บทที่ 2
ทฤษฎีและระบบงานที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เกม

เกมเป็นกิจกรรมอย่างหนึ่งที่มอบความบันเทิงให้แก่มนุษย์หรือบางครั้งอาจใช้ประโยชน์ในด้านการศึกษาโดยมีโครงสร้างหลักประกอบด้วยเป้าหมาย และกฎกติกาที่ถูกกำหนดโดยมนุษย์สำหรับการแข่งขันหรือพัฒนาทักษะด้านร่างกายการใช้พละกำลังหรือความคิดเพื่อเอาชนะ ตัวอย่างเกมยุคแรกที่ยังคงได้รับความนิยมในยุคปัจจุบัน ได้แก่ ไพ่นกกระจอก ชักเย่อ หมากรุก หมากล้อม เป็นต้น

2.1.2 เกม PC

เกม PC หมายถึงเกมที่เล่นบนเครื่องคอมพิวเตอร์ส่วนบุคคล เกมคอมพิวเตอร์ส่วนบุคคลเริ่มแรกมีการพัฒนาให้มีรูปแบบการเล่นและกราฟิกที่เรียบง่าย ก่อนที่จะมีรูปแบบสลับซับซ้อนอย่างที่เห็นในปัจจุบันเกมคอมพิวเตอร์ส่วนบุคคลถูกผลิตขึ้นมาโดยผู้พัฒนาเกมหนึ่งคนหรือมากกว่า ซึ่งส่วนใหญ่มักจะเป็นการรวมตัวกันของผู้เชี่ยวชาญหลายด้านที่รับผิดชอบในการดำเนินงานในตำแหน่งต่าง ๆ เช่น ตำแหน่ง Animator ที่รับหน้าที่ออกแบบการเคลื่อนไหวและฉากต่าง ๆ ตำแหน่ง Audio Engineer ที่รับผิดชอบเรื่องเสียงประกอบฉากของเกมหรือเสียงเอฟเฟคต่าง ๆ ที่เกิดขึ้นภายในเกม และตำแหน่ง Game Programmer รับผิดชอบการเขียนโค้ดให้กับเกม เป็นต้น

2.1.3 เกมแอ็คชัน (Action Game)

เกมแอ็คชันเน้นการใช้ประสาทสัมผัสทั้งสายตาและมือในการควบคุมตัวละคร เพื่อให้ผ่านด่านภายในเกมหรือเพื่อที่จะทำให้ผ่านเควสที่เกมได้กำหนด เกมแอ็คชันบางเกมไม่ใช่ว่าผู้เล่นต้องเล่นทำเควสตามด่านหรือต้องเอาชนะบอสให้ได้เพื่อจะผ่านด่านเท่านั้น แต่จะมีประเภทที่เกมจะให้เล่นไปเรื่อย ๆ ไม่มีจุดหมายที่แน่นอนว่าทำแบบนี้ผ่านหรือไม่ผ่าน แต่จะให้ผู้เล่นเก็บไอเทมตามด่านหรือทำการกำจัดมอนสเตอร์เพื่อรับคะแนนในการเล่นในแต่ละรอบแทน แอ็คชันเป็นเหมือนคำเรียกภาพรวมของเกมประเภทต่าง ๆ เช่น เกมแนวยิงปืน ผจญภัย หรือ RPG ที่เป็นแนวเกมแยกย่อยออกมาอีกที ตัวอย่างเกม เช่น Bayonetta, Devil May Cry 5, Nier: Automata เป็นต้น

ภาพประกอบที่ 2.1 Devil May Cry 5

ที่มา: https://game.mthai.com/app/uploads/2019/03/devil-may-cry-5-e3-2018-1.jpg

2.1.4 เกมแอ็คชันโร้คไลค์ (Action Rogue-Like)

เกมแอ็คชันโร้คไลค์คือเกมที่มีลักษณะสำคัญของการ Random หรือสุ่มไม่ว่าจะเป็นด่าน อาวุธ ไอเทม สกิล และศัตรูภายในเกมจะทำการสุ่มทุกครั้งที่คุณเริ่มเกมใหม่ทำให้ผู้เล่นต้องปรับตัวให้ยืดหยุ่น ตามไอเทมที่ได้รับมา เช่น รอบนี้อาจจะได้รับไอเทมที่เข้ากับด่านทำให้กำจัดมอนสเตอร์ได้ง่าย แต่รอบถัดไปได้รับไอเทมที่ไม่เข้ากับด่านที่กำลังเล่น ก็อาจจะทำให้การต่อสู้กับมอนสเตอร์ในรอบนั้นลำบากขึ้น อย่างไรก็ตามการที่ได้รับไอเทมที่ใช้งานยากไม่ได้แปลว่าจะทำให้ผ่านด่านในรอบนั้นไม่ได้เพราะมันขึ้น อยู่กับตัวผู้เล่นเองว่าจะสามารถปรับตัวให้เข้ากับอาวุธและด่านในรอบนั้นได้ดีแค่ไหน

ดังนั้น ความสนุกของมันคือ “การปรับตัว” ที่ผู้เล่นจะต้องพิจารณาว่า ตานี้เราสุ่มได้ไอเทมนี้มา เราจะเล่นยังไง แบบไหน ค่อย ๆ ไปดีไหม หรือว่าจะลุยแหลกเลยดี ตัวอย่างเกม เช่น Enter the Gungeon, Vampire Survivors, The Binding of Isaac, Dead Cells เป็นต้น

ภาพประกอบที่ 2.2 Vampire Survivors

ที่มา:

https://cdn.akamai.steamstatic.com/steam/apps/1794680/ss_054159adc52856d066d48bda02866da524c43439.jpg?t=1657145362

2.1.5 มีม (Meme)

มีมเป็นรูปแบบของความคิดทางวัฒนธรรม สัญลักษณ์ หรือการปฏิบัติ ที่สามารถส่งผ่านจากจิตใจคนหนึ่งไปสู่อีกคนหนึ่ง ผ่านการเขียน การพูด ท่าทาง พิธีกรรม ภาพล้อเลียน และมีคำศัพท์ต่าง ๆ ในภาพที่มีความหมายเชิงตลก หรือปรากฏการณ์ลอกเลียนแบบอื่น ๆ คำว่า meme ในภาษาอังกฤษมาจากการผสมของคำว่า "gene" (ยีน หรือ สิ่งสืบต่อพันธุกรรม) นักวิทยาศาสตร์ชาวอังกฤษ ริชาร์ด ดอว์กินส์ ได้คิดคำว่า "meme" ขึ้นมาในหนังสือ “The Selfish Gene (1976)” ในแนวคิดเกี่ยวกับการอภิปรายทฤษฎีวิวัฒนาการเกี่ยวกับการอธิบายการแพร่ของความคิดและปรากฏการณ์ทางวัฒนธรรม โดยรากศัพท์ของคำว่า meme มาจากศัพท์กรีก mimema (มีเมมา) ที่แปลว่าการลอกเลียน ในปี 2013 ดอว์กินส์ ได้กำหนดรูปแบบ Meme ทางอินเทอร์เน็ตว่าเป็นสิ่งที่เปลี่ยนแปลงโดยเจตนาโดยความคิดสร้างสรรค์ของมนุษย์ ซึ่งแตกต่างจากแนวคิดดั้งเดิมของเขาที่เกี่ยวข้องกับการกลายพันธุ์ ในเวลาต่อมามีมถูกใช้เรียกแทนมุกตลก ภาพล้อเลียน ภาพทำซ้ำ ที่แพร่หลายออกมา

2.1.6 อนิเมชั่น (Animation)

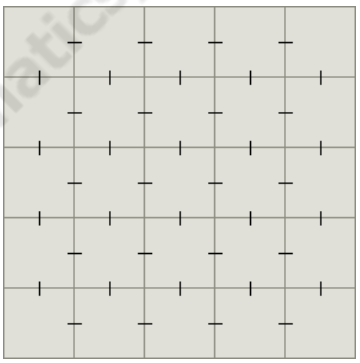
อนิเมชั่น คือ การนำเอาภาพนิ่งที่เรียกว่าสไปรต์ (Sprite) มาเรียงต่อกัน แล้วทำการเล่นภาพเหล่านั้นด้วยความเร็วที่เหมาะสมจะทำให้เห็นภาพติดตา เหมือนกำลังเคลื่อนไหวอยู่จริง ๆ แบ่งได้ 2 ประเภท คือ

(1) 2D Animation คือ ภาพเคลื่อนไหวแบบ 2 มิติ มองเห็นทั้งความสูงและความกว้างซึ่งจะมีความเหมือนจริงพอสมควร และในการสร้างจะไม่สลับซับซ้อนมากนัก เช่น ภาพเคลื่อนไหวที่ปรากฏตามเว็บต่าง ๆ รวมทั้ง Gif Animation เป็นต้น

(2) 3D Animation คือ ภาพเคลื่อนไหวแบบ 3 มิติ มองเห็นทั้งความสูง ความกว้าง และความลึก ภาพ ที่เห็นจะมีความสมจริงมากถึงมากที่สุด เช่น ภาพยนตร์ เรื่อง Toy Story, Shrek เป็นต้น

2.1.7 การค้นหาเส้นทาง (Pathfinding)

การทำงานโดยใช้การค้นหาแบบ Path Finding นั้น จะต้องใช้การคำนวณทางคณิตศาสตร์เข้าช่วย ซึ่งจะแตกต่างกันไปตามลักษณะของ Algorithm นั้น ๆ โดย Algorithm ส่วนใหญ่ที่ใช้ในการค้นหาแบบ Path Finding นั้นจะอธิบายด้วยการใช้กราฟที่มีการเชื่อมต่อเป็นโหนด ๆ โดยแต่ละโหนดจะมีเส้นทางเชื่อมไปหากัน การค้นหาเส้นทางเหล่าใช้เพื่อให้ตัวละคร npc เข้ามาหาที่ตำแหน่งของผู้เล่น

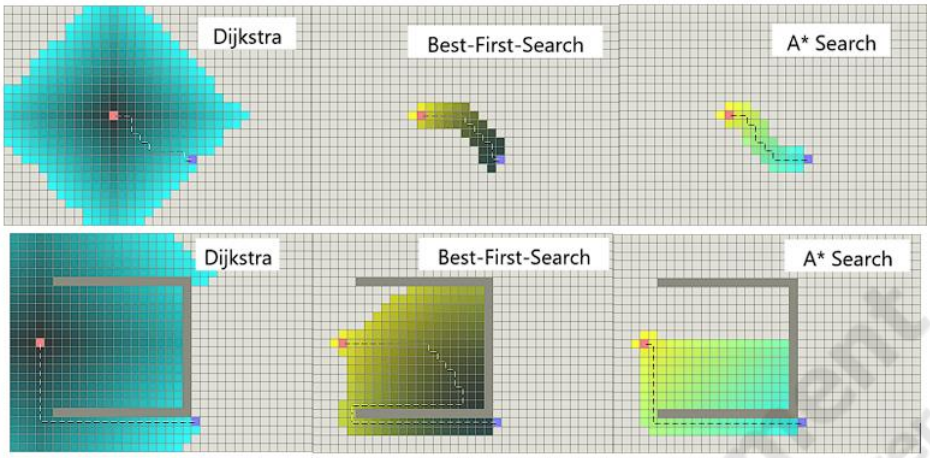


ภาพประกอบที่ 2.3 กราฟแสดงตัวอย่าง Algorithm

ที่มา: https://docs.godotengine.org/en/3.3/_images/player_coll_shape1.png

2.1.8 การค้นหาแบบ A* (A Star Algorithm)

A* นั้นนิยมมากที่สุดสำหรับการค้นหาเส้นทาง เนื่องจากค่อนข้างยืดหยุ่นและสามารถใช้ได้ในบริบทที่หลากหลายโดยที่ A* นั้นมีลักษณะคล้าย Algorithm ของ Dijkstra ที่สามารถใช้ค้นหาเส้นทางที่สั้นที่สุดได้และมันเร็วพอ ๆ กับ Greedy Best-First-Search



ภาพประกอบที่ 2.4 เปรียบเทียบการค้นหาเส้นทางของ Algorithm แบบต่าง ๆ

ที่มา: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

เอสตาร์ใช้การค้นหาตามแนวกว้างและหาเส้นทางที่มีระยะทางน้อยที่สุดจากโหนดแรกไปสู่โหนดเป้าหมายซึ่งอาจจะมีได้หลายโหนด โดยใช้ฟังก์ชันฮิวริสติกแบบค่าของระยะทาง (ใช้สัญลักษณ์ f(x)) เพื่อที่จะหาลำดับการผ่านโหนดในกราฟ โดยฟังก์ชันดังกล่าวเป็นผลรวมของสองฟังก์ชันดังนี้

(1) ค่าระยะทางทางจากโหนดเริ่มต้นมายังโหนดปัจจุบัน (ใช้สัญลักษณ์ g(x))

(2) ค่าประมาณฮิวริสติกที่ยอมรับได้ ของระยะทางจนถึงจุดหมาย (ใช้สัญลักษณ์ h(x))

หลักการทำงานของเอสตาร์คือ เมื่อเอสตาร์ท่องไปในกราฟ เอสตาร์จะเลือกเส้นทางที่มีค่าน้อยที่สุดที่มันทราบ โดยเก็บ Priority Queue ของเส้นทางอื่น ๆ ไว้ ถ้าระหว่างที่เอสตาร์ท่องไปแต่ละจุดแล้วเจอเส้นทางที่มีค่ามากกว่าเส้นทางอื่น ก็จะไปเลือกเส้นทางที่มีค่าน้อยกว่าใน Priority Queue แทน กระบวนการนี้จะทำต่อไปเรื่อย ๆ จนกระทั่งถึงจุดหมาย

2.2 ระบบงานที่เกี่ยวข้อง

2.2.1 Game Engine

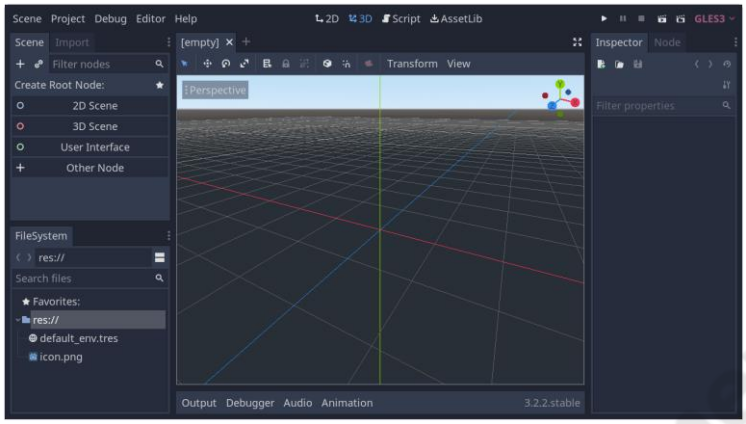
Game Engine คือ โปรแกรมที่ใช้สำหรับสร้างเกมโดยตัวโปรแกรมจะมีสิ่งอำนวยความสะดวกให้กับนักพัฒนามากมาย อย่างการ Rander ภาพ 2D หรือการสร้างโมเดล 3D สามารถนำภาพ 2D มาสร้างเป็น Animation ตัวละครของเกม มีระบบฟิสิกส์ให้สิ่งของภายในเกม มีระบบการเล่นเสียงเพลงพื้นหลังหรือเอฟเฟคประกอบการกระทำต่าง ๆ และสามารถสร้างเป็นฉาก 2D ต่าง ๆ จาก Tilemap กำหนดองค์ประกอบต่าง ๆ ภายในเกม ตัวอย่าง game engine ที่ใช้ในการพัฒนาเกม Godot, Armory,

Unreal Engine, Cry Engine, Defold, Corona, Unity, Frostbite Engine, RE Engine, Source Engine เป็นต้น

โดย engine แต่ละตัวก็มีความสามารถแตกต่างกันไปเช่นตัว engine ที่นิยมนำมาสร้างเกม 3D เป็นหลักอย่าง Armory แต่มีข้อจำกัดในการใช้งานเพราะต้องมีความชำนาญระดับหนึ่งถึงจะสามารถใช้งานได้อย่างมีปรสิทธิภาพ และหากต้องการที่จะสร้างเกมที่มีความเสมือนจริง Cry Engine ก็เป็นตัวเลือกหนึ่งที่มีความสามารถมากพอจะสร้างเกมเสมือนจริงที่มีความคมชัดของภาพในระดับสูงและกำหนดรายละเอียดของภาพได้ดีอีกหนึ่งตัว หรือแม้แต่ engine ที่ดัง ๆ อย่าง Unreal Engine ที่เหมาะกับทั้งมือใหม่และคนที่ชำนาญแล้วสามารถพัฒนาได้ทั้งเกม 2D และ 3D ลองรับการพัฒนาได้ตั้งแต่เกม 2D ไปจนถึงเกม 3D กราฟฟิกสูง และ engine อีกตัวคือ Unity ที่ใช้งานง่ายทำได้ทั้งเกม 2D และ 3D มี package ให้มากมายสามารถเลือกนำไปพัฒนาตามแต่เกมที่เรากำลังสร้าง และ engine ที่สามารถทำได้ทั้ง 2D และ 3D อีกตัวคือ Godot ถึงแม้ตัว engine จะมีข้อจำกัดและไม่ได้ดังมากเท่า 2 engine ข้างบนแต่ก็เป็นอีก engine หนึ่งที่ใช้งานง่ายและใช้พื้นที่น้อยสามารถใช้พัฒนาเกมที่ไม่ได้มีองค์ประกอบกว้างมากได้ออกมาดีไม่ต่างกัน

2.2.2 Godot

Godot (ออกเสียง Go-doh) สามารถทำงานได้หลายแพลตฟอร์ม (Cross platform) เช่น Window, Mac OS เป็นต้น และเป็นโปรแกรม open source ใช้งานฟรี ภายใต้สัญญา MIT ซึ่งไม่มีพันธะข้อผูกมัดหรือค่าสิทธิ (Royalty) ในการใช้งานแต่อย่างใด ผู้ใช้งานจะเป็นเจ้าของเกมที่ตนเองสร้างตลอดจนโค้ดทั้งหมดของตัวเอนจินโดยสมบูรณ์ เดิมที่ถูกพัฒนาอยู่ที่ Argentina โดยนักพัฒนาซอร์ฟแวร์ Juan Linietsky และ Ariel Manzur ในปี 2007 ให้กับบริษัทที่อยู่แถว Latin America ก่อนที่จะทำเป็นโปรแกรมฟรีให้คนทั่วไปใช้งาน ในปี 2014 Godot ได้เข้าร่วมกับ Software Freedom Conservancy ซึ่งเป็นองค์กรไม่แสวงหาผลกำไร ตัวโปรแกรมถูกสร้างมาเพื่อพัฒนาเกม 2D และ 3D ที่เกมสามารถรองรับการรันบน PC, Mobile และ web ได้ นอกจากพัฒนาเกมแล้ว ตัวโปรแกรมยังสามารถทำอนิเมชันได้ด้วย ภาษาโปรแกรมที่ Godot รองรับมี C#, GDscript, Visualscript, C++, C การเขียน script ให้กับตัวเกมที่นิยมจะใช้ 2 ภาษาคือ C# และ GDscript



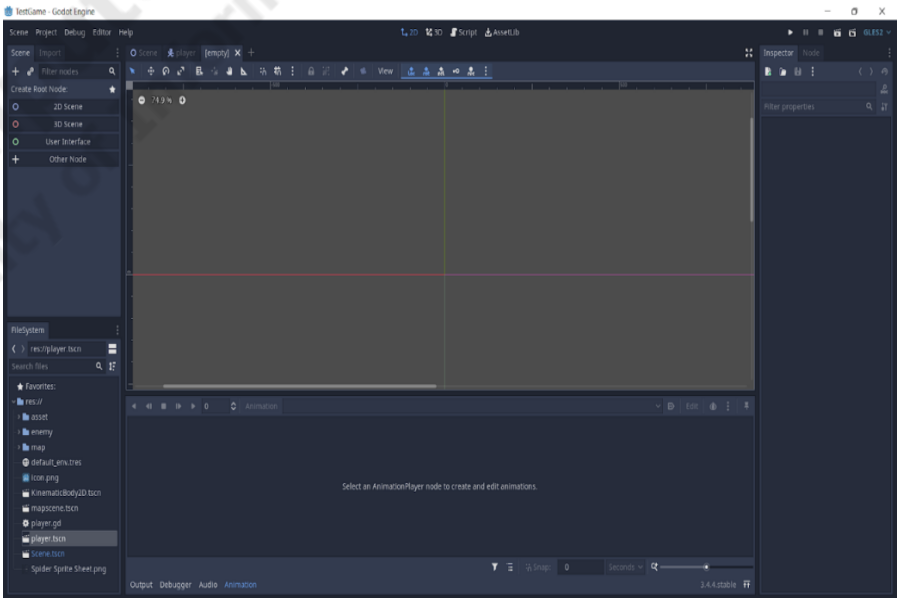
ภาพประกอบที่ 2.5 หน้าตาตัวโปรแกรม Godot

2.2.3 GDScript

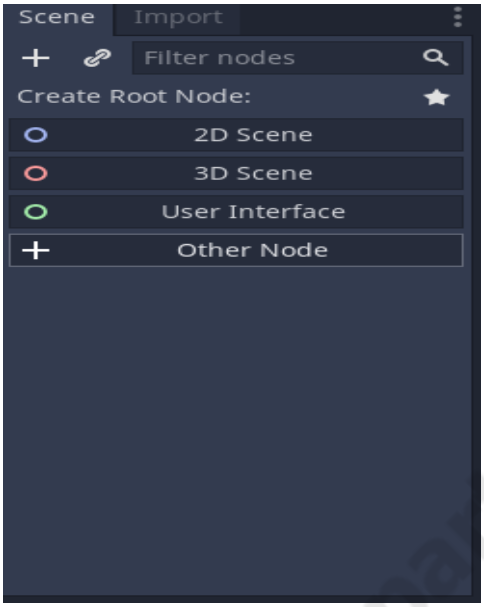
GDScript เป็นภาษาการเขียนโปรแกรมระดับสูงที่มีการเขียนแบบไดนามิก ใช้ไวยากรณ์ที่มีลักษณะคล้ายกับ Python โดยที่โค้ดถูกเยื้องแทนที่จะใช้วงเล็บและเซมิโคลอน ทำให้ง่ายต่อการเรียนรู้ โค้ดส่วนใหญ่สามารถเขียนและเปลี่ยนแปลงได้อย่างรวดเร็วและไม่ยุ่งยาก ทำให้เขียนโค้ดน้อย อ่านโค้ดได้ง่ายขึ้นและไม่จำเป็นต้องคอมไพล์เพื่อทดสอบ ดังตัวอย่าง

```
func _ready():
	print("Hello World")
```

2.2.4 การสร้างอนิเมชัน 2D ที่เกี่ยวข้อง

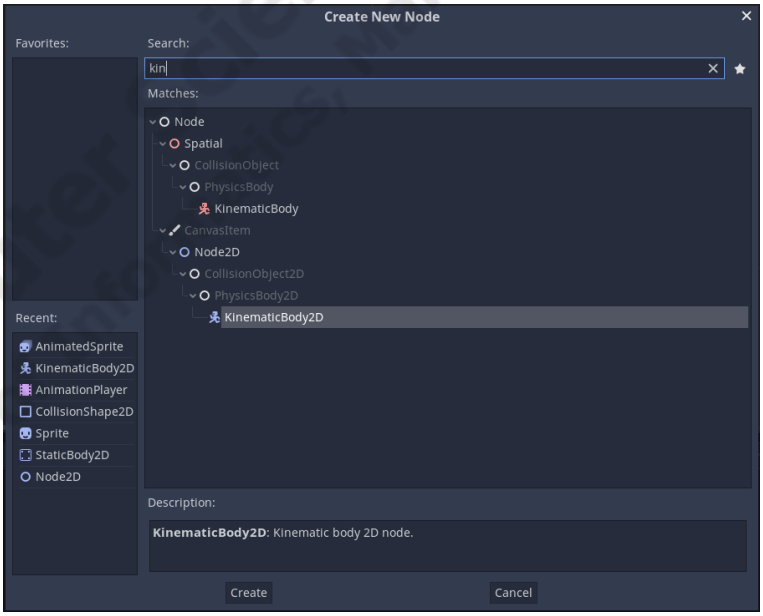


ภาพประกอบที่ 2.6 หน้า 2D



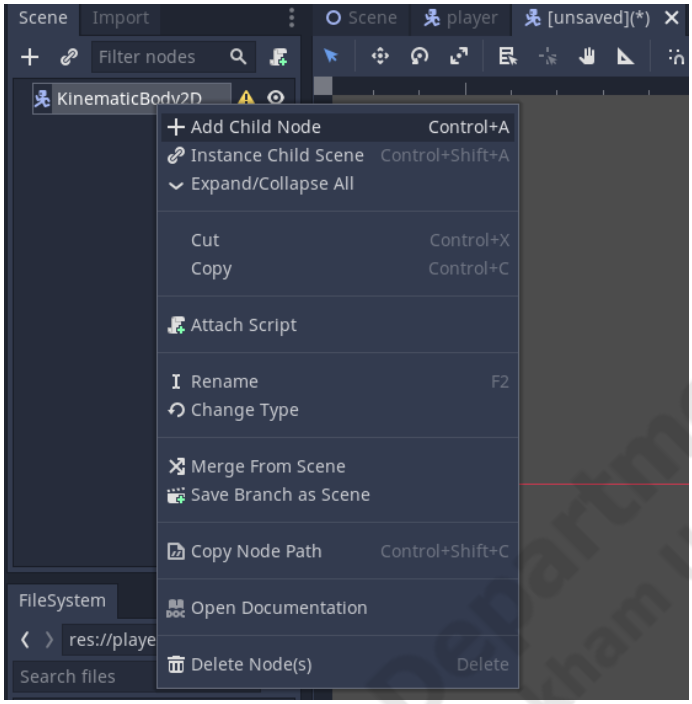
ภาพประกอบที่ 2.7 สร้างโหนดหลัก

ภายในโปรแกรม Godot จะมีโหนดชื่อว่า KinematicBody2D ใช้สร้างตัวละคร 2D ในเกม ทำการกดเพิ่มโหนดตามภาพด้านล่าง



ภาพประกอบที่ 2.8 สร้างโหนด KinematicBody2D

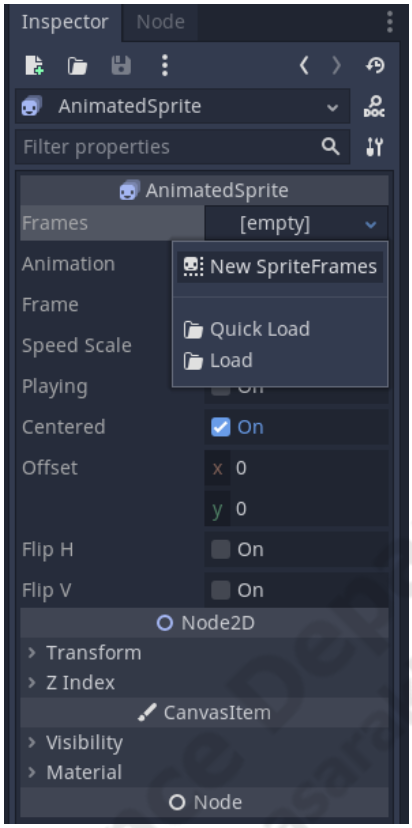
หลังจากที่สร้างโหนด KinematicBody2D แล้วให้ทำการ Add โหนด AnimatedSprite เพื่อใช้สร้างอนิเมชันภาพเคลื่อนไหวแบบ 2D



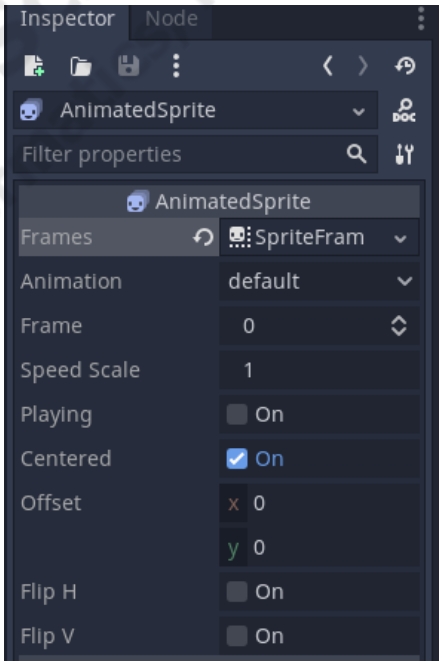
ภาพประกอบที่ 2.9 เพิ่มโหนดให้ KinematicBody2D

ภาพประกอบที่ 2.10 Add โหนด AnimatedSprite

หลังจากเพิ่มโหนด AnimatedSprite แล้วให้ทำการเลือกตรง Frames คลิกที่ New SpriteFrames แล้วทำการคลิกที่ Spriteframe อีกที



ภาพประกอบที่ 2.11 สร้าง New SpriteFrames



ภาพประกอบที่ 2.12 หลังจากสร้าง New SpriteFrames

หลังจากคลิกแล้วจะมีแท็บ SpriteFrames โผล่ขึ้นมาจากข้างล่าง

ภาพประกอบที่ 2.13 แท็บ SpriteFrames

คลิกที่ Add Frames from a Sprite Sheet ตัวที่ 2 ถัดจากรูปโฟลเดอร์

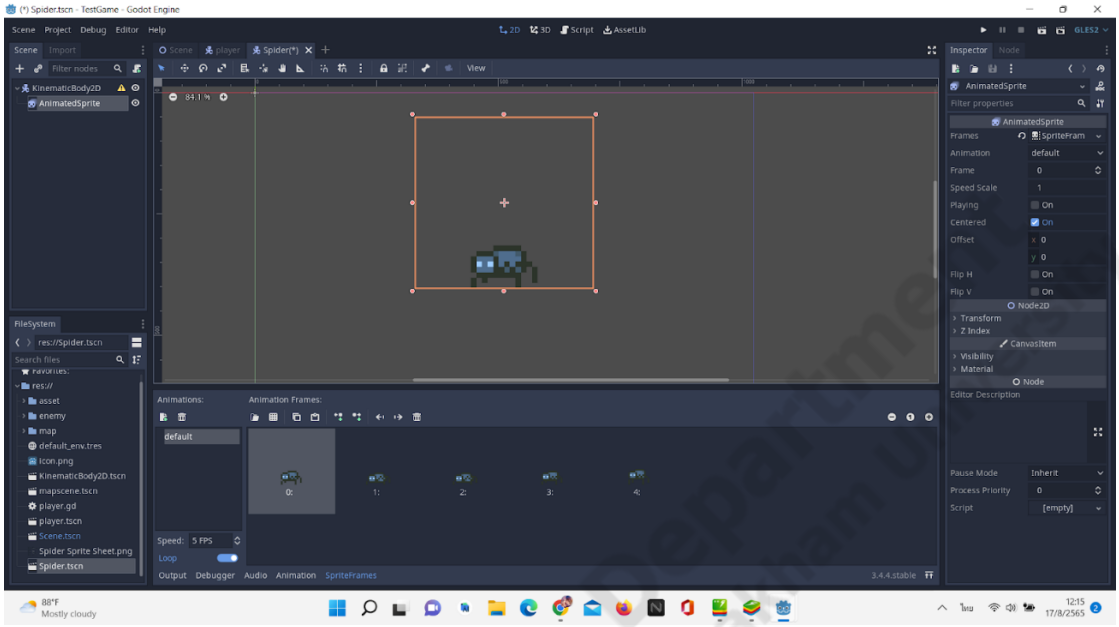
ภาพประกอบที่ 2.14 เลือกภาพตัวละคร 2D

ทำการเลือกรูปภาพตัวละคร 2D ที่ได้ทำการโหลดมาแล้วกด open

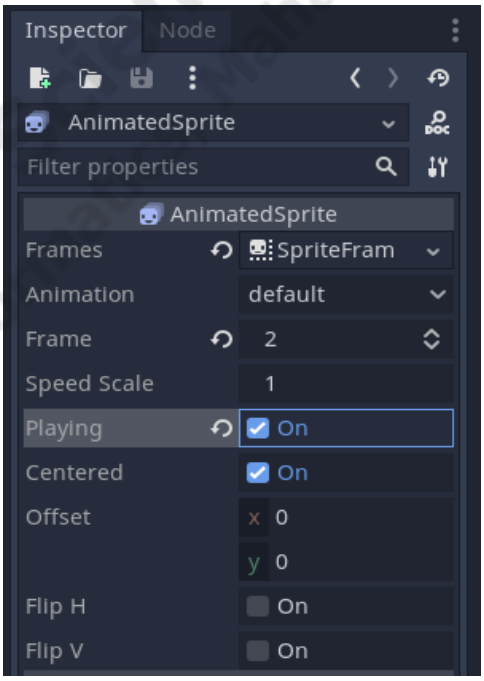
ภาพประกอบที่ 2.15 SpriteFrames

ทำการปรับ Horizontal และ Vertical ให้เหมาะสมกับภาพตัวละครของเรา แล้วใช้เมาร์คลิก

เลือกรูปภาพที่จะนำมาสร้างอนิเมชัน เมื่อเลือกได้แล้วทำการกดที่ Add Frame(s)

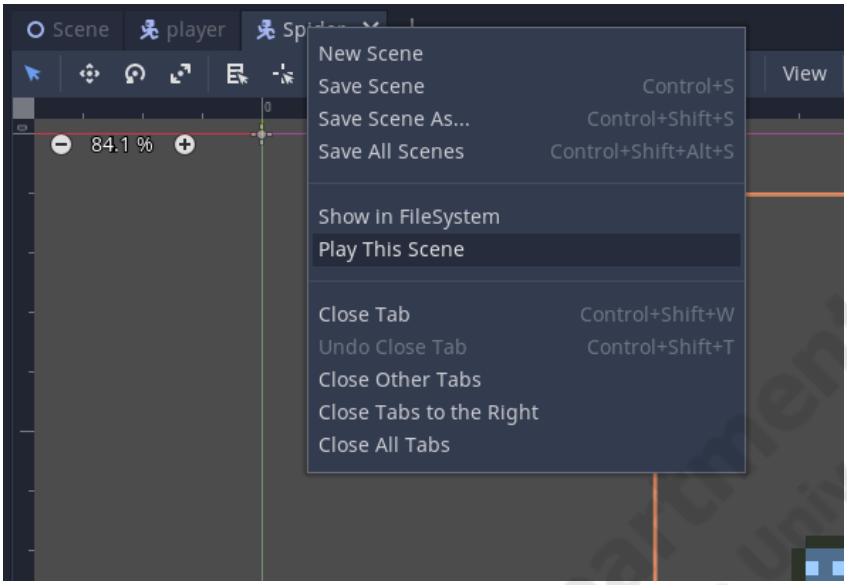


ภาพประกอบที่ 2.16 หลังจากกด Add Frame



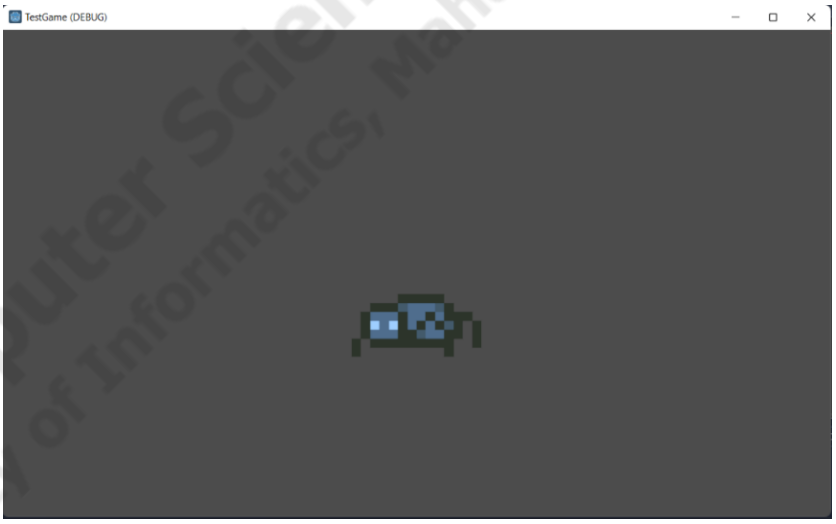
ภาพประกอบที่ 2.17 ทดสอบเล่นอนิเมชัน

ปรับขนาดตัวละครให้เหมาะสมกับการแสดงผลภายในเกมหลังจากนั้นกดที่ Playing เพื่อทดสอบการเล่นอนิเมชัน



ภาพประกอบที่ 2.18 ทดสอบรันเกม

ทำการทดสอบรันเกมเพื่อดูว่าตัวละครที่สร้างสามารถขยับได้ภายในเกมหรือไม่โดยกดคลิกขวาที่ scene ตัวละครที่เราสร้างแล้วกดที่ Play This Scene



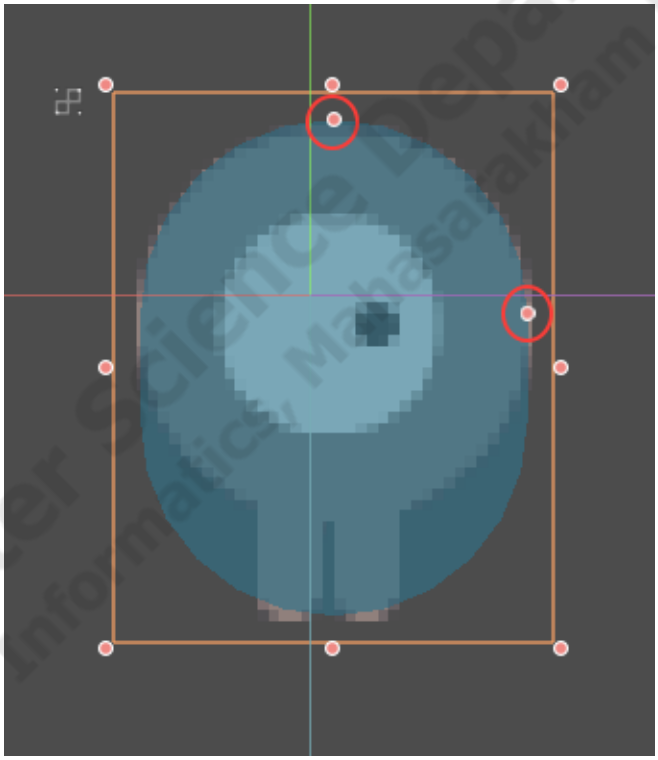
ภาพประกอบที่ 2.19 รันเกมดูการเคลื่อนไหวของตัวละคร

2.2.5 การตรวจจับการชนกันของวัตถุ (Collision Detection)

ในการพัฒนาเกม 2D หรือ 3D จะมีการตรวจจับการชนกันของวัตถุเพื่อที่จะกำหนดอีเว้นท์ที่จะเกิดขึ้นเมื่อวัตถุทั้ง 2 มีการเคลื่อนที่มาอยู่ในตำแหน่งที่ทับกัน เช่น กระสุนปืนที่เคลื่อนที่ไปโดนเป้าหมายจะทำดาเมจต่อเป้าหมาย เป็นต้น

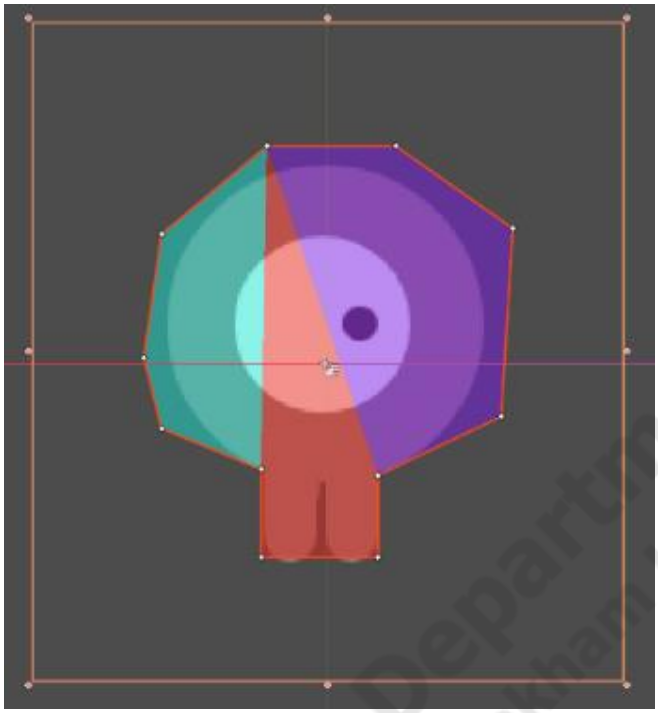
2.2.6 รูปร่างการชน (Collision Shapes)

ภายใน Godot จะมี รูปร่างการชน (Collision Shapes) ที่ใช้ในการตัวจับการชนกันของวัตถุ โดยที่สามารถวาดพื้นที่ ๆ ต้องการภายในวัตถุของเราได้เลย โดยพื้นที่ที่วาดขึ้นมาจะสามารกำหนดให้เป็นโหนดลูกของวัตถุนั้นเพื่อที่จะให้มันเคลื่อนที่ไปพร้อมกับวัตถุนั้นได้เมื่อเกิดอีเว้นท์ท์ต่าง ๆ รูปร่างของพื้นที่ตรวจจับการชนมีหลายรูปแบบ เช่น วงกลม วงรี สี่เหลี่ยม สามารถเลือกใช้ให้เหมาะสมกับงานได้ หรือหากต้องการความละเอียดในการจัดการรูปร่างให้มีรูปร่างหลายเหลี่ยม ใน Godot ก็มี Collision Polygon2D ให้สามารถกำหนดรูปร่างให้ละเอียดยิ่งขึ้น ตัวอย่างดังภาพประกอบที่ 2.20 และภาพประกอบที่ 2.21



ภาพประกอบที่ 2.20 CollisionShapes2D

ที่มา: https://docs.godotengine.org/en/3.3/_images/player_coll_shape1.png



ภาพประกอบที่ 2.21 CollisionPolygon2D

ที่มา: https://docs.godotengine.org/en/3.3/_images/player_coll_shape1.png

2.3 เกมที่คล้ายกัน

2.3.1 Vampire Survivors

ภาพประกอบที่ 2.22 หน้าจอหลักของเกม Vampire Survivors

ภาพประกอบที่ 2.23 หน้าเลือกตัวละครเกม Vampire Survivors

ตัวละครแต่ละตัวจะมีสกิลติดตัวและสเตตัสที่แตกต่างกัน ก่อนที่จะได้ตัวละครมาเล่นผู้เล่นต้องทำตามเงื่อนไขของเกมก่อน เช่น ต้องเล่นเกมเอาชีวิตในด่านให้ครบ 30 นาที หรือต้องชนะ Death ที่จะปรากฏตัวออกมาหลังจากผ่านไป 30 นาทีหากชนะจะได้ Death มาเล่น เมื่อทำตามเงื่อนไขแล้วตัวละครถึงจะมีมาให้ซื้อในหน้าเลือกตัวละคร

ภาพประกอบที่ 2.24 หน้าเลือกด่านเกม Vampire Survivors

ด่านแต่ละด่านจะมีให้ปรับระดับความยาก เมื่อปรับแล้วจะส่งผลกับเงินที่เราจะได้รับเมื่อเล่นจบเกมและมีให้ตัวเลือกที่ทำให้เกมที่เล่นเร็วขึ้นคือตัวละครจะเคลื่อนที่เร็วขึ้นและมอนสเตอร์ก็จะเคลื่อนที่เร็วขึ้นด้วย ด่านแต่ละด่านแตกต่างกันที่มอนสเตอร์และความสามารถของมอนสเตอร์

ภาพประกอบที่ 2.25 เกมเพลย์ภายในด่านเกม Vampire Survivors

ภายในด่านมอนสเตอร์จะเกิดขึ้นมาเรื่อย ๆ แล้วเข้ามาโจมตีผู้เล่นยิ่งเวลาเพิ่มขึ้นมากเท่าไหร่ มอนสเตอร์ก็จะเก่งขึ้นตามไปด้วย และเมื่อเราฆ่ามอนสเตอร์จะดรอปหิน EXP มาเพิ่มเลเวลตัวละครของเราเมื่อเลเวลอัพจะสุ่มดรอปสกิลให้เราเลือก 3 สกิล ทำซ้ำไปเรื่อย ๆ เราก็จะมีสกิลหลาย ๆ สกิลมาใช้เล่นสู้กับมอนสเตอร์ที่จะเก่งขึ้นตามเวลาของด่านนั้น โดยสกิลที่มีมากมายในเกมนั้นแต่ละรอบของด่านเราจะเลือกสกิลได้เพียง 12 สกิลเท่านั้นการเลือกสกิลซ้ำจะเป็นการอัพเลเวลให้สกิลนั้นให้มีประสิทธิภาพมากขึ้น และนอกจากการเก็บหิน EXP เพื่อให้เลเวลอัพแล้วเราสามารถฆ่ามินิบอสที่จะปรากฏตัวออกมาตามเวลาต่าง ๆ ของเกมเพื่อดรอปหีบสมบัติมาสุ่มดรอปสกิล 3 สกิลได้เหมือนกัน

ภาพประกอบที่ 2.26 ร้านค้าภายในเกม Vampire Survivors

เงินที่ได้จากการเล่นเกมในแต่ละด่านสามารถเอามาอัพสกิลติดตัวให้ตัวละครในร้านค้าเพื่อจะทำให้การเล่นครั้งต่อไปง่ายขึ้น

2.3.2 Holocure: Save the Fans!

โดยตัวเกมจะเป็นแนว Action Roguelike ที่ได้หยิบเหล่า Member จากโลก Hololive มาใช้สำหรับการต่อสู้ ซึ่งตัวเกมนั้นจะมีการปล่อยศัตรูออกมาแบบไม่หยุดพัก และทุกครั้งที่จบรอบจะได้รับความสามารถแตกต่างกันออกไป ซึ่งสามารถนำไปสร้าง Build สกิลของตัวเองได้ตามจินตนาการณ์ นอกจากนี้ยังสามารถนำเงินไปซื้อเพื่อ Upgrade ความสามารถต่าง ๆ ให้กับตัวเองได้อีกด้วย ตัวเกมนั้นได้รับแรงบันดาลใจมาจากเกมอินดี้ชื่อดังอย่าง Vampire Survivors โดยสามารถดาวน์โหลดได้ในลิ้งค์นี้ home page at itch.io.

คุณสมบัติ:

- ปลดล็อกและเล่นกับตัวละคร Hololive 20 ตัวผ่าน Gacha (สกุลเงินในเกมเท่านั้น)
- ทักษะเฉพาะตัวและการโจมตีพิเศษสำหรับตัวละครทุกตัว
- ปลดล็อกอาวุธและไอเทมแบบพาสซีฟมากมาย
- รีมิกซ์เพลง Hololive ที่สร้างขึ้นสำหรับเกมนี้โดย Eufrik
- เอาชีวิตรอดจากตัวละคร Hololive และมาสคอตสุดคลาสสิก

ภาพประกอบที่ 2.27 หน้าหลักของ Holocure: Save the Fans!

ภาพประกอบที่ 2.28 ภายในเกม Holocure: Save the Fans!

ภาพประกอบที่ 2.29 เลือกตัวละคร Holocure: Save the Fans!

ภาพประกอบที่ 2.30 ซื้อสกิลเสริม Holocure: Save the Fans!

ที่มา: https://kay-yu.itch.io/holocure

ตารางที่ 2.1 ตารางการเปรียบเทียบการฟังก์ชันทำงานของระบบ

ฟังก์ชันการทำงานของระบบ	Vampire Survivors	Holocure: Save the Fans!	Meme Survivors: Save the World!
ควบคุมการเคลื่อนที่ด้วยคีย์บอร์ด	/	/	/
การซื้อตัวละคร	/	x	x
ระบบสุ่มกาชาตัวละคร	x	/	x
อัพเกรดความสามารถโดยรวม	/	/	x
อัพเกรดความสามารถเฉพาะตัวละคร	x	/	/

ตารางที่ 2.1 ตารางการเปรียบเทียบการฟังก์ชันทำงานของระบบ (ต่อ)

ฟังก์ชันการทำงานของระบบ	Vampire Survivors	Holocure: Save the Fans!	Meme Survivors: Save the World!
การโจมตีพิเศษเฉพาะตัวละคร	x	/	/
ระบบรีฟันเงินคืน	/	/	/
ระบบผสมไอเทมและอาวุธ	/	/	x
โหมดเล่นจำกัดเวลา	/	/	/
โหมดเล่นไม่จำกัดเวลา	x	/	/
ระบบความสำเร็จ	/	x	x
เก็บคะแนนสูงสุดเพื่อโชว์	x	/	x