โครงการสร้างเกม

Pastoral Pathways

นางสาวณัฐนันท์ จิรวัฒน์

นายอดิศักดิ์ พวงนะที

โครงการนี้เป็นส่วนหนึ่งของวิชาโครงการ รหัสวิชา 30128-8501

ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

ประเภทวิชา อุตสาหกรรม สาขา เทคโนโลยีคอมพิวเตอร์

ภาคเรียนที่ 2 ปีการศึกษา 2567

วิทยาลัยเทคนิคจันทบุรี สถาบันการอาชีวศึกษาภาคตะวันออก

โครงการสร้างเกม

Pastoral Pathways

นางสาวณัฐนันท์ จิรวัฒน์

นายอดิศักดิ์ พวงนะที

โครงการนี้เป็นส่วนหนึ่งของวิชาโครงการ รหัสวิชา 30128-8501

ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)

ประเภทวิชา อุตสาหกรรม สาขา เทคโนโลยีคอมพิวเตอร์

ภาคเรียนที่ 2 ปีการศึกษา 2567

วิทยาลัยเทคนิคจันทบุรี สถาบันการอาชีวศึกษาภาคตะวันออก

ใบรับรองโครงการวิทยาลัยเทคนิคจันทบุรี

หัวข้อโครงการ : Pastoral Pathways

โดย : นางสาวณัฐนันท์ จิรวัฒน์

: นายอดิศักดิ์ พวงนะที

ครูที่ปรึกษา : นายฉัตรพฤกษ์ สิทิม

: นางสาวประภาพร บันเทา

สาขาวิชา : เทคโนโลยีคอมพิวเตอร์ วิทยาลัยเทคนิคจันทบุรี

ปีการศึกษา : 2567

แผนกเทคโนโลยีคอมพิวเตอร์ ให้นับโครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรประกาศนียบัตรวิชาชีพขั้นสูง (ปวส.) สาขาเทคโนโลยีคอมพิวเตอร์

.....หัวหน้าสาขาวิชาเทคนิคคอมพิวเตอร์

(นายศัตราวุธ ศรีชาติ)

คณะกรรมการสอบโครงงาน

.....ประธานกรรมการ

(นายฉัตรพฤกษ์ สีทิม)

.....กรรมการ

(นางสาวปรีดาภรณ์ อึ้งนภารัตน์)

.....กรรมการ

(นางสาวประภาพร บันเทา)

.....กรรมการ

(นายอาทิตย์ ชิดชอบ)

.....กรรมการ

(นายปิยพงษ์ เกตุภูงา)

.....

(นางพัทธนันท์ ภูแก้ว)

รองผู้อำนวยการฝ่ายวิชาการ

วันที่...../...../2567

หัวข้อโครงการ : Pastoral Pathways

โดย : นางสาวณัฐนันท์ จิรวัฒน์

: นายอดิศักดิ์ พวงนะที

ครูที่ปรึกษา : นายฉัตรพฤกษ์ สิทิม

: นางสาวประภาพร บันเทา

สาขาวิชา : เทคโนโลยีคอมพิวเตอร์

ปีการศึกษา : 2567

บทคัดย่อ

ในปัจจุบัน เกมออนไลน์มีบทบาทสำคัญในด้านการสร้างความบันเทิงและพัฒนาทักษะในหลากหลายด้าน โดยเฉพาะอย่างยิ่งเกมแนวจำลองสถานการณ์ (Simulation) ซึ่งเป็นหนึ่งในประเภทเกมที่ได้รับความนิยมอย่างแพร่หลาย เกมแนวทำฟาร์ม (Farming Simulation) เป็นหนึ่งในเกมประเภทนี้ที่มุ่งเน้นให้ผู้เล่นได้สัมผัสประสบการณ์การบริหารจัดการฟาร์มเสมือนจริง ไม่ว่าจะเป็นการเพาะปลูกพืชเลี้ยงสัตว์ การจัดการทรัพยากร และการวางแผนการใช้ชีวิตในฟาร์ม

เกม Pastoral Pathways ไม่เพียงแต่ช่วยสร้างความเพลิดเพลินให้แก่ผู้เล่น แต่ยังส่งเสริมให้ผู้เล่นฝึกฝนทักษะการวางแผนและการแก้ไขปัญหาอย่างมีระบบ ตลอดจนการพัฒนาทักษะการบริหารจัดการทรัพยากรอย่างมีประสิทธิภาพ นอกจากนี้ เกมยังมีส่วนช่วยในการพัฒนาความคิดสร้างสรรค์ การคิดวิเคราะห์ และการตัดสินใจในสถานการณ์จำลองที่คล้ายคลึงกับชีวิตจริง

คำสำคัญ : เกมแนวจำลองสถานการณ์, การวางแผน, การบริหารฟาร์ม, Pastoral Pathways

Research Title : Pastoral Pathways
Researcher : Nattanan Jirawat
: Adisak Phoungnatee
Researcher Consultants : Chattapluek Seethim
: Prapaporn Bantao
Field : Technology Computer
Year : 2567

Abstract

In the present day, online games can play a significant role in providing entertainment and enhancing various cognitive and practical skills. In particular, simulation games are among the most widely popular genres, offering immersive and interactive experiences. Farming simulation games are a subset of this genre that allow players to engage in the management of a virtual farm, including growing crops, raising animals, managing resources, and planning daily farm activities.

The game Pastoral Pathways not only provides enjoyment and relaxation for players but also encourages them to develop strategic planning and problem-solving skills in a systematic and engaging manner. Additionally, it enhances resource management efficiency, fosters creativity, improves analytical thinking, and strengthens decision-making abilities in simulated scenarios that closely resemble real-life agricultural and economic challenges.

Keywords: Simulation games, planning, farm management, Pastoral Pathways

กิตติกรรมประกาศ

โครงงานเกม Pastoral Pathways ที่นักศึกษาได้ค้นคว้าและทดลอง ในครั้งนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยความกรุณา และความช่วยเหลือเป็นที่อย่างสูงยิ่งจาก นายฉัตรพฤกษ์ สีทิม และนางสาวประภาพร บันเทา ที่ได้กรุณาให้ความปรึกษาแนะนำและตรวจสอบ แก้ไขข้อบกพร่องทุกขั้นตอนของการจัดทำโครงการ Pastoral Pathways คณะผู้จัดทำโครงการขอขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้ขอขอบพระคุณผู้ที่มีอุปการะคุณทุกท่านที่ได้ให้ความช่วยเหลือในการจัดทำโครงงานครั้งนี้ จนสามารถประสบความสำเร็จลุล่วงไปได้ด้วยดี

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญภาพ

สารบัญภาพ(ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ในยุคปัจจุบัน เกมออนไลน์เข้ามามีบทบาทสำคัญทั้งในด้านความบันเทิงและการพัฒนาทักษะของผู้เล่นในหลายๆ ด้าน โดยเฉพาะเกมแนวจำลองสถานการณ์ (Simulation) ที่ให้ผู้เล่นได้สัมผัสประสบการณ์ที่หลากหลายและสมจริง ซึ่งหนึ่งในแนวเกมที่ได้รับความนิยมอย่างสูงคือเกมแนวทำฟาร์ม (Farming Simulation) เช่นเกม Pastoral Pathways ซึ่งเกมนี้มุ่งเน้นให้ผู้เล่นได้สัมผัสประสบการณ์การบริหารจัดการฟาร์มเสมือนจริง โดยจำลองสถานการณ์การเพาะปลูกพืช การเลี้ยงสัตว์ และการจัดการทรัพยากรที่มีอยู่ ผู้เล่นจะต้องเรียนรู้และพัฒนาทักษะในด้านการวางแผน การตัดสินใจ และการแก้ปัญหาอย่างมีระบบ ซึ่งเป็นทักษะสำคัญที่สามารถนำไปใช้ประโยชน์ในชีวิตจริงได้อย่างมีประสิทธิภาพ

Pastoral Pathways ยังมีจุดเด่นในด้านการส่งเสริมความรู้เกี่ยวกับการเกษตรกรรมอย่างยั่งยืน โดยผู้เล่นสามารถเรียนรู้เกี่ยวกับวงจรชีวิตของพืชผลและสัตว์เลี้ยงได้แบบละเอียด รวมถึงการใช้ทรัพยากรธรรมชาติอย่างมีคุณค่าในการจัดการฟาร์มเสมือน ผู้เล่นจะได้เห็นภาพชัดเจนว่าการทำฟาร์มที่ยั่งยืนนั้นมีลักษณะอย่างไร ไม่ว่าจะเป็นการใช้ปุ๋ยอินทรีย์ ลดการใช้สารเคมี หรือการบริหารจัดการน้ำและพลังงานอย่างเหมาะสม เพื่อไม่ให้เกิดการทำลายสิ่งแวดล้อมและสร้างสมดุลในระบบนิเวศ นอกจากนี้ ตัวเกมยังมีระบบที่ช่วยให้ผู้เล่นสามารถเชื่อมต่อและแลกเปลี่ยนความรู้กับผู้เล่นคนอื่น ๆ ผ่านทางชุมชนออนไลน์ ทำให้เกิดเครือข่ายของคนที่มีความสนใจในด้านการเกษตรกรรม ส่งเสริมการเรียนรู้และแบ่งปันแนวคิดที่เป็นประโยชน์

การเล่นเกม Pastoral Pathways ไม่ได้มีเพียงแค่ความสนุกเพลิดเพลินเท่านั้น แต่ยังช่วยเสริมสร้างทักษะทางด้านความคิดสร้างสรรค์และการวิเคราะห์ โดยเฉพาะในสถานการณ์ที่มีความคล้ายคลึงกับชีวิตจริง ผู้เล่นจะต้องตัดสินใจในเรื่องการใช้ทรัพยากรอย่างชาญฉลาด การวางแผนการเพาะปลูก การดูแลสัตว์เลี้ยง และการจัดการทรัพยากรอื่นๆ ในฟาร์ม ซึ่งการตัดสินใจเหล่านี้จะส่งผลต่อความสำเร็จและการพัฒนาของฟาร์มในระยะยาว ความท้าทายเหล่านี้ทำให้ผู้เล่นต้องคิดวิเคราะห์และวางแผนอย่างรอบคอบ ส่งเสริมให้เกิดการเรียนรู้ที่จะบริหารจัดการฟาร์มด้วยวิธีที่มีประสิทธิภาพและเป็นมิตรต่อสิ่งแวดล้อม นอกจากนี้ เกมยังนำเสนอการทำฟาร์มที่หลากหลาย ไม่ว่าจะเป็นการปลูกพืชชนิดต่างๆ ที่ต้องการการดูแลเฉพาะด้าน หรือการเลี้ยงสัตว์ที่มีเอกลักษณ์เฉพาะตัว

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อออกแบบและสร้างเกม Pastoral Pathways

1.2.2 เพื่อวัดผลประสิทธิภาพของเกม Pastoral Pathways

1.2.3 เพื่อวัดผลความพึงพอใจจากผู้ใช้หลังการเล่นเกม Pastoral Pathways

1.3 ขอบเขตของโครงการ

1.3.1 เป้าหมายของ โครงการนี้มีเป้าหมายหลักในการพัฒนาเกมฟาร์มที่มุ่งเน้นให้ผู้เล่นสามารถเรียนรู้และสนุกสนานไปกับการจัดการฟาร์มเสมือนจริง โดยจะสร้างประสบการณ์การเล่นที่สนุกสนานและมีคุณค่าทางการศึกษา สำหรับนักเรียน นักศึกษา และผู้ที่สนใจในการทำฟาร์มและการเกษตร

1.3.2 ฟังก์ชันหลักของเกม

1) การปลูกพืช : ผู้เล่นสามารถเลือกปลูกพืชหลายชนิด เช่น ผักและผลไม้ โดยต้องดูแลให้พืชเติบโต เพื่อสร้างผลผลิตที่ขายหรือใช้ในฟาร์ม

2) การเลี้ยงสัตว์ : ผู้เล่นสามารถเลี้ยงสัตว์ เช่น วัว หมู และไก่ โดยต้องดูแลสุขภาพและให้อาหารเพียงพอ เพื่อให้สัตว์ผลิตสินค้าที่มีค่า เช่น นมและไข่

3) การจัดการทรัพยากร : ผู้เล่นต้องบริหารจัดการทรัพยากร เช่น น้ำและปุ๋ย ให้เหมาะสมกับความต้องการของพืชและสัตว์ เพื่อเพิ่มผลผลิต

4) การพัฒนาฟาร์ม : ผู้เล่นสามารถขยายและปรับปรุงฟาร์ม โดยการสร้างสิ่งปลูกสร้างใหม่เพื่อเพิ่มประสิทธิภาพในการผลิต

5) ระบบเศรษฐกิจ : ในเกมผู้เล่นสามารถซื้อขายผลิตภัณฑ์จากฟาร์ม สร้างรายได้และลงทุนในการพัฒนาฟาร์มต่อไป

6) ระบบตัวละคร NPC : มี NPC ประมาณ 7 คนที่ช่วยในการซื้อขายและให้คำแนะนำ เพื่อพัฒนาฟาร์มอย่างมีประสิทธิภาพ

7) การตกแต่งฟาร์ม : ผู้เล่นสามารถตกแต่งฟาร์มตามต้องการ โดยเลือกใช้เครื่องมือตกแต่งต่าง ๆ เพื่อสร้างบรรยากาศที่สวยงาม

8) ระบบสภาพอากาศและฤดูกาล : เกมจำลองสภาพอากาศและฤดูกาลที่ส่งผลต่อการปลูกพืชและเลี้ยงสัตว์ ผู้เล่นต้องปรับตัวให้เหมาะสมกับสภาพแวดล้อมที่เปลี่ยนแปลง

1.3.3 เกมจะถูกพัฒนาให้สามารถเล่นได้บนแพลตฟอร์ม PC

1.3.4 เครื่องมือที่ใช้ในโครงงาน

1) เครื่องมือในการสร้างชิ้นงาน

1.1) โปรแกรม Unity

1.2) โปรแกรม Aseprite

2) เครื่องมือที่ใช้ในการวัดคุณภาพของเกม

2.1) แบบสอบถามความพึงพอใจ

3) เครื่องมือในการหาประสิทธิภาพของเกม

3.1) เครื่องมือตรวจจับบักใน Unity

3.2) Stack Overflow

1.3.5 วิธีการเก็บรวบรวมข้อมูล

1) แบบสอบถามความพึงพอใจของผู้เล่น

1.3.6 สถิติที่ใช้ในการวิเคราะห์ข้อมูล

1) ค่าเฉลี่ย โดยใช้สูตร

ค่าเฉลี่ย

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

$\vec{x}$ = ค่าเฉลี่ยของคะแนน

$\sum x$ = ผลรวมของคะแนน

N = จำนวน

2) ส่วนเบี่ยงเบนมาตรฐานโดยใช้สูตร

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

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

S.D. แทน ส่วนเบี่ยงเบนมาตรฐาน

X แทน คะแนนแต่ละคน

N แทน จำนวน

1.4 ขอบเขตการวิจัย

1.4.1 ใช้โปรแกรม Unity , Aseprite, Visual Studio Code ในการสร้างตัวละคร ระบบเกม

1.4.2 ตัวละครดำเนินเกมมีเพียง 1 ตัวละคร

1.4.3 ใช้ภาษา C# ในการเขียนโค้ดเกม

1.5 ข้อจำกัดของโครงการ

1.5.1 ไม่สามารถเล่นบนระบบปฏิบัติการอื่นนอกเหนือ Window และ Mac ได้

1.5.2 Unity ไม่รองรับตัวภาษาไทยทำให้ฟอนต์บางตัวเกิดข้อผิดพลาด

1.5.3 เกมเป็นระบบผู้เล่นคนเดียว ไม่สามารถเล่นหลายคนได้

1.6 สมมติฐานของการวิจัย

คะแนนจากการทดสอบประสิทธิภาพของเกมอยู่ในระดับ ดี และความพึงพอใจของผู้เล่นหลังเล่นเกมจบอยู่ในระดับที่ ดี

1.7 คำจำกัดความที่ใช้ในงานวิจัย

1.7.1 Unity คือ โปรแกรมที่มีความสามารถหลากหลายได้แก่สร้างเกม 2 มิติการสร้างเกม 3 มิติ การสร้าง AR, VR สามารถส่งแอพพลิเคชั่นได้ทั้งระบบ Windows, iOS และ Android

1.7.2 C# คือ ภาษาโปรแกรมแบบหลายโมเดล ที่ใช้ระบบชนิดข้อมูลแบบรัดกุม (strong typing) และ C# สนับสนุนการเขียนโปรแกรมในเชิงคำสั่ง การเขียนโปรแกรมในเชิงประกาศ การเขียนโปรแกรมเชิงฟังก์ชัน การเขียนโปรแกรมในเชิงกระบวนการ การเขียนโปรแกรมใน เชิงวัตถุ (แบบคลาส) และการเขียนโปรแกรมในเชิงส่วนประกอบ

1.7.3 Aseprite เป็นโปรแกรมในการสร้างภาพพิกเซลที่ให้มีคุณสร้างภาพเคลื่อนไหว 2 มิติสำหรับวิดีโอเกม สามารถเขียนสไปรท์โดยใช้เลเยอร์และเฟรม

1.7.4 Stack Overflow คือ เว็บสำหรับนักเขียนโปรแกรมและนักพัฒนา ที่มีไว้สำหรับแบ่งปันโค๊ด ถาม หรือตอบคำถามกับนักเขียนโปรแกรมจากทั่วโลก

1.8 ประโยชน์ที่คาดว่าจะได้รับ

1.8.1 ได้เกม Pastoral Pathways

1.8.2 ได้ประสิทธิภาพของเกม Pastoral Pathways ทำงานได้ดี

1.8.3 ความพึงพอใจจากผู้ใช้หลังการเล่นเกม Pastoral Pathways อยู่ในระดับ ดี

1.9 ระยะเวลาที่ใช้ในการจัดทำโครงการ

ตารางที่ 1.1 แสดงระยะเวลาดำเนินการ

ขั้นตอนการดำเนินงาน	**สัปดาห์**																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1. กำหนดโครงการ	◄	►																
2. ออกแบบเกม		◄	►															
3. ศึกษารายละเอียด				◄	►	◄	►											
4. ออกแบบระบบเกม						◄	►	◄	►	◄	►							
5. เริ่มทำเกม								◄	►	◄	►	◄	►	◄	►			
6. ทดสอบเกม															◄	►		
7. ประเมินและสรุปผล																◄	►	
8. จัดทำเอกสาร																	◄	►

1.10 งบประมาณที่ใช้ในการจัดทำโครงการ

ตารางที่ 1.2 แสดงงบประมาณและทรัพยากร

ที่	รายการวัสดุ	จำนวน	หน่วย	ราคา/บาท
1	ทำเล่ม	1	-	1,000
รวม				1,000

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

ในการศึกษาผลงานที่เกี่ยวข้องกับการพัฒนาเกมคอมพิวเตอร์ โดยใช้โปรแกรม Unity ซึ่งนับว่าเป็นองค์ประกอบหนึ่งที่จะสำคัญมาก และจะทำให้มีโครงการมีความสมบูรณ์ครบถ้วน คณะผู้จัดทำได้แบ่งเอกสารและทฤษฎีที่เกี่ยวข้องออกเป็นหัวข้อดังนี้

2.1 ความหมายของเกมคอมพิวเตอร์

2.2 หลักการพัฒนาซอฟต์แวร์

2.3 โปรแกรมที่ใช้งาน

2.4 งานวิจัยที่เกี่ยวข้อง

2.1 ความหมายของเกมคอมพิวเตอร์ (ที่มา : https://archive.lib.cmu.ac.th/full/T/2560/)

เกมคอมพิวเตอร์เป็นสื่อบันเทิงที่ได้รับความนิยมอย่างแพร่หลายในปัจจุบัน โดยเฉพาะในกลุ่มเด็ก วัยรุ่น และผู้ใหญ่ ซึ่งเป็นเพราะเกมคอมพิวเตอร์มีความสามารถในการสร้างปฏิสัมพันธ์ระหว่างผู้เล่นกับคอมพิวเตอร์ในลักษณะที่ผู้เล่นสามารถสัมผัสประสบการณ์ที่สนุกสนาน เพลิดเพลิน และผ่อนคลายไปพร้อมกัน อีกทั้งยังช่วยพัฒนาทักษะด้านต่าง ๆ ของผู้เล่น ไม่ว่าจะเป็นทักษะทางกาย เช่น การตอบสนองอย่างรวดเร็ว ทักษะทางสติปัญญา เช่น การวิเคราะห์ การคิดอย่างเป็นระบบ และการแก้ปัญหา ตลอดจนทักษะทางอารมณ์ เช่น การควบคุมความรู้สึก และการทำงานร่วมกับผู้อื่นในเกมที่มีลักษณะเป็นกลุ่ม นอกจากนี้ เกมคอมพิวเตอร์ยังมีบทบาทในการส่งเสริมการเรียนรู้ในรูปแบบที่ไม่น่าเบื่อ ผู้เล่นสามารถเรียนรู้กฎ กติกา และวิธีการแก้ไขสถานการณ์ต่าง ๆ ในเกม โดยสิ่งเหล่านี้สามารถนำไปประยุกต์ใช้ในชีวิตประจำวันได้ ซึ่งเป็นเหตุผลที่เกมคอมพิวเตอร์กลายเป็นหนึ่งในสื่อที่ทรงอิทธิพลต่อการพัฒนาทั้งในด้านความบันเทิงและการศึกษา

สำหรับความหมายของเกมคอมพิวเตอร์นั้น นักวิชาการและผู้เชี่ยวชาญต่างมีมุมมองที่หลากหลาย เช่น วรพจน์ พวงสุวรรณ ได้ให้คำจำกัดความว่า "เกมคอมพิวเตอร์คือสื่อบันเทิงในรูปแบบดิจิทัลที่บรรจุอยู่ในแผ่นซีดีรอมหรือสื่อบันทึกข้อมูลรูปแบบอื่น ๆ ที่สามารถเล่นผ่านคอมพิวเตอร์ส่วนบุคคลได้ โดยเกมเหล่านี้มีจุดมุ่งหมายเพื่อสร้างความสนุกสนาน การพัฒนาศักยภาพของผู้เล่นในหลายมิติ และได้รับการพัฒนาอย่างต่อเนื่องเพื่อตอบสนองต่อความต้องการที่เปลี่ยนแปลงของผู้บริโภคในยุคดิจิทัล

2.2 หลักการพัฒนาซอฟต์แวร์ (ที่มา : https://archive.lib.cmu.ac.th/full/T/2560/)

ในหลักการพัฒนาซอฟต์แวร์ทางวิศวกรรมนั้นจำเป็นต้องใช้กระบวนการเชิงวิศวกรรมที่เรียกว่าวิศวกรรมซอฟต์แวร์สำหรับนักพัฒนาและบำรุงรักษาซอฟต์แวร์อย่างเป็นขั้นตอนเพื่อให้งานสำเร็จลุล่วงตามเวลาและบรรลุเป้าหมายที่ต้องการซึ่งซอฟต์แวร์ที่พัฒนานามาแล้วจะเข้าสู่วัฏจักรของการนำไปใช้งานแล้วนำมาปรับปรุงแก้ไขและย้อนนำกลับ มาใช้งานใหม่ตลอดระยะเวลาของการใช้งานซอฟต์แวร์นั้นๆไม่ว่า จะเนื่องมาจากข้อผิดพลาดของซอฟต์แวร์จากการเปลี่ยนแปลงข้อกำหนดหรือเงื่อนไขภายในซอฟต์แวร์หรือจากความต้องการของผู้ใช้งานซึ่ง โดยมากมักจะมีการเปลี่ยนแปลงภายหลังที่ได้ทดลองใช้ซอฟต์แวร์ไปแล้วในระยะหนึ่งดังนั้นในกำรออกแบบและพัฒนาซอฟต์แวร์หากมีการออกแบบเพื่อรองรับซึ่งการพัฒนาซอฟต์แวร์จะใช้หลักตามวัฏจักรของการพัฒนาระบบ (System development life cycle:SDLC) ประกอบด้วย 6 ขั้นตอนดังนี้

2.2.1 การวางแผนระบบ (System planning) เป็นการกำหนดจุดมุ่งหมายเกี่ยวกับวัตถุประสงค์ขอบข่ายของเนื้อหาและฟังก์ชันต่างๆที่จะมีในระบบอย่างคร่าวๆ ทำการค้นคว้าและรวบรวมข้อมูลซึ่งผลลัพธ์ที่ได้จากการวางแผนจะใช้ประกอบกับขั้นตอนการวิเคราะห์ระบบ

2.2.2 การวิเคราะห์ระบบ (System analysis) วิเคราะห์ความต้องการของผู้ใช้และเจ้าของระบบ เพื่อกำหนดรูปแบบของระบบและตรวจสอบรายละเอียดในทุกมิติ เพื่อให้ตรงกับความต้องการมากที่สุด โดยเอกสารที่ได้จากการวิเคราะห์ความต้องการคือชุดเอกสารข้อกำหนดความต้องการทางเทคนิคซึ่งจะไปใช้ในการออกแบบซอฟต์แวร์ต่อไป

2.2.3 การออกแบบระบบ (System design) ออกแบบเชิงเทคนิคตามข้อกำหนด เช่น การออกแบบเชิงแนวคิด ฐานข้อมูล การควบคุม การออกแบบหน้าจอ และส่วนแสดงผลต่าง ๆ ทั้งนี้เพื่อสร้างระบบให้สอดคล้องกับวัตถุประสงค์และเนื้อหาที่วางไว้

2.2.4 ดำเนินเรื่อง (Story board) วางลำดับเรื่องราวและภาพตามวัตถุประสงค์ โดยเรียงลำดับภาพและข้อความตั้งแต่ต้นจนจบ ช่วยให้การสร้างเนื้อหามีความต่อเนื่อง เป็นระบบ และง่ายต่อการบำรุงรักษาหรือแก้ไขในอนาคตอีกด้วย

2.2.5 การพัฒนาระบบ (System development) เริ่มเขียนโปรแกรมจริงตามแบบที่วางไว้ พร้อมทั้งทดสอบระบบในทุกส่วนเพื่อให้มั่นใจว่าแต่ละองค์ประกอบทำงานร่วมกันได้ดี การตรวจสอบในขั้นนี้สำคัญเพื่อให้ระบบมีความสมบูรณ์และมีข้อผิดพลาดน้อยที่สุด

2.2.6 การทดสอบคุณภาพของซอฟต์แวร์ (System testing) เป็นขั้นตอนของการดำเนินการตรวจสอบคุณภาพตามหลักของวิศวกรรมซอฟต์แวร์และการต่อประสานกับ ผู้ใช้เพื่อให้ มั่นใจได้ว่าซอฟต์แวร์ที่พัฒนาขึ้นมีคุณลักษณะของซอฟต์แวร์ที่ดี

2.3 โปรแกรมที่ใช้งาน (ที่มา: https://www.mindphp.com/คู่มือ/8286-program-unity.html)

2.3.1 Unity เป็นโปรแกรมที่มีความสามารถหลากหลายได้แก่สร้างเกม 2 มิติการสร้างเกม 3 มิติการสร้าง AR, VR สามารถส่งแอพพลิเคชั่นได้ทั้งระบบ Windows, iOS และ Android

รูปภาพที่ 2.1 โลโก้ Unity

ตัวเอนจินนั้นใช้ภาษา C# ในการพัฒนาโปรแกรมให้เป็นไปตามที่ผู้พัฒนาต้องการ โดยอินเตอร์เฟสของ Unity นั้นเข้าใจได้ง่ายสำหรับนักพัฒนามือใหม่และมีรายละเอียดเยอะ

รูปภาพที่ 2.2 หน้าอินเตอร์เฟสของ Unity

2.3.2 เครื่องมือของโปรแกรม Unity (ที่มา: https://docs.unity3d.com/TheEditor.html)

A : เมนูบาร์ (Menu Bar) สำหรับเรียกใช้คำสั่งต่างๆ

B : แถบเครื่องมือ (Tool Bar) สำหรับจัดการกับวัตถุ

C : พาเนล Hierarchy สำหรับจัดลำดับวัตถุ

D : พาเนล Scene สำหรับแสดงพื้นที่การทำงาน

E : พาเนล Project สำหรับจัดเก็บไฟล์ต่างๆที่ต้องใช้งาน เช่น รูปภาพ วีดีโอ

F : พาเนล Inspector สำหรับกำหนดค่าต่างๆ ให้กับวัตถุที่เลือกในพาเนล

2.3.3 ข้อดีและข้อเสียโปรแกรม Unity

1) ข้อดี มีตัวอย่าง Source code และคลิปสอนมากมาย ที่ทาง Unity ทำไว้ให้ศึกษา รวมถึงนักพัฒนาอีกหลายๆท่านที่มีคลิปสอนสร้าง AR, VR , ภาพกราฟิก หรือ เกมต่างๆได้หลากหลาย

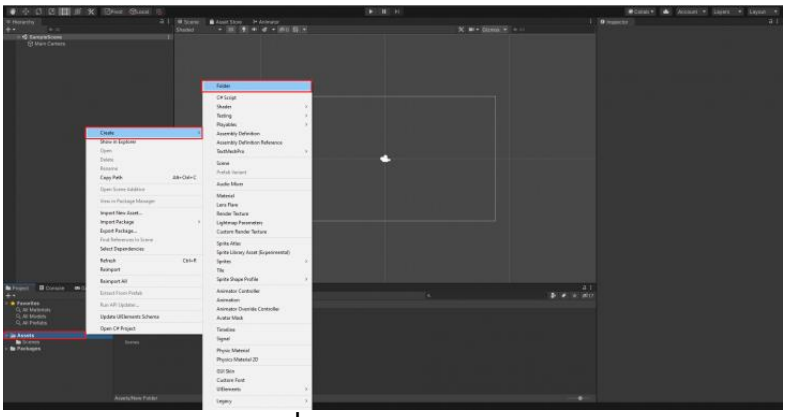
2) ข้อเสีย ใช้เนื้อที่ในการจัดเก็บโปรแกรม และไฟล์ ในปริมาณมากเกินไป

2.3.4 ขั้นตอนการเริ่มสร้างเกมจาก โปรแกรมสร้างเกม Unity

1) เปิดโปรแกรม ตั้งชื่อเกมแล้วเลือก Templates แบบ 2D

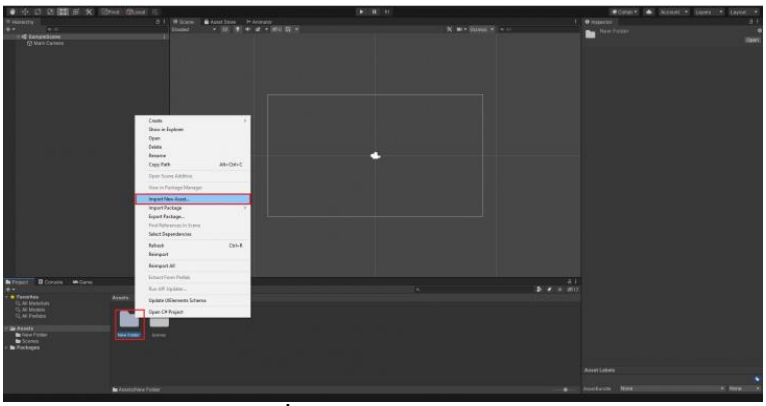
2) เลือก Template ที่จะใช้ หรือจะใช้โปรแกรม Unity มีให้ดาวน์โหลดก็ได้

3) เมื่อเข้ามาถึงหน้าโปรแกรม ไปที่แถบ Project เลือก Asset คลิกขวา ⟶ Create ⟶ Folder จากนั้นตั้งชื่อโฟล์เดอร์



รูปภาพที่ 2.3 ตัวอย่างการใช้งาน **1**

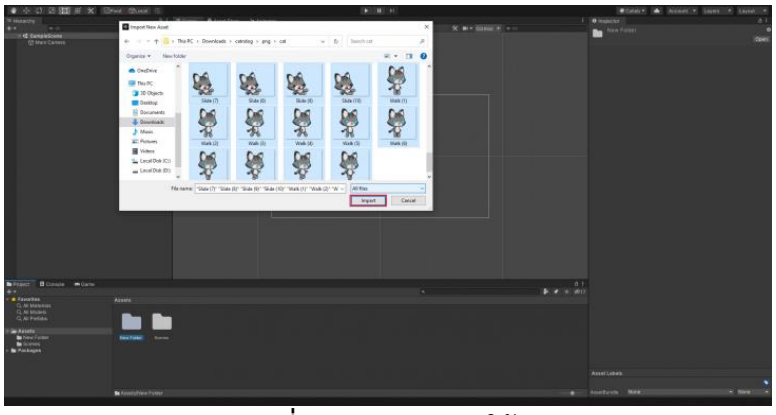
4) นำรูปภาพหรือตัวละครจาก Template ที่เราดาวน์โหลดไว้มาใช้งาน เลือกโฟล์เดอร์ที่สร้าง คลิกขวา เลือก Import New Asset แล้วเลือกรูปที่ต้องการใช้ แล้วกด Import



รูปภาพที่ 2.4 ตัวอย่างการใช้งาน 2

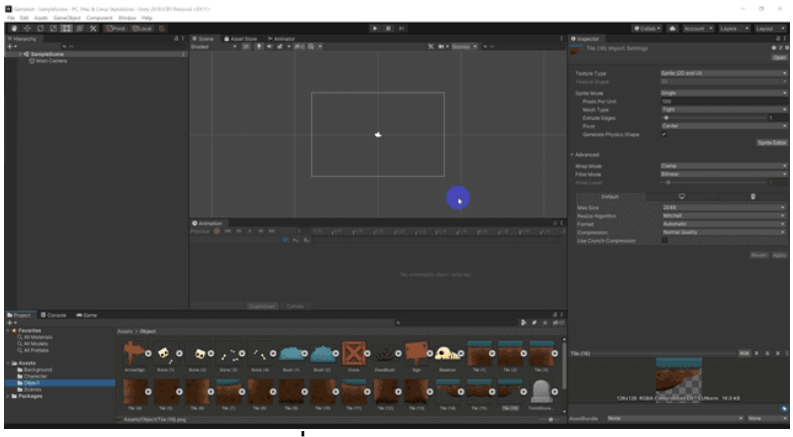
5) ลากตัวละครเข้าไปวางใน Scene ให้เริ่มจากตัว Idle ซึ่งเป็นตัวที่ยืนเฉยๆ

6) เมื่อเสร็จแล้วจะกดมาหน้า Scene เราจะไม่เห็นตัวละคร Order in Layer เป็น 2



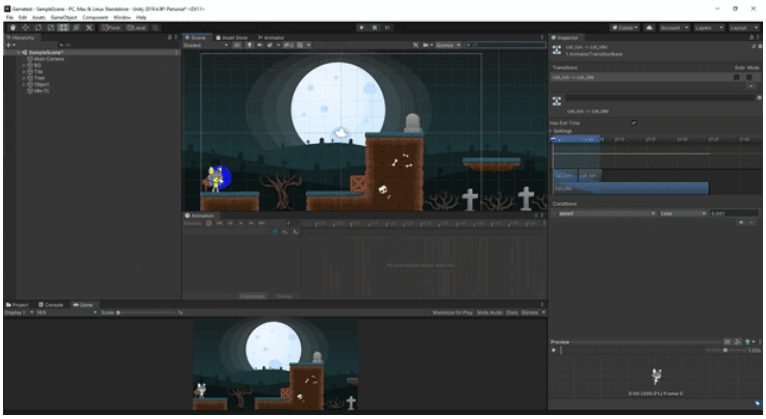
รูปภาพที่ 2.5 ตัวอย่างการใช้งาน 3

7) การที่จะทำให้ตัวละครยืนบนวัตถุอื่นๆ ได้ให้ไปตั้งค่าที่ Inspector ⟶ Add Component ⟶ เลือก Rigidbody 2D และ Circle Collider 2D



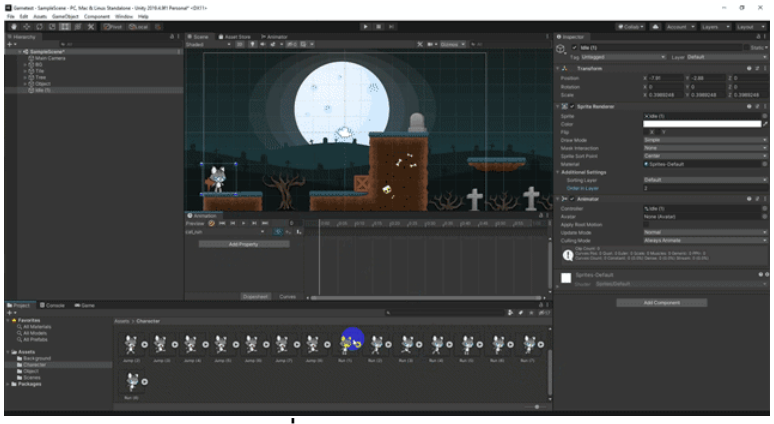
รูปภาพที่ 2.6 ตัวอย่างการใช้งาน 4

8) ต่อไปเราจะมาใส่กำหนดการเคลื่อนไหวให้ตัวละคร ให้เลือกที่ตัวละครก่อน จากนั้น ไปที่ Window ⟶ Animation ⟶ Animation



รูปภาพที่ 2.7 ตัวอย่างการใช้งาน 5

9) ไปคลิกเครื่องหมายที่ Dropdown (▼) เลือก Create New Clip ตั้งชื่อแล้วกด Save จากนั้นลากตัวละครมาใส่ในแถบ Animation แล้วลองกดเล่นดู ทำให้ครบทั้งวิ่ง และกระโดด



รูปภาพที่ 2.8 ตัวอย่างการใช้งาน 6

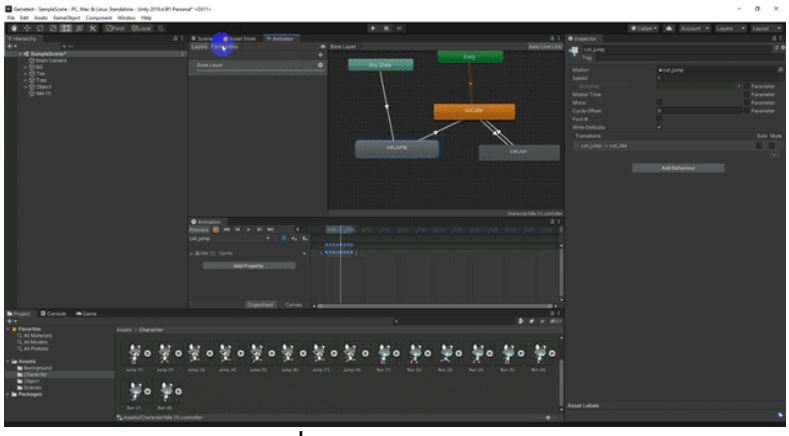
10) เมื่อเสร็จแล้วให้ไปที่ Animator เลือก Parameters เลือกเครื่องหมายบวก แล้วเลือกประเภทของตัวแปร จากนั้นตั้งชื่อ ซึ่งตัวแปรแต่ละประเภทคือ

1) Float ใช้กับตัวเลขที่เป็นทศนิยม

2) Int ใช้กับตัวเลขจำนวนเต็ม

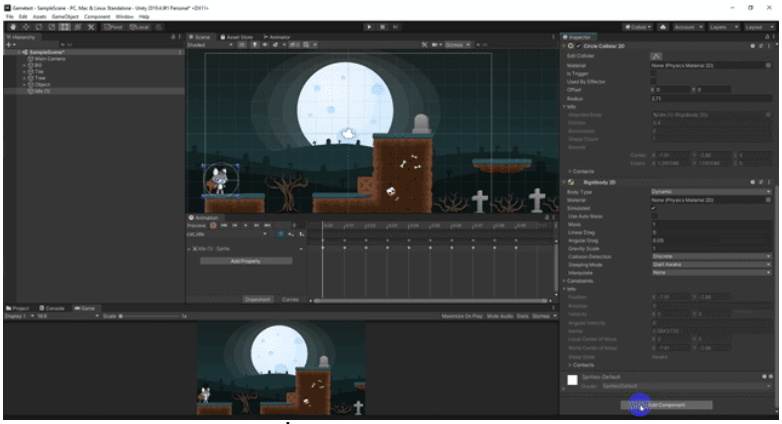
3) Bool ใช้เก็บข้อมูลที่เป็นค่า จริง (True) หรือเท็จ (False) เท่านั้น

11) ซึ่งเราจะกำหนดอันแรกเป็น Float ตั้งชื่อเป็น Speed และอันที่สองเป็น Bool ตั้งชื่อว่า Jump



รูปภาพที่ 2.9 ตัวอย่างการใช้งาน 7

12) ขั้นตอนต่อไปคือการทำให้ตัวละครเคลื่อนที่ไปข้างหน้าโดยเราจะต้องเขียนโค้ด ไปที่Add Component ⟶ New Script ⟶ ตั้งชื่อ Script ⟶ Create and Addจากนั้นดับเบิลคลิก ตรง Script แล้วเริ่มเขียนโค้ด

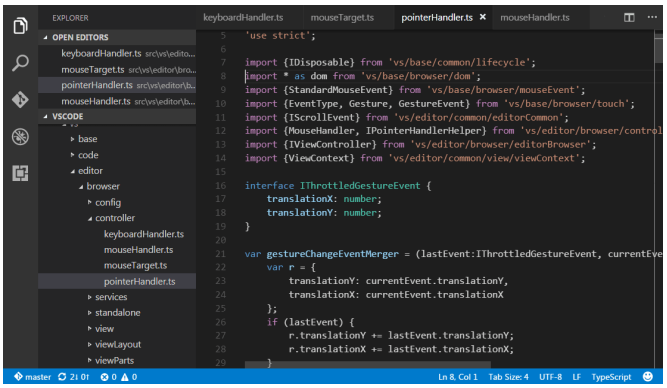


รูปภาพที่ 2.10 ตัวอย่างการใช้งาน 8

2.3.5 Visual Studio Code (ที่มา: https://code.visualstudio.com/docs/supporting/FAQ) หรือ VSCode เป็นโปรแกรม Code Editor ที่ใช้ในการแก้ไขและปรับแต่งโค้ด จากค่ายไมโครซอฟท์ มีการพัฒนาออกมาในรูปแบบของ OpenSource จึงสามารถนำมาใช้งานได้แบบฟรี ๆ ที่ต้องการความเป็นมืออาชีพ

รูปภาพที่ 2.11 โลโก้ Visual Studio Code

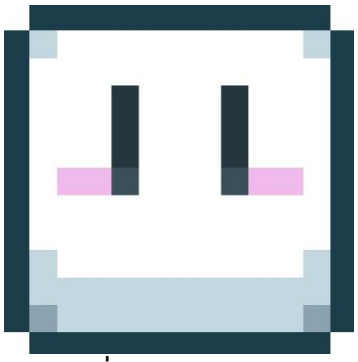
Visual Studio Code นั้น เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานข้ามแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows, macOS และ Linux สนับสนุนทั้งภาษา JavaScript, TypeScript และ Node.js สามารถเชื่อมต่อกับ Git ได้ นำมาใช้งานได้ง่ายไม่ซับซ้อน มีเครื่องมือส่วนขยายต่าง ๆ ให้เลือกใช้อย่างมากมาก ไม่ว่าจะเป็น 1.การเปิดใช้งานภาษาอื่นๆ ทั้ง ภาษา C++, C#, Java, Python, PHP หรือ Go 2.Themes 3.Debugger 4.Commands เป็นต้น



รูปภาพที่ 2.12 หน้าอินเตอร์เฟสของ Visual Studio Code

2.3.6 Aseprite (ที่มา: https://www.aseprite.org/faq)

คือโปรแกรมสำหรับการสร้างและแก้ไขภาพพิกเซล (pixel art) ที่ได้รับความนิยมในหมู่นักพัฒนาเกมและศิลปินดิจิทัล โดยเฉพาะในงานที่เกี่ยวข้องกับเกม 2 มิติ เช่น การออกแบบตัวละคร, สิ่งแวดล้อม, และอนิเมชันแบบพิกเซล (pixel animation)



รูปภาพที่ 2.13 โลโก้ Aseprite

Aseprite เหมาะสำหรับการสร้าง พิกเซลอาร์ต (Pixel Art) และ แอนิเมชัน 2D โดยเฉพาะ เหมาะกับนักพัฒนาเกม 2D และศิลปินที่ต้องการเครื่องมือเฉพาะทาง มีฟีเจอร์เด่นอย่างการสร้างเฟรมต่อเฟรม (Frame-by-Frame Animation), Onion Skinning, ระบบจัดการสีแบบพาเลต, การออกแบบ Sprite Sheet และการปรับแต่งพิกเซลอย่างแม่นยำ รองรับการส่งออกไฟล์หลายรูปแบบ เช่น .png, .gif และ Sprite Sheet ทำให้เป็นตัวเลือกยอดเยี่ยมสำหรับทั้งมือใหม่และมืออาชีพในงานพิกเซลอาร์ตและเกมพัฒนาเกม

รูปภาพที่ 2.14 อินเตอร์เฟสของ Aseprite

2.4 งานวิจัยที่เกี่ยวข้อง

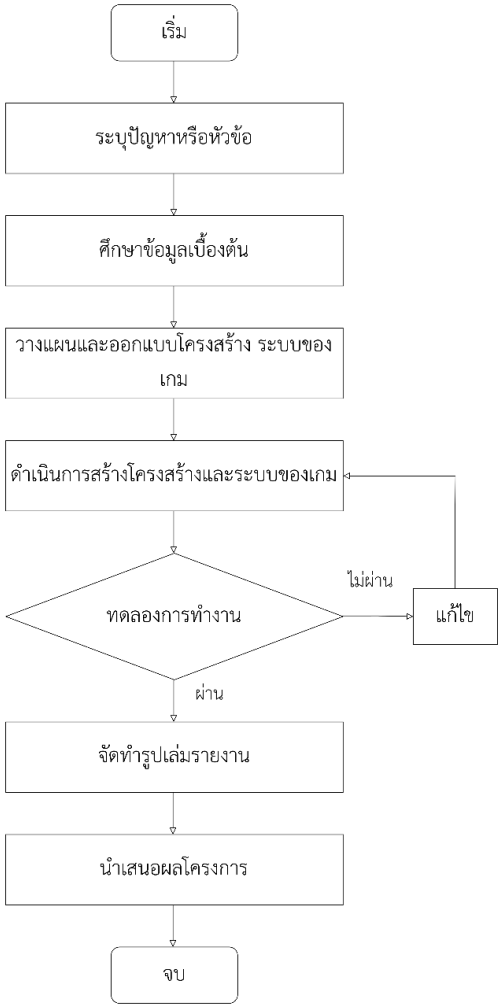
วิไลภรณ์ ภูกองชัย (2559 บทคัดย่อ) เกมสำหรับฟื้นฟูสมรรถภาพทางร่างกายของผู้ป่วยที่มีอาการกล้ามเนื้อแขนอ่อนแรงด้วยอุปกรณ์คิเน็ค งานวิจัยนี้เป็นการพัฒนาเกมสำหรับฟื้นฟูสมรรถภาพกล้ามเนื้อแขนของผู้ป่วยที่มีอาการอ่อนแรงด้วยเทคโนโลยีโมชั่นคอนโทรลเลอร์มีวัตถุประสงค์เพื่อบำบัดและฟื้นฟูสมรรถภาพกล้ามเนื้อแขนของผู้ป่วยผ่านการเล่นเกมแตะฟองอากาศ จะสามารถเล่นได้โดยที่ผู้ป่วยจะต้องแตะฟองอากาศที่แสดงบนหน้าจอคอมพิวเตอร์ให้ได้จำนวนเยอะที่สุด ระบบจะแสดงคะแนนเมื่อจบเกม นักกายภาพหรือผู้ดูแลผู้ป่วยสามารถนำคะแนนดังกล่าวมาใช้เพื่อประเมินความคืบหน้าการฟื้นฟูของผู้ป่วย

พรชนัน ดาราพงษ์ (2555 บทคัดย่อ) การออกแบบเกมดิจิตอลเพื่อส่งเสริมการเรียนรู้วิชาวิทยาศาสตร์สำหรับระดับชั้นประถมศึกษาปีที่ 3 การวิจัยในครั้งนี้มีทั้งหมด 3 ช่วง คือ ช่วงที่ 1 สังเคราะห์องค์ความรู้ภาคเอกชนและภาคสนามเพื่อนำมาวิเคราะห์ในงานออกแบบเกมดิจิตอลเสริมการเรียนรู้ระดับชั้นปฐมวัย ช่วงที่ 2 นำข้อมูลทั้งหมดที่ได้จากการวิเคราะห์มาออกแบบเกมดิจิตอลประเภทแนวแก้ไขปัญหาแทรกสาระความรู้วิชาวิทยาศาสตร์สำหรับชั้นประถมศึกษาปีมี่ 3 ช่วงที่ 3 นำเกมที่ได้ออกแบบไปทดลองกับกลุ่มตัวอย่างโดยการประเมินผลงานออกแบบนั้นใช้แบบทดสอบวัดผลความรู้ก่อนเล่นและหลังที่ได้เล่นเกม เกมที่ออกแบบเพื่อหาค่าเฉลี่ยที่ได้มาสรุปผลการทดลอง โดยผลการวิจัยพบว่าการออกแบบเกมดิจิตอลเพื่อส่งเสริมการเรียนรู้วิชาวิทยาศาสตร์สำหรับระดับชั้นประถมศึกษาปีที่ 3 คะแนนแบบทดสอบประเมินความรู้ทั้งหมดอยู่ในระดับที่ดี ซึ่งหมายความได้ว่า การพัฒนากราฟฟิกองค์ประกอบในเกมดิจิตอลเสริมการเรียนรู้นั้นส่งผลดีให้กับนักเรียนที่ได้ทดลองเล่นเป็นอย่างมาก

เอกพันธ์ นามสมบูรณ์ (2562 บทคัดย่อ) โครงงานนี้มีวัตถุประสงค์ เพื่อให้ผู้เล่นได้รู้จักองค์ประกอบรวมของการทำธุรกิจ ผ่านทางเกมFantasy World เพื่อสามารถให้ผู้เล่นได้เพลิดเพลิน และทำให้ผู้เล่นได้ฝึกการทำธุรกิจในโลกสมมุติขึ้นมา เพื่อฝึกการบริหารผ่านตัวละครโลกในเกม การค้ากับต่างเผ่าพันธุ์ที่สมมุติขึ้นภายในเกมได้ฝึกการหาข้อมูลในเกมการพัฒนาเกม FantasyWorld ดำเนินการวิเคราะห์ และออกแบบเกมโดยมีนิทาน และนิยาย ในโลกที่มีเวทย์มนต์เป็นต้นแบบ ร่วมถึงตำนานต่างๆที่ผู้คนสามารถพบเห็นได้บ่อย ทั้งในภาพยนต์ เกม การ์ตูน หรือสื่ออื่นๆ โดยใช้โปรแกรม RPG Maker MV ในการออกแบบตัวละคร แผนที่ เมือง ภูมิประเทศขึ้น โดยโปรแกรม RPG Maker MV ใช้ภาษา JavaScript ที่เป็นคำสั่งในโปรแกรมกำหนดมาให้จากการพัฒนาเกม FantasyWorld

บทที่ 3
วิธีดำเนินงานโครงการ

ในการจัดทำโครง Pastoral Pathways ผ่านโปรแกรม Unity มีขั้นตอนการสร้างในส่วนต่างๆ โดยทางกลุ่มผู้สร้างได้ร่วมกันวางแผนในการปฏิบัติงานและจัดแบ่งงานตามความ เหมาะสม ขั้นตอนในการดำเนินโครงการ แบ่งออกเป็นดังนี้



รูปภาพที่ 3.1 แสดงผังการดำเนินโครงการ

3.1 การศึกษาข้อมูลเบื้องต้น

3.1.1 ศึกษาค้นคว้าจากเอกสารตำราและสื่อต่างๆ ที่ให้ความรู้เกี่ยวข้องกับ เกมสองมิติ, Unity และอุปกรณ์คอมพิวเตอร์ รวบรวมข้อมูลให้ได้มากที่สุด

3.2 โปรแกรมที่ใช้ในการพัฒนา

3.2.1 โปรแกรมที่ใช้ในการพัฒนาเกม คือ โปรแกรม Unity 3D เป็นโปรแกรมที่มีความสามารถหลากหลายได้แก่สร้างเกม 2 มิติการสร้างเกม 3 มิติการสร้าง AR, VR สามารถส่งแอพพลิเคชั่นได้ทั้งระบบ Windows, iOS และ Android โดยจะใช้คู่กับโปรแกรม Vuforia ในการสร้างสื่อ AR , VR

3.2.2 โปรแกรมที่ใช้ในการวาดรูป คือ โปรแกรม Aseprite คือโปรแกรมสำหรับการสร้างและแก้ไขภาพพิกเซล (pixel art) ที่ได้รับความนิยมในหมู่นักพัฒนาเกมและ ศิลปินดิจิทัล โดยเฉพาะในงานที่มีความเกี่ยวข้องกับเกมประเภท 2 มิติ เช่น การออกแบบของตัวละคร,กับสภาพของสิ่งแวดล้อมและอนิเมชันแบบพิกเซล (pixel animation)

3.2.3 โปรแกรมที่จะในการใช้ในการแต่งรูป คือ โปรแกรม Adobe Photoshop โปรแกรม Adobe Photoshop หรือที่เรียกสั้นๆ ว่า Photoshop คือ โปรแกรมแต่งรูป (Photo Editing Software) ที่ให้คุณได้สามารถออกแบบและตกแต่งรูปภาพได้อย่างมืออาชีพ ไม่ว่าจะเป็นงานด้านสิ่งพิมพ์ หรือภาพสำหรับใช้งานบนเว็บไซต์ เป็นต้น

3.3 ขั้นตอนการดำเนินงาน

3.3.1 การคิดหัวข้อโครงการเพื่อนำเสนออาจารย์ที่ปรึกษาโครงการ

3.3.2 ศึกษาและค้นคว้าข้อมูลที่เกี่ยวข้องกับเรื่องที่สนใจคือเรื่องการใช้โปรแกรมว่ามีเนื้อหามากน้อยเพียงใด และต้องศึกษาค้นคว้าเพิ่มเติมเพียงใดจากเว็บไซต์ต่างๆ และเก็บข้อมูลไว้เพื่อจัดทำเนื้อหาต่อไป

3.3.3 ศึกษาการใช้โปรแกรมพัฒนาเกมจากเอกสาร หรือที่สร้างจากเว็บไซต์ต่างๆ ที่นำเสนอวิธีใช้

3.3.4 จัดทำโครงร่างโครงการเพื่อนำเสนออาจารย์ที่ปรึกษาผ่านเว็บบล็อกของตัวเองโดยได้นำไฟล์ข้อมูลไปฝากใน Google Drive

3.3.5 ปฏิบัติการจัดทำโครงการการพัฒนาเกมคอมพิวเตอร์ด้วยโปรแกรม unity โดยสร้างเกมคอมพิวเตอร์ตามรูปแบบที่คิดไว้

3.3.6 นำเสนอรายงาน ของความก้าวหน้ารายสัปดาห์ โดนอาจารย์ที่ปรึกษาจะเข้าไปตรวจความก้าวหน้าของเกมโดยอาจารย์ที่ปรึกษาจะให้ข้อมูลเสนอแนะต่างๆเพื่อให้ จัดทำเนื้อหาและการนำเสนอที่น่าสนใจต่อไป ทั้งนี้เมื่อได้รับคำแนะนำก็จะนำมาปรับปรุงแก้ไขให้เป็นที่น่าสนใจยิ่งขึ้น

3.3.7 จัดทำเอกสารรายงานโครงการคอมพิวเตอร์โดยนำเสนอรูปแบบไฟล์คอมพิวเตอร์และนำฝากข้อมูลดังกล่าวไว้ใน Google Drive

ตารางการวางแผน และการเตรียมการของโครงการนี้ใช้ ระยะเวลาในการพัฒนาตั้งแต่เดือนตุลาคม พ.ศ.2567 ถึงเดือน กุมภาพันธ์ พ.ศ. 2568

ตารางที่ 3.1 แสดงระยะเวลาดำเนินการ

ลำดับการทำงาน	**ระยะเวลการดำเนินงาน**				
	ปี พ.ศ. 2567			**ปี พ.ศ. 2568**	
	ต.ค	**พ.ย**	**ธ.ค**	**ม.ค**	**ก.พ**
1. การคิดหัวข้อโครงการ	←				
2. ศึกษาและค้นคว้าข้อมูล	←				
3. ศึกษาการใช้โปรแกรม	←				
4. จัดทำโครงร่างโครงการ	←	→			
5. ปฏิบัติการจัดทำโครงการ	←				→
6. นำเสนอรายงาน			←	→	
7. จัดทำเอกสารรายงาน			←		→

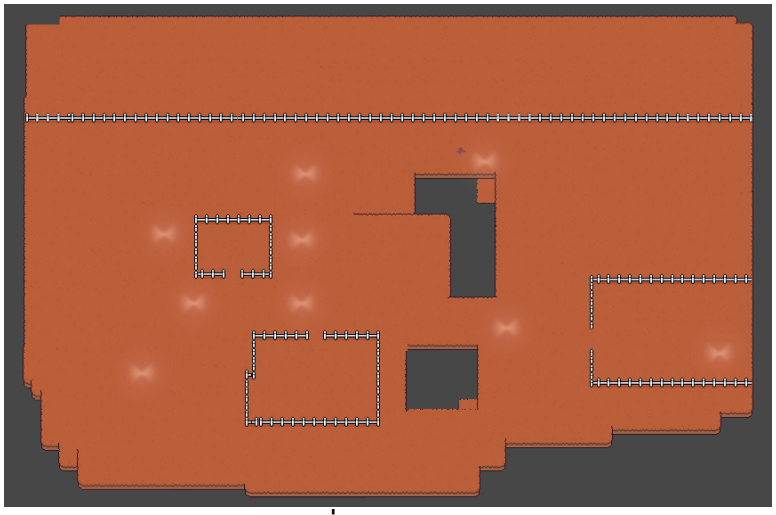
3.4 ดำเนินการออกแบบ

การดำเนินการออกแบบมีขั้นตอนดังนี้

รูปภาพที่ 3.2 ออกแบบเมนูเกม

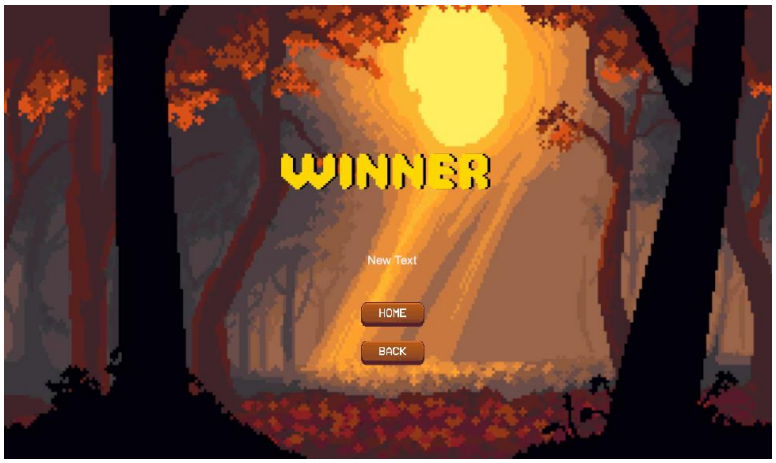
รูปภาพที่ 3.3 ออกแบบหน้า คัทซีน

รูปภาพที่ 3.4 ออกแบบตัวละคร



รูปภาพที่ 3.5 ออกแบบฉาก

รูปภาพที่ 3.6 จัดเรียงฉาก



รูปภาพที่ 3.7 ออกแบบฉากชนะ

รูปภาพที่ 3.8 ออกแบบชื่อเกม

3.5 การประเมินความพึงพอใจ

3.5.1 พิจารณาจากคะแนนตามเกณฑ์

เครื่องมือในการเก็บข้อมูลได้แก่ แบบสอบถาม เกม Pastoral Pathways ผ่านโปรแกรม Unity กลุ่มตัวอย่างงานมีขั้นตอนดังต่อไปนี้

สร้างแบบสอบถาม ซึ่งมีมาตรส่วนประมาณค่า (Rating scale) 5 ระดับ

1	หมายถึง	น้อยที่สุด
2	หมายถึง	น้อย
3	หมายถึง	ปานกลาง
4	หมายถึง	มาก
5	หมายถึง	มากที่สุด

การวิเคราะห์แบบสอบถามความคิดเห็นของนักเรียน จากเกม Pastoral Pathways โปรแกรม Unity กำหนดช่วงของค่าเฉลี่ยตามแนวของ จอห์น ดับบลิว เบสท์ และ เจมส์ วีคาห์น (Best John W. 7 Kahn James V., 1993,PP.181-182) ดังนี้

ค่าเฉลี่ย		ความหมาย
4.50 – 5.00	หมายถึง	มีความพึงพอใจอยู่ในระดับมากที่สุด
3.50 – 4.49	หมายถึง	มีความพึงพอใจอยู่ในระดับมาก
2.50 – 3.49	หมายถึง	มีความพึงพอใจอยู่ในระดับปานกลาง
1.50 – 2.49	หมายถึง	มีความพึงพอใจอยู่ในระดับน้อย
0.00 – 1.49	หมายถึง	มีความคิดเห็นอยู่ในระดับน้อยที่สุด

3.6 สถิติที่ใช้ในการวิเคราะห์ข้อมูล

3.6.1 ค่าสถิติร้อยละ (Percentage) มีสูตรดังนี้

สูตร $P = \frac{f}{n} \times 100$ เมื่อกำหนดให้

P แทน ค่าร้อยละ

F แทน จำนวนหรือความถี่ที่ต้องการหาค่าร้อยละ

n แทน จำนวนข้อมูลทั้งหมด

3.6.2 การหาค่าเฉลี่ย ($\bar{X}$)

สูตร $\bar{X} = \frac{\Sigma x}{n}$ เมื่อกำหนดให้

x แทน คะแนนเฉลี่ย

Σx แทน ผลรวมทั้งหมดของคะแนน

n แทน จำนวนคนทั้งหมด

3.6.3 การหาค่าเบี่ยงเบนมาตรฐาน (S.D. Standard Deviation)

สูตร $S.D. = \frac{\sqrt{n\Sigma_x 2(\Sigma x^2)}}{N(n-1)}$ เมื่อกำหนดให้

$S.D.$ คือ ค่าเบี่ยงเบนมาตรฐานของกลุ่มตัวอย่าง

$n\Sigma_x 2$ คือ ผลรวมยกกำลังสองของคะแนนทุกจำนวน

(Σx^2) คือ ผลรวมคะแนนทุกจำนวนยกกำลังสอง

N คือ จำนวนผู้ตอบแบบสอบถามทั้งหมด

บทที่ 4
ผลการดำเนินงานวิจัย

ในการจัดทำงานวิจัยชิ้นนี้ คณะผู้จัดทำได้ทำการสร้างเกม Pastoral Pathways โดยผ่านโปรแกรม unity โดยมีการเก็บรวบรวมข้อมูลการวิจัยดังนี้

4.1 ปัญหาที่พบระหว่างการวิจัย

4.2 ผลการประเมินความพึงพอใจของผู้เล่น

4.3 ประสิทธิภาพของเกม

4.1 ปัญหาที่พบระหว่างการวิจัย

ในระหว่างการวิจัย ทางคณะผู้จัดทำพบปัญหาระหว่างพัฒนาเกมมีดังนี้

4.3.1 ปัญหาด้านการพัฒนา UI และระบบเมนู ในระหว่างการพัฒนาเกมพบปัญหาที่เกี่ยวกับการออกแบบ UI (User Interface) และระบบเมนูที่มีผลต่อประสบการณ์การใช้งานของผู้เล่น โดยปัญหาหลักที่พบ ได้แก่ การออกแบบ UI ให้ใช้งานง่ายเป็นเรื่องที่ท้าทาย เนื่องจากต้องทำให้ผู้เล่นสามารถเข้าถึงฟังก์ชันต่างๆ ได้อย่างสะดวก เช่น เมนูหลัก เมนูตั้งค่า และอินเวนทอรี นอกจากนี้ ยังพบปัญหาการแสดงผล UI ที่ไม่รองรับหน้าจอทุกขนาด

4.3.2 ปัญหาด้านการทดสอบและบั๊กในเกม ในกระบวนการทดสอบเกมพบปัญหาต่างๆ ที่ส่งผลต่อคุณภาพของเกม ซึ่งต้องได้รับการแก้ไขก่อนเปิดให้ผู้เล่นใช้งาน โดยปัญหาหลักที่พบ ได้แก่ ตัวละครติดฉากหรือทะลุกำแพง เนื่องจากการตั้งค่าระบบฟิสิกส์ไม่ถูกต้อง ทำให้เกิดบั๊กการชนกันของวัตถุ ศัตรูไม่ตอบสนองต่อผู้เล่นหรือเคลื่อนที่ผิดพลาด เช่น ติดอยู่ในที่แคบหรือเดินวนซ้ำไปมา และยังพบว่าเควสและภารกิจบางส่วนไม่สามารถทำสำเร็จได้

4.3.3 ปัญหาด้านระบบไอเทมและอินเวนทอรีระบบไอเทมและอินเวนทอรีเป็นองค์ประกอบสำคัญของเกมที่ต้องได้รับการออกแบบให้มีความเสถียร แต่ในระหว่างการพัฒนาได้พบปัญหาต่างๆ เช่น การแสดงผลของไอเทมในช่องเก็บของไม่ถูกต้อง เช่น ไอเทมซ้อนทับกันผิดพลาด

4.2 ผลการประเมินความพึงพอใจของผู้เล่น

ผลการประเมินความพึงพอใจนั้นได้รับจากการสรุปคะแนนที่นักศึกษาและอาจารย์ผู้สอนเป็นผู้ให้คำแนะนำจากการนำเสนอและร่วมโครงการ ทางผู้วิจัยขอสรุปผลเป็นตัวอย่างผลคะแนนโครงการวิจัยกลุ่ม ดังนี้

ตารางที่ 4.1 ความพึงพอใจของผู้เล่น

หัวข้อการประเมิน	$\bar{x}$	$S.D.$	ระดับความพึงพอใจ
ด้านการออกแบบ			
1.การออกแบบมีความสวยงามเหมาะสม	2.90	0.70	ปานกลาง
2.ความน่าสนใจของภาพและเสียงของเกม	4.15	0.79	มาก
3.ความเหมาะสมของเนื้อหา	2.85	0.91	ปานกลาง
ด้านนำเสนอ			
4.ความน่าสนใจของเกม	3.00	0.71	ปานกลาง
5.ความชัดเจนของข้อมูล	4.30	0.64	มาก
6.ความเหมาะสมในการนำเสนอข้อมูล	4.75	0.54	มากที่สุด
ด้านประโยชน์			
7.ได้รับความรู้เกี่ยวกับการปลูกผัก	3.40	0.58	ปานกลาง
8.ได้รู้จักโปรแกรม Unity ที่ใช้ในการสร้างเกม	3.60	0.60	มาก
9.สามารถนำความรู้ที่ได้ไปใช้ในชีวิตประจำวันได้	4.00	0.67	มาก
ผลรวม	3.66	0.68	มาก

จากตารางที่ 4.1 เป็นการประเมินความพึงพอใจจากการใช้ผู้เล่น จากเกม Pastoral Pathways พบว่า ความรู้เกี่ยวกับการปลูกมีค่าเฉลี่ยสูงสุดอยู่ที่ ($\bar{x}$=4.75, S.D.=0.54) ซึ่งอยู่ในเกณฑ์ความพึงพอใจในระดับที่มากที่สุด รองลงมาคือ ความชัดเจนของข้อมูล โดยมีค่าเฉลี่ยอยู่ที่ ($\bar{x}$=4.30, S.D.=0.64) ซึ่งอยู่ในเกณฑ์ความพึงพอใจในระดับมาก และ การได้รู้จักโปรแกรม Unity ที่ใช้ในการสร้างเกม มีค่าเฉลี่ยต่ำสุดอยู่ที่ ($\bar{x}$=3.40, S.D.=0.58) ซึ่งอยู่ในเกณฑ์ความพึงพอใจในระดับปานกลาง โดยรวม ตัวเกมมีค่าเฉลี่ยความพึงพอใจอยู่ในระดับมาก ($\bar{x}$=3.64, S.D.=0.69)

4.3 ประสิทธิภาพของเกม

จากการทดสอบประสิทธิภาพของเกม Pastoral Pathways คณะผู้จัดทำจึงมีการบันทึกผลความเสถียรในตารางประสิทธิภาพดังตารางที่ 4.1 ดังนี้

ตารางที่ 4.2 ประสิทธิภาพของเกม

คุณสมบัติ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**ความเสถียร**
1.การแสดงผลของเกม	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	ร้อยละ100
2.การเคลื่อนไหวของตัวละคร	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	ร้อยละ100
3.การเปลี่ยนฉาก	✓	✓	✓	✓	✓	✓	✓	✓			ร้อยละ80
4.ความลื่นไหลของเกม	✓	✓	✓	✓	✓	✓	✓				ร้อยละ70
5.การทำงานของระบบเกม	✓	✓	✓	✓	✓	✓	✓	✓	✓		ร้อยละ 90

จากตารางที่ 4.2 พบว่าเกม Pastoral Pathways มีความเสถียรเฉลี่ยอยู่ที่ร้อยละ 88

บทที่ 5
สรุปผลและข้อเสนอแนะ

ในการสร้างโครงการเกม Pastoral Pathways มีวัตถุประสงค์เพื่อสร้างเกม Pastoral Pathways ให้มีความน่าสนใจ เพื่อหาประสิทธิภาพของ เกม Pastoral Pathways และ เพื่อสำรวจความพึงพอใจของผู้ใช้เกม Pastoral Pathways ผู้จัดทำขอเสนอสรุปผลการวิจัยและข้อเสนอแนะตามลำดับชองการวิจัยดังนี้

5.1 สรุปผลการวิจัย

โครงการ เกม Pastoral Pathways จัดทำขึ้นเพื่อสร้างเกมเพื่อความสนุกสนานและ ช่วยผ่อนคลายไม่เครียดจนเกินไป โดยก่อนที่เกมจะทำการเผยแพร่ให้ทดลองเล่นกันนั้น ทางคณะผู้จัดทำได้ทำการทดสอบประสิทธิภาพและความพึงพอใจกับบรรดานักศึกษาภายใน มาก่อนแล้วโดยมีผลการประเมินดังนี้

5.1.1 ประสิทธิภาพของเกม Pastoral Pathways การทดสอบประสิทธิภาพของเกม Pastoral Pathways พบว่าเกม Pastoral Pathways มีความเสถียรเฉลี่ยอยู่ที่ร้อยละ 88 ซึ่งนำไปใช้งานได้จริง

5.1.2 ผลประเมินความพึงพอใจจากการใช้ผู้เล่น จากเกม Pastoral Pathways พบว่า ความรู้เกี่ยวกับการปลูกมีค่าเฉลี่ยสูงสุดอยู่ที่ ($\overline{x}$=4.75, S.D.=0.54) ซึ่งอยู่ในเกณฑ์ความพึงพอใจในระดับที่มากที่สุด รองลงมาคือ ความชัดเจนของข้อมูล โดยมีค่าเฉลี่ยอยู่ที่ ($\overline{x}$=4.30, S.D.=0.64) ซึ่งอยู่ในเกณฑ์ความพึงพอใจในระดับมาก และ การได้รู้จักโปรแกรม Unity ที่ใช้ในการสร้างเกม มีค่าเฉลี่ยต่ำสุดอยู่ที่ ($\overline{x}$=3.40, S.D.=0.58) ซึ่งอยู่ในเกณฑ์ความพึงพอใจในระดับปานกลาง โดยรวม ตัวเกมมีค่าเฉลี่ยความพึงพอใจอยู่ในระดับมาก ($\overline{x}$=3.64, S.D.=0.69)

5.2 ข้อเสนอแนะ

5.2.1 เกมควรใช้งานในระบบปฏิบัติการได้

5.2.2 เกมควรเพิ่มภาษาในการใช้งานสำหรับอาจารย์ต่างชาติได้

5.2.3 ตัวเกมควรที่จะมีความหลากหลายของฉาก ปรับปรุงฉากแต่ละฉากให้สวยงาม

บรรณานุกรม

การเรียนภาษา C#. (2565) สืบค้นเมื่อ 10 ธันวาคม 2567 :
จาก https://www.w3schools.com/cs/index.php

การออกแบบเกมดิจิตอลเพื่อส่งเสริมการเรียนรู้วิชาวิทยาศาสตร์สำหรับระดับชั้นประถมศึกษาปีที่ 3. (2555) สืบค้นเมื่อ 10 ธันวาคม 2567 :
จาก http://www.sure.su.ac.th/xmlui/handle/123456789/3579

เกมสำหรับฟื้นฟูสมรรถภาพทางร่างกายของผู้ป่วยที่มีอาการกล้ามเนื้อแขนอ่อนแรงด้วยอุปกรณ์คิเน็ค. (2559) สืบค้นเมื่อ 10 ธันวาคม 2567 :
จาก http://dspace.bu.ac.th/bitstream/123456789/2314/1/welaiporn_phuk.pdf

เกม FantasyWorld. (2562) สืบค้นเมื่อ 11 ธันวาคม 2567 :
จาก https://ms.udru.ac.th/bc/assets/project_uploads/20201230103719.pdf

ขั้นตอนการเริ่มสร้างเกมจาก โปรแกรม Unity 3D. (2565) สืบค้นเมื่อ 11 ธันวาคม 2567:
จาก https://tips.thaiware.com/1379.html

หลักการออกแบบ UI. (2565) สืบค้นเมื่อ 11 ธันวาคม 2567 :
จาก https://www.designil.com/7-rules-beautiful-ui-design/

หลักการ UX/UI. (2565) สืบค้นเมื่อ 12 ธันวาคม 2567 :
จาก https://www.rmonlineservices.com/article/14/การออกแบบ-UX-UI-คืออะไร-ต่างกันอย่างไร

ภาคผนวก. ก

แบบเสนอขออนุมติ

แบบเสนอขออนุมัติโครงการ

Pastoral Pathways

ผู้เสนอโครงการ

นางสาวณัฐนันท์ จิรวัฒน์ **รหัสนักศึกษา 6631280007**

นายอดิศักดิ์ พวงนะที **รหัสนักศึกษา 6631280021**

แบบเสนอขออนุมัติโครงการนี้เป็นส่วนหนึ่งของวิชาโครงการ

รหัสวิชา 30128 - 8501 ประเภทวิชา อุตสาหกรรม

สาขา เทคโนโลยีคอมพิวเตอร์

ภาคเรียนที่ 2 ปีการศึกษา 2567

วิทยาลัยเทคนิคจันทบุรี สถาบันการอาชีวศึกษาภาคตะวันออก

ใบเสนอโครงการ

1. ชื่อโครงการ

Pastoral Pathways

2. ประเภทโครงการ

สิ่งประดิษฐ์ด้านนวัตกรรมและเทคโนโลยีปัญญาประดิษฐ์ อุปกรณ์อัจฉริยะ

3. ผู้เสนอโครงการ

3.1 นางสาวณัฐนันท์ จิรวัฒน์

3.2 นายอดิศักดิ์ พวงนะที

4. ครูที่ปรึกษา

4.1 นายฉัตรพฤกษ์ สีทิม

4.2 นางสาวประภาพร บรรเทา

5. ข้อมูลทั่วไป

ลักษณะทั่วไป

☑ เป็นผลงานโครงการที่คิดค้นขึ้นใหม่

☐ เป็นผลงานโครงการที่พัฒนาหรือปรับปรุงเพิ่มเติมจากของเดิม

6. หลักการและเหตุผล

ในปัจจุบัน เกมออนไลน์มีบทบาทสำคัญในด้านการสร้างความบันเทิงและพัฒนาทักษะในหลากหลายด้าน โดยเฉพาะอย่างยิ่งเกมแนวจำลองสถานการณ์ (Simulation) ซึ่งเป็นหนึ่งในประเภทเกมที่ได้รับความนิยมอย่างแพร่หลาย เกมแนวทำฟาร์ม (Farming Simulation) เป็นหนึ่งในเกมประเภทนี้ที่มุ่งเน้นให้ผู้เล่นได้สัมผัสประสบการณ์การบริหารจัดการฟาร์มเสมือนจริง ไม่ว่าจะเป็นการเพาะปลูกพืชเลี้ยงสัตว์ การจัดการทรัพยากร และการวางแผนการใช้ชีวิตในฟาร์ม

เกม Pastoral Pathways ไม่เพียงแต่ช่วยสร้างความเพลิดเพลินให้แก่ผู้เล่น แต่ยังส่งเสริมให้ผู้เล่นฝึกฝนทักษะการวางแผนและการแก้ไขปัญหาอย่างมีระบบ ตลอดจนการพัฒนาทักษะการบริหารจัดการทรัพยากรอย่างมีประสิทธิภาพ นอกจากนี้ เกมยังมีส่วนช่วยในการพัฒนาความคิดสร้างสรรค์ การคิดวิเคราะห์ และการตัดสินใจในสถานการณ์จำลองที่คล้ายคลึงกับชีวิตจริง

นอกจากนี้ การเล่นเกม Pastoral Pathways ยังเป็นโอกาสที่ดีในการเรียนรู้เกี่ยวกับการเกษตรกรรม การทำฟาร์มอย่างยั่งยืน และการใช้ทรัพยากรธรรมชาติอย่างรู้คุณค่า ผู้เล่นสามารถเรียนรู้และเข้าใจวงจรชีวิตของพืชผลและสัตว์เลี้ยง รวมถึงการนำแนวคิดการเกษตรมาใช้ในฟาร์มจำลอง โดยไม่กระทบต่อสิ่งแวดล้อม

7. วัตถุประสงค์ของการวิจัย

7.1 เพื่อออกแบบและสร้างเกม Pastoral Pathways

7.2 เพื่อวัดผลประสิทธิภาพของเกม Pastoral Pathways

7.3 เพื่อวัดผลความพึงพอใจจากผู้ใช้หลังการเล่นเกม Pastoral Pathways

8. ทฤษฎี/หลักวิชาการที่นำมาใช้ในการจัดทำโครงการ

8.1 ทฤษฎีทางด้านคอมพิวเตอร์

8.1.1 ทฤษฎีการใช้โปรแกรม Unity สามารถแบ่งออกเป็นหลายด้านตามองค์ประกอบสำคัญที่จำเป็นต้องเรียนรู้ในการพัฒนาเกมหรือโปรแกรมเชิงโต้ตอบ โดยทฤษฎีพื้นฐานเหล่านี้จะช่วยให้ผู้ใช้งานสามารถสร้างโปรเจ็กต์ด้วย Unity ได้อย่างมีประสิทธิภาพ

8.1.2 ทฤษฎีการออกแบบเกม (Game Design Theory) โดย Raph Koster มีการนำเสนอแนวคิดที่สำคัญเกี่ยวกับความสนุกในเกม ซึ่งเขาเน้นว่าความสนุกนั้นเกิดจากการเรียนรู้และการเข้าใจระบบต่าง ๆ ผ่านการเล่น ผู้เล่นจะรู้สึกสนุกเมื่อพวกเขาสามารถเข้าใจกลไกของเกมและทำการตัดสินใจที่ถูกต้องตามสถานการณ์ นอกจากนี้ เกมที่ดีควรนำเสนอความท้าทายที่หลากหลายเพื่อให้ผู้เล่นสามารถพัฒนาทักษะและเรียนรู้จากความผิดพลาดได้ การมีความท้าทายที่เหมาะสมช่วยให้ผู้เล่นรู้สึกว่าตนได้พัฒนาและก้าวหน้าในเกม ซึ่งทำให้เกิดความพึงพอใจและสนุกสนาน

8.2.3 MDA Framework (Mechanics, Dynamics, Aesthetics) เป็นทฤษฎีการออกแบบเกมที่ได้รับความนิยม โดยพัฒนาโดย Robert Zubek ร่วมกับนักออกแบบเกมคนอื่น ๆ แนวคิดนี้ช่วยในการวิเคราะห์การออกแบบเกมผ่านการแยกแยะระหว่างกลไก (Mechanics) ซึ่งหมายถึงกฎและระบบพื้นฐานของเกมที่กำหนดว่าผู้เล่นสามารถทำอะไรได้บ้าง พลศาสตร์ (Dynamics) คือการทำงานร่วมกันของกลไกเหล่านี้เมื่อผู้เล่นมีปฏิสัมพันธ์กับเกม โดยจะเปลี่ยนแปลงไปตามการตัดสินใจของผู้เล่นและผลกระทบที่เกิดขึ้น และเอสเธติกส์ (Aesthetics) ซึ่งหมายถึงประสบการณ์ทางอารมณ์และความรู้สึกที่ผู้เล่นได้รับจากเกม เช่น การเล่าเรื่อง บรรยากาศ และการออกแบบภาพกราฟิก การใช้ MDA Framework ทำให้นักออกแบบสามารถวิเคราะห์และปรับปรุงองค์ประกอบของเกมเพื่อสร้างประสบการณ์ที่มีคุณค่าให้กับผู้เล่นได้อย่างมีประสิทธิภาพ

8.2.4 Aseprite เป็นโปรแกรมที่นิยมใช้ในการสร้างและแก้ไขภาพพิกเซล (pixel art) โดยเฉพาะในวงการเกมและศิลปะดิจิทัล โปรแกรมนี้มีหลักการสำคัญที่ควรพิจารณา ได้แก่ การทำงานกับเลเยอร์ที่รองรับการแยกส่วนต่าง ๆ ของภาพ เช่น ตัวละครและพื้นหลัง ทำให้สามารถทำงานแบบไม่ทำลายได้; เครื่องมือวาดหลากหลาย เช่น พู่กันและยางลบ ช่วยให้การสร้างสรรค์ภาพเป็นเรื่องง่ายและปรับแต่งได้ตามต้องการ; การจัดการพาเลตสีที่ช่วยสร้างและเลือกสีได้อย่างสะดวก;

9. ขั้นตอนการทำงานของผลงานโครงการ

9.1 กำหนดแนวคิดหลักของเกม

9.2 ปรึกษาครูที่ปรึกษาโครงการ

9.3 เสนอขออนุมัติโครงการ

9.4 ดำเนินงานตามโครงการ

9.4.1 กำหนดโครงการ

9.4.2 ออกแบบเกม

9.4.3 ศึกษารายละเอียด

9.4.4 ออกแบบเกม

9.4.5 เริ่มทำเกม

9.4.6 ทดสอบเกม

9.4.7 ประเมินผลและสรุปผล

9.4.8 จัดทำเอกสารโครงการ

9.5 จัดทำเอกสาร

9.6 แสดงผลโครงการ

9.7 ประเมินผลโครงการ

10. ขอบเขตของโครงการ

10.1 เป้าหมายของ โครงการนี้มีเป้าหมายหลักในการพัฒนาเกมฟาร์มที่มุ่งเน้นให้ผู้เล่นสามารถเรียนรู้และสนุกสนานไปกับการจัดการฟาร์มเสมือนจริง โดยจะสร้างประสบการณ์การเล่นที่สนุกสนานและมีคุณค่าทางการศึกษา สำหรับนักเรียน นักศึกษา และผู้ที่สนใจในการทำฟาร์มและการเกษตร

10.2 ฟังก์ชันหลักของเกม

10.2.1 การปลูกพืช: ผู้เล่นสามารถเลือกปลูกพืชที่หลากหลายและดูแลให้เติบโต เพื่อสร้างผลผลิตที่สามารถขายหรือใช้ในฟาร์ม

10.2.2 การเลี้ยงสัตว์: ผู้เล่นสามารถเลี้ยงสัตว์ เช่น วัว หมู และไก่

10.2.3 การจัดการทรัพยากร: ผู้เล่นต้องบริหารจัดการทรัพยากร เช่น น้ำ ปุ๋ย และอาหารสัตว์ เพื่อให้ฟาร์มมีประสิทธิภาพ

10.2.4 การพัฒนาฟาร์ม: ผู้เล่นสามารถขยายและปรับปรุงฟาร์ม โดยการสร้างสิ่งปลูกสร้างและอุปกรณ์ต่าง ๆ

10.2.5 ระบบเศรษฐกิจในเกม: ผู้เล่นสามารถซื้อขายผลิตภัณฑ์จากฟาร์ม เพื่อสร้างรายได้และใช้ในการพัฒนาฟาร์มต่อไป

10.2.6 ระบบตัวละคร NPC: ในเกมจะมี NPC ประมาณ 7 คน ซึ่งมีบทบาทสำคัญในการซื้อขายผลิตภัณฑ์จากฟาร์ม รวมถึงให้คำแนะนำและช่วยเหลือผู้เล่นในการพัฒนาฟาร์มและบริหารจัดการทรัพยากร

10.2.7 การตกแต่งฟาร์ม: ผู้เล่นสามารถตกแต่งฟาร์มของตนได้ตามต้องการ โดยเลือกใช้เครื่องมือตกแต่งต่าง ๆ เช่น รั้ว ทางเดิน และต้นไม้ เพื่อสร้างฟาร์มที่สวยงามและเป็นเอกลักษณ์

10.2.8 ระบบสภาพอากาศและฤดูกาล: เกมจะมีการจำลองสภาพอากาศและฤดูกาลที่แตกต่าง

10.3 เกมจะถูกพัฒนาให้สามารถเล่นได้บนแพลตฟอร์ม PC

10.4 สถิติที่ใช้ในการวิเคราะห์ข้อมูล

10.4.1 ค่าเฉลี่ย โดยใช้สูตร

$$\vec{\mathrm{X}} = \frac{\sum x}{N}$$

$\vec{\mathrm{X}}$ = ค่าเฉลี่ยของคะแนน

$\sum x$ = ผลรวมของคะแนน

N = จำนวน

10.4.2 ส่วนเบี่ยงเบนมาตรฐานโดยใช้สูตร

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

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

S.D. แทน ส่วนเบี่ยงเบนมาตรฐาน

X แทน คะแนนแต่ละคน

N แทน จำนวน

11. ประโยชน์ที่คาดว่าจะได้รับ

11.1 ได้เกม Pastoral Pathways

11.2 ได้ประสิทธิภาพของเกม Pastoral Pathways ทำงานได้ดี

11.3 ความพึงพอใจจากผู้ใช้หลังการเล่นเกม Pastoral Pathways อยู่ในระดับ ดี

12. ระยะเวลาที่ใช้ในการจัดทำโครงการ

ระยะเวลาที่ใช้ในการจัดทำโครงการตั้งแต่วันที่ 7 ตุลาคม 2567 ถึง 14 กุมภาพันธ์ 2568

ตารางที่ ก.1 แสดงระยะเวลาดำเนินการ

ขั้นตอนการดำเนินงาน	สัปดาห์																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1. กำหนดโครงการ	◄	►																
2. ออกแบบเกม		◄	►															
3. ศึกษารายละเอียด				◄	►													
4. ออกแบบระบบเกม						◄	►											
5. เริ่มทำเกม								◄	►									
6. ทดสอบเกม															◄	►		
7. ประเมินและสรุปผล																◄	►	
8. จัดทำเอกสาร																	◄	►

ลงชื่อ ..... **ลงชื่อ.....**

(นางสาวณัฐนันท์ จิรวัฒน์) (นายอดิศักดิ์ พวงนะที)

ผู้จัดทำโครงการ ผู้จัดทำโครงการ

ความคิดเห็นของครูที่ปรึกษา/ครูผู้สอนโครงการ/คณะกรรมการผู้ตรวจสอบโครงการ

.....

ลงชื่อ.....

(นายฉัตรพฤกษ์ สีทิม)

ครูผู้สอนโครงการ

ความคิดเห็นของหัวหน้าสาขาเทคโนโลยีคอมพิวเตอร์

โครงการนี้เห็นสมควร ☐ อนุมัติ ☐ ไม่อนุมัติ

ลงชื่อ.....

(นายศัตราวุธ ศรีชาติ)

หัวหน้าสาขาเทคโนโลยีคอมพิวเตอร์

วันที่ ..../..../....

ความคิดเห็นของรองผู้อำนวยการฝ่ายวิชาการ

โครงการนี้เห็นสมควร ☐ อนุมัติ ☐ ไม่อนุมัติ

ลงชื่อ.....

(นายชูชาติ รูปงาม)

รองผู้อำนวยการฝ่ายวิชาการ

วันที่ ..../..../....

ความคิดเห็นของผู้อำนวยการ

โครงการนี้เห็นสมควร ☐ อนุมัติ ☐ ไม่อนุมัติ

ลงชื่อ.....

(นายกมล เรียงไธสง)

ผู้อำนวยการวิทยาลัยเทคนิคจันทบุรี

วันที่ ..../..../....

ภาคผนวก. ข

ผลแบบประเมินหาความพึงพอใจ

แบบประเมินความพึงพอใจ

เกม Pastoral Pathways

คำชี้แจง 1) แบบประเมินชุดนี้จัดทำขึ้นโดยนักเรียนประกาศนียบัตรวิชาชีพชั้นปีที่ 3 สาขาวิชาเทคนิคคอมพิวเตอร์ วิทยาลัยเทคนิคจันทบุรี โดยมีวัตถุประสงค์เพื่อสร้างเกม โดยใช้โปรแกรม Unity

ตอนที่ 1 ข้อมูลทั่วไปของผู้ตอบแบบสอบถาม

เพศ ☐ ชาย ☐ หญิง **อายุ** ☐ 17 – 20 ปี ☐ 21 – 25 ปี

ชั้นปี ☐ ปวช.1 ☐ ปวช.2 ☐ ปวช.3

ตอนที่ 2 แบบประเมินความพึงพอใจของผู้เล่นเกม

หัวข้อการประเมิน	**ระดับความพึงพอใจ**				
	5	**4**	**3**	**2**	**1**
ด้านการออกแบบเกม					
1.1 การออกแบบมีความสวยงามเหมาะสม					
1.2 ความน่าสนใจของภาพและเสียงของเกม					
1.3 เนื้อหามีความเหมาะสม					
ด้านนำเสนอ					
2.1 ความน่าสนใจของเกม					
2.2 ความชัดเจนของข้อมูล					
2.3 ความเหมาะสมในการนำเสนอข้อมูล					
ด้านประโยชน์					
3.1 ได้รับความรู้เกี่ยวกับอุปกรณ์คอมพิวเตอร์ผ่านเกม					
3.2 ได้รู้จักโปรแกรม Unity ที่ใช้ในการสร้างเกม					
3.3 สามารถนำความรู้ที่ได้ไปใช้ในชีวิตประจำวันได้					

ข้อเสนอแนะ.....

.....

ภาคผนวก. ค

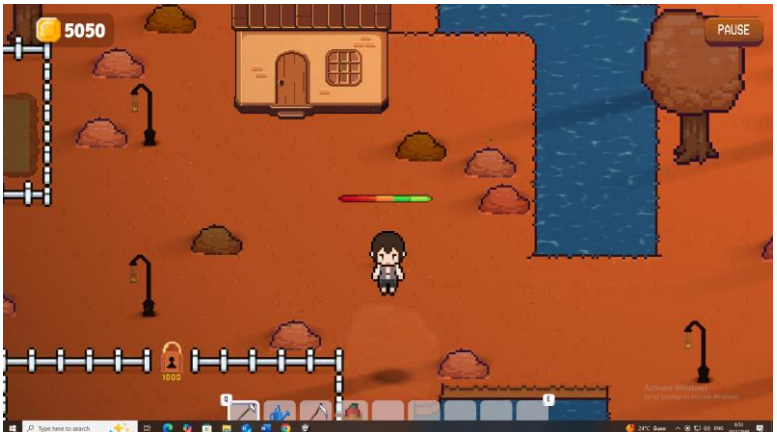
คู่มือการใช้งาน

คู่มือการใช้งานเกม Pastoral Pathways

Pastoral Pathways ยังมีจุดเด่นในด้านการส่งเสริมความรู้เกี่ยวกับการเกษตรกรรมอย่างยั่งยืน โดยผู้เล่นสามารถเรียนรู้เกี่ยวกับวงจรชีวิตของพืชผลและสัตว์เลี้ยงได้แบบละเอียด รวมถึงการใช้ทรัพยากรธรรมชาติอย่างมีคุณค่าในการจัดการฟาร์มเสมือน ผู้เล่นจะได้เห็นภาพชัดเจนว่าการทำฟาร์มที่ยั่งยืนนั้นมีลักษณะอย่างไร ไม่ว่าจะเป็นการใช้ปุ๋ยอินทรีย์ ลดการใช้สารเคมี หรือการบริหารจัดการน้ำและพลังงานอย่างเหมาะสม เพื่อไม่ให้เกิดการทำลายสิ่งแวดล้อมและสร้างสมดุลในระบบนิเวศ นอกจากนี้ ผู้เล่นจะต้องค่อยเก็บเงินในการปลดล็อกและ ผ่านออกไปเพื่อจบเกม

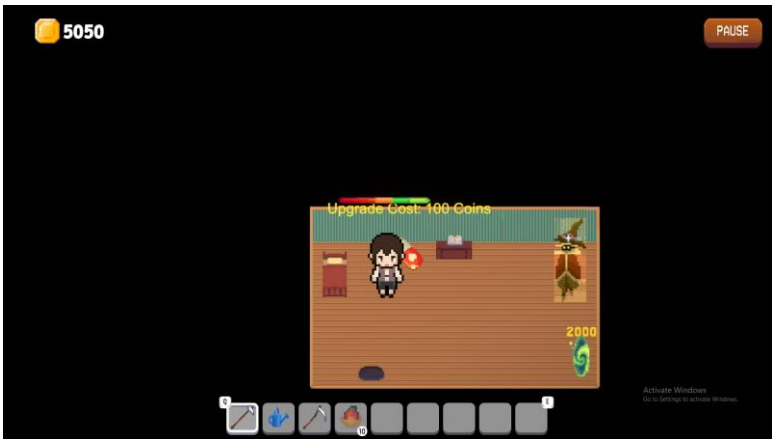
1.การใช้งานเกม Pastoral Pathways

1.1 กด WASD ในการเคลื่อนที่ไปมาในเกม



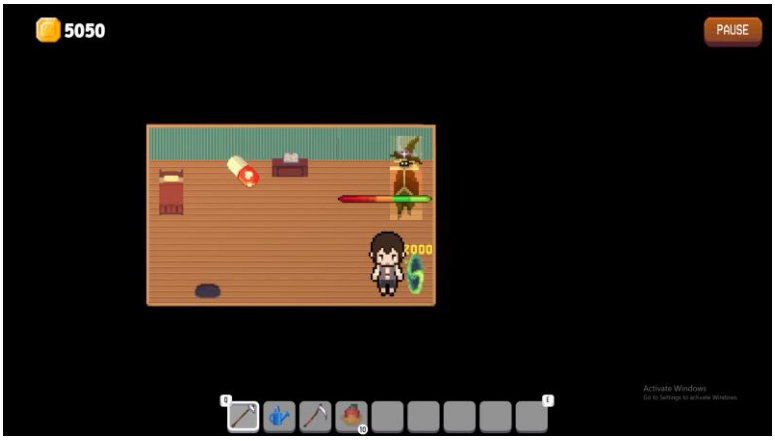
รูปภาพที่ ค.1 การเคลื่อนไหวในเกม

1.2 กดคลิกซ้ายในการใช้สิ่งของต่างๆ



รูปภาพที่ ค.2 คลิกซ้ายเพื่อใช้สิ่งของ

1.3 กด F เพื่อกดใช้งานประตูกับเตียงนอน

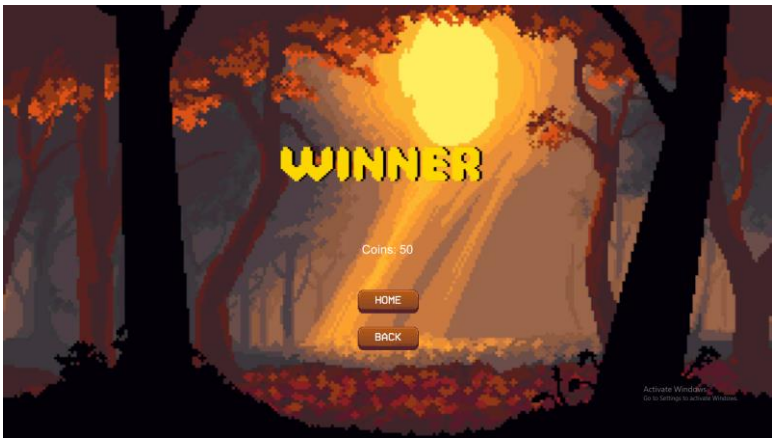


รูปภาพที่ ค.3 กด F เพื่อใช้งาน

1.4 กด shift เพื่อเป็นการวิ่ง

รูปภาพที่ ค.4 กด shift เพื่อเป็นการวิ่ง

1.5 เก็บ Coin เพื่อเข้าประตูเพื่อชนะ



รูปภาพที่ ค.5 จบเกมเมื่อเก็บ Coin ถึงที่กำหนด

2. การติดตั้ง

2.1 กดเข้าลิงก์ดาวน์โหลด

2.2 ทำการแตกไฟล์ WinRAR หรือ 7Zip

2.3 เปิดไฟล์ Pastoral Pathways.exe

3. ข้อควรระวัง

3.1 หากในระหว่างขั้นตอนดาวน์โหลดมีปัญหาจะไม่สามารถเล่นเกมได้

ภาคผนวก. ง

Source Code

ในส่วนของเกมสรุปการเขียนเกม Source Code มีไฟล์ทั้งหมด 5 ไฟล์ดังนี้

1. PlayerController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.InputSystem;
using UnityEngine.SceneManagement;
namespace HappyHarvest
{
    public class PlayerController : MonoBehaviour
    {
        public static PlayerController Instance;
        [SerializeField] private float YOffset = 10f;
        public Sprite[] staminaSprites;
        [SerializeField] private Vector3 CustomStartPosition; // เพิ่มตัวแปรสำหรับกำหนดตำแหน่งเริ่มต้น
        [SerializeField] private float StaminaBarScaleMultiplierX = 1.0f; // ปรับขนาดในแกน X
        [SerializeField] private float StaminaBarScaleMultiplierY = 1.0f; // ปรับขนาดในแกน Y
        public float MaxStamina = 100f;
        private bool m_IsNearBed = false; // เพิ่มตัวแปรตรวจสอบว่าอยู่ใกล้เตียงหรือไม่
         private Collider2D m_CurrentBedCollider = null;
        public float CurrentStamina { get; private set; }
        public float StaminaRecoveryRate = 5f;
        public float StaminaUsageRate = 15f; // ใช้ต่อวินาที
        public bool IsSprinting => m_IsSprinting;
        [SerializeField] private SpriteRenderer StaminaBarFill; // ส่วนที่เติมของ Stamina Bar
```

```
[SerializeField] private float SprintMultiplier = 1.5f;
private float staminaBarSmoothSpeed = 10f; // ความเร็วในการปรับแถบให้สมูท
private float targetStaminaValue = 0f;
private bool m_IsSprinting = false;
private bool m_CanSprint = true;
public InputActionAsset InputAction;
public float Speed = 4.0f;

public SpriteRenderer Target;
public Transform ItemAttachBone;



public int Coins
{
    get => m_Coins;
    set
    {
        m_Coins = value;
        UIHandler.UpdateCoins(Coins);
    }
}

public InventorySystem Inventory => m_Inventory;
public Animator Animator => m_Animator;

//This is private as we don't want to be able to set coins without going through the accessor above that ensure
//the UI is updated, but is tagged as SerializedField so it appear in the editor so designer can set the starting
//amount of coins
[SerializeField]
```

```
[Header("Player Stats")]
private int m_Coins = 10;

[SerializeField]
private InventorySystem m_Inventory;
private Door m_CurrentTargetDoor = null;
private DoorEnter m_CurrentTargetDoorEnter = null; // ประตูที่ผู้เล่นกำลังยืนอยู่ใกล้
private InputAction m_UseAction;
private Rigidbody2D m_Rigidbody;

private Vector3 m_SpawnPosition;

private InputAction m_MoveAction;
private InputAction m_NextItemAction;
private InputAction m_PrevItemAction;
private InputAction m_UseItemAction;

private Vector3 m_CurrentWorldMousePos;
private Vector2 m_CurrentLookDirection;
private Vector3Int m_CurrentTarget;

private TargetMarker m_TargetMarker;

private bool m_HasTarget = false;
private bool m_IsOverUI = false;

private bool m_CanControl = true;

private Animator m_Animator;

private InteractiveObject m_CurrentInteractiveTarget = null;
```

```
    private Collider2D[] m_CollidersCache = new Collider2D[8];

    private Dictionary<Item, ItemInstance> m_ItemVisualInstance = new();

    private int m_DirXHash = Animator.StringToHash("DirX");
    private int m_DirYHash = Animator.StringToHash("DirY");
    private int m_SpeedHash = Animator.StringToHash("Speed");

    void Awake()
    {
        if (GameManager.Instance.Player != null)
        {
            Destroy(gameObject);
            return;
        }

        m_Rigidbody = GetComponent<Rigidbody2D>();
        m_Animator = GetComponentInChildren<Animator>();
        m_TargetMarker = Target.GetComponent<TargetMarker>();
        m_TargetMarker.Hide();

        //we can only set DontDestroyOnLoad root object, so ensure its root (Level
Designer sometime place the character
        //prefab already in the scene and can sometime tuck it under other object
in the hierarchy)
        gameObject.transform.SetParent(null);

        GameManager.Instance.Player = this;
        DontDestroyOnLoad(gameObject);
        CurrentStamina = MaxStamina;
        if (Instance == null)
```

```
            {
                Instance = this; // ถ้ายังไม่มีการตั้งค่า Instance, ให้ตั้งค่าให้ตัวเอง
                DontDestroyOnLoad(gameObject); // ถ้าต้องการให้ PlayerController ไม่หายไป
เมื่อโหลดฉากใหม่
            }
            else
            {
                Destroy(gameObject); // ถ้ามี Instance อยู่แล้ว จะทำการลบตัวใหม่
            }
        }

        void Start()
        {

            m_Rigidbody = GetComponent<Rigidbody2D>();

            // สร้าง input action สำหรับการใช้/ทำลายประตู
    m_UseAction = new InputAction("Use", binding: "<Keyboard>/f");
    m_UseAction.Enable();
    m_UseAction.performed += ctx =>
    {
        Debug.Log("F key pressed");
        TryDestroyDoor();
    };
            //Retrieve the action from the InputAction asset, enable them and add the
callbacks.

            //Move action doesn't have any callback as it will be polled in the
movement code directly.
            m_MoveAction = InputAction.FindAction("Gameplay/Move");
            m_MoveAction.Enable();
```

```
m_NextItemAction = InputAction.FindAction("Gameplay/EquipNext");
m_PrevItemAction = InputAction.FindAction("Gameplay/EquipPrev");

m_NextItemAction.Enable();
m_NextItemAction.performed += context =>
{
    ToggleToolVisual(false);
    m_Inventory.EquipNext();
    ToggleToolVisual(true);
};

m_PrevItemAction.Enable();
m_PrevItemAction.performed += context =>
{
    ToggleToolVisual(false);
    m_Inventory.EquipPrev();
    ToggleToolVisual(true);
};

m_UseItemAction = InputAction.FindAction("Gameplay/Use");
m_UseItemAction.Enable();

m_UseItemAction.performed += context => UseObject();

m_CurrentLookDirection = Vector2.right;

m_Inventory.Init();

foreach (var entry in m_Inventory.Entries)
{
```

```
            if (entry.Item != null)
                CreateItemVisual(entry.Item);
        }
        ToggleToolVisual(true);

        UIHandler.UpdateInventory(m_Inventory);
        UIHandler.UpdateCoins(m_Coins);
    }

    private void Update()
    {
        HandleStamina();
        if (m_IsNearBed && m_UseAction.triggered) // ตรวจสอบว่าอยู่ใกล้เตียงและกด F
        {
            RestoreStamina();
        }
        if (m_CurrentTargetDoor != null && m_UseAction.triggered)
        {
            TryDestroyDoor();
        }
        if (m_CurrentTargetDoorEnter != null && m_UseAction.triggered)
{
    TryEnterDoor();
}
        m_IsOverUI = EventSystem.current.IsPointerOverGameObject();
        m_CurrentInteractiveTarget = null;
        m_HasTarget = false;

        if (!IsMouseOverGameWindow())
        {
            UIHandler.ChangeCursor(UIHandler.CursorType.System);
```

```
            return;
        }

        if (!m_CanControl || m_IsOverUI)
        {
            if (m_IsOverUI) UIHandler.ChangeCursor(UIHandler.CursorType.Interact);
            return;
        }

        m_CurrentWorldMousePos =
Camera.main.ScreenToWorldPoint(Mouse.current.position.ReadValue());
        //check if we are above an interactive object
        var overlapCount =
Physics2D.OverlapPointNonAlloc(m_CurrentWorldMousePos, m_CollidersCache, 1 <<
31);
        for (int i = 0; i < overlapCount; ++i)
        {
            var obj = m_CollidersCache[i].GetComponent<InteractiveObject>();
            if (obj != null)
            {
                m_CurrentInteractiveTarget = obj;
                m_HasTarget = false;
                UIHandler.ChangeCursor(UIHandler.CursorType.Interact);
                return;
            }
        }

        //If we reached here, we are not above UI or an interactive object, so set
the cursor to the normal one
        UIHandler.ChangeCursor(UIHandler.CursorType.Normal);
```

```
            var grid = GameManager.Instance.Terrain?.Grid;

            //some scene may not have a terrain (interior scene)
            if (grid != null)
            {
                var currentCell = grid.WorldToCell(transform.position);
                var pointedCell = grid.WorldToCell(m_CurrentWorldMousePos);

                currentCell.z = 0;
                pointedCell.z = 0;

                var toTarget = pointedCell - currentCell;

                if (Mathf.Abs(toTarget.x) > 1)
                {
                    toTarget.x = (int)Mathf.Sign(toTarget.x);
                }

                if (Mathf.Abs(toTarget.y) > 1)
                {
                    toTarget.y = (int)Mathf.Sign(toTarget.y);
                }


                m_CurrentTarget = currentCell + toTarget;
                Target.transform.position =
GameManager.Instance.Terrain.Grid.GetCellCenterWorld(m_CurrentTarget);

                if (m_Inventory.EquippedItem != null
                    && m_Inventory.EquippedItem.CanUse(m_CurrentTarget))
                {
```

```
                m_HasTarget = true;
                m_TargetMarker.Activate();
            }
            else
            {
                m_TargetMarker.Hide();
            }
        }

        if (Keyboard.current.f5Key.wasPressedThisFrame)
        {
            SaveSystem.Save();
        }
        else if (Keyboard.current.f9Key.wasPressedThisFrame)
        {
            SaveSystem.Load();
        }
    }
void TryEnterDoor()
{
    if (m_CurrentTargetDoorEnter != null)
    {
        // ตรวจสอบว่า coin ของผู้เล่นเพียงพอหรือไม่
        if (m_CurrentTargetDoorEnter.CanEnter(m_Coins))
        {
            Debug.Log("Entering door to 'End' scene");

            // ลด coin เมื่อผู้เล่นเข้า
            m_Coins -= m_CurrentTargetDoorEnter.requiredCoins;
            UIHandler.UpdateCoins(m_Coins); // ปรับ UI coin
```

```
            // สามารถใช้ SceneManager เพื่อเปลี่ยนฉาก
            SceneManager.LoadScene("Winner");
        }
        else
        {
            Debug.Log("Not enough coins to enter the door.");
        }
    }
    else
    {
        Debug.Log("No door to enter.");
    }
}

        void TryDestroyDoor()
{
    if (m_CurrentTargetDoor != null)
    {
        Debug.Log("Attempting to destroy door");

        // ลด Coin ก่อน
        if (m_CurrentTargetDoor.CanDestroy(m_Coins))
        {
            m_Coins -= m_CurrentTargetDoor.requiredCoins;
            UIHandler.UpdateCoins(m_Coins); // ปรับ UI ให้ทันที
            m_CurrentTargetDoor.DestroyDoor();
        }
        else
        {
            Debug.Log("Not enough coins to destroy the door.");
        }
```

```
    }
    else
    {
        Debug.Log("No door nearby.");
    }
}



        // เช็คว่า player ยืนอยู่ใกล้ประตูหรือไม่
void OnTriggerEnter2D(Collider2D other)
{
    // Door Trigger Code
    Door door = other.GetComponent<Door>();
    if (door != null)
    {
        m_CurrentTargetDoor = door;
        Debug.Log("Nearby door detected");
        UIHandler.ChangeCursor(UIHandler.CursorType.Interact);
    }

    DoorEnter doorEnter = other.GetComponent<DoorEnter>();
    if (doorEnter != null)
    {
        m_CurrentTargetDoorEnter = doorEnter;
        Debug.Log("Nearby door enter detected");
        UIHandler.ChangeCursor(UIHandler.CursorType.Interact);
    }

    // Bed Trigger Code
    if (other.CompareTag("Bed"))
```

```
    {
        m_IsNearBed = true;
        m_CurrentBedCollider = other; // Store the bed's collider
        Debug.Log("Near bed!");
        UIHandler.ChangeCursor(UIHandler.CursorType.Interact);
    }
}

void OnTriggerExit2D(Collider2D other)
        {
            // Door Exit Code
            if (m_CurrentTargetDoor != null && other.GetComponent<Door>() ==
m_CurrentTargetDoor)
            {
                m_CurrentTargetDoor = null;
                UIHandler.ChangeCursor(UIHandler.CursorType.Normal);
            }
            if (m_CurrentTargetDoorEnter != null && other.GetComponent<DoorEnter>()
== m_CurrentTargetDoorEnter)
            {
                m_CurrentTargetDoorEnter = null;
                UIHandler.ChangeCursor(UIHandler.CursorType.Normal);
            }

            // Bed Exit Code
            if (other == m_CurrentBedCollider)
            {
                m_IsNearBed = false;
                m_CurrentBedCollider = null;
                Debug.Log("Left bed!");
                UIHandler.ChangeCursor(UIHandler.CursorType.Normal);
```

```
        }
    }
    void UseObject()
    {
        if(m_IsOverUI)
            return;

        if (m_CurrentInteractiveTarget != null)
        {
            m_CurrentInteractiveTarget.InteractedWith();
            return;
        }

        if (m_Inventory.EquippedItem != null &&
m_Inventory.EquippedItem.NeedTarget() && !m_HasTarget) return;

        UseItem();
    }

    void FixedUpdate()
    {
        var move = m_MoveAction.ReadValue<Vector2>();

        //note: == and != for vector2 is overriden to take in account floating point
imprecision.
        if (move != Vector2.zero)
        {
            SetLookDirectionFrom(move);
        }
        else
        {
```

```
            //we aren't moving, look direction is based on the currently aimed toward
point
            if (IsMouseOverGameWindow())
            {
                Vector3 posToMouse = m_CurrentWorldMousePos - transform.position;
                SetLookDirectionFrom(posToMouse);
            }
        }

        var movement = move * Speed;
        var speed = movement.sqrMagnitude;

        m_Animator.SetFloat(m_DirXHash, m_CurrentLookDirection.x);
        m_Animator.SetFloat(m_DirYHash, m_CurrentLookDirection.y);
        m_Animator.SetFloat(m_SpeedHash, speed);

        m_Rigidbody.MovePosition(m_Rigidbody.position + movement *
Time.deltaTime);
        var moveInput = m_MoveAction.ReadValue<Vector2>();
    float speedMultiplier = m_IsSprinting ? SprintMultiplier : 1.0f;

    var velocity = moveInput * Speed * speedMultiplier;
    var currentSpeed = velocity.sqrMagnitude;

    m_Animator.SetFloat(m_DirXHash, m_CurrentLookDirection.x);
    m_Animator.SetFloat(m_DirYHash, m_CurrentLookDirection.y);
    m_Animator.SetFloat(m_SpeedHash, currentSpeed);

    m_Rigidbody.MovePosition(m_Rigidbody.position + velocity * Time.deltaTime);
}
```

```
void HandleStamina()
{
        var move = m_MoveAction.ReadValue<Vector2>();

        if (move != Vector2.zero && Keyboard.current.shiftKey.isPressed &&
CurrentStamina > 0 && m_CanSprint && m_CanControl)
        {
            m_IsSprinting = true;
            CurrentStamina -= StaminaUsageRate * Time.deltaTime;

            if (CurrentStamina <= 0)
            {
                CurrentStamina = 0;
                m_IsSprinting = false;
                m_CanSprint = false;
            }
        }
        else
        {
            m_IsSprinting = false;
            // เอาส่วนการฟื้นฟู stamina อัตโนมัติออก
            // CurrentStamina += StaminaRecoveryRate * Time.deltaTime;  <-- เอาออก

            if (CurrentStamina > MaxStamina)
            {
                CurrentStamina = MaxStamina;
                m_CanSprint = true;
            }
        }
    // อัปเดตรูปภาพตามเปอร์เซ็นต์ Stamina
    if (StaminaBarFill != null && staminaSprites.Length == 6)
```

```
    {
        float staminaRatio = CurrentStamina / MaxStamina;

        if (staminaRatio >= 0.83f)
            StaminaBarFill.sprite = staminaSprites[5]; // 100%
        else if (staminaRatio >= 0.67f)
            StaminaBarFill.sprite = staminaSprites[4]; // 83%
        else if (staminaRatio >= 0.50f)
            StaminaBarFill.sprite = staminaSprites[3]; // 67%
        else if (staminaRatio >= 0.33f)
            StaminaBarFill.sprite = staminaSprites[2]; // 50%
        else if (staminaRatio >= 0.17f)
            StaminaBarFill.sprite = staminaSprites[1]; // 33%
        else
            StaminaBarFill.sprite = staminaSprites[0]; // 17%
    }
}
void RestoreStamina()
        {
            CurrentStamina = MaxStamina; // ฟื้นฟู Stamina เต็ม
            Debug.Log("Stamina restored!");
            GameManager.Instance.SetTimeTo6AM();
        }
        bool IsMouseOverGameWindow()
        {
            return !(0 > Input.mousePosition.x || 0 > Input.mousePosition.y ||
Screen.width < Input.mousePosition.x || Screen.height < Input.mousePosition.y);
        }

        void SetLookDirectionFrom(Vector2 direction)
        {
```

```
    if (Mathf.Abs(direction.x) > Mathf.Abs(direction.y))
    {
        m_CurrentLookDirection = direction.x > 0 ? Vector2.right : Vector2.left;
    }
    else
    {
        m_CurrentLookDirection = direction.y > 0 ? Vector2.up : Vector2.down;
    }
}

public bool CanFitInInventory(Item item, int count)
{
    return m_Inventory.CanFitItem(item, count);
}

public bool AddItem(Item newItem)
{
    CreateItemVisual(newItem);
    return m_Inventory.AddItem(newItem);
}

public void SellItem(int inventoryIndex, int count)
{
    if (inventoryIndex < 0 || inventoryIndex > Inventory.Entries.Length)
        return;

    var item = Inventory.Entries[inventoryIndex].Item;

    if (item == null || !(item is Product product))
        return;
```

```
    int actualCount = m_Inventory.Remove(inventoryIndex, count);

    m_Coins += actualCount * product.SellPrice;
    UIHandler.UpdateCoins(m_Coins);
    UIHandler.PlayBuySellSound(transform.position);
}

public bool BuyItem(Item item)
{
    if (item.BuyPrice > m_Coins)
    {
        return false;
    }

    m_Coins -= item.BuyPrice;
    UIHandler.UpdateCoins(m_Coins);
    UIHandler.PlayBuySellSound(transform.position);
    AddItem(item);
    return true;
}

public void ChangeEquipItem(int index)
{
    //Disable the current item if there is one
    ToggleToolVisual(false);

    m_Inventory.EquipItem(index);

    ToggleToolVisual(true);
}
```

```
        public void ToggleControl(bool canControl)
        {
            m_CanControl = canControl;
            if (canControl)
            {
                m_MoveAction.Enable();
                m_NextItemAction.Enable();
                m_PrevItemAction.Enable();
                m_UseItemAction.Enable();
            }
            else
            {
                m_MoveAction.Disable();
                m_NextItemAction.Disable();
                m_PrevItemAction.Disable();
                m_UseItemAction.Disable();
            }
        }

public void UseItem()
{
    if (m_Inventory.EquippedItem == null)
        return;

    var equippedItem = m_Inventory.EquippedItem;

    // ตรวจสอบว่า Stamina เพียงพอสำหรับการใช้งาน
    if (equippedItem.StaminaCost > CurrentStamina)
    {
        Debug.Log("Not enough stamina to use the item.");
        return; // หยุดการใช้งานถ้า Stamina ไม่เพียงพอ
```

```
}

// ลดค่า Stamina ตาม StaminaCost
CurrentStamina -= equippedItem.StaminaCost;

var previousEquipped = m_Inventory.EquippedItem;

m_Inventory.UseEquippedObject(m_CurrentTarget);

if (m_ItemVisualInstance.ContainsKey(previousEquipped))
{
    var visual = m_ItemVisualInstance[previousEquipped];

    m_Animator.SetTrigger(visual.AnimatorHash);

    if (visual.Animator != null)
    {
        if (!visual.Instance.activeInHierarchy)
        {
            var current = visual.Instance.transform;
            while (current != null)
            {
                current.gameObject.SetActive(true);
                current = current.parent;
            }
        }

        visual.Animator.SetFloat(m_DirXHash, m_CurrentLookDirection.x);
        visual.Animator.SetFloat(m_DirYHash, m_CurrentLookDirection.y);
        visual.Animator.SetTrigger("Use");
    }
```

```
    }

    if (m_Inventory.EquippedItem == null)
    {
        StartCoroutine(DelayedObjectDisable(previousEquipped));
    }
}


    IEnumerator DelayedObjectDisable(Item item)
    {
        yield return new WaitForSeconds(1.0f);
        ToggleVisualExplicit(false, item);
    }

    public void Save(ref PlayerSaveData data)
    {
        data.Position = m_Rigidbody.position;
        data.Coins = m_Coins;
        data.Inventory = new List<InventorySaveData>();
        m_Inventory.Save(ref data.Inventory);
    }

    public void Load(PlayerSaveData data)
    {
        m_Coins = data.Coins;
        m_Inventory.Load(data.Inventory);

        m_Rigidbody.position = data.Position;
    }
```

```
    void ToggleToolVisual(bool enable)
    {
        if (m_Inventory.EquippedItem != null &&
m_ItemVisualInstance.TryGetValue(m_Inventory.EquippedItem, out var itemVisual))
        {
            itemVisual.Instance.SetActive(false);
        }
    }

    void ToggleVisualExplicit(bool enable, Item item)
    {
        if (item != null && m_ItemVisualInstance.TryGetValue(item, out var
itemVisual))
        {
            itemVisual.Instance.SetActive(false);
        }
    }

    void CreateItemVisual(Item item)
    {
        if (item.VisualPrefab != null && !m_ItemVisualInstance.ContainsKey(item))
        {
            var newVisual = Instantiate(item.VisualPrefab, ItemAttachBone, false);
            newVisual.SetActive(false);

            m_ItemVisualInstance[item] = new ItemInstance()
            {
                Instance = newVisual,
                Animator = newVisual.GetComponentInChildren<Animator>(),
                AnimatorHash = Animator.StringToHash(item.PlayerAnimatorTriggerUse)
            };
```

```
            }
        }
    }

    class ItemInstance
    {
        public GameObject Instance;
        public Animator Animator;
        public int AnimatorHash;
    }

    [System.Serializable]
    public struct PlayerSaveData
    {
        public Vector3 Position;

        public int Coins;
        public List<InventorySaveData> Inventory;
    }
}
```

2. Storage

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

namespace HappyHarvest
{
    public class Storage
    {
        public List<InventorySystem.InventoryEntry> Content { get; private set; }

        public Storage()
        {
            Content = new List<InventorySystem.InventoryEntry>();
        }

        public void Store(InventorySystem.InventoryEntry entry)
        {
            //we won't have thousands of objects types stored, so there should be no performance problem on simply searching
            //for the key. But as usual : profile. If profiling show this to be massively underperformant, switch over to a
            //lookup data format like Dictionary.
            var idx = Content.FindIndex(inventoryEntry => inventoryEntry.Item.Key == entry.Item.Key);
            if (idx != -1)
            {
                Content[idx].StackSize += entry.StackSize;
            }
            else
            {
```

```
                Content.Add(new InventorySystem.InventoryEntry()
                {
                    Item = entry.Item,
                    StackSize = entry.StackSize
                });
            }
        }

        // Will return how much was actually retrieve (in case more than what's stored is asked)
        public int Retrieve(int contentIndex, int amount)
        {
            Debug.Assert(contentIndex < Content.Count, "Tried to retrieve a non existing entry from storage");

            int actualAmount = Mathf.Min(amount, Content[contentIndex].StackSize);

            Content[contentIndex].StackSize -= actualAmount;

            return actualAmount;
        }
    }
}
```

3. DayCycleHandler

```
using System;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Rendering.Universal;

using UnityEngine.UIElements;


#if UNITY_EDITOR

using UnityEditor;

using UnityEditor.UIElements;

#endif


namespace HappyHarvest

{


    /// <summary>

    /// Handle the cycle of Day and Night. Everything that need to change across time will register itself to this handler

    /// which will update it when it update (e.g. ShadowInstance, Interpolator etc.).

    /// The ticking of that system can be stopped, this is useful e.g. if the game is put in pause (or need to do cutscene

    /// etc..)
```

```
/// </summary>
[DefaultExecutionOrder(10)]
public class DayCycleHandler : MonoBehaviour
{
    public Transform LightsRoot;

    [Header("Day Light")]
    public Light2D DayLight;
    public Gradient DayLightGradient;

    [Header("Night Light")]
    public Light2D NightLight;
    public Gradient NightLightGradient;

    [Header("Ambient Light")]
    public Light2D AmbientLight;
    public Gradient AmbientLightGradient;

    [Header("RimLights")]
    public Light2D SunRimLight;
    public Gradient SunRimLightGradient;
```

```
    public Light2D MoonRimLight;
    public Gradient MoonRimLightGradient;

    [Tooltip("The angle 0 = upward, going clockwise to 1 along the day")]
    public AnimationCurve ShadowAngle;
    [Tooltip("The scale of the normal shadow length (0 to 1) along the day")]
    public AnimationCurve ShadowLength;

    private List<ShadowInstance> m_Shadows = new();
    private List<LightInterpolator> m_LightBlenders = new();

    private void Awake()
    {
        GameManager.Instance.DayCycleHandler = this;
    }

    /// <summary>
    /// We use an explicit ticking function instead of update so the GameManager can potentially freeze or change how
    /// time pass
    /// </summary>
    public void Tick()
```

```
        {
            UpdateLight(GameManager.Instance.CurrentDayRatio);
        }

        public void UpdateLight(float ratio)
        {
            DayLight.color = DayLightGradient.Evaluate(ratio);
            NightLight.color = NightLightGradient.Evaluate(ratio);

#if UNITY_EDITOR
            //the test between the define will only happen in editor and not in build, as it is assumed those will be set
            //in build. But in editor we may want to test without those set. (those were added later in development so
            //some test scene didn't have those set and we wanted to be able to still test those)
            if(AmbientLight != null)
#endif
                AmbientLight.color = AmbientLightGradient.Evaluate(ratio);

#if UNITY_EDITOR
            if(SunRimLight != null)
```

```
#endif
                SunRimLight.color = SunRimLightGradient.Evaluate(ratio);

#if UNITY_EDITOR
            if(MoonRimLight != null)
#endif
                MoonRimLight.color = MoonRimLightGradient.Evaluate(ratio);

            LightsRoot.rotation = Quaternion.Euler(0,0, 360.0f * ratio);

            UpdateShadow(ratio);
        }

        void UpdateShadow(float ratio)
        {
            var currentShadowAngle = ShadowAngle.Evaluate(ratio);
            var currentShadowLength = ShadowLength.Evaluate(ratio);

            var opposedAngle = currentShadowAngle + 0.5f;
            while (currentShadowAngle > 1.0f)
                currentShadowAngle -= 1.0f;
```

```
            foreach (var shadow in m_Shadows)
            {
                var t = shadow.transform;
                //use 1.0-angle so that the angle goes clo
                t.eulerAngles = new Vector3(0,0, currentShadowAngle * 360.0f);
                t.localScale = new Vector3(1, 1f * shadow.BaseLength *
currentShadowLength, 1);
            }

            foreach (var handler in m_LightBlenders)
            {
                handler.SetRatio(ratio);
            }
        }

        public void Save(ref DayCycleHandlerSaveData data)
        {
            //data.TimeOfTheDay = m_CurrentTimeOfTheDay;
        }

        public void Load(DayCycleHandlerSaveData data)
```

```
        {
            //m_CurrentTimeOfTheDay = data.TimeOfTheDay;
            //StartingTime = m_CurrentTimeOfTheDay;
        }

        public static void RegisterShadow(ShadowInstance shadow)
        {
#if UNITY_EDITOR
            //in the editor when not running, we find the instance manually. Less efficient but not a problem at edit time
            //allow to be able to previz shadow in editor
            if (!Application.isPlaying)
            {
                var instance = GameObject.FindObjectOfType<DayCycleHandler>();
                if (instance != null)
                {
                    instance.m_Shadows.Add(shadow);
                }
            }
            else
            {
#endif
```

```
            GameManager.Instance.DayCycleHandler.m_Shadows.Add(shadow);
#if UNITY_EDITOR
        }
#endif
    }

    public static void UnregisterShadow(ShadowInstance shadow)
    {
#if UNITY_EDITOR
        //in the editor when not running, we find the instance manually. Less efficient but not a problem at edit time
        //allow to be able to previz shadow in editor
        if (!Application.isPlaying)
        {
            var instance = GameObject.FindObjectOfType<DayCycleHandler>();
            if (instance != null)
            {
                instance.m_Shadows.Remove(shadow);
            }
        }
        else
        {
```

```
#endif
            if(GameManager.Instance?.DayCycleHandler != null)
                GameManager.Instance.DayCycleHandler.m_Shadows.Remove(shadow);
#if UNITY_EDITOR
        }
#endif
    }

    public static void RegisterLightBlender(LightInterpolator interpolator)
    {
#if UNITY_EDITOR
        //in the editor when not running, we find the instance manually. Less efficient but not a problem at edit time
        //allow to be able to previz shadow in editor
        if (!Application.isPlaying)
        {
            var instance = FindObjectOfType<DayCycleHandler>();
            if (instance != null)
            {
                instance.m_LightBlenders.Add(interpolator);
            }
        }
```

```
            else
            {
#endif
            GameManager.Instance.DayCycleHandler.m_LightBlenders.Add(interpolator);
#if UNITY_EDITOR
            }
#endif
        }

        public static void UnregisterLightBlender(LightInterpolator interpolator)
        {
#if UNITY_EDITOR
            //in the editor when not running, we find the instance manually. Less
efficient but not a problem at edit time
            //allow to be able to previz shadow in editor
            if (!Application.isPlaying)
            {
                var instance = FindObjectOfType<DayCycleHandler>();
                if (instance != null)
                {
                    instance.m_LightBlenders.Remove(interpolator);
                }
```

```
            }
            else
            {
#endif
            if(GameManager.Instance?.DayCycleHandler != null)

GameManager.Instance.DayCycleHandler.m_LightBlenders.Remove(interpolator);
#if UNITY_EDITOR
            }
#endif
        }
    }
    [System.Serializable]
    public struct DayCycleHandlerSaveData
    {
        public float TimeOfTheDay;
    }



#if UNITY_EDITOR
    // Wrapping a custom editor between UNITY_EDITOR define check allow to keep it in the same
```

```
    // file as this part will be stripped when building for standalone (where Editor class doesn't exist).
    // Don't forget to also wrap the UnityEditor using at the top of the file between those define check too.

    // Show a slider that allow to test a specific time to help define colors.
    [CustomEditor(typeof(DayCycleHandler))]
    class DayCycleEditor : Editor
    {
        private DayCycleHandler m_Target;
        public override VisualElement CreateInspectorGUI()
        {
            m_Target = target as DayCycleHandler;
            var root = new VisualElement();
            InspectorElement.FillDefaultInspector(root, serializedObject, this);
            var slider = new Slider(0.0f, 1.0f);
            slider.label = "Test time 0:00";
            slider.RegisterValueChangedCallback(evt =>
            {
                m_Target.UpdateLight(evt.newValue);
                slider.label = $"Test Time {GameManager.GetTimeAsString(evt.newValue)} ({evt.newValue:F2})";
```

```
            SceneView.RepaintAll();
        });
        //registering click event, it's very catch all but not way to do a change check for control change
        root.RegisterCallback<ClickEvent>(evt =>
        {
            m_Target.UpdateLight(slider.value);
            SceneView.RepaintAll();
        });
        root.Add(slider);
        return root;
    }
}
#endif
}
```

4. GameManager

```
using System;
using System.Collections.Generic;
using Cinemachine;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.Tilemaps;



namespace HappyHarvest

{

    /// <summary>

    /// The GameManager is the entry point to all the game system. It's execution order is set very low to make sure

    /// its Awake function is called as early as possible so the instance if valid on other Scripts.

    /// </summary>

    [DefaultExecutionOrder(-9999)]

    public class GameManager : MonoBehaviour

    {

        private static GameManager s_Instance;



#if UNITY_EDITOR
```

```
        //As our manager run first, it will also be destroyed first when the app will be exiting, which lead to s_Instance
        //to become null and so will trigger another instantiate in edit mode (as we dynamically instantiate the Manager)
        //so this is set to true when destroyed, so we do not reinstantiate a new one
        private static bool s_IsQuitting = false;
#endif
        public static GameManager Instance
        {
            get
            {
#if UNITY_EDITOR
                if (s_Instance == null)
                {
                    //in editor, we can start any scene to test, so we are not sure the game manager will have been
                    //created by the first scene starting the game. So we load it manually. This check is useless in
                    //player build as the 1st scene will have created the GameManager so it will always exists.
                    Instantiate(Resources.Load<GameManager>("GameManager"));
                }
#endif
```

```
            return s_Instance;
        }
    }

    public TerrainManager Terrain { get; set; }
    public PlayerController Player { get; set; }
    public DayCycleHandler DayCycleHandler { get; set; }
    public WeatherSystem WeatherSystem { get; set; }
    public CinemachineVirtualCamera MainCamera { get; set; }
    public Tilemap WalkSurfaceTilemap { get; set; }

    public SceneData LoadedSceneData { get; set; }

    // Will return the ratio of time for the current day between 0 (00:00) and 1 (23:59).
    public float CurrentDayRatio => m_CurrentTimeOfTheDay / DayDurationInSeconds;

    [Header("Market")]
    public Item[] MarketEntries;

    [Header("Time settings")]
```

```
[Min(1.0f)]

public float DayDurationInSeconds;

public float StartingTime = 0.0f;


[Header("Data")]

public ItemDatabase ItemDatabase;

public CropDatabase CropDatabase;


public Storage Storage;


private bool m_IsTicking;


private List<DayEventHandler> m_EventHandlers = new();

private List<SpawnPoint> m_ActiveTransitions = new List<SpawnPoint>();


private float m_CurrentTimeOfTheDay;


private void Awake()

{

    if (s_Instance == null)

    {
```

```
    s_Instance = this;
    DontDestroyOnLoad(gameObject);
}
else
{
    Destroy(gameObject);
}

// ตรวจสอบว่ามี Prefab GameManager หรือไม่
if (Instance == null)
{
    var prefab = Resources.Load<GameManager>("GameManager");
    if (prefab != null)
    {
        Instantiate(prefab);
    }
    else
    {
        Debug.LogError("GameManager Prefab not found in Resources.");
    }
}
```

```
            m_IsTicking = true;


            ItemDatabase.Init();
            CropDatabase.Init();


            Storage = new Storage();


            m_CurrentTimeOfTheDay = StartingTime;


            //we need to ensure that we don't have a day length at 0, otherwise we will get stuck into infinite loop in update
            //(and a day with 0 length makes no sense)
            if (DayDurationInSeconds <= 0.0f)
            {
                DayDurationInSeconds = 1.0f;
                Debug.LogError("The day length on the GameManager is set to 0, the length need to be set to a positive value");
            }
        }


        private void Start()
```

```
        {
            m_CurrentTimeOfTheDay = StartingTime;

            UIHandler.SceneLoaded();
        }

#if UNITY_EDITOR
        private void OnDestroy()
        {
            s_IsQuitting = true;
        }
#endif

        private void Update()
        {
            if (m_IsTicking)
            {
                float previousRatio = CurrentDayRatio;
                m_CurrentTimeOfTheDay += Time.deltaTime;

                while (m_CurrentTimeOfTheDay > DayDurationInSeconds)
```

```
    m_CurrentTimeOfTheDay -= DayDurationInSeconds;


foreach (var handler in m_EventHandlers)
{
    foreach (var evt in handler.Events)
    {
        bool prev = evt.IsInRange(previousRatio);
        bool current = evt.IsInRange(CurrentDayRatio);


        if (prev && !current)
        {
            evt.OffEvent.Invoke();
        }
        else if (!prev && current)
        {
            evt.OnEvents.Invoke();
        }
    }
}


if (DayCycleHandler != null)
```

```
            DayCycleHandler.Tick();
        }
    }

    public void LoadGameOverScene()
    {
        SceneManager.LoadScene("GameOver");
    }

    public void Pause()
    {
        m_IsTicking = false;
        Player.ToggleControl(false);
    }

    public void Resume()
    {
        m_IsTicking = true;
        Player.ToggleControl(true);
    }
```

```
public void RegisterSpawn(SpawnPoint spawn)
{
    if (Player == null && spawn.SpawnIndex == 0)
    { //if we have no player, we need to create one
        Instantiate(Resources.Load<PlayerController>("Character"));
        spawn.SpawnHere();
    }

    m_ActiveTransitions.Add(spawn);
}

public void UnregisterSpawn(SpawnPoint spawn)
{
    m_ActiveTransitions.Remove(spawn);
}

public void MoveTo(int targetScene, int targetSpawn)
{
    Pause();
    SaveSystem.SaveSceneData();
    UIHandler.FadeToBlack(() =>
```

```
        {
            var asyncop = SceneManager.LoadSceneAsync(targetScene,
LoadSceneMode.Single);
            asyncop.completed += operation =>
            {
                m_IsTicking = true;
                foreach (var active in m_ActiveTransitions)
                {
                    if (active.SpawnIndex == targetSpawn)
                    {
                        active.SpawnHere();
                        SaveSystem.LoadSceneData();
                    }
                }
                UIHandler.SceneLoaded();
                UIHandler.FadeFromBlack(() =>
                {
                    Player.ToggleControl(true);
                });
            };
        });
    }
```

```
/// <summary>
/// Will return the current time as a string in format of "xx:xx"
/// </summary>
/// <returns></returns>
public string CurrentTimeAsString()
{
    return GetTimeAsString(CurrentDayRatio);
}

/// <summary>
/// Return in the format "xx:xx" the given ration (between 0 and 1) of time
/// </summary>
/// <param name="ratio"></param>
/// <returns></returns>
public static string GetTimeAsString(float ratio)
{
    var hour = GetHourFromRatio(ratio);
    var minute = GetMinuteFromRatio(ratio);

    return $"{hour}:{minute:00}";
}
```

```
        public static int GetHourFromRatio(float ratio)
        {
            var time = ratio * 24.0f;
            var hour = Mathf.FloorToInt(time);
            return hour;
        }
        public static int GetMinuteFromRatio(float ratio)
        {
            var time = ratio * 24.0f;
            var minute = Mathf.FloorToInt((time - Mathf.FloorToInt(time)) * 60.0f);

            return minute;
        }

        public static void RegisterEventHandler(DayEventHandler handler)
        {
            if (Instance == null)
            {
                Debug.LogError("GameManager.Instance is null. Ensure GameManager
exists in the Scene.");
                return;
```

```
}
if (handler == null || handler.Events == null)
{
    Debug.LogError("DayEventHandler or its Events array is null.");
    return;
}
foreach (var evt in handler.Events)
{
    if (evt == null)
    {
        Debug.LogWarning("A DayEvent is null. Skipping.");
        continue;
    }

    if (evt.IsInRange(GameManager.Instance.CurrentDayRatio))
    {
        evt.OnEvents?.Invoke();
    }
    else
    {
        evt.OffEvent?.Invoke();
```

```
            }
        }

        Instance.m_EventHandlers.Add(handler);
    }

    public static void RemoveEventHandler(DayEventHandler handler)
    {
        Instance?.m_EventHandlers.Remove(handler);
    }
    public void SetTimeTo6AM()
  {
    m_CurrentTimeOfTheDay = DayDurationInSeconds * 0.25f; // 0.25f represents
6:00 AM (1/4 of the day)
    Debug.Log("Time set to 6:00 AM");

    // Update DayCycleHandler if it exists.
    DayCycleHandler?.Tick(); // Ensure visual time update.
  }
  }
}
```

5. DoorEnter

```
using UnityEngine;
using UnityEngine.SceneManagement;

namespace HappyHarvest
{
    public class DoorEnter : MonoBehaviour
    {
        public int requiredCoins = 100;
        public string targetSceneName;

        private GameObject messageObject;
        private Vector3 initialPosition;
        private float floatingSpeed = 1.0f;

        private void Start()
        {
            messageObject = new GameObject("DoorMessage");
            messageObject.transform.SetParent(transform);
            messageObject.transform.localPosition = new Vector3(0, 1.5f, 0);
            initialPosition = messageObject.transform.localPosition;
            messageObject.SetActive(false);
        }

        public bool CanEnter(int playerCoins)
        {
            return playerCoins >= requiredCoins;
        }

        public void EnterDoor(PlayerController player)
```

```
        {
            LoadTargetScene();
            Destroy(messageObject);
        }

        private void LoadTargetScene()
        {
            if (!string.IsNullOrEmpty(targetSceneName))
            {
                SceneManager.LoadScene(targetSceneName);
            }
            else
            {
                Debug.LogError("Target scene name not set for the door!");
            }
        }
        private void Update()
        {
            if (messageObject.activeSelf)
            {
                messageObject.transform.localPosition = initialPosition + new Vector3(0,
Mathf.Sin(Time.time * floatingSpeed) * 0.5f, 0);
            }
        }
        private void OnTriggerEnter2D(Collider2D other)
        {
            if (other.CompareTag("Player"))
            {
                messageObject.SetActive(true);
            }
        }
```

```
    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            messageObject.SetActive(false);
        }
    }
    private void OnTriggerStay2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            PlayerController player = other.GetComponent<PlayerController>();
            if (player != null)
            {
                if (CanEnter(player.Coins))
                {
                    if (Input.GetKeyDown(KeyCode.F))
                    {
                        EnterDoor(player);
                    }
                }
                else
                {
                    Debug.Log("Not enough coins to enter.");
                }
            }
        }
    }
}
```

ประวัติผู้จัดทำ

ประวัติผู้จัดทำโครงการ

ชื่อ นางสาวณัฐนันท์ จิรวัฒน์

เกิดเมื่อวันที่ 15 มีนาคม พ.ศ 2548

บ้านเลขที่ 26/4 หมู่ 4 ต.บางกะจะ อ.เมือง จ.จันทบุรี

หมายเลขโทรศัพท์ 082-783-0354

E-mail : nattananjirawat3@gmail.com

การศึกษา

ชั้นประถมศึกษา โรงเรียนสฤษดิเดช

ชั้นมัธยมศักษา โรงเรียนอนุบาลจันทบุรี

ชั้นประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคจันทบุรี

ขณะนี้กำลังศึกษาอยู่ในชั้นประกาศนียบัตรวิชาชีพชั้นสูงที่วิทยาลัยเทคนิคจันทบุรี

ประวัติผู้จัดทำโครงการ

ชื่อ นายอิดศักดิ์ พวงนะที

เกิดเมื่อวันที่ 14 กุมภาพันธ์ พ.ศ 2548

บ้านเลขที่ 55/28 หมู่ 5 ต.ท่าช้าง อ.เมือง จ.จันทบุรี

หมายเลขโทรศัพท์ 061-201-5072

E-mail : adisakphoungnatee@gmail.com

การศึกษา

ชั้นประถมศึกษา โรงเรียนบ้านแก้ว

ชั้นมัธยมศักษา โรงเรียนเบญจมานุสรณ์

ชั้นประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคจันทบุรี

ขณะนี้กำลังศึกษาอยู่ในชั้นประกาศนียบัตรวิชาชีพชั้นสูงที่วิทยาลัยเทคนิคจันทบุรี