โครงงานวิทยาศาสตร์

เรื่อง flaxibility: Drag and Drop Factory Managing Puzzle Game

โดย

1. นาย เจษฎา คณะทอง ชั้น 5/1 เลขที่ 12
2. นาย ภูวิศ พุฒิไพโรจน์ ชั้น 5/8 เลขที่ 21

ครูและอาจารย์ที่ปรึกษาโครงงาน

ครูที่ปรึกษา นาย วินัย รัตนพล

อาจารย์ที่ปรึกษา ศ.ดร. ชินพงศ์ อังสุโชติเมธี คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชาโครงงาน STEM 2 (ว30284)

ภาคเรียนที่ 2 ปีการศึกษา 2564

โรงเรียน มอ.วิทยานุสรณ์ อำเภอหาดใหญ่ จังหวัดสงขลา

ชื่อโครงงาน flaxibility: Drag and Drop Factory Managing Puzzle Game
สาขา เทคโนโลยีและคอมพิวเตอร์
ผู้จัดทำ 1. นาย เจษฎา คณะทอง ชั้น 5/1 เลขที่ 12
2. นาย ภูวิศ พุฒิไพโรจน์ ชั้น 5/8 เลขที่ 21
ครูที่ปรึกษา นาย วินัย รัตนพล
อาจารย์ที่ปรึกษา ศ.ดร. ชินพงศ์ อังสุโชติเมธี คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์
โรงเรียน มอ.วิทยานุสรณ์ 424/18 ถ.กาญจนวนิช ต.คอหงส์ อ.หาดใหญ่ จ.สงขลา 90110
โทรศัพท์ 074-201628 **โทรสาร** 074-201629
ปีการศึกษา 2564

บทคัดย่อ

เกม flaxibility เป็นเกมที่ถูกพัฒนาภายใต้โครงงานวิทยาศาสตร์ สาขาคอมพิวเตอร์ เรื่อง flaxibility: Drag and Drop Factory Managing Puzzle Game โดยมีวัตถุประสงค์คือการวิเคราะห์และออกแบบเกม เพื่อเป็นสื่อในการสอนการคิดแบบเป็นขั้นเป็นตอน และเพื่อสร้างเกมที่สามารถมอบความสนุกให้กับผู้เล่นได้ Flaxibility ถูกสร้างขึ้นโดยมี Setting เป็นคุณลุงคนหนึ่งที่บริหารธุรกิจเดิมล้มเหลว ทำให้มาเปิดโรงงานใหม่เป็นโรงงานผลิตผ้า แต่ลุงคนนั้นกลับเป็นคนขี้เหนียวจึงไม่ออกแบบโรงงานสำหรับระยะยาวจนต้องรื้อและสร้างโรงงานใหม่อยู่หลายครั้งซึ่งการสร้างโรงงานใหม่นี้ก็จะเป็นหน้าที่ของผู้เล่น ในด้านเกมเพลย์เกมนี้เป็นเกมแนว Puzzle factory builder เนื่องจากมีข้อจำกัดด้าน เวลา จึงไม่สามารถพัฒนาเกมได้ตามที่ออกแบบไว้ ทั้งนี้ ตัวเกมยังสามารถพัฒนาต่อยอดเพื่อเพิ่มความน่าสนใจได้ อีก เกม flaxibility พัฒนาโดยใช้ Godot engine

กิตติกรรมประกาศ

โครงงานเทคโนโลยีและคอมพิวเตอร์ เรื่อง flaxibility: Drag and Drop Factory Managing Puzzle Game ได้สำเร็จลุล่วงตามเป้าหมายและจุดประสงค์ของโครงงานได้ สามารถดำเนินการไปได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์และสนับสนุนเป็นอย่างดี ขอขอบคุณอาจารย์ที่ปรึกษา ศ.ดร. ชินพงศ์ อังสุโชติเมธี และครูที่ปรึกษา นาย วินัย รัตนพล ที่ได้ให้คำแนะนำ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่างๆ ของโครงงานมาโดยตลอด และตรวจทานความถูกต้องของเอกสาร จนโครงงานนี้ได้สำเร็จลุล่วงสมบูรณ์ตามจุดประสงค์ ผู้ศึกษาจึงขอกราบพระคุณเป็นอย่างสูง

ขอขอบคุณเพื่อนร่วมงานทุกคนที่อำนวยความสะดวกและช่วยเหลือการทำโครงงานในครั้งนี้และขอขอบพระคุณผู้จัดทำทฤษฎีต่างๆ ของวิทยานิพนธ์ งานวิจัย วารสาร บทความ และคู่มืออ้างอิงสำหรับ Godot Engine ที่ผู้จัดทำนำมาอ้างอิงในการทำโครงงานฉบับนี้ รวมถึงชุมชนผู้พัฒนา Godot Engine สำหรับการพัฒนาความสามารถใหม่ๆ ให้ได้ผู้จัดทำให้อย่างไม่แสวงผลกำไร รวมถึงผู้ใช้งาน Godot Forum และ Godot Engine Discord ที่ช่วยไขข้อสงสัยและช่วยแก้ปัญหาที่ผู้จัดทำพบเจอไว้ ณ โอกาสนี้ด้วย

1. นาย เจษฎา คณะทอง
2. นาย ภูวิศ พุฒิไพโรจน์

ผู้จัดทำ

วันที่ 28 มีนาคม 2565

สารบัญ

สารบัญตาราง

สารบัญภาพประกอบ

สารบัญภาพประกอบ

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในแต่ละเป้าหมายจะสำเร็จหรือไม่ ขึ้นอยู่กับคุณภาพและประสิทธิภาพของการคิด และการกระทำ ดังนั้นการคิดอย่างเป็นระบบหรือการคิดเป็นขั้นเป็นตอนเพื่อผลลัพธ์อย่างเป็นอัตโนมัติที่มีประสิทธิภาพจึงเป็นทักษะที่สำคัญเพื่อนำไปสู่การบรรลุเป้าหมาย (ธวัชชัย,2563)

เราจึงต้องการสร้างสื่อที่มีการส่งเสริมการเรียนรู้แบบเป็นขั้นตอน หรือ algorithm เราจึงตัดสินใจที่จะสร้างเกมเพราะเป็นสื่อที่เข้าถึงง่าย ทั้งยังมีการเติบโตที่เร็วในภาคอุตสาหกรรม (พิชชากร. 2563; NALISA, 2564) เพื่อมานำเสนอการคิดแบบเป็นขั้นเป็นตอน

1.2 สมมติฐานและตัวแปรที่เกี่ยวข้องในการศึกษาโครงงาน

สมมติฐานสามารถสร้างสื่อการสอนการคิดแบบเป็นขั้นเป็นตอนผ่านเกมได้

ตัวแปรต้น เกม

ตัวแปรตาม ความสามารถในการสอน

ตัวแปรควบคุม Godot engine ทีมผู้พัฒนา

1.3 จุดมุ่งหมายของการศึกษาโครงงาน

1. เพื่อสร้างสื่อการสอนการคิดแบบเป็นขั้นเป็นตอนอย่างง่าย
2. เพื่อสร้างเกมที่สามารถใช้สอนการคิดแบบเป็นขั้นเป็นตอนได้

1.4 ประโยชน์ที่คาดว่าจะได้รับจากการศึกษาโครงงาน

1. สามารถสร้างสื่อการสอนการคิดแบบเป็นขั้นเป็นตอนอย่างง่าย
2. สามารถสร้างเกมที่สอดคล้องกับจุดมุ่งหมายที่ 1 คือ สามารถใช้สอนการคิดแบบเป็นขั้นเป็นตอนอย่างง่ายได้

1.5 ขอบเขตของการศึกษาโครงงาน

พัฒนาเกมบน Godot engine ตั้งแต่วันที่ 26/11/2021 จนถึงวันที่ 31/3/2022

1.6 นิยามศัพท์เฉพาะที่ใช้ในการศึกษาโครงงาน

1. Game engine เครื่องมือที่ใช้สร้างและพัฒนาเกม

2. Godot หนึ่งใน game engine ภายใต้ MIT license พัฒนาโดย Juan Linietsky และ Ariel Manzur
3. การคิดแบบเป็นขั้นเป็นตอน การที่สามารถจัดลำดับความสำคัญ สิ่งใดพึงทำก่อนหลังตามลำดับเพื่อให้งานนั้นๆออกมามีคุณภาพ
4. Node หน่วยที่เล็กที่สุดของ Godot มีหน้าที่ทำสิ่งต่างๆแตกต่างกันไปตามชนิด อาทิ label ใช้สำหรับเพิ่มข้อความ sprite ใช้สำหรับเพิ่มรูปภาพต่างๆเป็นต้น นอกจากนี้ยังสามานรถจัดกลุ่มและรวม Node ให้เป็น scene ได้
5. Scene คือการจัดรวมกันของ Node เพื่อใช้ประโยชน์ต่างๆไม่ว่าจะขนาดเล็ก เช่นใช้สร้างตัวละคร สิ่งของประกอบภายในเกม ไปจนถึงขนาดใหญ่อย่างเช่นด่านต่างๆภายในตัวเกม

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

Godot Engine

Godot Engine เป็นชุดเครื่องมือต่างๆ เพื่อช่วยสร้างเกม 2D และ 3D เกมที่สร้างสามารถส่งออกไปยังแพลตฟอร์มต่างๆ ได้มากมาย ได้แก่ระบบปฏิบัติการบนคอมพิวเตอร์ (Linux, macOS, Windows), ระบบปฏิบัติการบนโทรศัพท์มือถือ (Android, iOS), บนเว็บ (HTML5), หรือบนเกมคอนโซลต่างๆ

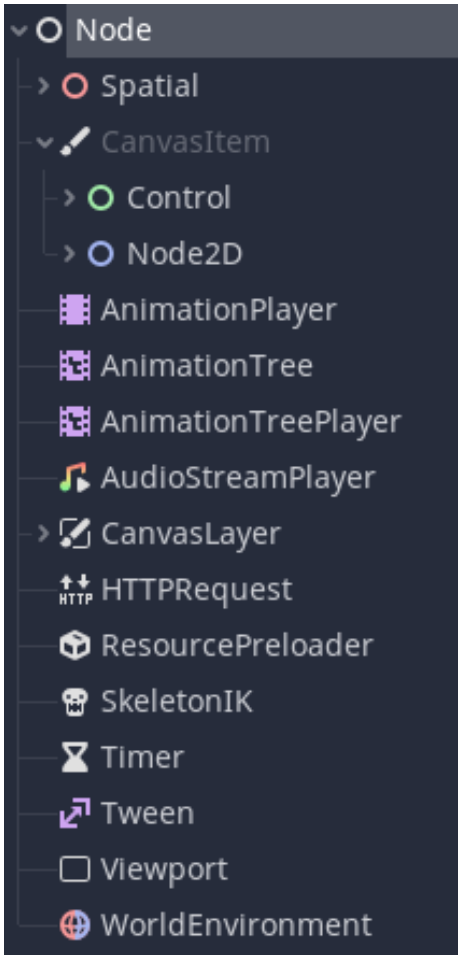
Godot Engine ได้เผยแพร่แบบโอเพ่นซอร์สในเดือนกุมภาพันธ์ ค.ศ. 2014 ภายใต้ MIT license หลังจากได้รับการพัฒนาโดย Juan Linietsky และ Ariel Manzur มาหลายปี ทำให้ไม่มีการเก็บค่าลิขสิทธิ์หรือรืออเงินส่วนแบ่งกับผู้ใช้งาน ในปัจจุบัน Godot Engine มีการพัฒนาอย่างเป็นอิสระ และมีการขับเคลื่อนโดยชุมชนผู้ใช้งาน

ภาษาที่ใช้พัฒนาเกมบน Godot Engine มีให้ผู้พัฒนาเลือก 3 ชนิด ได้แก่ GDScript เป็นภาษาแบบ high-level และ dynamically typed ที่มี Syntax คล้าย Python ซึ่งจะเป็นภาษาที่มีความใกล้ชิดกับ Godot Engine มากที่สุด, C# เป็นภาษาแบบ high-level และ statically typed, หรือ VisualScript เป็นภาษาที่ใช้การจินตนาการคำสั่งต่างๆ เป็น block ที่เชื่อมต่อกัน ทำให้การเข้าถึงการพัฒนาเกมได้ง่ายขึ้นสำหรับผู้พัฒนาที่ไม่มีพื้นฐานการพัฒนาซอฟต์แวร์

ภาพที่ 2.1 Logo ของ Godot Engine

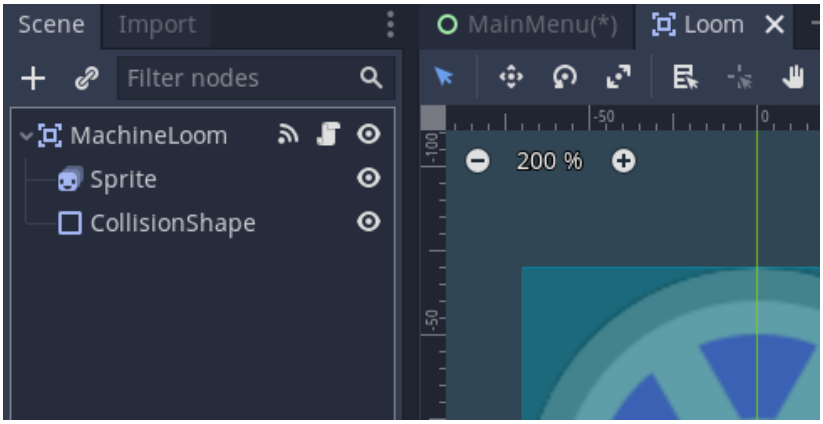
การพัฒนา Software ต่างๆ จำเป็นจะต้องมีการมองการทำงานอย่างเป็นนามธรรม หรือการ Abstraction เช่นเดียวกับเครื่องมือพัฒนาเกมอย่าง Godot Engine ที่มีการมองเกมหนึ่งเป็นแผนผังของ Nodes ที่รวมกลุ่มกันเป็น Scenes ซึ่งแต่ละ Node สามารถสื่อสารกับโดยใช้ Signals

Nodes เป็นหน่วยที่เล็กที่สุดใน Godot Engine ซึ่งจะแบ่งเป็น Node ที่ทำหน้าที่แตกต่างกันหลายชนิด และเป็น Node ที่ทำงานบนเกม 2D, 3D หรือเป็นส่วนที่ให้ หรือรับข้อมูลจากผู้ใช้ ซึ่งผู้พัฒนาเกมจะสามารถนำ Node หลายชนิดมารวมกันเป็น Scene



ภาพที่ 2.2 ชนิด Node หลักของ Godot Engine

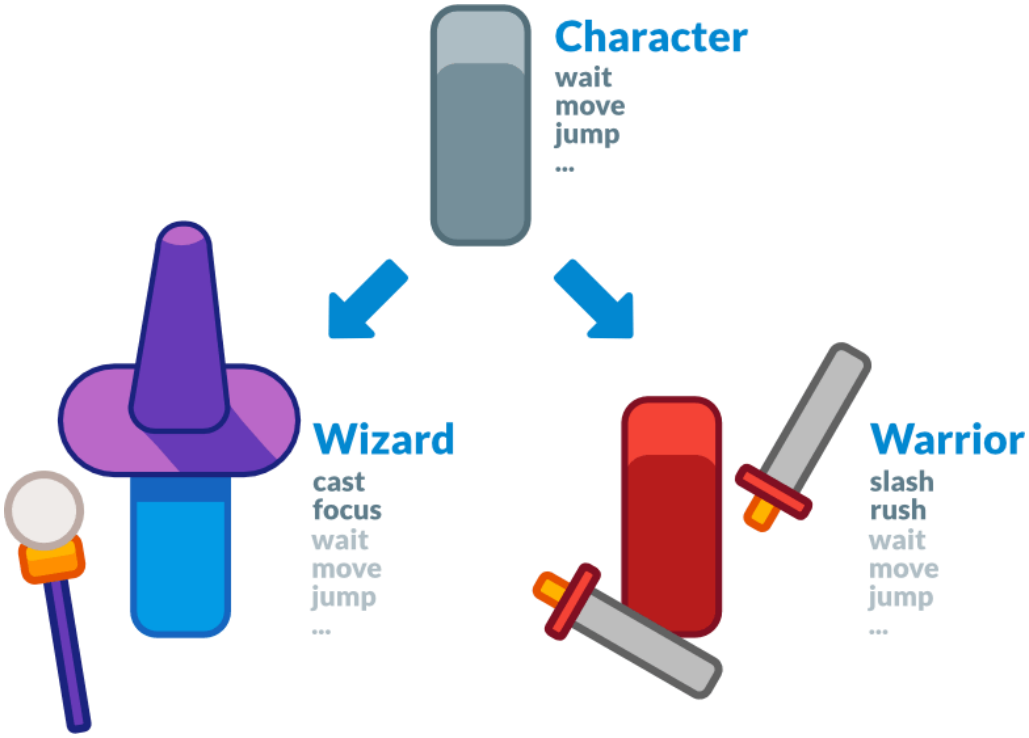
Scene หนึ่งสามารถมีหน้าที่ได้หลากหลาย เช่น Scene ตัวละคร, เครื่องมือ, เมนูหลักของเกม, หรือด่านหนึ่งของเกม Scene ทุกตัวจะมี Root Node เป็น Node หลัก และสามารถเพิ่ม Node อื่นๆ เป็น Child Node ของมันได้ Scene หนึ่งสามารถนำไปใช้ได้หลายครั้งโดยการทำ Scene Instance เมื่อนำ Scene Instance ไปวางใน Scene อื่น จะขึ้นเป็น Node ชนิดตาม Root Node ของ Scene นั้นๆ และจะไม่มี Child Node ปรากฏ การ Scene หรือ Node ต่างๆ มาใช้งานร่วมกับจะทำให้เกิด Scene tree



ภาพที่ 2.3 Scene Tree “Loom”มี Area2D “MachineLoom” เป็น Root Node และมี AnimatedSprite “Sprite” กับ CollisionShape2D “CollisionShape” เป็น Child Node

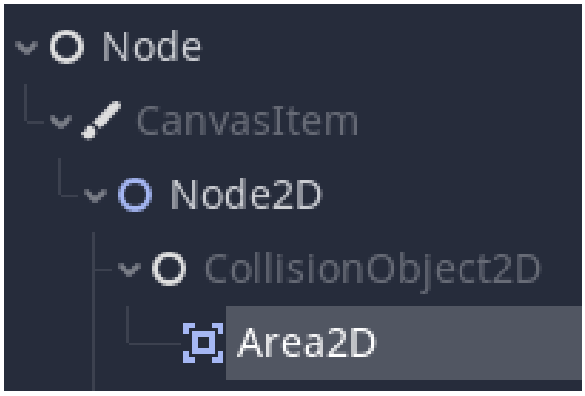
Godot ยึดหลักการพัฒนาซอฟต์แวร์แบบ object-oriented ผ่านการระบบการแบ่ง Scene และ Node ผู้พัฒนาสามารถเรียบเรียง และรวบรวมส่วนประกอบต่างๆ เป็นโครงสร้างกึ่งสำเร็จรูป ตัวอย่างเช่น เมื่อต้องการสร้างเมืองที่ ประกอบด้วย Node ตะเกียงผุพัง และในตะเกียงจะมี Node ไฟกระพริบ เมื่อผู้พัฒนาต้องการที่จะเปลี่ยนสีของ Node ไฟกระพริบ ผู้พัฒนาสามารถเปลี่ยนสี Node ไฟกระพริบ แล้วจะทำให้ตะเกียงทุกดวงในเมือง เปลี่ยนสีตามไปด้วย

นอกจากนั้น ผู้พัฒนาสามารถใช้ Scene หนึ่งเป็นฐาน และนำฐานนั้น ไปต่อยอดเป็น Scene ใหม่ที่ใช้ทั้งคำสั้งจากฐาน และคำสั่งเฉพาะที่ต่อยอดออกไป เช่น เมื่อต้องการสร้างอาชีพของตัวละครพ่อมด ก็สามารถต่อยอดมาจากฐานที่เป็นตัวละครเปล่าที่สามารถเคลื่อนไหวได้ เพิ่มความสามารถเฉพาะของพ่อมด คือการร่ายมนต์ หรือความสามารถอื่นๆ โดยที่ไม่ทำให้ฐานเปลี่ยนไป



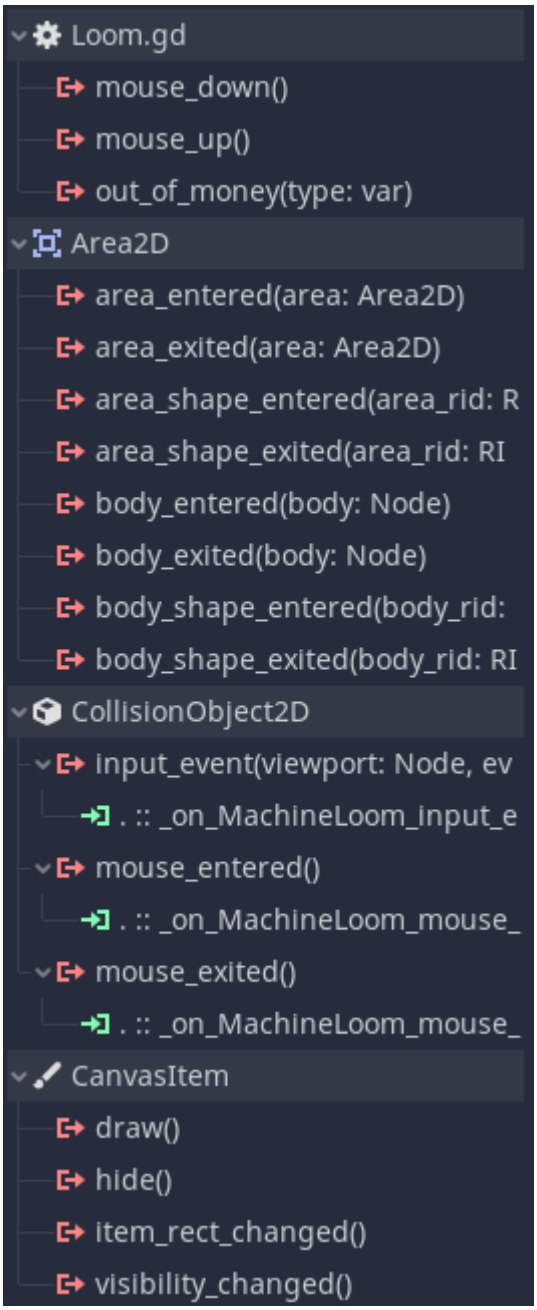
ภาพที่ 2.4 การต่อยอด Scene “Character” เป็น “Wizzard” และ “Warrior”

Node พื้นฐานของ Godot Engine ก็มีการใช้การต่อยอดเช่นเดียวกัน กล่าวคือ Node ชนิดเดียวกันที่เป็นส่วนหนึ่งของแผนผังชนิด Node ก็จะถือว่าเป็นการต่อยอดจาก Node ที่มีลำดับอยู่สูงกว่า แต่เพิ่มความสามารถของตัวเองเข้าไปด้วย เช่น Area2D จะมีคุณสมบัติ และความสามารถของ CollisionObject2D, Node2D, CanvasItem, และ Node



ภาพที่ 2.5 แผนผังการต่อยอดของ Area2D

Node จะมีการส่ง Signal เมื่อมีเหตุการณ์บางอย่างเกิดขึ้น ซึ่งจะแตกต่างกันตามชนิดของ Node ซึ่งจะรวม Signal ของ Node ที่ต่อยอดมาด้วย และผู้พัฒนาสามารถสร้าง Signal เพิ่มตามความต้องการได้ ซึ่งจะช่วยให้ Node สื่อสารกับ Node อื่นๆ หรือ Script ได้โดยไม่ต้อง Hardcode การสื่อสาร และเพิ่มความยืดหยุ่นในการ Code และโครงสร้างของ Scene ก่อนจะใช้ Signal ใดๆ ก็จำเป็นจะต้อง Connect Signal กับ Function ใน Script ก่อนเสมอ ไม่เช่นนั้น Signal จะไม่ทำตามคำสั่ง



ภาพที่ 2.6 Signal ส่วนหนึ่ง ของ Scene “Loom”

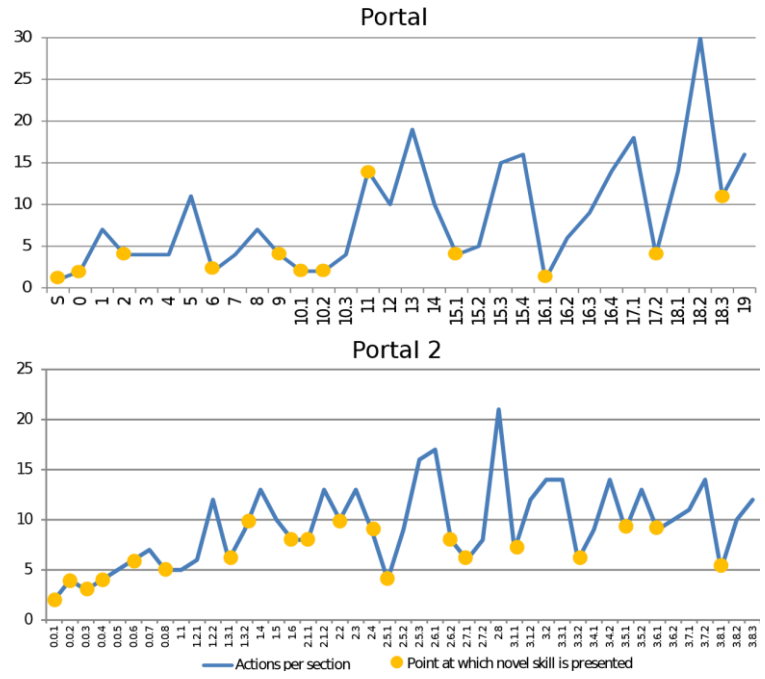
จากภาพข้างต้น จะเห็นว่า Node CollisionObject2D มี Signal หลายชนิดให้เลือกใช้ เช่นเมื่อเมาส์เข้าหรือออกพื้นที่ของ CollisionObject2D จะส่ง Signal mouse_entered และ mouse_exited ตามลำดับ ซึ่งจะเห็นว่ามีลูกศรสีเขียวอยู่ข้างล่าง Signal นั้นๆ เป็นการแสดงว่า Signal นั้นได้ Connect กับ Function นั้นๆ เรียบร้อยแล้ว นอกจากนั้นจะมี Signal ของ Loom.gd เป็น Signal ที่กำหนดขึ้นเอง และมี Signal จาก Node ที่เป็นฐานการต่อยอดอย่าง Signal ของ CanvasItem

Learning Curves: Analysing Pace and Challenge in Four Successful Puzzle Games

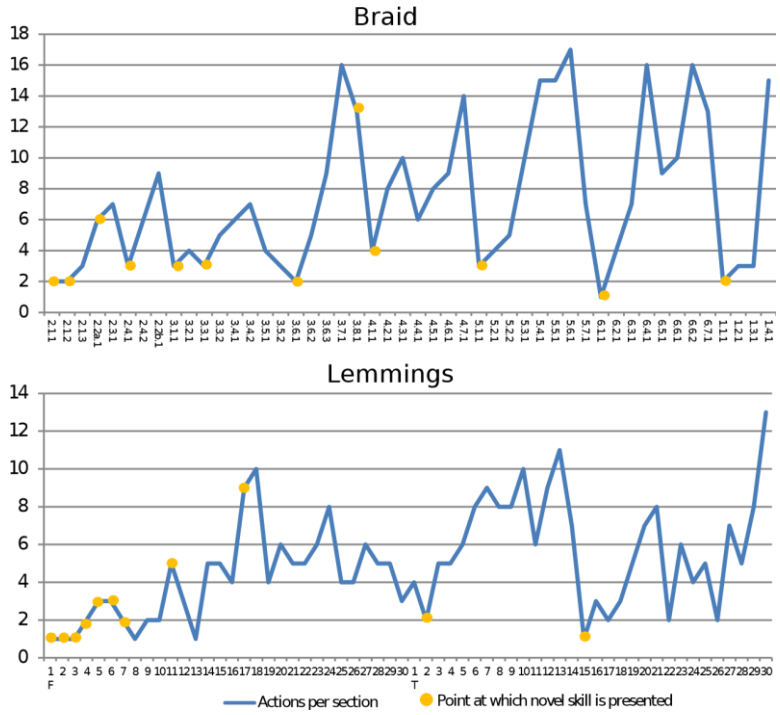
งานวิจัยนี้พิจารณาส่วนหนึ่งของการออกแบบเกมที่เป็นสิ่งสำคัญในการเรียนรู้วิธีการเล่นเกม และความสนุกของเกม สิ่งนั้นคือความเร็วที่ผู้เล่นจะเจอกับความท้าทาย หรือความยากของเกม คำถามนี้มีการถกเถียงและวิเคราะห์กันอย่างมากมายโดยผู้ออกแบบเกม หรือผู้วิจัยต่างๆ โดยมองจากหลากหลายมุมมอง และได้มีการเข้าใจกันว่าความท้าทายของเกม ควรจะเทียบเท่าความชำนาญของผู้เล่น และควรจะเพิ่มขึ้น เมื่อความชำนาญของผู้เล่นเพิ่มขึ้น

การวิเคราะห์ความเร็ว และความท้าทายของเกมในงานวิจัยนี้ จะเน้นการวิเคราะห์ทางปฏิบัติเพื่อทำความเข้าใจการแก้ไขปัญหาโดยใช้จิตวิทยาในการวิเคราะห์พฤติกรรม โดยส่วนใหญ่จะเป็นการนำพฤติกรรมที่ทราบอยู่แล้ว มาใช้งานร่วมกัน เกมจึงต้องมีโครงสร้างที่ช่วยผู้เล่นแก้ปัญหา กล่าวคือ ก่อนที่ผู้เล่นเรียนรู้สิ่งที่ซับซ้อนของเกม ผู้เล่นควรจะรู้ส่วนประกอบต่างๆ ของสิ่งซับซ้อนก่อนที่จะนำมารวมกัน นี่เป็นพื้นฐานการออกแบบของเกมที่นำมาวิเคราะห์ในงานวิจัยนี้ และจะวิเคราะห์ความซับซ้อนของเกมที่เปลี่ยนแปลงเมื่อเล่นด่านที่ยากขึ้น และจะเกี่ยวข้องกับการแนะนำหรือสอนสิ่งใหม่ในเกม

งานวิจัยนี้ได้พบว่าเกมแก้ไขปัญหาที่ได้รับความนิยม ที่มีพื้นฐานการเรียนรู้เกมและนำไปใช้ มีแนวโน้มที่จะสอนผู้เล่นอย่างมีระบบ ทุกเกมที่นำมาวิเคราะห์จะมีบางอย่างที่เหมือนกันอยู่ คือ เกมจะแนะนำส่วนประกอบต่างๆ แยกกัน ผ่านการแก้ปัญหาง่ายๆ หลังจากนั้นจะให้ใช้ส่วนประกอบใหม่ ร่วมกับส่วนประกอบที่รู้อยู่แล้ว และความซับซ้อนของแต่ละด่านจะเพิ่มขึ้นเรื่อยๆจนกว่าจะแนะนำส่วนประกอบอื่นๆ



ภาพที่ 2.7 จำนวนการกระทำขั้นต่ำของผู้เล่น (minimum player action) ที่ใช้ในการไขปริศนาในเกม Portal (ส่วนบน) and Portal 2 (ส่วนล่าง)



ภาพที่ 2.8 จำนวนการกระทำขั้นต่ำของผู้เล่น (minimum player action) ที่ใช้ในการไขปริศนาในเกม Braid (ส่วนบน) and Lemmings (ส่วนล่าง)

ภาพที่ 2.9 Portal 2 (ซ้าย) Braid (ขวา)

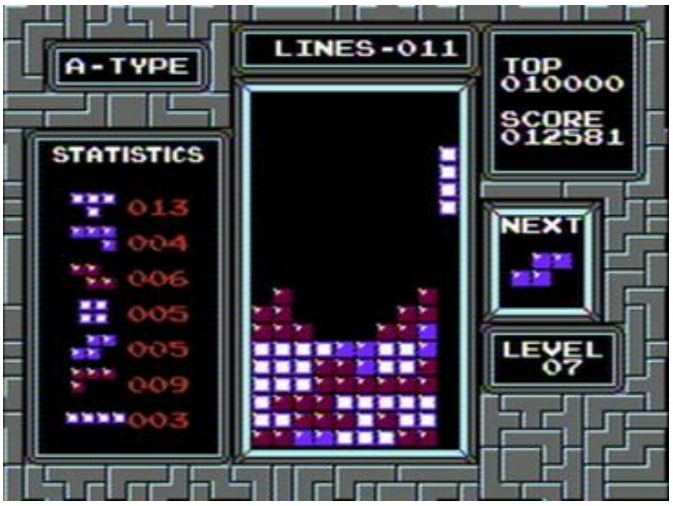
งานวิจัยนี้ รวมกับงานวิจัยอื่นๆในด้านการออกแบบเกม ทำให้เห็นการออกแบบความเร็วการเพิ่มขึ้นของความซับซ้อน หรือความยากของเกมอย่างเหมาะสม โดยนักออกแบบเกมควรออกแบบเกมโดยเมื่อผู้เล่นเจอกับส่วนประกอบใหม่ของเกม และนำส่วนประกอบนั้นๆ ไปฝึกซ้อมกับส่วนประกอบอื่นๆของเกมที่ผู้เล่นรู้อยู่แล้ว ทำให้ผู้เล่นมีความรู้เกี่ยวกับเกมเพิ่มขึ้นเรื่อยๆ จนจะได้ใช้ความรู้ทั้งหมดในปัญหาเดียว

Transhistorical perspective of the puzzle video game genre

ช่วงเริ่มต้นเกมแนวปริศนาจะแบ่งได้เป็น 3 ประเภท คือ

1. เกมที่เน้นใช้เพื่อการเรียนรู้ หรือใช้ความจำ เกมประเภทนี้มีอยู่เพื่อป้องกันเกมจากข้อครหาที่ทำให้เกิดความรุนแรงและหายนะอื่นๆ
2. เกมแบบดั้งเดิมที่นำมาสร้างในโลกดิจิทัล เช่นเกมพวก Mahjong, Dominoes, Rubik's cube ,หรือหมากรุก เกมเหล่านี้จะเป็นเกมที่บังคับให้ผู้เล่นคิด จึงจัดอยู่ในเกมแนวปริศนาเช่นกัน
3. เกมที่สร้างขึ้นโดยผู้พัฒนาเป็นงานอดิเรก และนี่เป็นประเภทที่ส่งผลต่อวงการเกมแนวปริศนาเป็นอย่างมาก เช่นเกม Over the eighties Heiankyo Alien (1980), Sokoban (1981), Door Door (1983), Lode Runner (1983), Tetris (1984), และ Chain-Shot! (1985) เกมพวกนี้ได้พัฒนาโดยผู้พัฒนาเกมทีมเล็กๆ หรือเรียกว่าผู้พัฒนาอินดี้ ซึ่งส่วนใหญ่อยู่ในประเทศญี่ปุ่น

Tetris เป็นเกมที่ได้รับการยอมรับว่ามีอิทธิพลต่อวงการเกมปริศนากับคนทั่วไปมาก แต่ก็ไม่ใช่ว่า Tetris เป็นเกมปริศนาที่ดีเพราะต้องใช้การตอบสนองของผู้เล่นมากกว่าการใช้กลยุทธ์ในการแก้ปัญหา

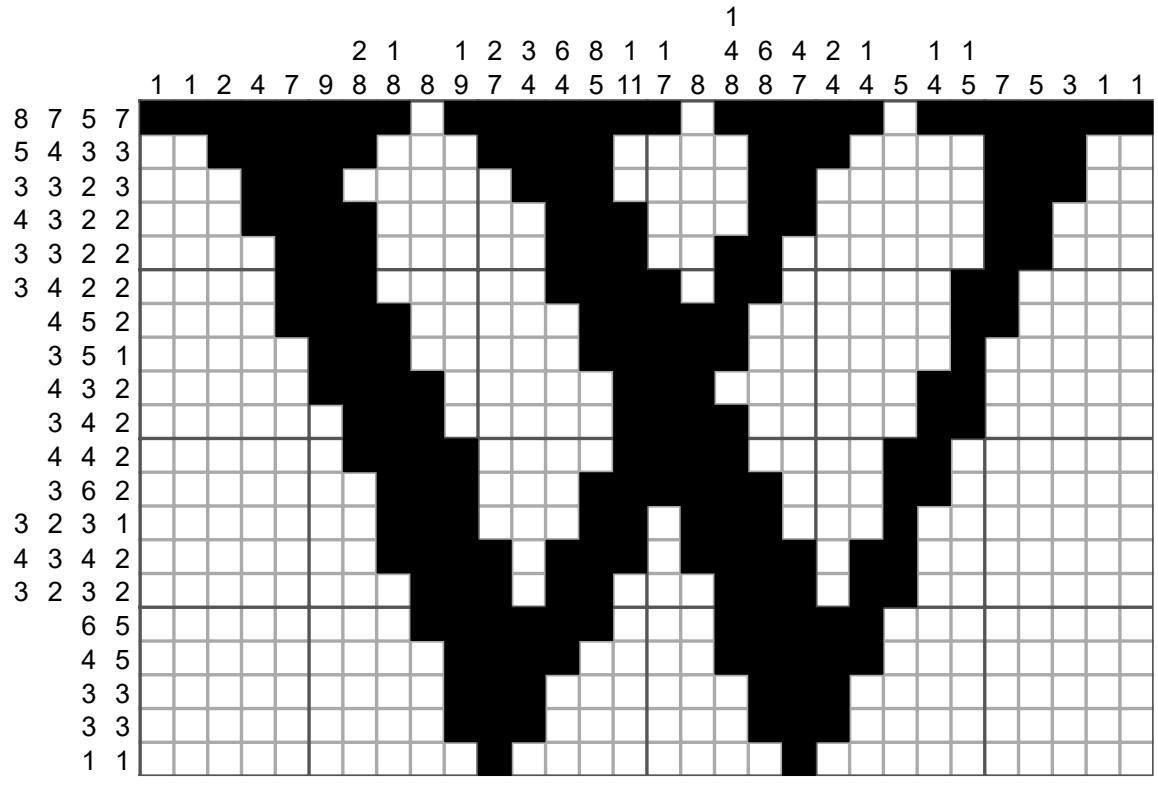


ภาพที่ 2.10 Tetris

ในยุคต่อมาเกมแนวนี้เริ่มที่จะย้ายจากเครื่องเล่นเกมที่บ้านเป็นเครื่องเล่นเกมพกพาด้วยความที่เป็นเกมที่มีความรวดเร็ว ทั้งยังมีการการเปิดตัวของ Game Boy และ Game Gear ทำให้เกิดเกมต่างๆมากมาย เช่น *Klax* (1989), *Columns* (1990), Wonderswan (*Gunpey*, 1999), PSP (*Lumines*, 2004) แต่ก็ยังมีเกมปริศนาให้เล่นบน Home Console อย่าง Super Nintendo Entertainment System หรือ Sega Mega Drive มีเกมเช่น Panel de Pon (1995), Lemmings (1991), The Incredible Machine (1993), Myst (1993) หรือว่าเกมใน Arcade ที่เน้นการแข่งกับ AI หรือผู้เล่นคนอื่นเพื่อเพิ่มรายได้ เช่น Puyo Puyo (1992), Puzzle Bobble (1994), Magical Drop (1995), Super Puzzle Fighter (1996)

ต่อมา Game Arcade เริ่มที่จะปิดตัวลง PlayStation ควบคุมตลาด Game Console ถึงแม้ว่าการไม่มีคู่แข่งจะส่งผลเสียต่อผู้บริโภค แต่ก็ทำให้ค่ายเกมได้มีการทดลองสิ่งใหม่ๆได้ง่าย เพราะสามารถขายเกมให้ทำกำไรได้ไม่ยาก มีเกมเช่น Intelligent Qube (1997), Devil Dice (1998), No One Can Stop Mr. Domino! (1998), Kula World (1998), Suzuki Bakuhatsu (2000), Chu Chu Rocket! (1999), Wetrix (1998) และเกมในยุคนี้เริ่มที่จะเล่นบนคอมพิวเตอร์มากขึ้นเพราะการเข้ามาของ Internet เกมบนคอมพิวเตอร์จะเป็นเกมที่เรียนรู้ได้ง่าย และสามารถเล่นในสถานที่ทำงานได้ ทำให้มีผู้ใช้งานคอมพิวเตอร์หลักล้านคนที่ไม่เคยแตะเกมมาก่อน ได้มาเป็นผู้เล่นเกมโดยเริ่มที่เกมปริศนาเช่น Bejeweled (2001), Collapse! (1998)

ยุคต่อมาจะมีผู้เล่นเกมครั้งแรก หรือเล่นแบบครั้งคราวด้วยการเปิดตัวของ Nintendo Wii ที่ขายได้อย่างเทน้ำเทท่า และ Nintendo DS ที่มียอดขายไม่เท่า Wii แต่ก็ประสบความสำเร็จเป็นเครื่องเล่นเกมพกพาที่มีความนิยมมากที่สุด ทำให้เกมปริศนาในยุคนี้มีมากมาย ตัวอย่างเช่น Slitherlink (2006), Picross (2007) Professor Layton and the Curious Village (2007), Puzzle Quest: Challenge of the Warlords (2007) แต่ผู้เล่นอีกส่วนก็ไม่ได้ซื้อเครื่องเล่นเกม แต่ใช้โทรศัพท์มือถือแทน อย่างไรเกมปริศนาก็ได้รับความนิยมบนโทรศัพท์เช่นกัน อย่างเกม Angry Birds (2009), Cut The Rope (2010), Candy Crush Saga (2012), Threes (2014)



ภาพที่ 2.11 Picross (Nonogram)

ถึงแม้ว่าปริศนา กับวิดีโอเกมจะเกิดขึ้นมาพร้อมๆกัน เกมปริศนาก็ยังไม่มีท่าทีว่าจะกลายเป็นกระแสหลักของวงการเกม โดยมีข้อยกเว้นคือ Catherine (2011) เกมปริศนาที่มีความนิยมส่วนมากยังอยู่ในโทรศัพท์มือถือ และส่วนใหญ่จะเป็นเกมที่พวกเขาเปิดขึ้นมาเล่น ผู้เล่นเกมปริศนายังคงต้องการนักพัฒนาอินดี้หรือเกมเล็กๆที่จำหน่ายบนเครื่องเล่นเกมของ Nintendo พึ่งเป็นผู้เล่นกลุ่มเล็กๆ มาตรฐานของเกมปริศนายุคนี้ก็จะเป็นเกม Portal (2007), Braid (2008), Paper's Please (2013), Captain Toad (2014), Hakoboy! (2015), Stephen's Sausage Roll (2016), Cosmic Express (2017), Gorogoa (2017), Statik (2017)

ภาพที่ 2.12 Paper's Please

ภาพที่ 2.13 Stephen's Sausage Roll

บทที่ 3

วิธีการดำเนินงาน

3.1 วัสดุอุปกรณ์ (ระบุชนิด/รุ่นของเครื่องมือที่ใช้)

1. Git
2. GitHub
3. Godot Game Engine
4. Visual Studio Code
5. Inkscape
6. After Effects
7. ffmpeg
8. AnimationRecorder
9. Imagemagick
10. Freesound.org
11. Audacity
12. Godot Forum

3.2 วิธีการดำเนินการ

1. การออกแบบเกม

   ผู้เล่นจะรับบทเป็น

Level	Available Resources		Objective
	Machine	Resources	
1	Conveyor belt, Loom	Cottons, Money	3 Thread
2	Conveyor belt, Sewing Machine, Loom	Cottons, Money	5 Shirt
3	Conveyor belt, Sewing Machine, Dyeing Machine	Threads, Dyes, Thread	3 Dyed Shirt 2 Shirt
4	Conveyor belt, Sewing Machine, Packager	Threads	5 Packaged Thread 5 Packaged Shirt

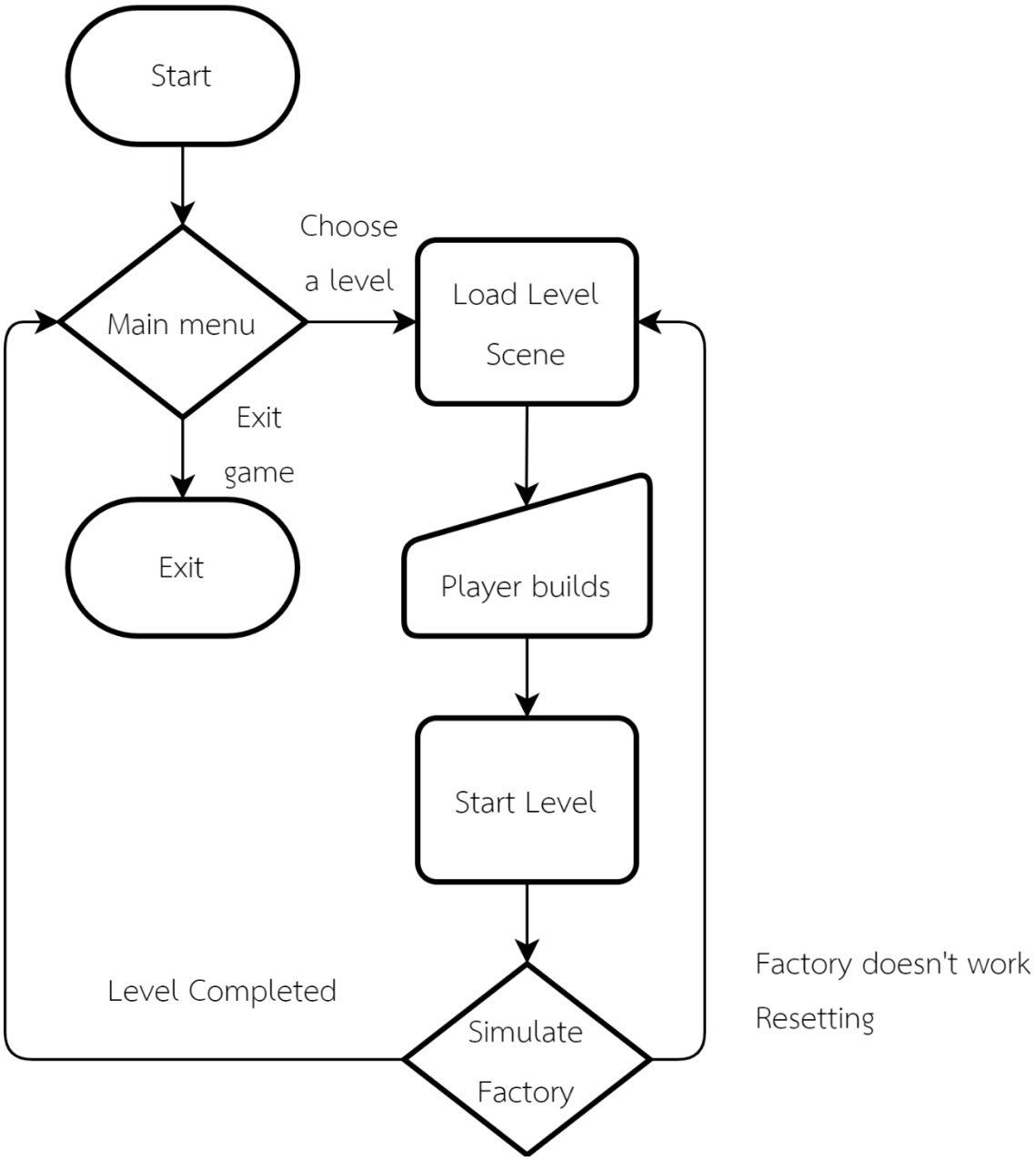
5	Conveyor belt, Sewing Machine, Silkscreener, Packager	Threads, Dyes	5 Packaged Shirt 5 Packaged Screened Shirt 5 Packaged Dyed Shirt
6	Conveyor belt, Sewing Machine, Silkscreener, Packager	Threads, Dyes	10 Packaged Screened A Dyed B Shirt 10 Packaged Screened B Dyed A Shirt

ตารางที่ 3.1.1 ข้อมูลของด่าน 1-6

Machine Type	Price (Coins)	Efficiency	Function
Conveyor	5	40/min	Manipulate item locations
Conveyor Splitter/Merger	10		
Loom	30	20/min	2 Cotton -> 1 Thread
Sewing Machine	40	20/min	2 Threads -> 1 shirt
Silkscreener	40	15/min	1 Shirt + 1 Dye -> 1 Screened shirt
Dyeing Machine	40	15/min	1 Shirt + 1 Dye -> 1 Dyed shirt
Packager	100	30/min	1 Shirt -> 1 Packaged Shirt

ตารางที่ 3.1.2 ข้อมูลของเครื่องจักรในเกม

2. ขั้นตอนในการดำเนินงาน
    1. ออกแบบเกม
    2. ออกแบบ Sprite
    3. พัฒนาเกม
    4. สร้าง Animation
    5. เล่น และทดสอบเกม



ภาพที่ 3.2.1 แผนผังการทำงานของเกม

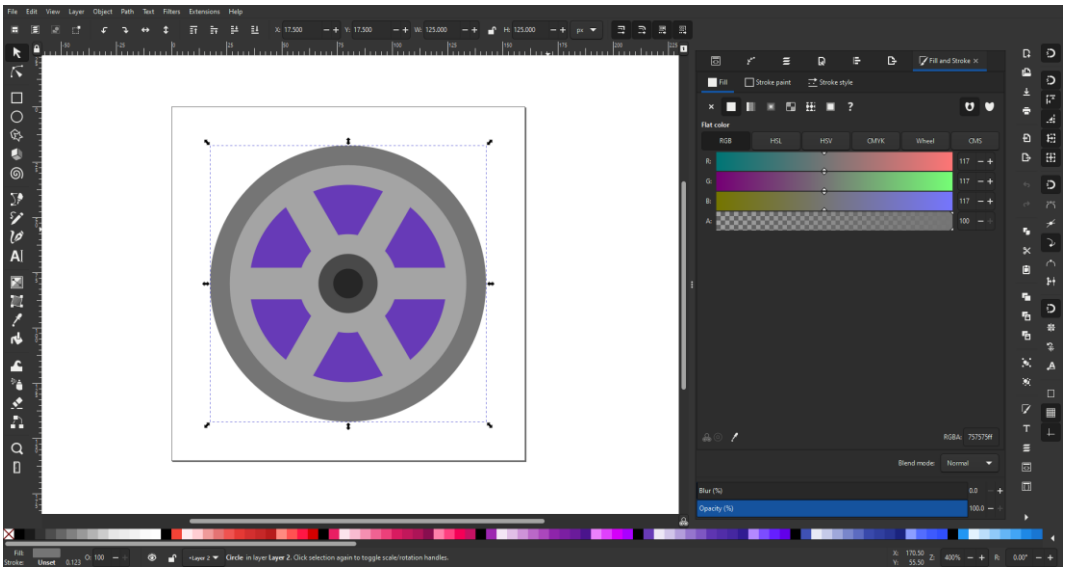
บทที่ 4

ผลการศึกษา

flaxibility ได้ใช้ระบบการติดตาม Source Code “Git” โดยผู้ที่มีความสนใจจะศึกษาการทำงานของเกม สามารถรับไฟล์ได้จาก GitHub ผ่านทาง https://phuwit.sunad.party/flaxibility หรือทาง https://github.com/phuwit/flaxibility

Sprites

Sprite ในเกมนี้จะเป็นรูปแบบ Vector Art สกุล svg ที่สร้างจากโปรแกรม Inkscape ซึ่ง Godot Engine รองรับไฟล์ประเภทนี้ได้อย่างดี ไฟล์ประเภทนี้มีข้อดีคือการแก้ไขได้ง่าย และสามารถเห็นการเปลี่ยนแปลงได้ทันทีใน Godot Engine เครื่องจักรมีการออกแบบอย่างเรียบง่าย ไม่มีรายละเอียดมาก มีการใช้สีขาว, เทา, ดำ, กับม่วงจาก Material Design Palette 2014 เป็นหลัก



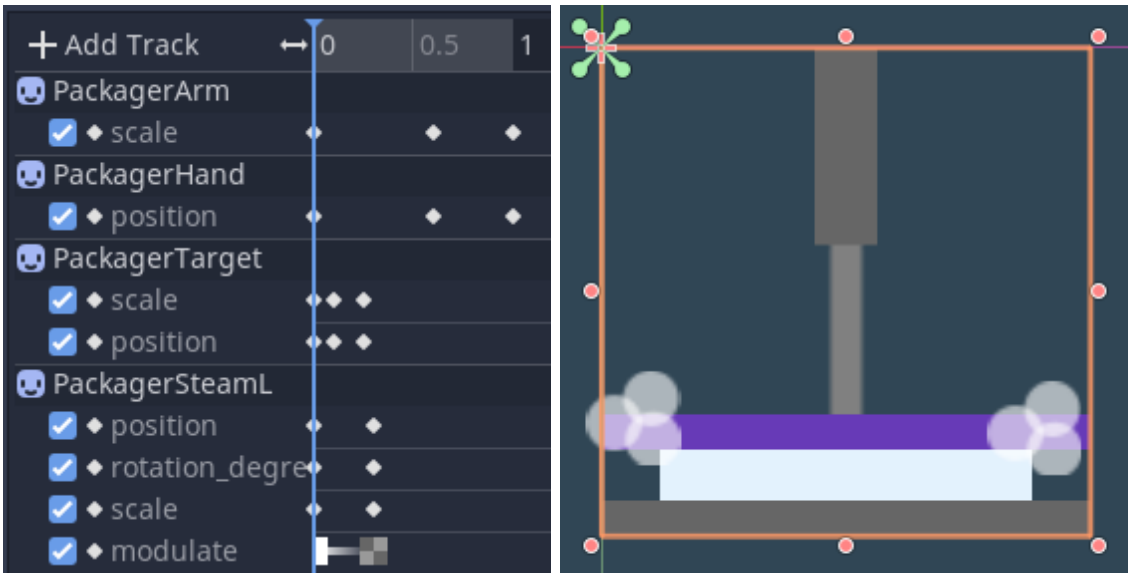
ภาพที่ 4.1 Inkscape UI

Logo ของเกมได้มีการออกแบบอย่างเรียบง่ายเช่นกัน และออกแบบเป็นเส้นยาวๆ เส้นหลักกับเส้นเล็กๆ ที่เป็นส่วนที่แยกออดมาจากตัวอักษรหลัก มีการร่างด้วย Procreate จนมีการออกแบบที่น่าพอใจ แล้วค่อยมาวาดอีกครั้งให้เป็น Vector Path

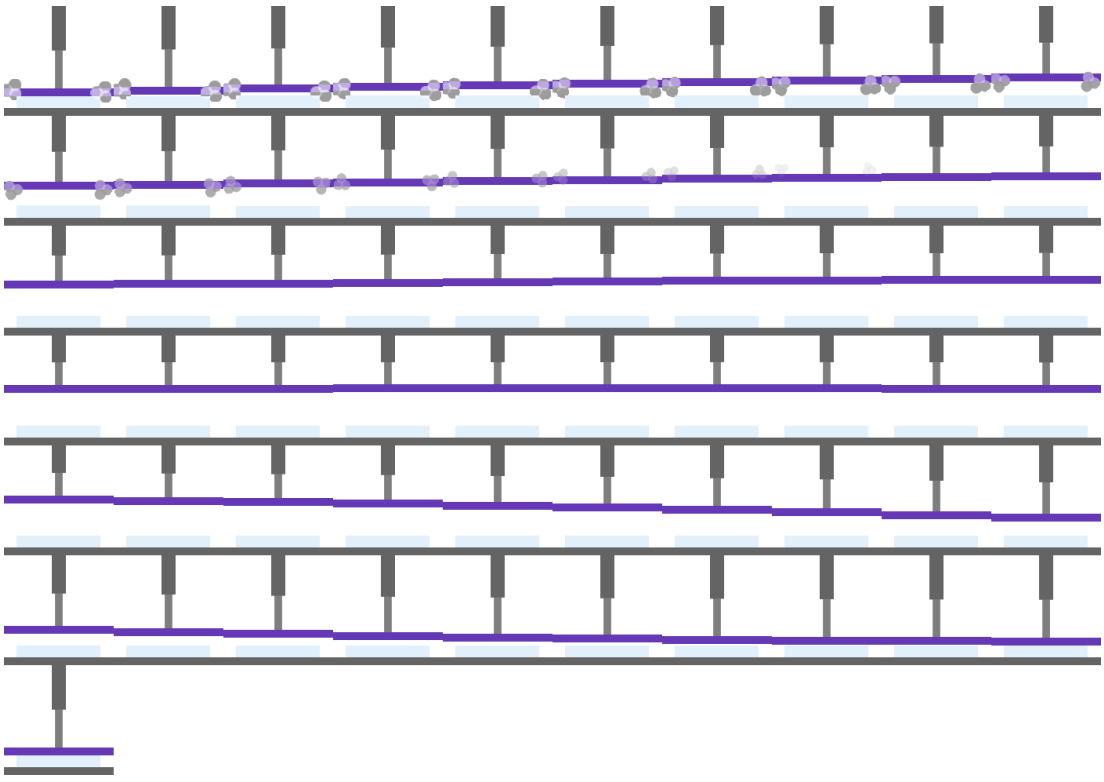
ภาพที่ 4.2 Logo

Animations

การทำภาพเคลื่อนไหว หรือการทำ Animation ของเกมนี้ทำขึ้นจากการแยกส่วนต่างๆ ของเครื่องจักร แล้วนำมาทำ Keyframe เพื่อเปลี่ยนแปลงคุณสมบัติต่างๆ ของชิ้นส่วนหนึ่ง เช่นขนาด, องศาการหมุน, การยืด, สี ,หรือความโปร่งใส จากนั้นจะมีการใช้ AnimationRecorder เพื่อสร้าง Spritesheet ของ Animation ดังกล่าว

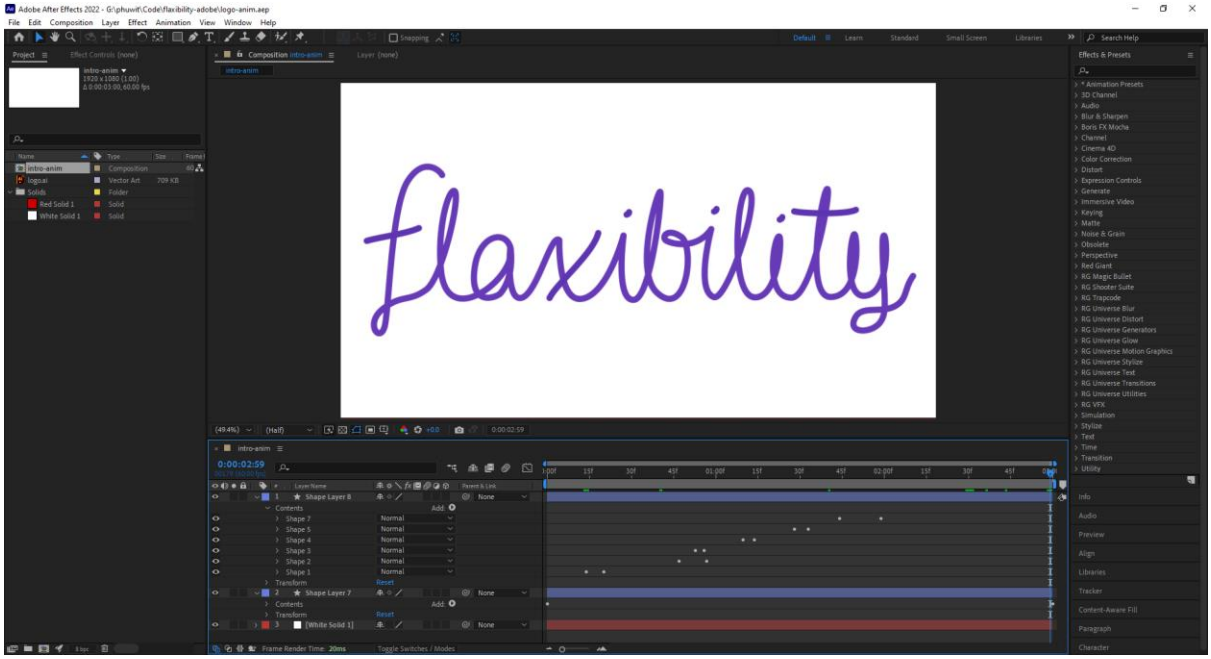


ภาพที่ 4.3 Packager Animation Keyframes



ภาพที่ 4.4 Packager Spritesheet

Logo Animation ได้มีการสร้างโดยใช้ After Effects แล้ว Export มาเป็น H.264 หลังจากนั้นจึงจะใช้ ffmpeg แปลงชนิดไฟล์เป็น Theora ที่ Godot สามารถอ่าน และเล่นได้



ภาพที่ 4.5 Logo Animation After Effects Project File

```
ffmpeg -i intro.mp4 -codec:v libtheora -qscale:v 10 intro.ogv
```

ภาพที่ 4.6 การใช้ ffmpeg แปลงชนิดไฟล์จาก H.264 เป็น Theora

Sounds

Sound Effects ที่ใช้อยู่ได้รับมาจาก freesound.org มีการดัดแปลงเสียง Click.wav ของผู้ใช้ kwahmah_02 โดยดัดแปลง +10 Semitones และ -10 Semitones โดยใช้ Audacity หลังจากนั้น ไฟล์เสียงทุกไฟล์จะผ่านการแปลงชนิดให้เป็น Vorbis โดยใช้ ffmpeg ซึ่งจะมีการเล่นเสียงเมื่อผู้เล่นกดปุ่ม หรือหยิบเครื่องจักรขึ้นมา

ภาพที่ 4.7 ดัดแปลงเสียง +10 Semitones โดยใช้ Audacity

```
ffmpeg -i audio.wav -codec:a libvorbis -qscale:a 8 audio.ogg
```

ภาพที่ 4.8 การใช้ ffmpeg แปลงชนิดไฟล์จาก Wave เป็น Vorbis

Menu

เมื่อผู้เล่นเปิดเกมมา ผู้เล่นจะเจอกับหน้า Splash Screen Logo Animation ผู้เล่นสามารถกดปุ่มใดก็ได้เพื่อข้าม หรือรอจน Animation จบ เกมจะเปลี่ยนหน้าไปเป็น Main Menu

```
func _on_VideoPlayer_finished():
    get_tree().change_scene('res://Menu/MainMenu.tscn')


func _input(event:InputEvent):
    if (event is InputEventKey)\
    or (event is InputEventMouseButton):
        get_tree().change_scene('res://Menu/MainMenu.tscn')
```

ภาพที่ 4.9 Splash Screen Source Code

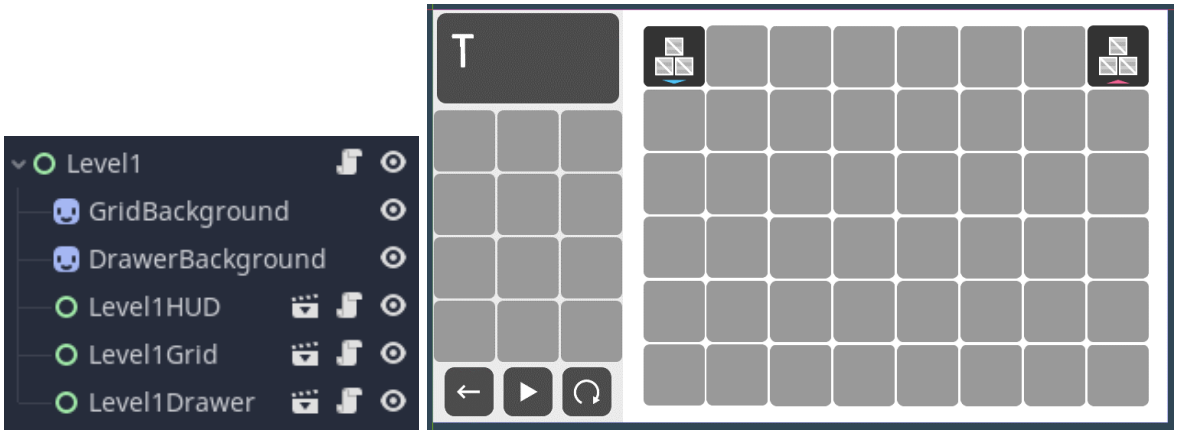
เมื่อสลับเข้ามาหรือออกจากหน้า Main Menu จะมี Animation Fade และผู้เล่นสามารถเลือกกดปุ่มเล่น หรือออกจากเกม เมื่อกดปุ่มเล่นเข้าไปเกมก็จะสลับไปยังด่านนั้นๆ หรือออกจากเกม

ภาพที่ 4.10 Scene/Scene Tree “Main Menu”

ผู้พัฒนาได้มีการจ้างการสร้างเพลงสำหรับเกมนี้ด้วยผ่าน fiverr.com เป็นเพลงพื้นหลังให้ผู้เล่นรู้สึกดื่มด่ำกับเกมมากขึ้น ซึ่งผู้จัดทำได้รับเพลงหลังจากหมดระยะเวลาในการพัฒนาแล้ว จึงยังไม่ได้ใช้งานในเกม

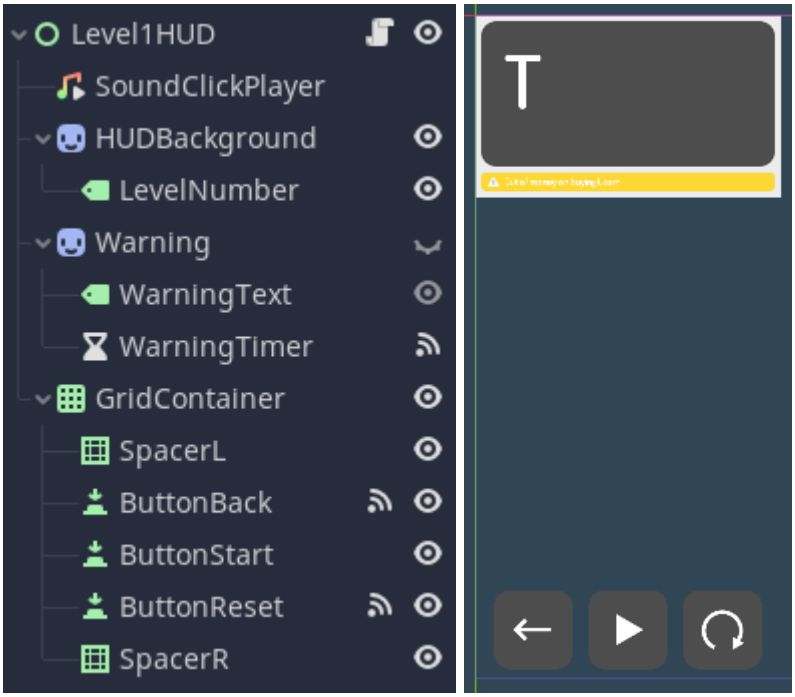
Levels

การออกแบบ UI ของแต่ละด่านจะแบ่งเป็น 3 ส่วนหลักๆ คือ HUD, Drawer, และ Grid



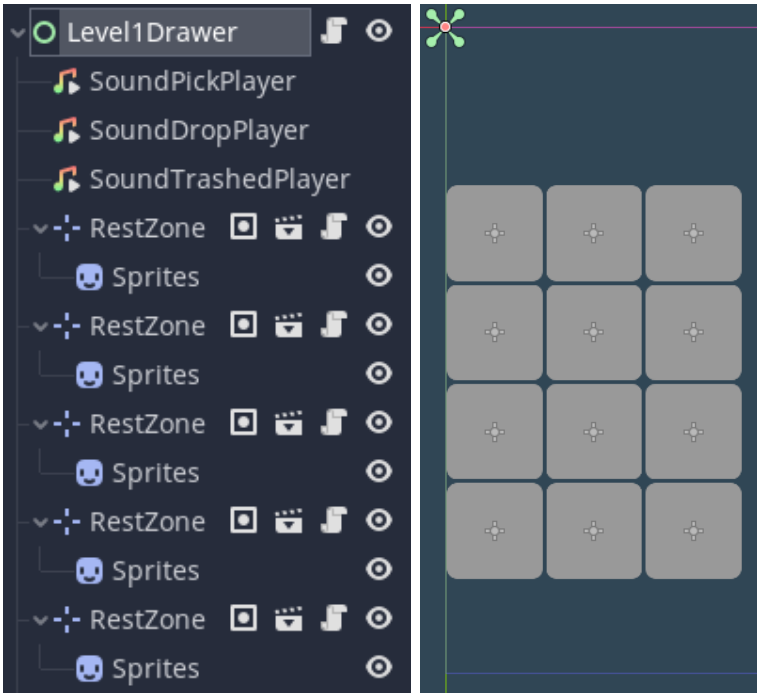
ภาพที่ 4.11 Scene/Scene Tree on Every Level

HUD จะเป็นส่วนที่แสดงจุดมุ่งหมาย คำเตือน ทรัพยากรที่มี และเงินของผู้เล่น รวมทั้งปุ่มควบคุมการเริ่มด่าน การเริ่มด่านใหม่ และการออกจากด่าน



ภาพที่ 4.12 Scene/Scene Tree on Every HUD

Drawer เป็นส่วนที่ผู้เล่นจะซื้อเครื่องจักรไปวางฝั่ง Grid จะมีการควบคุมการสร้างเครื่องจักรใหม่ และการหักเงินผู้เล่นเมื่อเครื่องจักรได้มีการซื้อไปแล้ว ถ้าผู้เล่นไม่ต้องการเครื่องจักรใดแล้ว สามารถนำมาขายและรับเงินคืนได้ Drawer จะมีการเก็บตำแหน่ง Rest Zone เป็น 2D Array เพื่อช่วยในการระบุตำแหน่ง และช่วยให้การพัฒนาง่ายขึ้น



ภาพที่ 4.13 Scene/Scene Tree on Every Drawer (Node Count Varies)

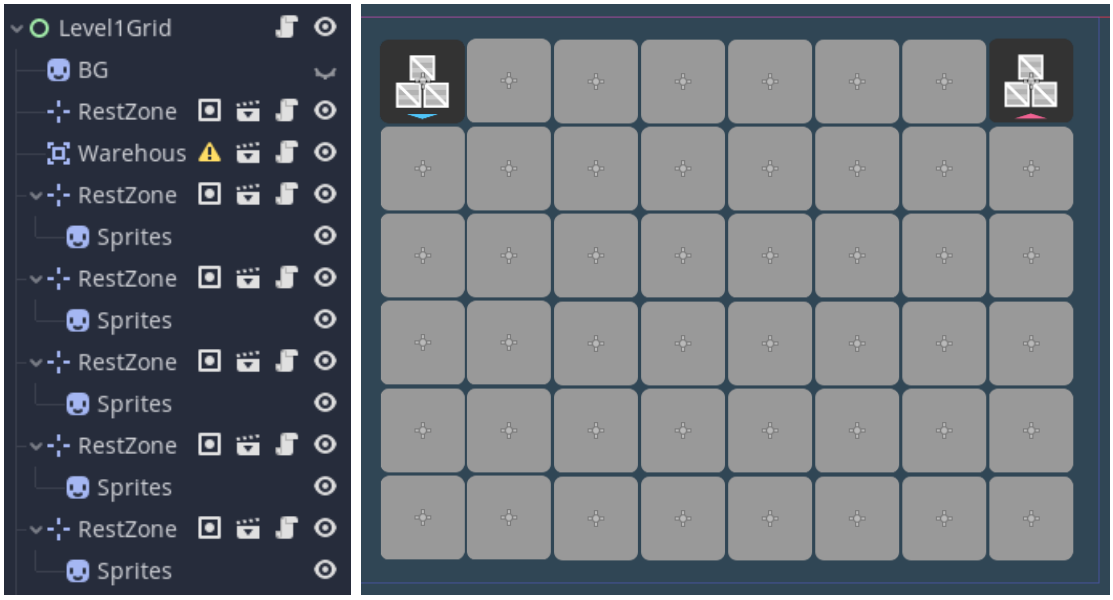
```
func spawn_machine(machine, restNode, reduceMoney):
    var newMachine = machine.instance()
    var machineCost = newMachine.cost
    var machineType = newMachine.type
    var containerName = 'Container' + machineType
    if reduceMoney == true:
        var moneyOpsSuccessful = money_ops(machineCost)
        if moneyOpsSuccessful == false:
            newMachine.queue_free()
            pass
    get_node(containerName).add_child(newMchine)
    add_to_group(machineType)
    connect_conveyor_signals(newMachine)
    newMachine.snap_to(restNode)
```

ภาพที่ 4.14 Machine Spawning Source Code

```
func money_ops(amount):
    if amount <= Global.money:
        Global.money -= amount
        return true
    else:
        return false
```

ภาพที่ 4.15 Money Operations(Deduction/Refund) Source Code

Grid เป็นส่วนโรงงานขนาด 8x6 ของผู้เล่นที่สามารถวางเครื่องจักรที่ซื้อมาและออกแบบโรงงานตามวัตถุประสงค์ของด่าน จะมี Warehouse In/Out เพื่อจ่ายทรัพยากร แบะรับสินค้าจากโรงงานของผู้เล่น และมีการเก็บตำแหน่ง Rest Zone เป็น 2D Array เช่นกัน



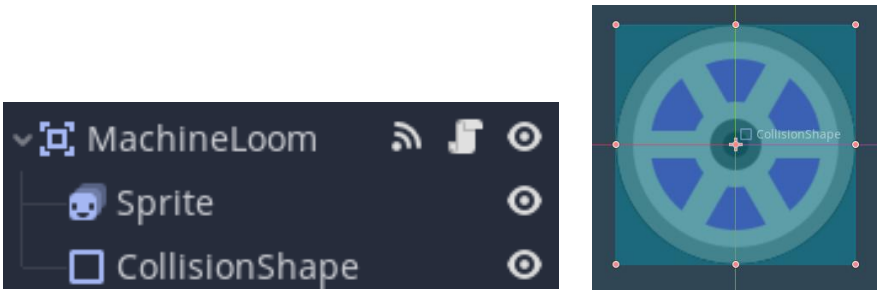
ภาพที่ 4.16 Scene/Scene Tree on Every Grid

```
func generate_pos_array(width, height, allNodes):
    var array = []
    var currentIndex = 0
    width += 1
    height += 1
    for y in range(height):
        array.append([])
        array[y].resize(width)
        for x in range(width):
            array[y][x] = allNodes[currentIndex]
            allNoes[currentIndex].posY = y
            allNodes[currentIndex].posX = x
            currentIndex += 1
    return array
```

ภาพที่ 4.17 Rest Zone Array Generation Source Code

Drag And Drop

ระบบ Drag And Drop ของ flaxibility มีฐานมาจาก godot_tutorial_content ของ bramreth ซึ่งเรามีการการดัดแปลงจากต้นฉบับอย่างมาก เครื่องจักรทุกเรื่องจะแยกเป็น Scene ที่มี Area2D เป็น Root Node ซึ่งจะมี Child Node เป็นAnimated Sprite ตามชนิดของเครื่องจักร และ CollisionShape2D ขนาด 140x140 pixels สำหรับเครื่องจักรทั่วไป และขนาด 160x160 pixels สำหรับเครื่องจักรชนิด Conveyors



ภาพที่ 4.18 Scene/Scene Tree “Loom”

เกมจะรับ Input เมื่อ Cursor อยู่ในเขตของ CollisionShape2D และผู้เล่นได้กดปุ่มเม้าส์ซ้าย กลไกของระบบ Drag And Drop ก็จะทำงาน โดยจะเกมเปลี่ยนแปลงตำแหน่งของเครื่องจักรให้อยู่ในเดียวกับ Cursor ทุก Frame จนกว่าผู้เล่นจะปล่อยมือจากปุ่ม เมื่อปล่อยจะทำให้กลไกการ Snap ไปที่ Rest Zone ทำงาน เกมจะคำนวณระยะห่างของเครื่องจักร กับตำแหน่งของ Rest Zone ทุกตำแหน่ง ถ้าระยะต่ำกว่าที่กำหนดไว้ เครื่องจักรจะ Snap ไปที่จุดนั้นๆ โดยการเปลี่ยนตำแหน่งแบบ Linear Interpolation (lerp)

```
func _on_ConveyorStraight_input_event(_viewport:Node, event:InputEvent, _shape_idx:int):
    if (event is InputEventMouseButton):
            if (event.pressed == true)\
                and (event.button_index == BUTTON_RIGHT):
                    clickR = true
                    rotate_conveyor()
            elif (event.pressed == false)\
                and (event.button_index == BUTTON_RIGHT):
                    clickR = false
            get_tree().set_input_as_handled()

func rotate_conveyor():
    var rotationsDirections = ['north', 'east', 'south', 'west']
    var newRotationsIndex = rotationsDirections.find(conveyorRotation) + 1
    if newRotationsIndex > (rotationsDirections.size() - 1):
        newRotationsIndex -= rotationsDirections.size()
    conveyorRotation = rotationsDirections[newRotationsIndex]
    self.rotation_degrees += 90
```

ภาพที่ 4.19Conveyor Rotating Add-on Source Code

```
func _process(delta):
    if (clickL == true):
        global_position = get_global_mouse_position()
    else:
        global_position = lerp(global_position, restNodePos, 10 * delta)

func _on_MachineLoom_input_event(_viewport:Node, event:InputEvent, _shape_idx:int):
    if (event is InputEventMouseButton):
        if (Global.money >= cost):
            if (mouseOver == true)\
            and (event.button_index == BUTTON_LEFT)\
            and (event.pressed == true):
                clickL = true
            elif (event.pressed == false):
                clickL = false
                snap_to_nearest_rest_node()
            get_tree().set_input_as_handled()

func snap_to_nearest_rest_node():
    for child in Global.allRestNodes:
        var distanceToRest = global_position.distance_to(child.global_position)
        if distanceToRest < shortestDist and child.selected == false:
            snap_to(child)

func snap_to(restNode):
    if currentNode:
        currentNode.selected = false
    restNode.machine = self
    restNode.select()
    currentNode = restNode
    restNodePos = restNode.global_position
```

ภาพที่ 4.20 Core Drag and Drop Source Code

Conveyors

เครื่องจักรธรรมดาจะมีตัวแปร Object Input/Output และ Conveyor จะมีตัวแปร Holding ก่อนที่จะมีความสามารถในการย้ายสิ่งของ Conveyor ต้องมีการทราบเครื่องจักรที่อยู่รอบๆ ก่อน

```
func get_nearby_absolute_position():
    absoluteNorthY = currentPosY - 1
    absoluteNorthX = currentPosX

    absoluteEastY = currentPosY
    absoluteEastX = currentPosX + 1

    absoluteSouthY = currentPosY
    absoluteSouthX = currentPosX + 1

    absoluteWestY = currentPosY
    absoluteWestX = currentPosX - 1
```

ภาพที่ 4.21 Getting Nearby Rest Node Position Source Code

หลังจากนั้นจะมีการเทียบการหมุนของ Conveyor กับ Node รอบๆ เกมจะได้ทราบว่าต้องรับ และส่งของไปทางใด แล้วจะมีการเช็คว่าเป็น Rest Node ที่มีอยู่บน Grid และ มีเครื่องจักรอยู่ จึงจะส่งของมูลเครื่องจักรให้ Conveyor

```
match conveyorRotation:
        'north':
            targetPosY = absoluteNorthY
            targetPosX = absoluteNorthX
        'east':
            targetPosY = absoluteEastY
            targetPosX = absoluteEastX
        'south':
            targetPosY = absoluteSouthY
            targetPosX = absoluteSouthX
        'west':
            targetPosY = absoluteWestY
            targetPosX = absoluteWest
```

ภาพที่ 4.22 Choosing Target Nodes based on Rotation Source Code

```
func check_legal_pos(posY, posX):
    if (posY ≥ 0) and (posX ≥ 0)\
    and (posY ≤ Global.gridColumn)\
    and (posX ≤ Global.gridRows):
        return true
    else:
        return false
```

ภาพที่ 4.23 Checking if the Rest Node is Legal Source Code

```
func get_node_from_pos(posY, posX):
    if (check_legal_pos(posY, posX) == false):
        return null
    elif (Global.restNodesGridPos[posY][posX] ≠ null):
        return Global.restNodesGridPos[posY][posX]
```

ภาพที่ 4.24 Returning Machine Object Source Code

เมื่อมีการเช็คเรียบร้อยแล้ว การย้ายสิ่งของจะเริ่มขึ้น ซึ่งจะแบ่งเป็น 2 แบบ คือการรับของ หรือการส่งของ ซึ่งจะแตกต่างกันตามชนิดของ Source และ Target การรับของจะทำงานเมื่อ Source เป็นชนิด Conveyor หรือ Warehouse Out การส่งของจะทำงานเมื่อ Target เป็นชนิด Machine หรือ Warehouse In

```
if source:
    if (source.type.begins_with('Conveyor')):
        source.remove_child_in_HoldingContainer()
        holding = source.holding
        source.holding = null
    elif (source.type == 'Warehouse'):
        if (source.interfaceMode == 'out')\
            and (source.stock > 0):
            holding = source.stockTemplatePacked.instance()
            source.stock -= 1
    elif (source.output != null):
        holding = source.output
        source.output = null
        get_node("HoldingLabel").text = holding
```

ภาพที่ 4.25 Taking Items Source Code

```
if target:
    if (target.type == 'Warehouse'):
        if (source.interfaceMode == 'in'):
            source.stock += 1

    elif (target.input == null):
        target.input = holding
        holding = null
        get_node("HoldingLabel").text = 'text'
```

ภาพที่ 4.26 Sending Items Source Code

บทที่ 5
อภิปราย และสรุปผลการศึกษา

5.1 อภิปรายผลการศึกษา

จาการสร้างเกม flaxibility เป็นเวลา 4 เดือนโดยประมาณ พบว่าเกมยังมีปัญหาบางส่วนและผิดกับที่คาดไว้ในตอนแรกค่อนข้างมาก ทั้งนี้เกมยังคงสามารถพัฒนาและต่อยอดเพิ่มเติมได้อีกเพื่อกลุ่มเป้าหมายที่แตกต่างกันไป

5.2 สรุปผลการศึกษา

การพัฒนาเกมเพื่อเป็นสื่อในการสอนการคิดแบบเป็นขั้นเป็นตอนภายใต้ชื่อ flaxibility: Drag and Drop Factory Managing Puzzle Game มีความเป็นไปได้ที่จะสอดคล้องตามวัตถุประสงค์ของโครงงาน เพียงแต่ต้องปรับปรุงตัวเกมให้เสถียรขึ้น

5.3 ข้อเสนอแนะ

- โครงงานนี้ไม่จำเป็นจะต้องเป็นเกมที่พัฒนาทุกระบบตั้งแต่ต้น ซึ่งถ้าทำเป็นเป็น Mod สำหรับเกมอื่น เช่น Minecraft อาจจะช่วยให้การพัฒนาง่ายขึ้น และทำให้เกมมีคุณภาพมากขึ้น โดยผู้เล่นอาจจะคุ้นชินกับการเล่นเกมนั้นๆ และทำให้มีผู้สนใจเล่นมากขึ้น
- การพัฒนาเกมมีข้อจำกัดด้านเวลา จึงต้องลดขอบเขตของเกมลงมาพอควร จึงจะทันกำหนดเวลา

เอกสารอ้างอิง

Bellard, Fabrice, and the FFmpeg contributors. "FFmpeg Wiki." Last modified March 7, 2022. https://trac.ffmpeg.org/wiki.

Bramwell. "Smooth Drag N Drop: Godot Guide." *YouTube*. June 30, 2020. https://www.youtube.com/watch?v=iSpWZzL2i1o.

Google LLC. "The Color System - Material Design." Last modified 2014. https://material.io/design/color/the-color-system.html.

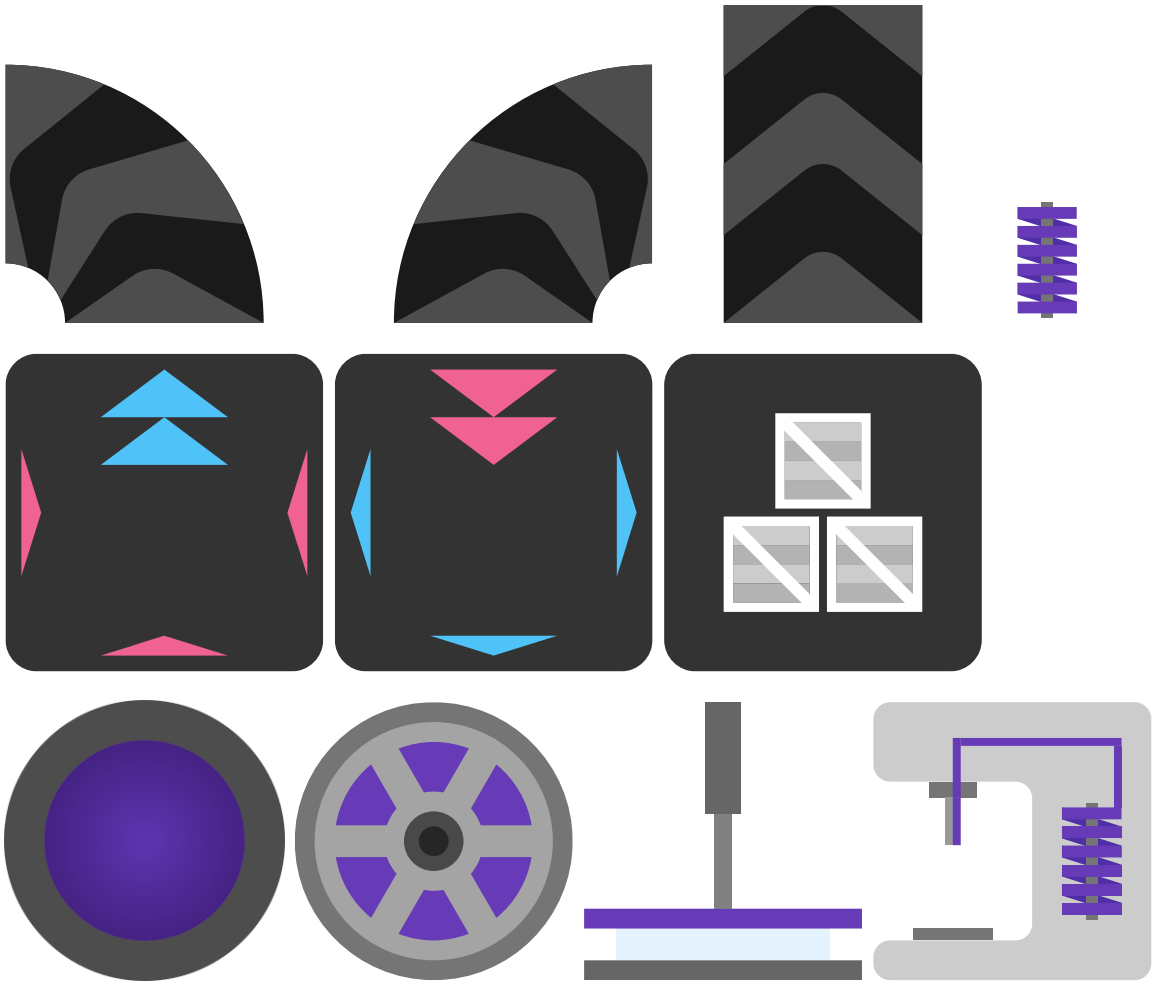
Linietsky, Juan, Ariel Manzur, and the Godot community. "Godot Engine - Free and open source 2D and 3D game engine." Last modified March 23, 2022. https://godotengine.org.

Linietsky, Juan, Ariel Manzur, and the Godot community. "Godot Engine Documentation." Last modified March 25, 2022. https://docs.godotengine.org/en/stable/.

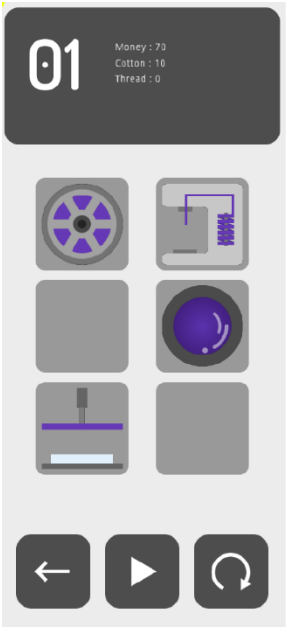
Linietsky, Juan, Ariel Manzur, and the Godot community. "Godot Engine Git repository." GitHub. Accessed March 28, 2022. https://github.com/godotengine/godot.

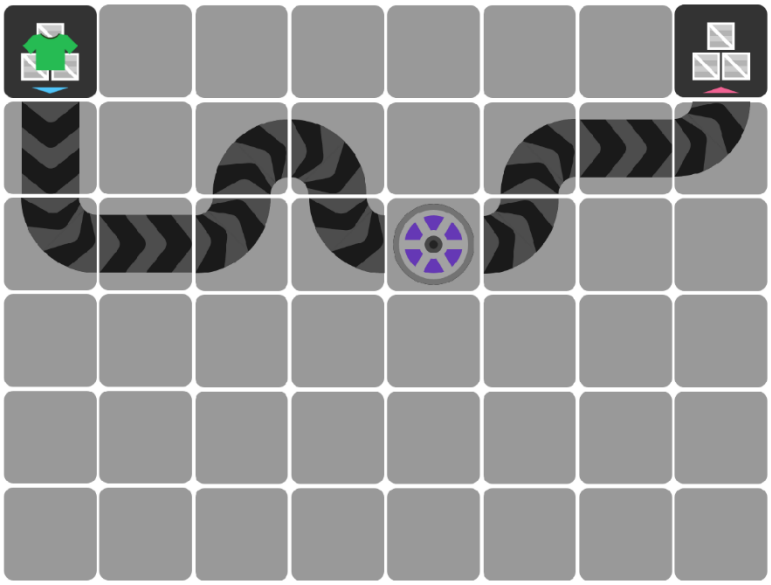
Tabarani, Ahmed E. "AhmedElTabarani/AnimationRecorder: A Tool That Recording Any Animation of AnimationPlayer." GitHub. Last modified November 6, 2021. https://github.com/AhmedElTabarani/AnimationRecorder.

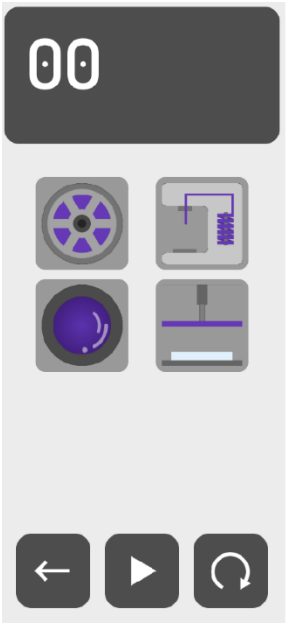
ภาคผนวก

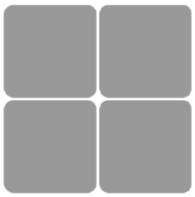


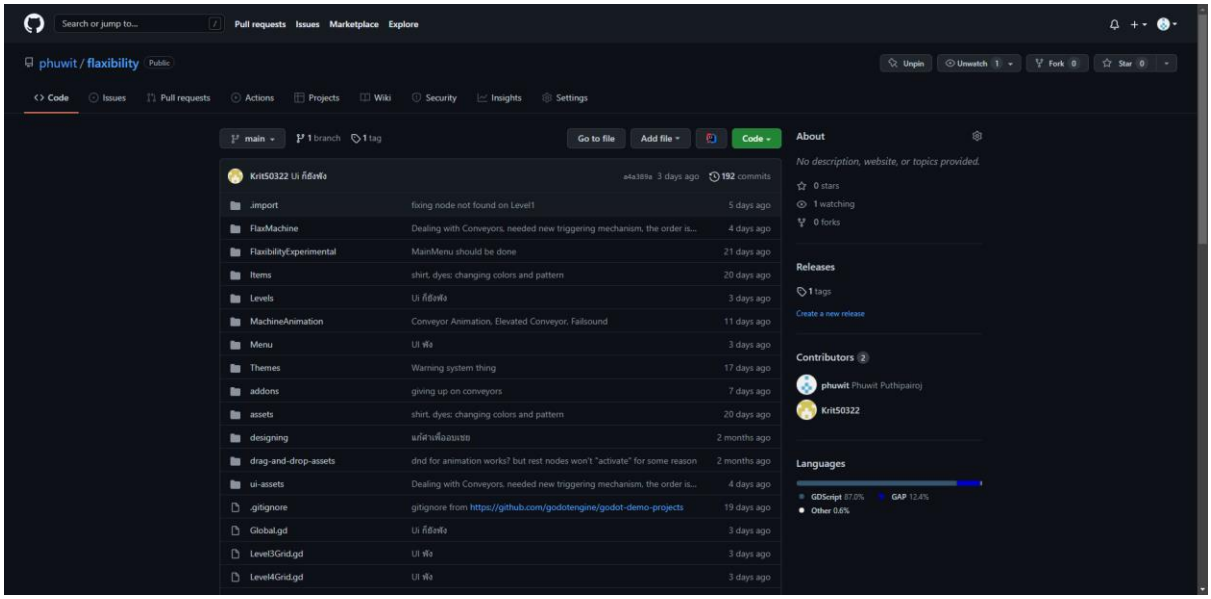
flexibility

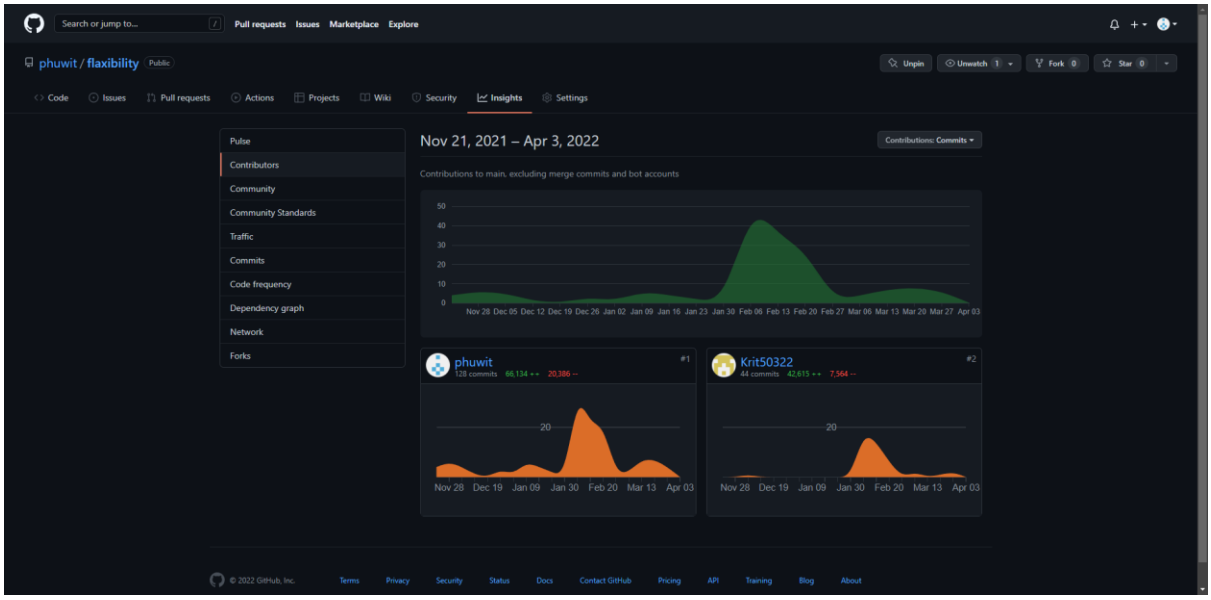

01
Money : 70
Cotton : 10
Thread : 0

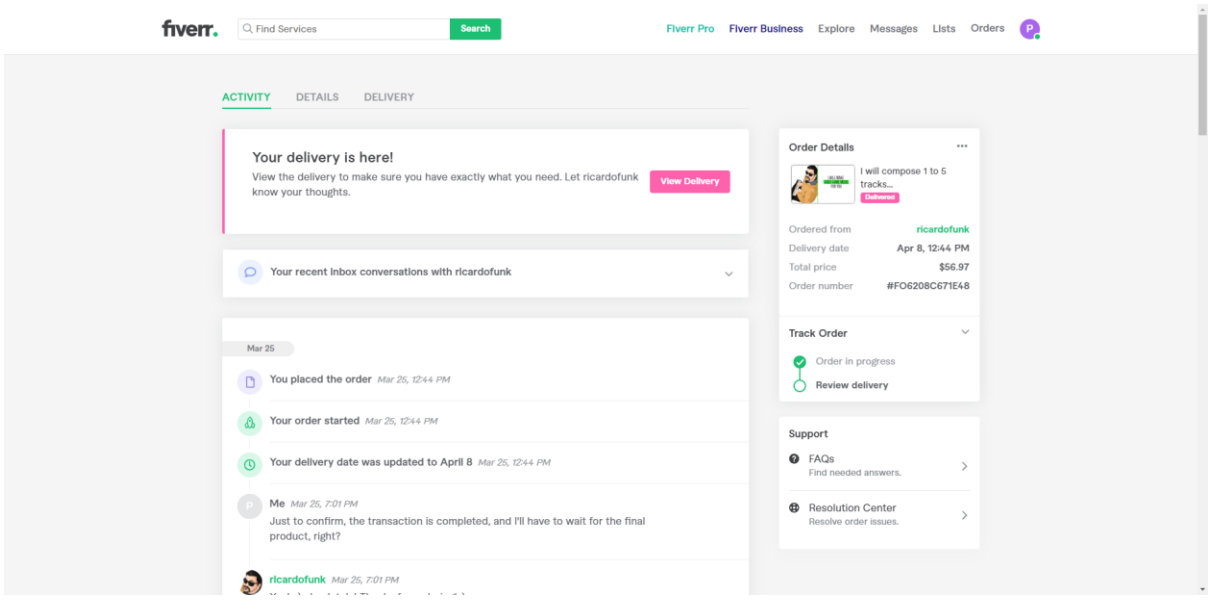



00




Search or jump to...
Pull requests
Issues
Marketplace
Explore
phuwit / flaxibility
Public
Unpin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
1 tag
Go to file
Add file
Code
Krit50322 Ui ก็ยังฟัง
a4a389a 3 days ago
192 commits
.import
fixing node not found on Level1
5 days ago
FlaxMachine
Dealing with Conveyors, needed new triggering mechanism, the order is...
4 days ago
FlaxibilityExperimental
MainMenu should be done
21 days ago
Items
shirt, dyes: changing colors and pattern
20 days ago
Levels
Ui ก็ยังฟัง
3 days ago
MachineAnimation
Conveyor Animation, Elevated Conveyor, Failsound
11 days ago
Menu
UI ฟัง
3 days ago
Themes
Warning system thing
17 days ago
addons
giving up on conveyors
7 days ago
assets
shirt, dyes: changing colors and pattern
20 days ago
designing
แก้คำเพื่อออมแขย
2 months ago
drag-and-drop-assets
dnd for animation works? but rest nodes won't "activate" for some reason
2 months ago
ui-assets
Dealing with Conveyors, needed new triggering mechanism, the order is...
4 days ago
.gitignore
gitignore from https://github.com/godotengine/godot-demo-projects
19 days ago
Global.gd
Ui ก็ยังฟัง
3 days ago
Level3Grid.gd
UI ฟัง
3 days ago
Level4Grid.gd
UI ฟัง
3 days ago
About
No description, website, or topics provided.
0 stars
1 watching
0 forks
Releases
1 tags
Create a new release
Contributors 2
phuwit Phuwit Puthipairoj
Krit50322
Languages
GDScript 87.0%
GAP 12.4%
Other 0.6%


Search or jump to...
Pull requests
Issues
Marketplace
Explore
phuwit / flaxibility
Public
Unpin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Pulse
Contributors
Community
Community Standards
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
Nov 21, 2021 – Apr 3, 2022
Contributions: Commits
Contributions to main, excluding merge commits and bot accounts
50
40
30
20
10
0
Nov 28 Dec 05 Dec 12 Dec 19 Dec 26 Jan 02 Jan 09 Jan 16 Jan 23 Jan 30 Feb 06 Feb 13 Feb 20 Feb 27 Mar 06 Mar 13 Mar 20 Mar 27 Apr 03
phuwit
#1
128 commits
66,134 ++
20,386 --
20
Nov 28 Dec 19 Jan 09 Jan 30 Feb 20 Mar 13 Apr 03
Krit50322
#2
44 commits
42,615 ++
7,564 --
20
Nov 28 Dec 19 Jan 09 Jan 30 Feb 20 Mar 13 Apr 03
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


fiverr.
Find Services
Search
Fiverr Pro
Fiverr Business
Explore
Messages
Lists
Orders
ACTIVITY
DETAILS
DELIVERY
Your delivery is here!
View the delivery to make sure you have exactly what you need. Let ricardofunk know your thoughts.
View Delivery
Your recent inbox conversations with ricardofunk
Mar 25
You placed the order Mar 25, 12:44 PM
Your order started Mar 25, 12:44 PM
Your delivery date was updated to April 8 Mar 25, 12:44 PM
Me Mar 25, 7:01 PM
Just to confirm, the transaction is completed, and I'll have to wait for the final product, right?
ricardofunk Mar 25, 7:01 PM
Order Details
I will compose 1 to 5 tracks...
Delivered
Ordered from ricardofunk
Delivery date Apr 8, 12:44 PM
Total price $56.97
Order number #FO6208C671E48
Track Order
Order in progress
Review delivery
Support
FAQs
Find needed answers.
Resolution Center
Resolve order issues.