**บทที่ 4**

**การดำเนินการพัฒนาและการทดสอบระบบ**

ในส่วนของบทนี้คือการดำเนินการพัฒนาระบบเกมผจญภัยสัตว์ป่าตามกระบวนการที่ได้ศึกษาและการออกแบบในบทที่ผ่านมา โดยทำการพัฒนาและทดสอบว่าผลลัพธ์ที่ได้จะตรงตามที่ออกแบบไว้ ซึ่งมีรายละเอียดทั้งหมดดังนี้

4.1 ผลการดำเนินการพัฒนาตัวละครและองค์ประกอบภายในเกม

4.2 การพัฒนาเกม

4.3 การทดสอบระบบ

4.4 ผลการประเมินความพึงพอใจของผู้ใช้งาน

**4.1 ผลการดำเนินการพัฒนาตัวละครและองค์ประกอบภายในเกม**

ในขั้นตอนนี้เป็นการดำเนินการพัฒนาตัวละครและองค์ประกอบต่าง ๆ ในเกมตามที่ได้ออกแบบไว้ ดังนี้

**4.1.1 หน้าจอเมนู**

****

**ภาพที่ 4.1** หน้าจอเมนูหลัก

2) หน้าจอเลือกสัตว์ ประกอบไปด้วย

- รูปสัตว์ ทั้ง6 ชนิด

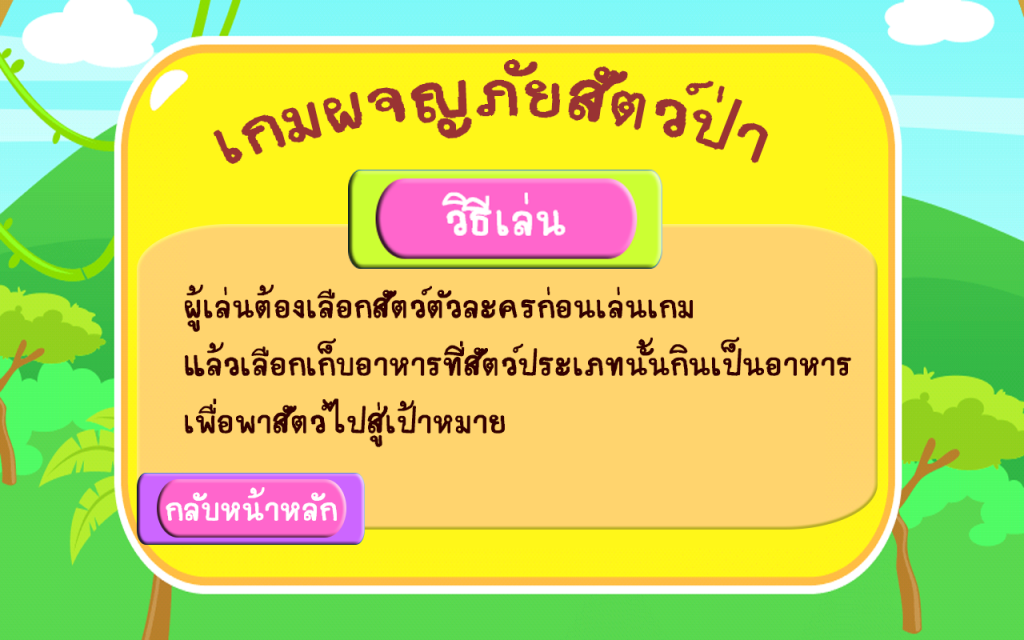
- สัตว์กินพืชเป็นอาหาร 3 ชนิด

- สัตว์กินเนื้อเป็นอาหาร 3 ชนิด



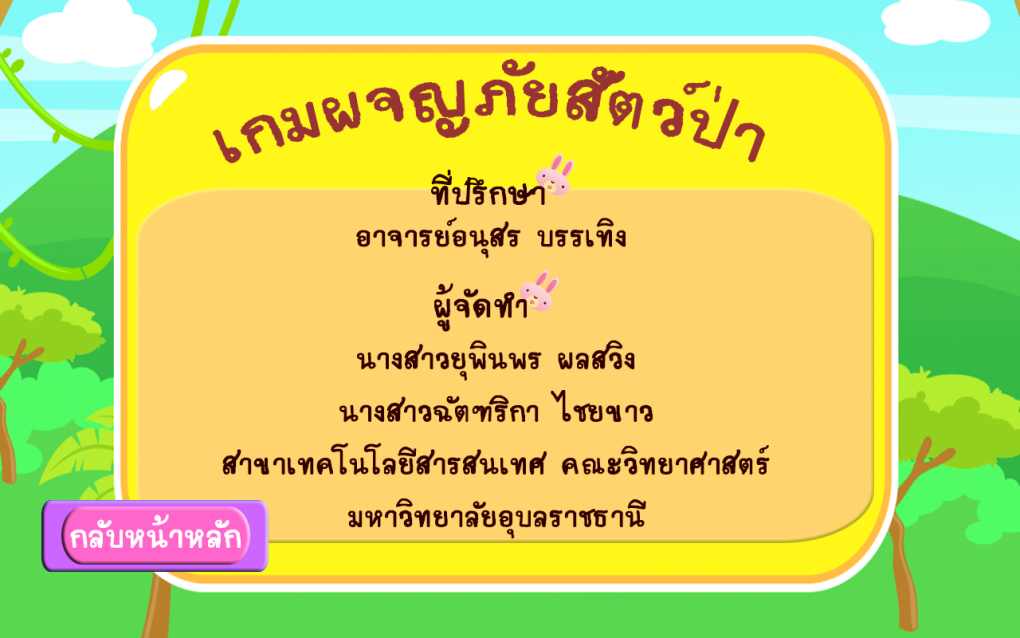
**ภาพที่ 4.2** หน้าจอเลือกสัตว์ที่จะเล่น

4) หน้าจอวิธีเล่น



ภาพที่ 4.3 หน้าจอเมนูวิธีเล่น

5) หน้าจอเมนูผู้จัดทำ ประกอบไปด้วยข้อมูลผู้ทำ ดังภาพที่ 4.4

****

**ภาพที่ 4.4** หน้าจอเมนูผู้จัดทำ

**4.1.2 ตัวละคร**

ในส่วนนี้เป็นการทำตัวละครต่าง ๆ ตามที่ได้ออกแบบไว้ในบทที่ 3 เพื่อใช้สำหรับการพัฒนาการบังคับทิศทางของตัวละครในเกม ดังตารางที่ 4.5

**ตารางที่** 4.5 ท่าทางของตัวละคร

|  |  |
| --- | --- |
| รายละเอียดตัวละคร | ภาพตัวละคร |
| 1.กระต่าย  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินพืชเป็นอาหาร เช่น  แครอท สามารถวิ่งเร็วได้ และอาศัยอยู่ในป่าอุดมสมบรูณ์ | **D:\IT\4\term2\Project_animal\วาดAI\animal\labbit.png** |
| 2.สิงโต  เป้าหมาย: การเก็บไอเท็มที่เป็นให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่เป็นนักล่า กินเนื้อสัตว์อื่นเป็นอาหาร เช่น กระต่าย ไก่ป่า กวาง ม้าลาย เป็นต้น และสามารถเดิน กระโดด ได้ ชอบอาศัยอยู่ในป่าเขตร้อน | D:\IT\4\term2\Project_animal\วาดAI\animal\Lion-12.png |
| 3. กวาง  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินพืช ทั้งใบและยอด อาศัยอยู่ทุ่งโล่ง ชายป่า และสามารถเดิน กระโดด ได้ ในป่าอุดมสมบรูณ์ | **D:\IT\4\term2\Project_animal\วาดAI\กวาง.png** |
| 4.ช้าง  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินพืชเป็นอาหาร เช่น อ้อย ต้นกล้วย และสามารถเดิน เหยียบศัตรูได้ อาศัยอยู่ในป่าอุดมสมบรูณ์ | **D:\IT\4\term2\Project_animal\วาดAI\elephant.png** |

**ตารางที่** 4.5 ท่าทางของตัวละคร (ต่อ)

|  |  |
| --- | --- |
| รายละเอียดตัวละคร | ภาพตัวละคร |
| 5.หมาป่า  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินเลือด เนื้อของสิ่งมีชีวิตทุกประเภทและสามารถเดิน กระโดด ได้ อาศัยอยู่ในป่าดิบชื้น | **D:\IT\4\term2\Project_animal\วาดAI\Fox1.png** |
| 6.เสือ  เป้าหมาย: การเก็บไอเท็มทีให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินเนื้อเป็นอาหาร เช่น กวาง หมูป่า และสามารถเดิน กระโดด ได้ อาศัยอยู่ในป่าเขตร้อน | **D:\IT\4\term2\Project_animal\วาดAI\Tiger11.png** |

**4.1.3 ฉากด่านต่างๆ**

ในส่วนนี้ผู้พัฒนาได้นำฉาก ตัวละคร และสิ่งกีดขวางต่าง ๆ มาจัดวางองค์ประกอบตามที่ได้ออกแบบไว้ โดยในแต่ละด่านจะมีฉากที่แตกต่างกัน และสิ่งกีดขวางเพิ่มตามที่ได้ออกแบบไว้ ดังตารางที่ 4.6

**ตารางที่** 4.6 หน้าจอด่านต่าง ๆ

|  |  |
| --- | --- |
| รายละเอียดแต่ละด่าน | ภาพด่านในเกม |
| 1.ด่านที่ 1 : ป่าที่มีความอุดมสมบรูณ์  ภายในด่านประกอบด้วยไอเท็มที่จะช่วยในการผ่านด่าน 1ชนิดและมีอุปสรรคคือพื้นต่างระดับ |  |

**ตารางที่** 4.6 หน้าจอด่านต่าง ๆ (ต่อ)

|  |  |
| --- | --- |
| รายละเอียดแต่ละด่าน | ภาพด่านในเกม |
| 2.ด่านที่ 2 : เป็นฉากที่มีสภาพอากาศแบบเขตร้อน ภายในด่านประกอบด้วยไอเท็มช่วยให้ผ่านด่าน 3 ชนิดที่และมีไอเท็มเพิ่มลดชีวิต อุปสรรคคือพื้นต่างระดับ |  |
| 3. ด่านที่ 3 : เป็นฉากที่มีสภาพอากาศหนาว ภายในด่านประกอบด้วยไอเท็มที่ช่วยให้ผ่านด่าน 4 ชนิดและมีไอเท็มเพิ่มลดชีวิต อุปสรรคคือพื้นต่างระดับและมีสัตว์ชนิดอื่นมาทำร้าย |  |
| 4. ด่านที่ 4 : เป็นฉากที่มีสภาพอากาศแบบฝนตก ภายในด่านประกอบด้วยไอเท็มที่ช่วยให้ผ่านด่าน 5 ชนิดและมีไอเท็มเพิ่มลดชีวิต อุปสรรคคือพื้นต่างระดับ มีสัตว์อื่นมาทำร้ายและมีกิ่งไม้หล่นลงมา หากกิ่งไม้หล่นลงมาโดนตัวจะทำให้ชีวิตลดลงเร็วขึ้น |  |

**ตารางที่** 4.6 หน้าจอด่านต่าง ๆ (ต่อ)

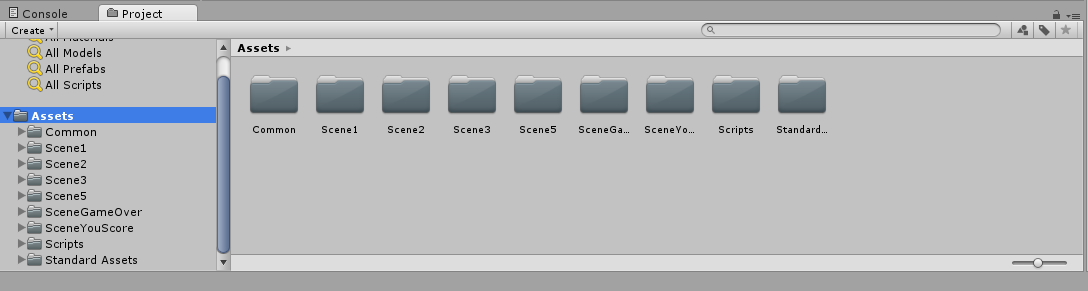
|  |  |
| --- | --- |
| รายละเอียดแต่ละด่าน | ภาพด่านในเกม |
| 5. ด่านที่ 5 : เป็นฉากจะเปลี่ยนไปตามป่าที่สัตว์เหล่านั้นอาศัยอยู่ ซึ่งมีเวลาจำกัดในการเก็บคะแนน เพื่อทำคะแนนให้ได้สูงสุด จะมีฉากต่างๆ ดังนี้  • ฉากป่าที่มีความอุดมสมบูรณ์ สัตว์ที่อาศัยอยู่ ได้แก่ กระต่าย กวาง ช้าง  • ฉากป่าเขตร้อน สัตว์ที่อาศัยอยู่ ได้แก่ สิงโต เสือ  • ฉากป่าดิบชื้น สัตว์ที่อาศัยอยู่ ได้แก่ หมาป่า | D:\IT\4\term2\Project_animal\picture-S5\Scene5.png |

**4.2 การพัฒนาเกม**

ในหัวข้อนี้จะเป็นขั้นตอนหลักในการพัฒนาเกมในทุกด่านซึ่งจะใช้วิธีการพัฒนาแบบเดียวกัน โดยแบ่งเป็น 6 ขั้นตอนดังนี้

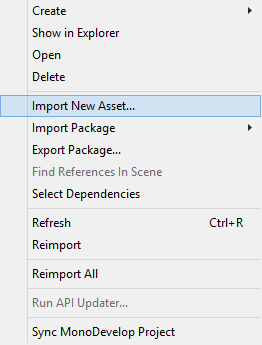
**4.2.1 การนำเข้าข้อมูลในการพัฒนาเกม**

ในส่วนนี้จะเป็นการนำเข้าข้อมูลรูปภาพ ที่ต้องใช้เข้ามาใส่ไว้ใน Project Assets ลงในโฟลเดอร์ที่ได้สร้างขึ้นไว้ ดังภาพที่ 4.1



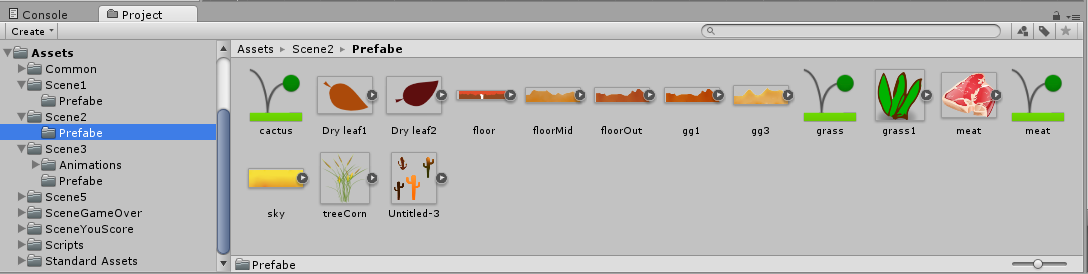
**ภาพที่ 4.1** โฟลเดอร์สำหรับจัดเก็บข้อมูลต่าง ๆ ใน Project Panel

1. คลิกขวา เลือก Import New Asset.. ดังภาพที่ 4.2



**ภาพที่ 4.2** การนำเข้าข้อมูล

1. เลือกโฟลเดอร์ที่ต้องเก็บไฟล์ภาพที่จะใช้ทำฉากในแต่ละด่าน ดังภาพที่ 4.3



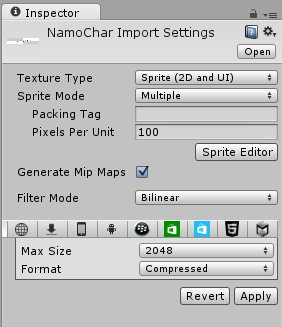
**ภาพที่ 4.3** โฟลเดอร์สำหรับจัดเก็บภาพในแต่ละด่าน

1. เลือกโฟลเดอร์ที่ต้องเก็บไฟล์ภาพหรือตัวละครต่างๆ ดังภาพที่ 4.4



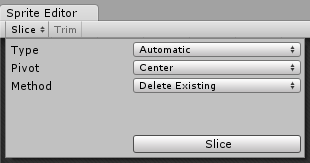
**ภาพที่ 4.4** โฟลเดอร์สำหรับจัดเก็บสัตว์หรือตัวละครต่างๆ

1. ตั้งค่าที่ Inspector Panel ดังภาพ แล้วกด Apply ดังภาพที่ 4.5



**ภาพที่ 4.5** การตั้งค่าภาพที่ Inspector Panel

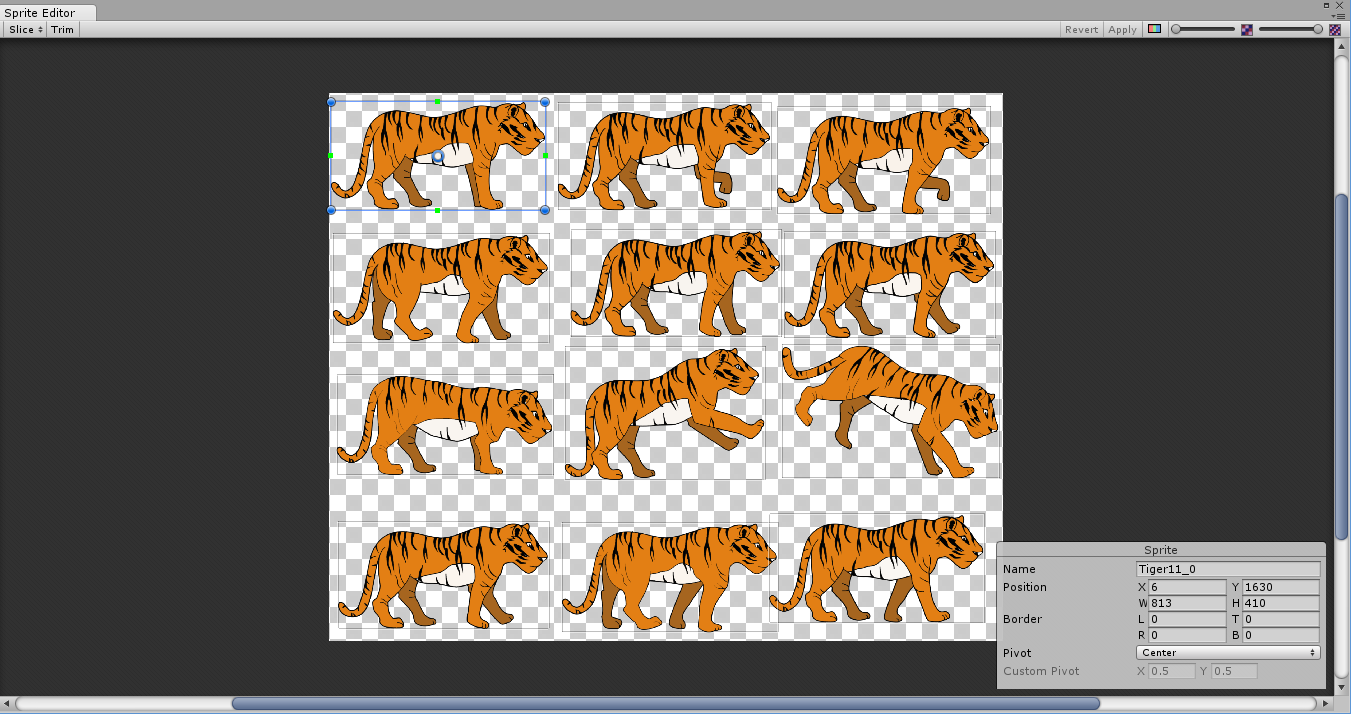
1. คลิกที่ปุ่ม Sprite Editor ใน Inspector Panel แล้วคลิกที่คำว่า Slice ดังภาพที่ 4.6



**ภาพที่ 4.6** การตั้งค่า Sprite Editor

1. คลิกที่ปุ่ม Apply โปรแกรมก็จะทำการตัดแบ่งส่วนของภาพให้อยู่ในรูปแบบของ Sprite

ดังภาพที่ 4.7

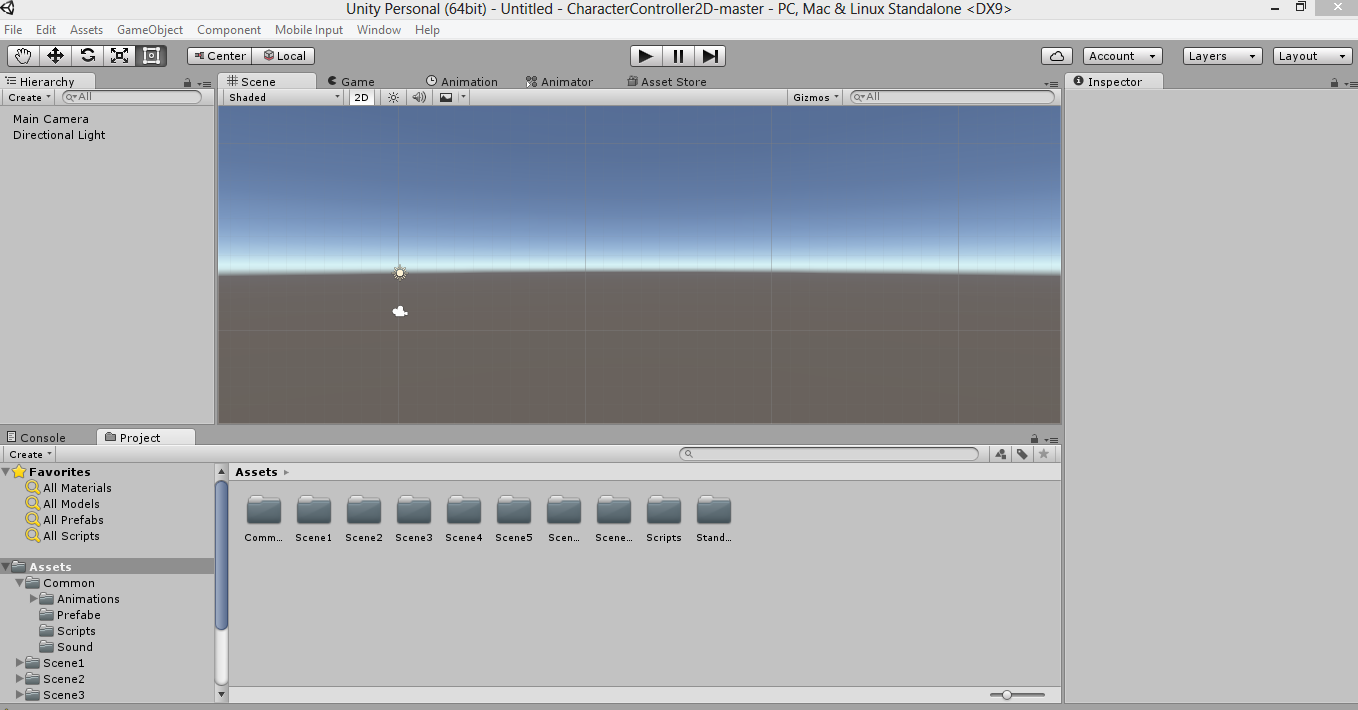


**ภาพที่ 4.7** การแบ่งภาพเป็น Sprite

**4.2.2 เมนูหลัก**

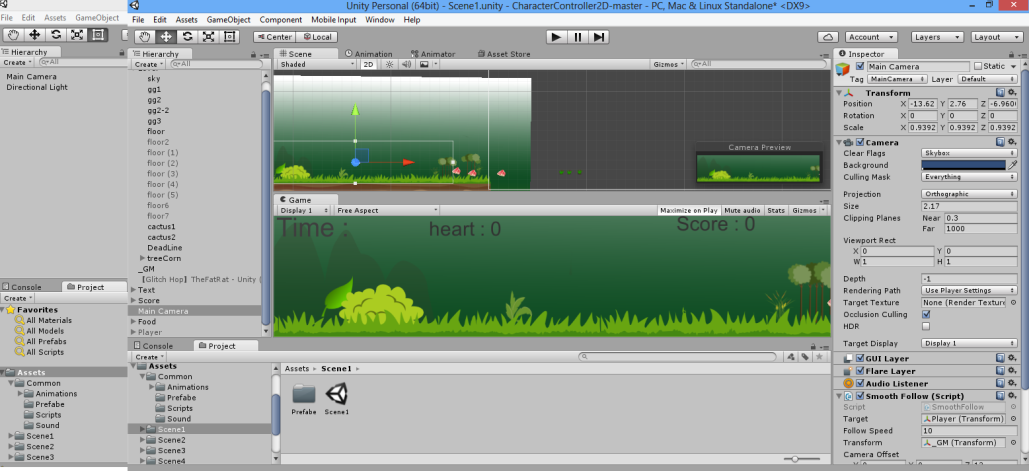
ในส่วนนี้เป็นการพัฒนาหน้าจอเมนูหลักของเกม ซึ่งเมนูหลักประกอบด้วย เมนูจัดการผู้เล่น เมนูวิธีการเล่นเกม เมนูผู้จัดทำ เมนูเลือกด่าน เมนูออกจากเกม

1. เมื่อเปิดโปรแกรมขึ้นมาครั้งแรก ก็จะมีหน้าสีน้ำเงินว่างเปล่า หากต้องการบันทึกให้กดปุ่ม Ctrl และ S หรือกดที่เมนู File แล้วเลือก Save เพื่อทำการบันทึก โดยตั้งชื่อว่า CaracterController2D หรือชื่อที่ต้องการ ดังภาพที่ 4.8



**ภาพที่ 4.8** หน้าจอ CaracterController2D

1. นำรูปภาพที่ import เข้ามาใน project มาแล้วจัดวางองค์ประกอบไว้ใน Scene Panel ดังภาพที่ 4.9

****

**ภาพที่ 4.9** การจัดวางองค์ประกอบของหน้า CaracterController2D ฉากที่1ของเกม

3) ตั้งค่า Main Camera ในส่วนของ Component ที่ Inspector Panel และ คลิกที่ Add Component ใส่source code ScaleWidthCamera.cs เข้าไป เพื่อกำหนดขนาดหน้าจอให้รองรับขนาดหน้าจอโทรศัพท์ที่ต่างกัน ดังภาพที่ 4.10



**ภาพที่ 4.10** การกำหนดค่าของ Component ที่ใช้งาน

|  |
| --- |
| 1. using UnityEngine; 2. [ExecuteInEditMode] 3. public class ScaleWidthCamera : MonoBehaviour 4. { 5. public int targetWidthSize = 480; //ขนาดความกว้างของหน้าจอ 6. public float pixelsToUnits = 100; //อัตราส่วนต่อหน่วย 7. void Update() 8. { 9. //กำหนดค่าตัวแปรคำนวณขนาดความกว้างของหน้าจอโทรศัพท์ 10. int height = Mathf.RoundToInt(targetWidthSize / (float)Screen.width \* Screen.height); 11. //กำหนดค่าตัวแปรคำนวณขนาดจอกับขนาดกล้อง 12. GetComponent<Camera>().orthographicSize = height / pixelsToUnits / 2; 13. } 14. } |

**ภาพที่ 4.11** ตัวอย่างsource code ScaleWidthCamera.cs

จากภาพที่ 4.11 บรรทัดที่ 1 เป็นการเรียกใช้ UnityEngine ซึงเป็น Library เฉพาะของโปรแกรม Unity โดยการเรียกใช้ Library พื้นฐานจะมีมาให้เมื่อมีการสร้างไฟล์ C# สคริปต์

บรรทัดที่ 3 เป็นการสร้าง Class โดยมีการสืบทอดกับ MonoBehaviour ซึ่งเป็นคลาสของ Unity เพื่อเรียกใช้ Method ต่าง ๆ เช่น Start, Awake, Update, FixedUpdate เป็นต้น

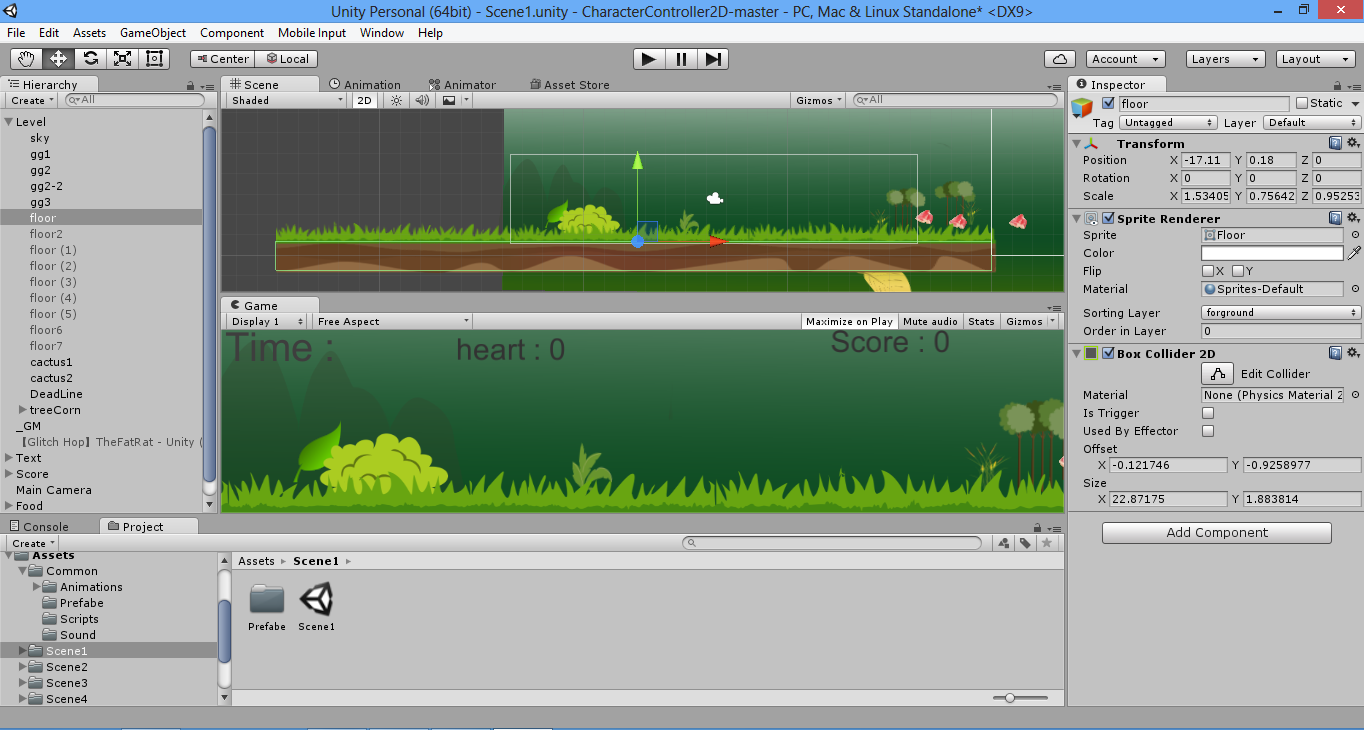
บรรทัดที่ 5 และ 6 เป็นการสร้าง Attribute สำหรับกำหนดขนาดหน้าจอ และอัตราส่วนต่อขนาดหน้าจอ

บรรทัดที่ 7 เป็นการสร้าง Method Update ซึ่งเป็น Method ที่สืบทอดมาจากคลาส MonoBehaviour โดย Method Update นี้จะทำงานแบบ Realtime เมื่อเกมทำงานใช้สำหรับการตรวจสอบหรือกำหนดค่าที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา โดยในที่นี้เป็นการเรียกใช้ Camera Component เพื่อกำหนดค่าขนาดหน้าจอของ Camera ให้เท่ากับหน้าจอของโทรศัพท์

**4.2.3 การพัฒนาการเคลื่อนไหวของตัวละครในเกม**

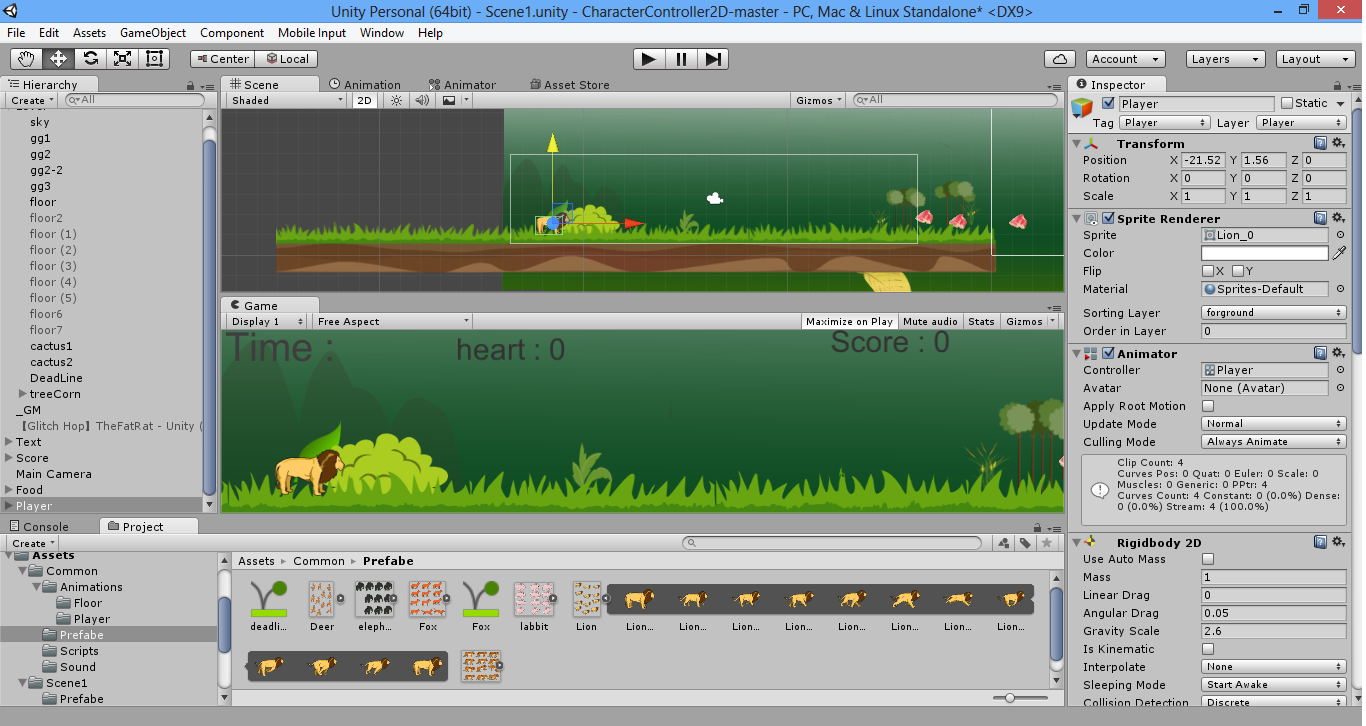
ในหัวข้อนี้จะเป็นการนำรูปภาพตัวละครที่เป็นท่าทางต่าง ๆ ที่ได้ทำไว้มาพัฒนาเป็นแอนนิเมชันสำหรับนำไปใช้ในเกม โดยมีวิธีดังนี้

1. ใส่องค์ประกอบต่าง ๆ ในหน้าจอด่านต่าง ๆ และใส่ Component ชื่อ Box Collider 2D สำหรับตรวจสอบการชนกันของ Game Object และเป็นพื้นที่สำหรับการเคลื่อนที่ของตัวละคร ที่ Game Object ที่มีชื่อว่า floor ซึ่งเป็นส่วนที่เป็นพื้นดินในฉาก โดยตั้งค่าที่ Inspector Panel ดังภาพที่ 4.12



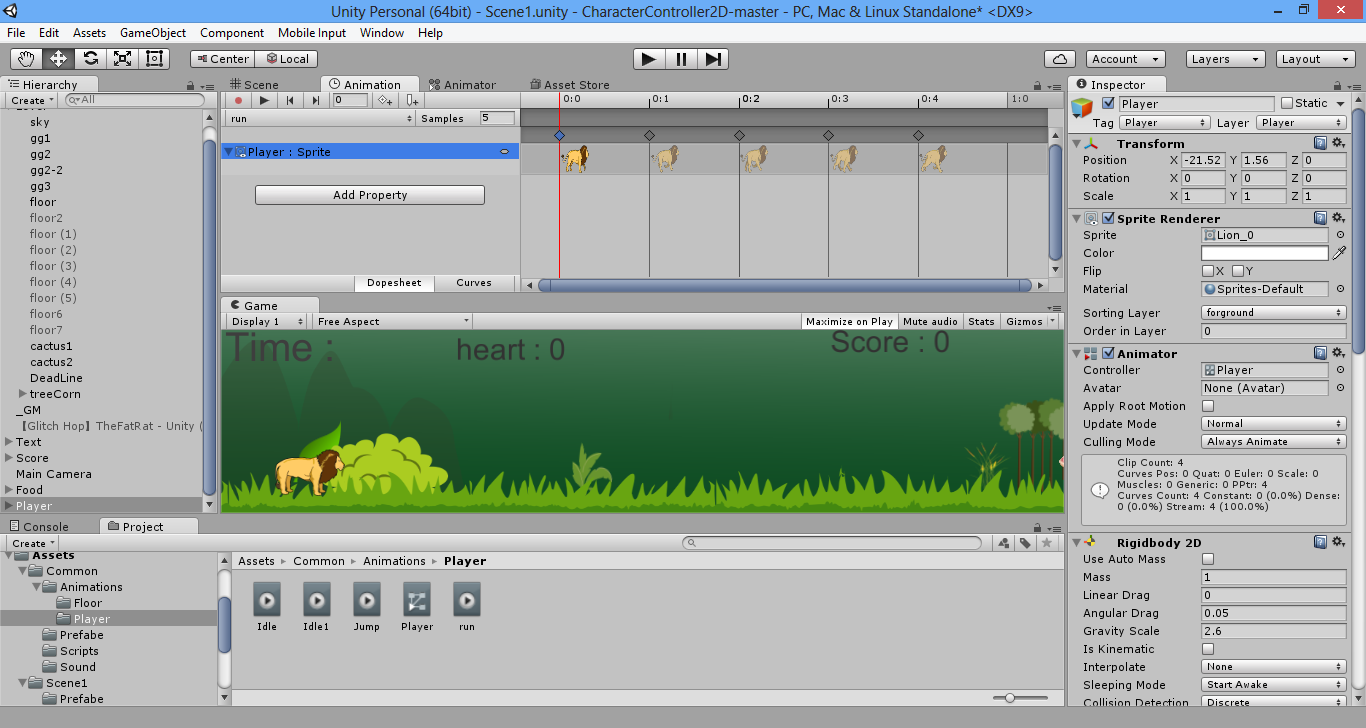
**ภาพที่ 4.12** การใส่ Box Collider 2D

1. นำรูปภาพที่ Import เข้ามาแล้วใน Project โดยนำ Sprite ที่เป็นท่าทางต่าง ๆ รูปแรกเข้ามาแทรกลงที่ Hierarchy Panel แล้วคลิกที่ Sprite แล้วเพิ่ม Component ชื่อ Animator ที่ Inspector ของ Sprite ตัวละคร ดังภาพที่ 4.13



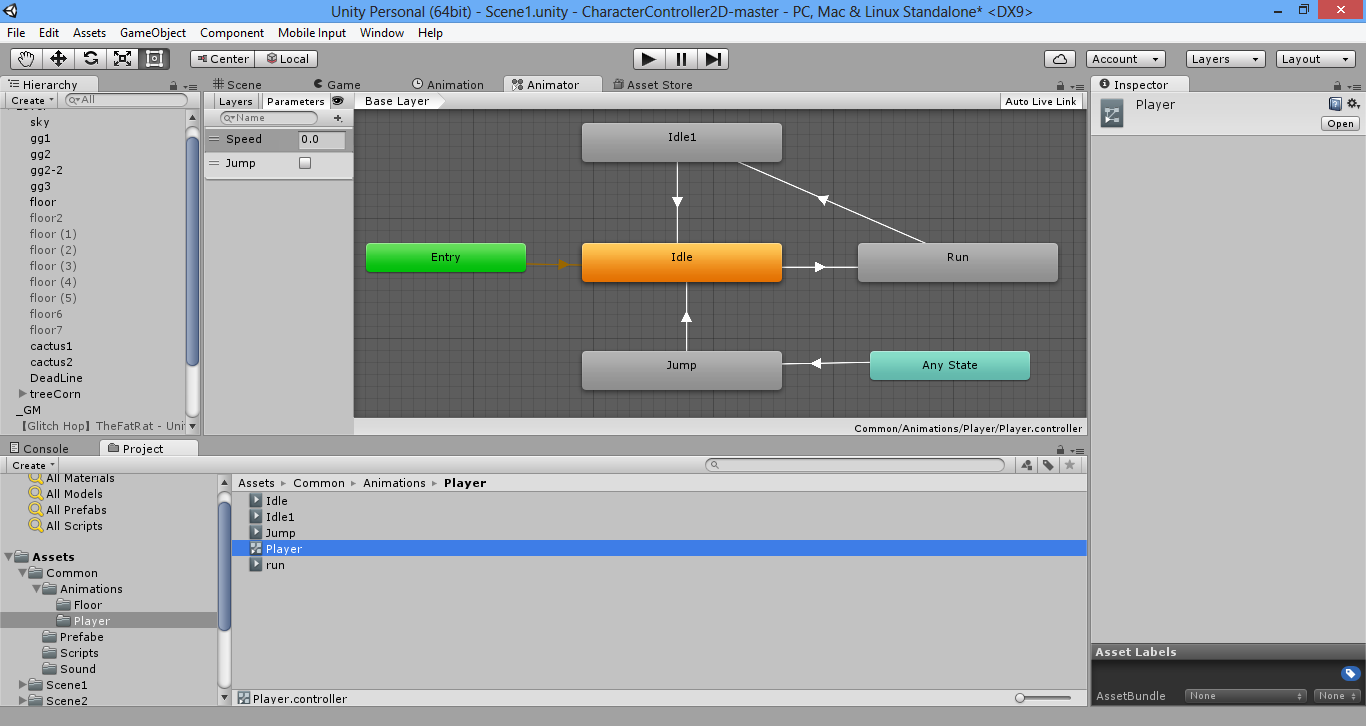
**ภาพที่ 4.13** การทำแอนนิเมชันด้วย Sprite

1. เข้าไปที่ Animation Panel คลิก Create New Clip..ตั้งชื่อว่า Run แล้วลาก Sprite ที่เป็นท่าทางนั้นทั้งหมดลงที่ Key Frame ของ Animation Panel แล้วตั้งค่า Samples ให้เท่ากับ 5 โดยทุกท่าทางการเคลื่อนไหวของตัวละครจะใช้วิธีการทำแบบเดียวกันกับวิธีที่ 1 และ 2 ดังภาพที่ 4.14



**ภาพที่ 4.14** การนำ Sprite วางบน Key Frame

1. เข้าไปที่ Animator Component แล้วสร้าง Parameters ตามชื่อของแต่ละ State และสร้าง Transition ให้แต่ละ State เชื่อมต่อกับ Idle State ซึ่งเป็น Default State โดยแต่ละเส้นของ Transition จะต้องใส่ Parameters ที่สร้างไว้ตามชื่อของ State พร้อมทั้งใส่ Component Script โดยใส่ AnimationControl.cs เข้าไปที่ Game Object ชื่อ Player ดังภาพที่ 4.15



**ภาพที่ 4.15** การจัดการ Animator State

|  |
| --- |
| 1. using UnityEngine; 2. using System.Collections; 3. public class AnimationControl : MonoBehaviour { 4. public PlayerMovement PlayerMv; //เรียกใช้คลาส PlayerMovement 5. public static int MoveNum; //ค่ารวมของการเดิน 6. public int LR; //ค่าการเดินซ้ายขวา 7. public int IdleAttBt; //ค่าการโจมตี 8. public int WalkAttbt; //ค่าการเดินโจมตี 9. private Animator anim; //เรียกใช้ Component Animator 10. //ค่าเริ่มต้นในการเรียกใช้ Component 11. void Awake (){ 12. PlayerMv = GetComponent<PlayerMovement>(); 13. anim = GetComponent<Animator>(); 14. } 15. void FixedUpdate (){ 16. //รับค่าการเคลื่อนไหวเพื่อหาค่ารวมในการตรวจสอบเงื่อนไขการเคลื่อนไหว 17. MotionStatusNumber(LR,IdleAttBt,WalkAttbt, PlayerMv.Jbt); 18. //แปลงค่าที่ใช้ร่วมกัน 19. //ตรวจสอบเงื่อไขการเดิน 20. if(PlayerMv.HoriInput==1 || PlayerMv.HoriInput == -1){ 21. LR = 3; 22. } 23. else if(PlayerMv.HoriInput == 0) 24. { 25. LR = 0; 26. anim.ResetTrigger("Walk"); 27. } 28. //ตรวจสอบเงื่อนไขการโจมตี 29. if (AttackShooting.Shooting == 1 || AttackShooting.Shooting == 2) 30. { 31. IdleAttBt = 7; 32. } 33. else if (AttackShooting.Shooting == 0) |

**ภาพที่ 4.16** ตัวอย่าง source code AnimationControl.cs

|  |
| --- |
| 1. { 2. IdleAttBt = 0; 3. anim.ResetTrigger("IdleAtt"); 4. } 5. //ตรวจสอบเงื่อนไขการโจมตีแบบเดิน 6. if(LR ==3 && IdleAttBt == 7) 7. { 8. WalkAttbt = 10; 9. } 10. else if(LR == 0 && IdleAttBt == 0) 11. { 12. WalkAttbt = 0; 13. anim.ResetTrigger("WalkAtt"); 14. } 15. if (PlayerMv.jumpStatus == false) 16. anim.ResetTrigger("Jump"); 17. //กำหนดเงื่อนไขการคลื่นไหวทั้งหมด 18. switch(MoveNum) 19. { 20. case 3: anim.SetTrigger("Walk"); //เดิน 21. break; 22. case 5: anim.SetTrigger("Jump"); //กระโดด 23. break; 24. case 7: anim.SetTrigger("IdleAtt"); //ยืนโจมตี 25. break; 26. case 8: anim.SetTrigger("IdleAtt"); //กระโดดโจมตี 27. //anim.SetTrigger("Jump"); 28. break; 29. case 10: anim.SetTrigger("WalkAtt"); //เดินโจมตี 30. break; 31. } 32. if(PlayerHealth.health < 10) |

**ภาพที่ 4.16** ตัวอย่าง source code AnimationControl.cs (ต่อ)

|  |
| --- |
| 1. { 2. anim.SetTrigger("Die"); //ตาย 3. } 4. } 5. //คำนวณหาค่าการเคลื่อนไหว สำหรับนำไปตรวจสอบเงื่อนไขการกำหนดการเคลื่อนไหวตัวละคร 6. public int MotionStatusNumber(int LRBt, int idleAtt, int WalkAtt, int Jumpbt ) 7. { 8. MoveNum = LRBt + idleAtt + Jumpbt; 9. return MoveNum; 10. } 11. } |

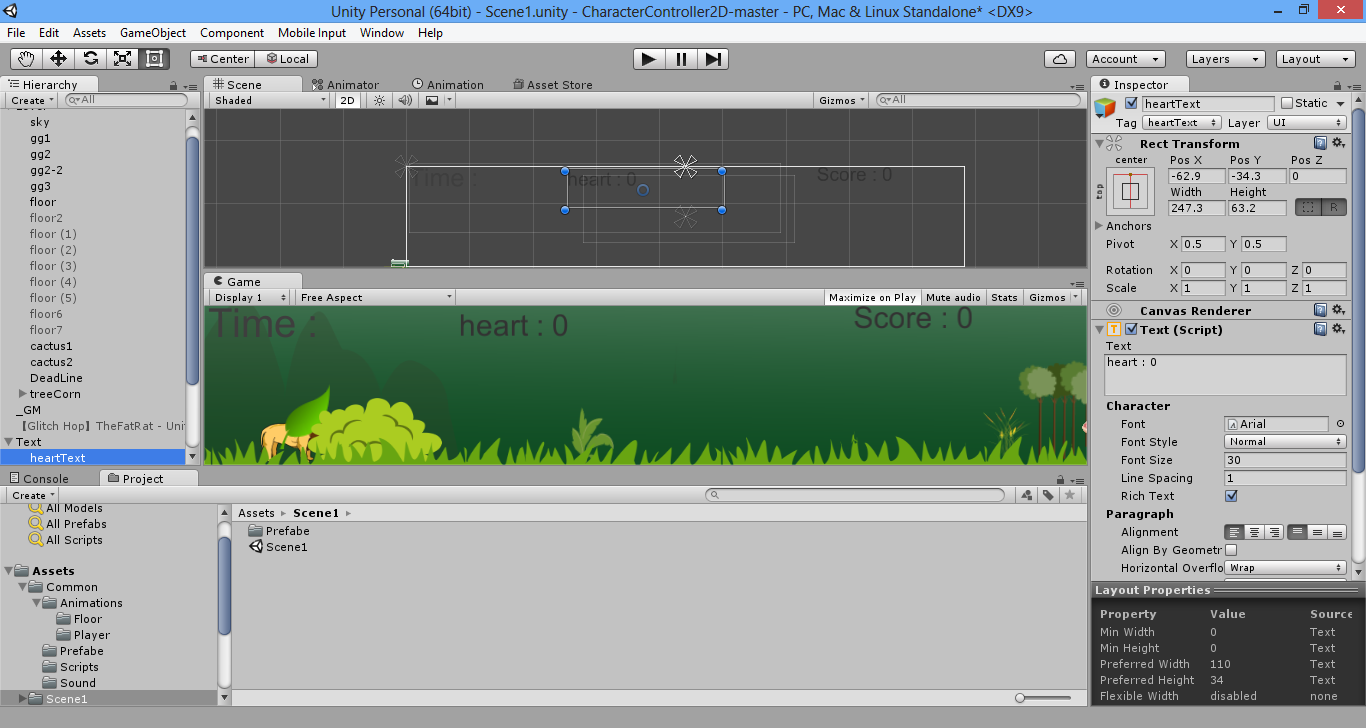
**ภาพที่ 4.16** ตัวอย่าง source code AnimationControl.cs (ต่อ)

จากตารางที่ 4.6 บรรทัดที่ 20 และ 51เป็นตรวจสอบเงื่อนไขการเดิน การโจมตี การกระโดด ที่รับค่าจาก Class AttackShooting และ PlayerMovement แล้วการส่งค่าการเคลื่อนไหวต่างๆ เข้าไปใน Method ชื่อ MotionStatusNumber ในบรรทัดที่ 71 โดย Method นี้จะทำการ Return ค่า MoveNum ไปตรวจสอบเงื่อนไขในบรรทัดที่ 51 แล้วกำหนด State ท่าทางการเดินของตัวละครให้เล่นตามค่าที่ได้

**4.2.4 การพัฒนาระบบพลังชีวิตของตัวละครหลัก**

ในขั้นตอนนี้เป็นการพัฒนาในส่วนของพลังชีวิตของผู้เล่น ซึ่งเป็นส่วนสำคัญในการกำหนดเงื่อนไขการเล่น โดยมีขั้นตอนการดำเนินการดังนี้

1. แทรก Script Component โดยเลือกไฟล์ PlayerHealth.cs ลงใน Game Object ชื่อ Player ซึ่งเป็นตัวละครในเกม พร้อมทั้งใส่องค์ประกอบต่าง ๆ ลงใน Inspector ของ Player ดังภาพที่ 4.17



**ภาพที่ 4.17** การเพิ่มแท็คหัวใจชีวิตให้กับตัวละคร

|  |
| --- |
| 1. using UnityEngine; 2. using UnityEngine.UI; 3. using System.Collections; 4. public class PlayerHealth : MonoBehaviour 5. { 6. public static float health = 100; //พลังชีวิต 7. public float Repeat = 2; //ค่าสำหรับตรวจสอบการชนกับ Enemy 8. public float hurtForce = 20; //แรงผลักเมื่อชนกับ Enemy 9. public float damage = 20; //จำนวนพลังที่ลดเมื่อชนกับ Enemy ในแต่ละครั้ง 10. public Scrollbar healthBar; //แถบพลังชีวิต 11. public Material hitEffect; //Material สีแดง 12. public Material hitEffectDefault; //Material สีขาว 13. private float lastHit; //ช่วงครั้งสุดท้ายที่ชนกับ Enemy 14. //เรียกใช้ Class 15. public PuaseMenu GameOver; 16. public AttackShooting AttBlock; 17. public PlayerMovement MoveBlock; 18. public static bool Isdeath; //สถานะการตาย 19. public Rigidbody2D bombPlayer; //เลือด 20. void Awake() 21. { 22. GameOver = this.GetComponent<PuaseMenu>(); //เรียกใช้ Component 23. AttBlock = this.GetComponent<AttackShooting>(); 24. MoveBlock = this.GetComponent < PlayerMovement>(); 25. } 26. void Update(){ 27. healthBar.size = health / 100; 28. //ตรวจสอบพลังชีวิต |

**ภาพที่ 4.18** ตัวอย่าง source code PlayerHealth.cs

|  |
| --- |
| 1. if(health < 1){ 2. Isdeath = true; 3. } 4. if(Isdeath == true){ 5. Instantiate(bombPlayer, transform.position, transform.rotation); 6. Invoke("InvokeDeath", 2); 7. Isdeath = false; 8. }} 9. void OnCollisionStay2D(Collision2D col) { //ตรวจสอบการชนแบบต่อเนื่อง 10. //เงื่อนไขเมื่อชนกับ Enemy1 และ Enemy2 11. if (col.gameObject.tag == "Enemy1" || col.gameObject.tag == "Enemy2" ){ 12. float vertical = col.gameObject.transform.position.y - transform.position.y; 13. float Horizontal = col.gameObject.transform.position.x - transform.position.x; 14. gameObject.GetComponent<SpriteRenderer>().material = hitEffect; 15. Invoke("setDeafualt", 1); 16. if (PuaseMenu.Puase == false){ 17. if (Time.time > lastHit + Repeat) { //พลังลดลง 18. if (health > 1){ 19. KeepDamage(col.transform); 20. lastHit = Time.time; 21. } 22. else if (health < 1 ){ 23. Isdeath = true; 24. }}}}} 25. void OnCollisionEnter2D(Collision2D col){ //ตรวจสอบการชนแบบต่อครั้ง 26. if (col.gameObject.tag == "Ground" ){ 27. gameObject.GetComponent<SpriteRenderer>().material = hitEffectDefault; |

**ภาพที่ 4.18** ตัวอย่าง source code PlayerHealth.cs (ต่อ)

|  |
| --- |
| 1. } 2. if (col.gameObject.tag == "Enemy1" || col.gameObject.tag == "Enemy2"){ 3. musicControl.itemCheckingSound = 5; 4. }} 5. //ตรวจสอบหลังจากที่ชนแล้ว 6. void OnCollisionExit2D(Collision2D col){ 7. if (col.gameObject.tag == "Enemy1" || col.gameObject.tag == "Enemy2" ){ 8. gameObject.GetComponent<SpriteRenderer>().material = hitEffectDefault; 9. }} 10. void InvokeDeath(){ //เข้าสู่สถานะการตาย 11. if (LifeScore.Life > 0){ 12. LifeScore.Life -= 1; 13. if (PowerScore.ScoreAtt2 < 10){ 14. PowerScore.ScoreAtt2 = 10; 15. } 16. health = 100; 17. PlayerPrefs.SetInt("ATTACK" + saveDataPrefs.slotSelectId, PowerScore.ScoreAtt2); 18. PlayerPrefs.SetInt("LIFE" + saveDataPrefs.slotSelectId, LifeScore.Life); 19. Application.LoadLevel(Application.loadedLevel); 20. }} 21. void KeepDamage(Transform enemy1){ //พลังชีวิตลด 22. Vector3 hurtVector = transform.position - enemy1.position + Vector3.up \* 5f; 23. GetComponent<Rigidbody2D>().AddForce(hurtVector \* hurtForce); 24. health -= damage; 25. } 26. public void setDeafualt(){ //ค่าตัวละครไม่ให้เป็นสีแดง 27. gameObject.GetComponent<SpriteRenderer>().material = hitEffectDefault; 28. } 29. } |

**ภาพที่ 4.18** ตัวอย่าง source code PlayerHealth.cs (ต่อ)

จากภาพที่ 4.18 บรรทัดที่ 38-48 เป็นการกำหนดเงื่อนไขการตรวจสอบสถานะของตัวละคร ถ้าพลังชีวิตหมดก็จะเรียกใช้ Method InvokeDeath ที่บรรทัดที่ 107 โดยถ้าจำนวนชีวิตของผู้เล่นไม่หมดเกมก็จะกลับมาเริ่มใหม่

บรรทัดที 53 Method OnCollisionStay2D เป็น Method สำหรับตรวจสอบค่าการชน โดยมีการรับค่า Collision2D เพื่อตรวจสอบการชนของ Collider Component เมื่อมีการชนตัวละครก็จะมีสีแดงกระพริบ

บรรทัดที่ 84 Method OnCollisionEnter2D เป็น Method สำหรับตรวจสอบการชนแบบต่อครั้ง เมื่อ Collider2D Component เมื่อชนกับ Enemy ก็จะส่งค่า itemCheckingSound ไปที่ Class musicControl เพื่อตรวจสอบเงื่อนไขเกี่ยวกับเสียง

บรรทัดที่ 98 Method OnCollisionExit2D ก็จะตรวจสอบเงื่อนไขหลังจากจนกับ Enemy ก็จะคืนค่า Material ของตัวละครให้เป็นปกติ

**4.3 การทดสอบระบบ**

ในขั้นตอนนี้จะเป็นการทดสอบการทำงานของเกม โดยทดสอบจากผลที่คาดหวัง โดยใช้เทคนิคการทดสอบแบบ Black box เพื่อตรวจสอบความถูกของการทำงานทั้งหมดในเกมดังนี้

**4.3.1 ทดสอบปุ่มกดเมนูต่างๆ** ดังตารางที่ 4.1

**ตารางที่ 4.1** การทดสอบปุ่มเมนูต่าง ๆ

|  |  |  |  |
| --- | --- | --- | --- |
| **รายการที่ทดสอบ** | **ผลที่คาดหวัง** | **ผลที่ได้** | **หมายเหตุ** |
| ปุ่มเริ่มเกม | สามารถเข้าสู่เมนูผู้เล่นได้ |  |  |
| ปุ่มวิธีเล่น | สามารถเข้าสู่หน้าวิธีเล่นได้ |  |  |
| ปุ่มผู้จัดทำ | สามารถเข้าสู่หน้าผู้จัดทำได้ |  |  |
| ปุ่มออกจากเกม | สามารถเข้าสู่หน้าเมนูการออกจากเกมได้ |  |  |
| ปุ่มย้อนกลับ | สามารถกลับสู่เมนูหลักได้ |  |  |
| ปุ่มตกลง | สามารถสร้างผู้เล่นในหน้าเมนูผู้ใช้หรือเปลี่ยนชื่อผู้เล่นได้ |  |  |
| ปุ่มลบข้อมูล | สามารถลบข้อมูลผู้เล่นได้ |  |  |
| ปุ่มถัดไป | สามารถเข้าสู่หน้าเมนูเลือกด่านได้ |  |  |
| ปุ่มกลับสู่เมนูผู้เล่น | สามารถกลับสู่หน้าเมนูผู้เล่นได้ |  |  |
| ปุ่มเลือกด่าน | สามารถเข้าสู่ด่านต่างๆได้ |  |  |
| ปุ่มตกลง | สามารถออกจากเกมได้ |  |  |
| ปุ่มยกเลิก | สามารถยกเลิกการออกจากเกมได้ |  |  |
| ปุ่มเล่นต่อ | สามารถเกมเล่นต่อได้ |  |  |
| ปุ่มเริ่มใหม่ | สามารถเล่นเริ่มเกมใหม่ได้ |  |  |
| กลับสู่เมนูเกม | สามารถกลับสู่หน้าเลือกด่านได้ |  |  |
| ปุ่มหยุด | สามารถหยุดตัวละครในเกมได้ |  |  |

**4.3.2 ทดสอบปุ่มกดในเกม** ดังตารางที่ 4.2

**ตารางที่ 4.2** การทดสอบปุ่มกดในเกม

|  |  |  |  |
| --- | --- | --- | --- |
| **รายการที่ทดสอบ** | **ผลที่คาดหวัง** | **ผลที่ได้** | **หมายเหตุ** |
| ปุ่มเดินหน้า | สามารถบังคับตัวละครให้เดินไปข้างหน้าได้ |  |  |
| ปุ่มถอยหลัง | สามารถบังคับตัวละครให้เดินถอยหลังได้ |  |  |
| ปุ่มโจมตีด้วยผ้ายันต์ | สามารถทำลายผีที่มาขัดขวางให้หายไปได้ |  |  |
| ปุ่มโจมตีด้วยพลังธรรมะ | สามารถทำลายผีที่มาขัดขวางให้หายไปได้ |  |  |
| ปุ่มกระโดด | สามารถกระโดดได้ |  |  |

**4.3.3 ทดสอบปุ่มกดเมนูต่างๆ** ดังตารางที่ 4.3

**ตารางที่ 4.3** การทดสอบเงื่อนไขในเกม

|  |  |  |  |
| --- | --- | --- | --- |
| รายการที่ทดสอบ | ผลที่คาดหวัง | ผลที่ได้ | หมายเหตุ |
| การเก็บธรรมะ | สามารถเก็บธรรมะและเมื่อเก็บตรงกับความหมายที่หน้าจอพลังการโจมตีจะเพิ่มขึ้น |  |  |
| คะแนน | เมื่อทำลายผีคะแนนจะเพิ่มและในด่านที่เล่นนั้นถ้าเก็บคะแนนได้ 5,000 คะแนนก็จะเพิ่มจะเพิ่มจำนวนชีวิตได้ |  |  |
| พลังชีวิต | เมื่อพลังชีวิตหมดจำนวนชีวิตจะลดลง 1 ชีวิตได้ |  |  |
| จำนวนชีวิต | จำนวนชีวิตสามารถลดลงและเพิ่มขึ้นตามเงื่อนไขได้ |  |  |

**4.4 ผลการกรอกสอบถามความพึงพอใจของผู้ใช้งาน**

จากการประเมินความพึงพอใจของเกมผจญภัยสัตว์ป่า โดยการเก็บข้อมูลสำรวจความคิดเห็นด้วยวิธีการวัดทัศนคติของลิเคิร์ท (Likert Scale) โดยการกรอกแบบสอบถามจากกลุ่มตัวอย่างนักเรียนชั้นประถมศึกษาปีที่ 1-3 โรงเรียนบ้านหนองกุง จำนวน 25 คน สรุปดังตารางที่ 4.4

**ตารางที่ 4.4** ผลการกรอกแบบสอบถามความพึงพอใจของผู้ใช้งาน

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **รายการ** | **ระดับความพึงพอใจในการใช้งาน** | | | | | | | |
| มากที่สุด (5) | มาก (4) | ปานกลาง (3) | น้อย (2) | น้อยที่สุด (1) | รวม | ค่าเฉลี่ย | ระดับคุณภาพ |
| 1.ความสวยงามของการออกแบบตัวละครและฉากภายในเกมมีความเหมาะสม |  |  |  |  |  |  |  |  |
| ร้อยละ |  |  |  |  |  |  |
| 2.สามารถเรียนรู้หลักธรรมทางพระพุทธศาสนาได้ในขณะที่เล่นเกม |  |  |  |  |  |  |  |  |
| ร้อยละ |  |  |  |  |  |  |
| 3.ขนาดของตัวอักษรเหมาะสม |  |  |  |  |  |  |  |  |
| ร้อยละ |  |  |  |  |  |  |
| 4.ช่วยให้เกิดความเข้าใจในบทเรียนในวิชาพระพุทธศาสนามากขึ้น |  |  |  |  |  |  |  |  |
| ร้อยละ |  |  |  |  |  |  |
| 5.เนื้อหาสาระมีความเหมาะสม |  |  |  |  |  |  |  |  |
| ร้อยละ |  |  |  |  |  |  |
| 6.รูปแบบของเกมมีความน่าสนใจ |  |  |  |  |  |  |  |  |
| ร้อยละ |  |  |  |  |  |  |
| **ระดับค่าเฉลี่ยรวม** | | | | | | |  |  |

ข้อเสนอแนะ

* ควรเพิ่มพลังธรรมะที่ใช้ในการโจมตีได้เยอะขึ้น
* อยากให้ติดตั้งเกมนี้ที่โรงเรียน