneManager.LoadScene("MainMenuScene");
    }
}
```

ประวัติผู้จัดทำ

ประวัติผู้จัดทำโครงการ

ชื่อ นายพงษ์พัฒน์ สีแก

เกิดเมื่อวันที่ 22 เดือน กุมภาพันธ์ พ.ศ 2547

ที่อยู่ 136/1 หมู่7 ต.หนองบัว อ.เมือง จ.จันทบุรี

หมายเลขโทรศัพท์ 094-6474846

E-Mail : rewpongpat12@gmail.com

การศึกษา

ชั้นประถมศึกษา จากโรงเรียนวัดหนองบัว ปทุมาภิพัฒน์

ชั้นมัธยมศึกษาตอนต้น จากโรงเรียนเบญจมานุสรณ์

ชั้นประกาศนียบัตรวิชาชีพ จากวิทยาลัยเทคนิคจันทบุรี

ขณะนี้กำลังศึกษาอยู่ในชั้นประกาศนียบัตรวิชาชีพชั้นสูงที่วิทยาลัยเทคนิคจันทบุรี

ประวัติผู้จัดทำโครงการ

ชื่อ นายนิรุทธ์ อินวัฒนา

เกิดเมื่อวันที่ 8 เดือน มิถุนายน พ.ศ 2546

ที่อยู่ 42 หมู่ที่1 ต.หนองบัว อ.เมือง จ.จันทบุรี

หมายเลขโทรศัพท์ 095-1155020

E-Mail : familykitt4@gmail.com

การศึกษา

ชั้นประถมศึกษา จากโรงเรียนบ้านสระใหญ่

ชั้นมัธยมศึกษาตอนต้น จากโรงเรียนบ้านสระใหญ่

ชั้นประกาศนียบัตรวิชาชีพ จากวิทยาลัยเทคนิคจันทบุรี

ขณะนี้กำลังศึกษาอยู่ในชั้นประกาศนียบัตรวิชาชีพชั้นสูงที่วิทยาลัยเทคนิคจันทบุรี

แบบเสนออนุมัติโครงการ

เกม Eerie Woods

ผู้เสนอโครงการ

นายนิรุทธ์ อินวัฒนา

นายพงษ์พัฒน์ สีแก

แบบเสนออนุมัติโครงงานนี้เป็นส่วนหนึ่งของวิชาโครงงาน

รหัสวิชา 3012-8501 ประเภทวิชา อุตสหกรรม สาขาวิชา เทคโนโลยีคอมพิวเตอร์

ภาคเรียนที่ 2 ปีการศึกษา 2566

วิทยาลัยเทคนิคจันทบุรี สถาบันการอาชีวะศึกษาภาคตะวันออก

ใบเสนอโครงการ

1. ชื่อโครงการ

เกม Eerie Woods

2. ประเภทโครงการ

สิ่งประดิษฐ์ด้านวัตกรรมซอฟต์แวร์

3. ผู้สนอโครงการ

นายพงษ์พัฒน์ สีแก

นายนิรุทธ์ อินวัฒนา

4. ครูที่ปรึกษา

นางสาวประภาพร บันเทา

นายฉัตรพฤกษ์ สีทิม

5. ข้อมูลทั่วไป

ลักษณะทั่วไป

☑เป็นผลงานโครงการที่คิดค้นขึ้นใหม่

☐เป็นผลงานโครงการที่พัฒนาหรือปรับปรุงเพิ่มเติมจากของเดิม

6. หลักการและเหตุผล

ในปัจจุบันเทคโนโลยีและเกมเป็นสิ่งที่ทุกคนสามารถเข้าถึงได้ง่ายไม่ว่าจะเป็นเด็ก เยาวชน หรือผู้ใหญ่เนื่องจากเกมนั้นหาเล่นได้ง่ายและให้ความสนุกเพลินกับผู้เล่นผู้จัดทำโครงงานได้เห็นถึงความสำคัญในจุดนี้จึงมีความคิดที่จะสร้างเกมที่ให้ความสนุกและความตื่นเต้นกับผู้เล่นเกมกับผู้เล่นด้วย

เนื่องจากในปัจจุบันเกมนั้นเข้าถึงได้ง่ายจึงมีเกมที่เนื่อหาเกมไม่ดี เนื้อหารุนแรงและไม่เหมาะสมกับเด็กและเยาวชนและอาจทำให้เกิดการใช้ความรุนแรงได้ผู้จัดทำโครงงานจึงมีความคิดที่จะสร้างเกมที่สร้างสรรค์และช่วยพัฒนาทักษะการเล่นเกมของผู้เล่นเนื่องจากผู้จัดทำโครงงานมีความสนใจและชื่นชอบในการเล่นเกมและได้มีการศึกษาโค้ดและการใช้งานโปรแกรมในการสร้างเกมผู้จัดทำจึงมีจุดมุ่งหมายในการสร้างเกมนี้ขึ้นมา

จากเหตุผลข้างต้นจึงเป็นสาเหตุให้ผู้จัดทำโครงงานมีความต้องการที่จะสร้างเกม Eerie Woods ที่ให้ความสนุกสนานตื่นเต้นแก่ผู้เล่นโดยใช้โปรแกรม Unity 3D และภาษา C# ขึ้นมาเพื่อ

ต้องการให้ผู้เล่นที่ได้พัฒนาทักษะการเล่นเกมของผู้เล่นและได้ประโยชน์จากการเล่นเกมมากกว่าการใช้ความรุนแรง

7. วัตถุประสงค์ของการวิจัย

7.1 เพื่อสร้างเกม Eerie Woods

7.2 เพื่อศึกษาการใช้โปรแกรม Unity 3D และ ภาษา C#

7.3 เพื่อหาประสิทธิภาพของเกม Eerie Woods

8. ทฤษฎี/หลักการที่นำมาใช้ในการจัดทำโครงการ

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

นายนัฐวุฒิ หวลละห้อย นายภูวนาท แสงฤทธิ์ และนายรัฐพล สุขสว่างการจัดทำโครงการครั้งนี้มีวัตถุประสงค์เพื่อสร้างเกมสามมิติผีในฝัน ด้วยโปรแกรม Unity (Ghost in Dream 3D Game) ศึกษาความพึงพอใจของนักเรียนประกาศนียบัตรวิชาชีพ สาขาวิชาคอมพิวเตอร์ธุรกิจวิทยาลัยเทคนิคระยองที่มีต่อเกมสามมิติ ในฝันด้วยโปรแกรม Unity (Ghost in Dream 3D Game) และเผยแพร่เกมสามมิติผีในฝัน ด้วยโปรแกรม Unity (Ghost in Dream 3D Game) ผ่านโครงการประกวดโครงการวิชาชีพ ขมรมวิชาชีพคอมพิวเตอร์ธุรกิจ กลุ่มตัวอย่างที่ใช้คือนักเรียนประกาศนียบัตรวิชาชีพ สาขาคอมพิวเตอร์ธุรกิจ วิทยาลัยเทคนิคระยอง 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

อภินันท์ ภิรมย์ ภาคย์ สธนเสาวภาคย์ และภัทราวัลย์ คำปลิว งานวิจัยนี้มีวัตถุประสงค์เพื่อ 1) พัฒนาเกมสามมิติเรื่อง เกมทางออกอยู่ไหน 2) ศึกษาความพึงพอใจของนักศึกษา ที่ มีต่อเกมสามมิติ เรื่อง เกมทางออกอยู่ไหน กลุ่มเป้าหมาย นักศึกษาสาขาวิศวกรรมคอมพิวเตอร์และสาขาเครือข่ายคอมพิวเตอร์คณะวิศวกรรมศาสตร์ มหาวิทยาลัยราชภัฏมหาสารคาม จ านวน 80 คน เครื่องมือที่ใช้ในการวิจัย ได้แก่ แบบสอบถามความพึงพอใจของนักศึกษา สถิติที่ใช้ในการวิจัย คือ ค่าเฉลี่ย และ

ส่วนเบี่ยงเบนมาตรฐานผลการวิจัยสรุปได้ดังนี้ 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

9. ขั้นตอนการทำงานของผลงานโครงการ

9.1 เตรียมข้อมูลและอุปกรณ์ในการสร้าง

9.2 ปรึกษาครูที่ปรึกษาโครงการ

9.3 เสนอขออนุมัติโครงการ

9.4 ดำเนินงานตามโครงการ

9.5 จัดทำเอกสาร

9.6 แสดงผลโครงการ

9.7 ประเมินผลโครงการ

10. ขอบเขตของผลงานโครงการ

10.1 ประชากรและกลุ่มตัวอย่าง

10.1.1 ประชากร คือ ผู้ทดลองเล่นเกม Eerie Woods

10.1.2 กลุ่มตัวอย่าง คือ นักศึกษาแผนกเทคโนโลยีคอมพิวเตอร์ระดับ ปวส.2/1

10.2 ตัวแปรที่ศึกษา

10.2.1 ตัวแปรต้น เกม Eerie Woods

10.2.2 ตัวแปรตาม ความพึงพอใจของผู้ทดลองเล่นเกม Eerie Woods ความเสถียรของเกมและประสิทธิภาพของเกม

10.3 เครื่องมือที่ใช้ในโครงการ

10.3.1 แบบสอบความพึงพอใจของผู้เล่นเกม Eerie Woods และประสิทธิภาพของตัวเกม Eerie Woods

10.4 วิธีการเก็บรวบรวมข้อมูล

10.4.1 กำหนดกลุ่มประชากรและกลุ่มตัวอย่าง

10.4.2 ศึกษาข้อมูลเบื้องต้น

10.4.3 ออกแบบและสร้างเกม Eerie Woods

10.4.4 นำไปทดลองเล่นจริง

10.4.5 นำแบบสอบถามความพึงพอใจให้กลุ่มผู้ทดลองได้ลงความคิดเห็น

10.4.6 เก็บรวบรวมแบบสอบถามทั้งหมดและนำมาประมวลผลให้สมบูรณ์ต่อไป

10.5 สถิติที่ใช้ในการวิเคราะห์ข้อมูล

10.5.1 ค่าเฉลี่ย โดยใช้สูตร

ค่าเฉลี่ย

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

$\vec{x}$ = ค่าเฉลี่ยของคะแนน

$\sum x$ = ผลรวมของคะแนน

N = จำนวน

10.5.2 ส่วนเบี่ยงเบนมาตรฐานโดยใช้สูตร

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

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

S.D แทน ส่วนเบี่ยงเบนมาตรฐาน

X แทน คะแนนแต่ละคน

N แทน จำนวน

11. ประโยชน์ที่คาดว่าจะได้รับ

11.1 ได้เกม Eerie Woods

11.2 ได้ทักษะการใช้โปรแกรม Unity 3D และ ภาษา C#

11.3 ประสิทธิภาพของเกม Eerie Woods มีความแม่นยำร้อยละ 80

12. ระยะเวลาที่ใช้ในการจัดทำโครงการ

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																	◄	─ ►

13. งบประมาณและทรัพยากร

ที่	รายการพัสดุ	จำนวน	หน่วย	ราคาบาท
1	ค่าจัดทำเอกสาร	1	2000	2000
รวม				2000 .-

ลงชื่อ.....
(นายพงษ์พัฒน์ สีแก)
ผู้รับผิดชอบโครงงาน

ลงชื่อ.....
(นายนิรุทธ์ อินวัฒนา)
ผู้รับผิดชอบโครงงาน

ความคิดเห็นของครูที่ปรึกษา/ครูผู้สอนโครงการ/คณะกรรมการผู้ตรวจสอบโครงการ

ลงชื่อ.....

(นายฉัตรพฤกษ์ สีทิม)

ครูผู้สอนโครงการ

ความคิดเห็นของหัวหน้าสาขาเทคโนโลยีคอมพิวเตอร์

โครงการนี้เห็นสมควร ☐ อนุมัติ ☐ ไม่อนุมัติ

ลงชื่อ.....

(นายศัตราวุธ ศรีชาติ)

หัวหน้าสาขาวิชาเทคโนโลยีคอมพิวเตอร์

วันที่...../...../2566

ความคิดเห็นของรองผู้อำนวยการฝ่ายวิชาการ

โครงการนี้เห็นสมควร ☐ อนุมัติ ☐ ไม่อนุมัติ

ลงชื่อ.....

(นายชูชาติ รูปงาม)

รองผู้อำนวยการฝ่ายวิชาการ

วันที่...../...../2566

ความคิดเห็นของผู้อำนวยการวิทยาลัยเทคนิคจันทบุรี

โครงการนี้เห็นสมควร ☐ อนุมัติ ☐ ไม่อนุมัติ

ลงชื่อ.....

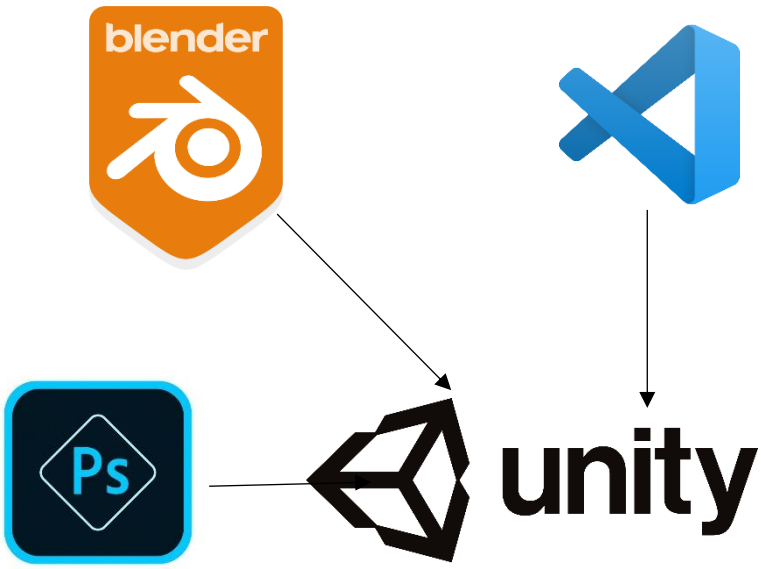
(นายกมล เรียงไธสง)

ผู้อำนวยการวิทยาลัยเทคนิคจันทบุรี

วันที่...../...../2566

ใบเสนอโครงงาน

เกม Eerie Woods



ลักษณะเกม

เกมแนว Survival Horror เนื้อเรื่อง มีเด็กกลุ่มหนึ่งไปเที่ยวในป่าช่วงวันหยุดยาว แล้วเกิดอุบัติเหตุทางรถยนตร์ ตัวเอกสลบพอตื่นขึ้นมา กับพบว่าเพื่อนๆหายแล้ว เห็นรอยเท้าเดินเขาไปในป่า ตัวเอกจึงต้องออกตามหาเพื่อนๆ ซึ่งตัวเอกไม่รู้เลยว่าในป่านั้นมีสิ่งน่ากลัวรอพวกเขาอยู่ ตัวเอกจึงต้องเอาตัวรอดจากอันตรายในป่าแล้วหาจากรอดออกป่า

ขอบเขตของโปรแกรม

1. เป็นเกม 3D
2. มุมมอง Fix camera
3. เกมแนว Survival Horror
4. ใช้โปรแกรม Unity ในการทำ
5. มีการแก้ไขปริศนา มี 3 ปริศนา
6. ใช้ภาษา C# ในการพัฒนา
7. ใช้ photoshop ในการทำฉาก
8. ใช้ blender ในการปั้นโมเดล
9. เปิดช่องเก็บของได้

10. กด W A S D ในการเดิน

11. กดคลิกซ้าย เพื่อโจมตี

12. กดคลิกขวา เพื่อเล็งอาวุธ

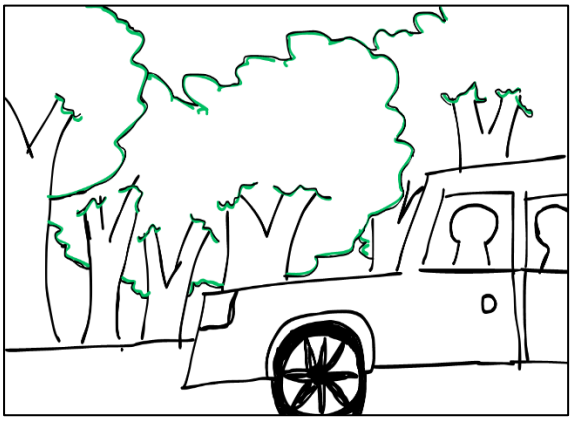
13. กด F เพื่อเก็บของ

14. กด TAB เพื่อแสดงที่เก็บของ

15. เล่นบน PC

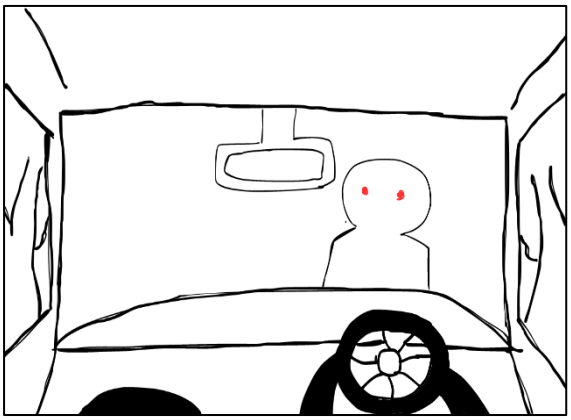
16. มี 1 ด่าน แต่หลายพื้นที่ เช่น ป่า,หมู่บ้าน,คฤหาสน์

Story Board



มีเด็กหนุ่ม2 คนไปเที่ยวในป่าช่วงวันหยุดยาว เพื่อจะไปตั้งแคมป์ ในขณะที่กำลังขับรถในตอนรุ่งเช้านั้น

ภาพที่1

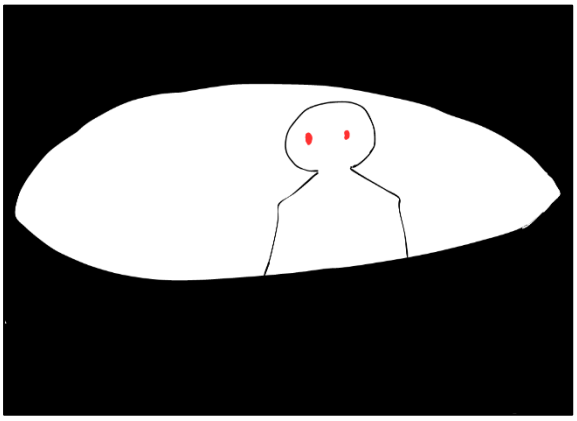


แต่มีคนตัดหน้ารถของทั้ง 2 คน ตัวเอกจึงตัดสินใจหักหลบคนที่ตัดหน้ารถทำให้รถเสียการทรงตัว

ภาพที่2

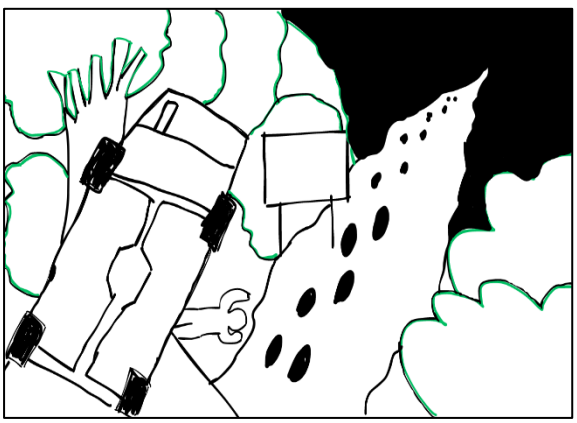
ภาพที่3

รถพุ่งไปชนกลับต้นไม้ข้างทาง ทำให้รถพลิกคว่ำ



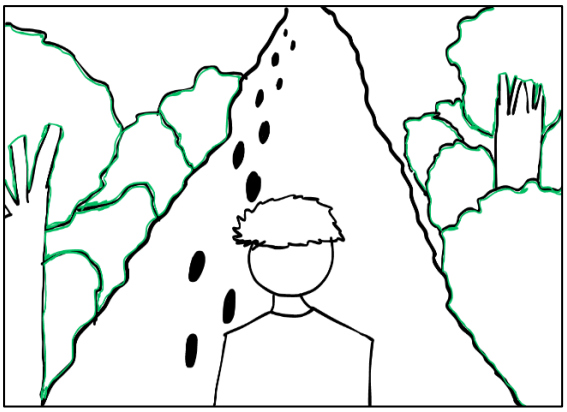
ภาพที่4

ตัวเอกหันไปหาเพื่อนที่เบาะข้างๆแต่กับไม่เห็นเพื่อนของตัวเอง ตัวเอกจึงพยายามคลานออกจากตัวรถ ตัวเอกจึงเห็นคน 1 คน ที่กำลังอุ้มเพื่อนของตัวเองอยู่ แต่ด้วยบาดแผลรถพลิกคว่ำทำให้ตัวเอกสลบไป



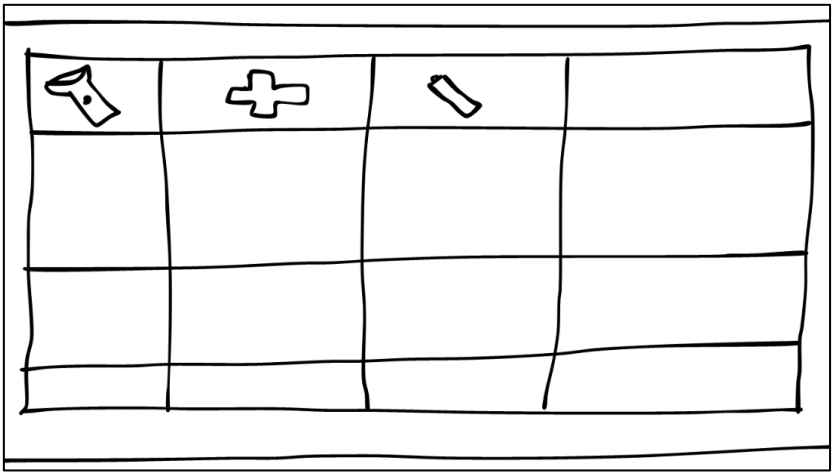
ภาพที่5

20 นาทีต่อมา ตัวเอกตื่นขึ้นแล้วพบกับรอยเท้าของคนเดินไปตามทางลึกเข้าไปในป่า

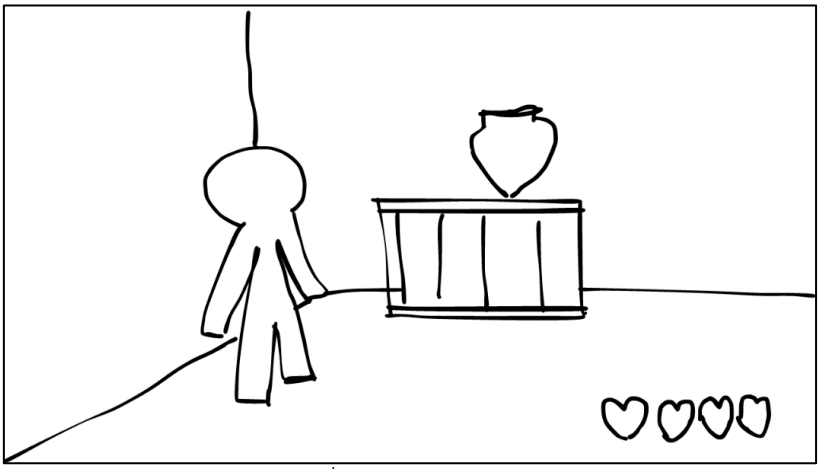


ตัวเอกจึงเดินตามรอยเท้าเข้าไปในป่าเพื่อตามหาเพื่อนของตัวเองและหาทางออกจากป่า

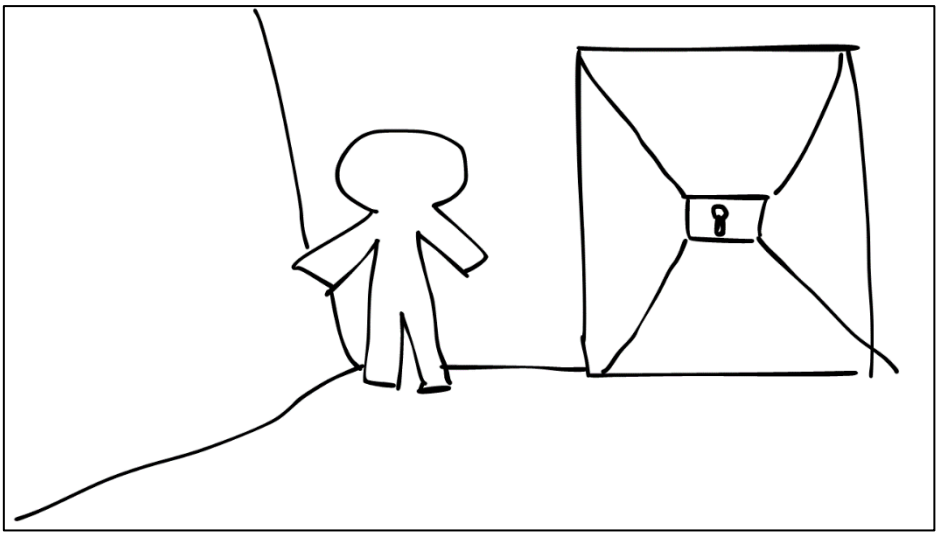
ภาพที่6



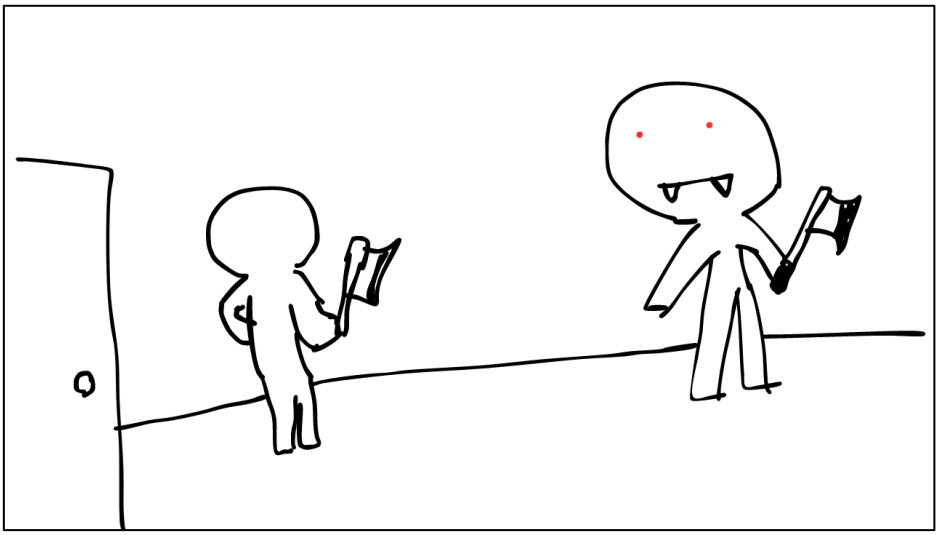
ภาพที่7 ช่องเก็บของ



ภาพที่8 มุมกล้องและฉาก



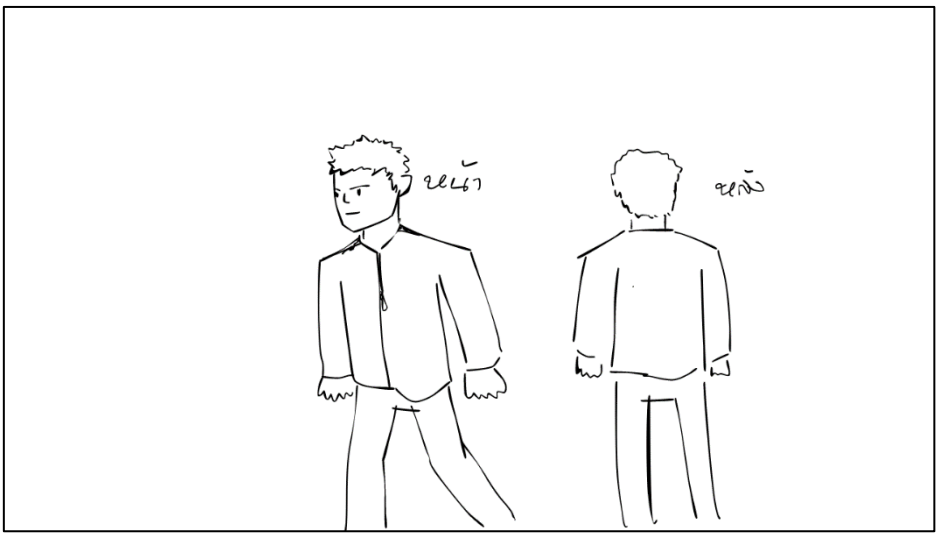
ภาพที่9 หาของมาเปิดประตู



ภาพที่10 ระบบการต่อสู้จะเป็นการเล็งและโจมตี

ภาพที่11 หน้า Main Menu

ภาพที่12 หน้า GAME OVER



ภาพที่13 การออกแบบตัวละคร

ใบรับรองอนุปริญญานิพนธ์
สาขาเทคโนโลยีคอมพิวเตอร์ วิทยาลัยเทคนิคจันทบุรี

หัวข้อวิจัย	:	Eerie Woods
ผู้ดำเนินการวิจัย	:	นายพงพัฒน์ สีแก
		นายนิรุทธ์ อินวัฒนา
ที่ปรึกษา	:	นายฉัตรพฤกษ์ สีทิม
		นางสาวประภาพร บันเทา
สาขาวิชา	:	เทคโนโลยีคอมพิวเตอร์
สาขางาน	:	คอมพิวเตอร์ระบบเครือข่าย
ปี พ.ศ.	:	2566

สาขาวิชาเทคโนโลยีคอมพิวเตอร์ ให้นับอนุปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรประกาศนียบัตรชีพขั้นสูง สาขาวิชาเทคโนโลยีคอมพิวเตอร์

.....หัวหน้าสาขาวิชาเทคโนโลยีคอมพิวเตอร์
(นายศัตราวุธ ศรีชาติ)

คณะกรรมการสอบโครงการ

.....ประธานกรรมการ
(นายฉัตรพฤกษ์ สีทิม)

.....กรรมการ
(นายปิยพงษ์ เกตุภูงา)

.....กรรมการ
(นางสาวประภาพร บันเทา)

แบบสอบถามความพึงพอใจ

ตอนที่ 1 ข้อมูลทั่วไปของผู้ตอบแบบสอบถาม

ชื่อ-นามสกุล : ..... ชั้น ..... แผนก .....

ตอนที่ 2 แบบประเมินความพึงพอใจของผู้ทดสอบเล่นเกม

ระดับความคิดเห็น

5 คะแนน	หมายถึง	ดีมาก
4 คะแนน	หมายถึง	ดี
3 คะแนน	หมายถึง	ปานกลาง
2 คะแนน	หมายถึง	พอใช้
1 คะแนน	หมายถึง	ปรับปรุง

หัวข้อ	ระดับความพึงพอใจ				
	5	4	3	2	1
1. ความสมบูรณ์ของตัวเกม					
2. ความละเอียดในแต่ละส่วนของพื้นที่ในเกม					
3. ความเหมาะสมของอุปสรรคในเกม					
4. ระยะเวลาในการเล่นเกม					
5. ความเหมาะสมและการใช้งาน					

ตอนที่ 3 ข้อคิดเห็น / ข้อเสนอแนะ

.....

.....

.....