**บทที่ 3**

**วิธีการดำเนินงาน**

**บทนี้จะอธิบายการออกแบบเนื้อหา บทบรรยายเนื้อเรื่อง ตัวละคร การออกแบบหน้าจอของเกมผจญภัยสัตว์ป่า โดยมีรายละเอียด ดังนี้**

3.1 การออกแบบเนื้อเรื่อง

3.2 บรรยายเนื้อเรื่องเกม (Cut Scene)

3.3 การออกแบบหน้าจอ

3.4 การออกแบบตัวละครและฉาก

3.5 การพัฒนาเกม

3.6 การพัฒนาระบบ

3.7 การออกแบบการทดสอบระบบและแบบประเมินความพึงพอใจของผู้ใช้งาน

**3.1 การออกแบบเนื้อเรื่อง**

3.1.1 มีการเคลื่อนไหวของตัวละครหรือวัตถุภายในเกม

3.1.2 รองรับผู้เล่น 1 คน

3.1.4 ภายในเกมมีด่านทั้งหมด 5 ด่าน

3.1.5 สามารถเลือกสัตว์ป่าทั้งหมด 6 ตัว โดยแบ่งเป็นสัตว์ป่ากินพืช 3 ตัวและสัตว์ป่าที่กินเนื้อเป็น

อาหาร 3 ตัว

3.1.5 เกมที่พัฒนาสามารถทำงานบนแท็บเล็ตที่มีระบบปฏิบัติการแอนดรอยด์ 4.0 ขึ้นไป

**3.2 บรรยายเนื้อเรื่องเกม (Cut Scene)**

* + 1. **เกริ่นนำเรื่อง**

1. เนื้อหาของเกมใช้สำหรับการฝึกการเรียนรู้ของเด็กที่มีช่วงอายุ 6-9 ปี
2. ภายในเกมจะเป็นสัตว์ป่า ที่กำลังอพยพหนีจากภัยธรรมชาติไปยังป่าที่มันอาศัยอยู่โดยสัตว์เหล่านั้น โดยผู้เล่นสามารถเลือกสัตว์ที่ต้องการเล่นได้ 6 ชนิด โดยแต่ละชนิดจะมีความสามารถที่ไม่เหมือนกัน
3. มีไอเท็มภายในเกม ดังนี้

* มีพลังชีวิตอยู่ด้านบน
* มีไอเท็มผักและเนื้อ
* ไอเท็มอาหารของสัตว์ขนิดต่างๆ

1. ก่อนเล่นเกมผู้เล่นจะต้องเลือกเป็นสัตว์เพื่อเดินทางไปให้ถึงจุดหมาย โดยจะมีสัตว์ป่าให้เลือก 6 ชนิด ดังนี้

* เสือ
* สิงโต
* หมาป่า
* กวาง
* กระต่าย
* ช้าง

5) ฉากภายในเกมจะเป็นป่าที่มีการสอดแทรกความรู้เกี่ยวกับสัตว์ป่า โดยมีเนื้อหา ดังนี้

* + แบ่งตามเกณฑ์การกินอาหาร ได้แก่ สัตว์กินพืช กินเนื้อ
  + แบ่งตามลักษณะภูมิประเทศ เช่น สัตว์ในขั้วโลก ทะเลทราย เขตร้อน
  + แบ่งตามรูปร่าง เช่น สัตว์บางชนิดมี 2 ขา 4 ขา
    1. **ด่านที่ 1**

เป็นฉากป่าที่มีความอุดมสมบรูณ์ ภายในด่านประกอบด้วยไอเท็มที่จะช่วยในการผ่านด่าน 1ชนิดและมีอุปสรรคคือพื้นต่างระดับ

* + 1. **ด่านที่ 2**

เป็นฉากที่มีสภาพอากาศแบบเขตร้อน ภายในด่านประกอบด้วยไอเท็มช่วยให้ผ่านด่าน 3 ชนิดที่และมีไอเท็มเพิ่มลดชีวิต อุปสรรคคือพื้นต่างระดับ

* + 1. **ด่านที่ 3**

เป็นฉากที่มีสภาพอากาศหนาว ภายในด่านประกอบด้วยไอเท็มที่ช่วยให้ผ่านด่าน 4 ชนิดและมีไอเท็มเพิ่มลดชีวิต อุปสรรคคือพื้นต่างระดับและมีสัตว์ชนิดอื่นมาทำร้าย

* + 1. **ด่านที่ 4**

เป็นฉากที่มีสภาพอากาศแบบฝนตก ภายในด่านประกอบด้วยไอเท็มที่ช่วยให้ผ่านด่าน 5 ชนิดและมีไอเท็มเพิ่มลดชีวิต อุปสรรคคือพื้นต่างระดับ มีสัตว์อื่นมาทำร้ายและมีกิ่งไม้หล่นลงมา หากกิ่งไม้หล่นลงมาโดนตัวจะทำให้ชีวิตลดลงเร็วขึ้น

* + 1. **ด่านที่ 5**

เป็นฉากจะเปลี่ยนไปตามป่าที่สัตว์เหล่านั้นอาศัยอยู่ มีเวลาจำกัดในการเก็บคะแนน เพื่อทำคะแนนให้ได้สูงสุด ซึ่งจะมีฉาก ดังนี้

* + ฉากป่าที่มีความอุดมสมบูรณ์ สัตว์ที่อาศัยอยู่ ได้แก่ กระต่าย กวาง ช้าง
  + ฉากป่าเขตร้อน สัตว์ที่อาศัยอยู่ ได้แก่ สิงโต เสือ
  + ฉากป่าดิบชื้น สัตว์ที่อาศัยอยู่ ได้แก่ หมาป่า

**3.3 การออกแบบหน้าจอ**

3.3.1 เมนูหลัก

ประกอบไปด้วย 4 เมนู คือ เริ่มเกม วิธีเล่น ข้อมูลผู้จัดทำ ออกจากเกม

****

**ภาพที่ 3.1** การออกแบบเมนูหลัก

3.3.2 หน้าจอเลือกสัตว์

สัตว์กินเนื้อ

สัตว์กินพืช

**ภาพที่ 3.2** การออกแบบหน้าจอเลือกสัตว์ที่จะเล่น

3.3.3 วิธีการเล่น

หน้านี้จะอธิบายวิธีการเล่นเกม และกฎกติกาต่าง ๆ ภายในเกม

วิธีการเล่น

เดินไปทางขวา

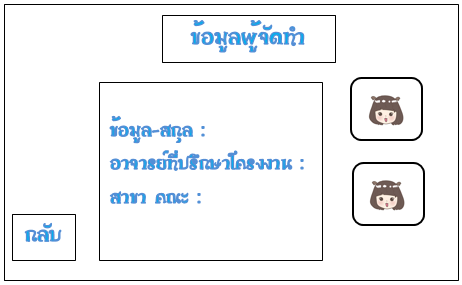
เดินไปทางซ้าย

กระโดด

**ภาพที่ 3.3** การออกแบบหน้าจอวิธีการเล่นเกม

3.3.4 หน้าจอข้อมูลผู้จัดทำ

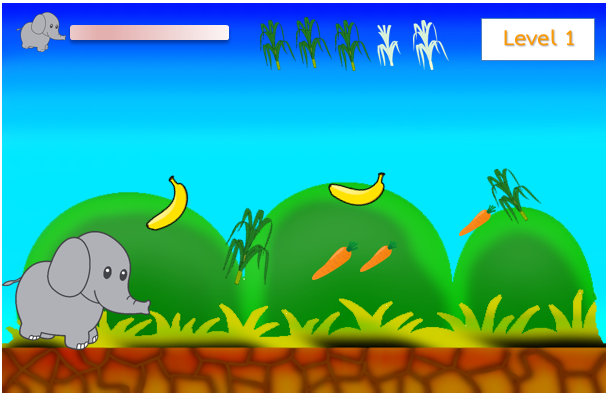
หน้านี้เป็นการออกแบบหน้าจอข้อมูลผู้จัดทำ โดยมีรายละเอียดคร่าว ๆ ดังนี้

****

**ภาพที่** **3.4** การออกแบบหน้าจอข้อมูลผู้จัดทำ

3.3.5 การออกแบบในส่วนหน้าจอในการเล่นเกม

ในแต่ละด่านผู้เล่นจะต้องเดินทางไปให้ถึงจุดหมาย โดยจะต้องเก็บไอเท็มที่เป็นอาหารที่สัตว์ตัวนั้นๆ กินเป็นอาหาร ให้ถูกต้อง หากเก็บครบก็จะผ่านด่านต่อไป โดยในแต่ละด่านจะมีอุปสรรค์ที่คอยกีดขวางต่างกันไป ผู้เล่นต้องเดินและกระโดดข้าม เพื่อไปยังไอเท็มที่ต้องการ จนกว่าจะครบตามเป้าหมายที่กำหนดในแต่ละด่านถึงจะผ่านไปด่านต่อไปได้

****

ขวา

ซ้าย

Jump

**ภาพที่ 3.5** การออกแบบหน้าจอในการเล่นเกม

**3.4 การออกแบบตัวละครและฉาก**

3.4.1 การออกแบบตัวละคร

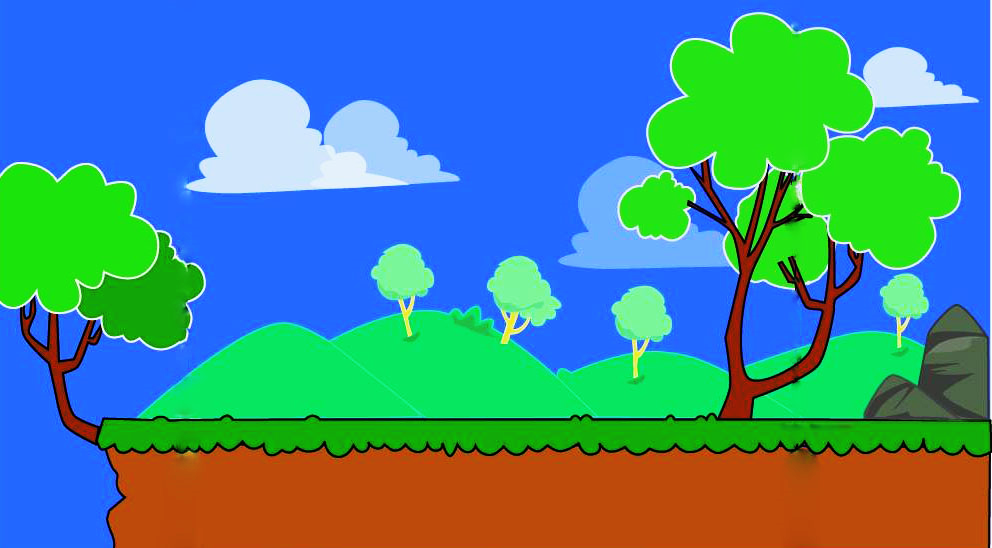
การออกแบบตัวละครเป็นการนำเอาสัตว์ป่า ที่มีข้อมูลอ้างอิงจากหนังสือเรียนมาออกแบบท่าทางที่ใช้ภายในเกม โดยผู้เล่นสามารถเลือกสัตว์ที่ต้องการเล่นได้ 6 ชนิด ดังนี้

|  |  |
| --- | --- |
| ภาพตัวละคร | รายละเอียดตัวละคร |
|  | ชื่อ : สิงโต  เป้าหมาย: การเก็บไอเท็มที่เป็นให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่เป็นนักล่า กินเนื้อสัตว์อื่นเป็นอาหาร เช่น กระต่าย ไก่ป่า กวาง ม้าลาย เป็นต้น และสามารถเดิน กระโดด ได้ ชอบอาศัยอยู่ในป่าเขตร้อน |
|  | ชื่อ : เสือ  เป้าหมาย: การเก็บไอเท็มทีให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินเนื้อเป็นอาหาร เช่น กวาง หมูป่า และสามารถเดิน กระโดด ได้ อาศัยอยู่ในป่าเขตร้อน |
|  | ชื่อ : หมาป่า  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินเลือด เนื้อของสิ่งมีชีวิตทุกประเภทและสามารถเดิน กระโดด ได้ อาศัยอยู่ในป่าดิบชื้น |

|  |  |
| --- | --- |
|  | ชื่อ : กระต่าย  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินพืชเป็นอาหาร เช่น  แครอท สามารถวิ่งเร็วได้ และอาศัยอยู่ในป่าอุดมสมบรูณ์ |
|  | ชื่อ : กวาง  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินพืช ทั้งใบและยอด อาศัยอยู่ทุ่งโล่ง ชายป่า และสามารถเดิน กระโดด ได้ ในป่าอุดมสมบรูณ์ |
|  | ชื่อ : ช้าง  เป้าหมาย: การเก็บไอเท็มให้ครบตามที่กำหนดไว้ในแต่ละด่าน  ลักษณะเฉพาะ: เป็นสัตว์ป่าที่กินพืชเป็นอาหาร เช่น อ้อย ต้นกล้วย และสามารถเดิน เหยียบศัตรูได้ อาศัยอยู่ในป่าอุดมสมบรูณ์ |

3.4.2 การอกแบบฉาก

1. ด่านที่ 1 : ป่าที่มีความอุดมสมบรูณ์













**ภาพที่** **1** ฉากด่านที่ 1

1. ด่านที่ 2 : เป็นฉากที่มีสภาพอากาศแบบเขตร้อน











**ภาพที่** **2** ฉากด่านที่ 2

1. ด่านที่ 3 : เป็นฉากที่มีสภาพอากาศหนาว









****



**ภาพที่** **3** ฉากด่านที่ 3

1. ด่านที่ 4 : เป็นฉากที่มีสภาพอากาศแบบฝนตก











**ภาพที่** **4** ฉากด่านที่ 4

1. ด่านที่ 5 : เป็นฉากจะเปลี่ยนไปตามป่าที่สัตว์เหล่านั้นอาศัยอยู่



Score=00







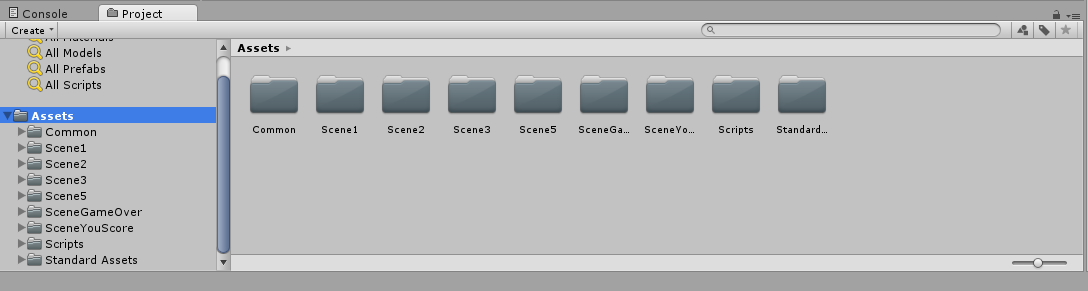


**ภาพที่** **5** ฉากด่านที่ 5

**3.5 การพัฒนาเกม**

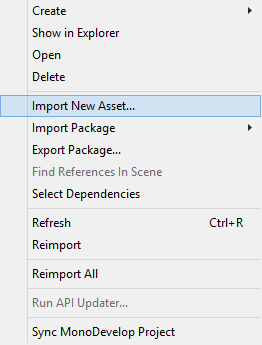
**3.5.1 การนำเข้าข้อมูลในการพัฒนาเกม**

ในส่วนนี้จะเป็นการนำเข้าข้อมูลรูปภาพ ที่ต้องใช้เข้ามาใส่ไว้ใน Project Assets ลงในโฟลเดอร์ที่ได้สร้างขึ้นไว้ ดังภาพที่ 3.7



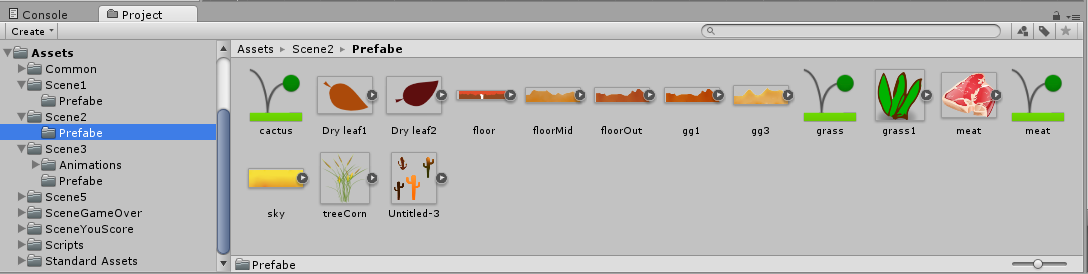
**ภาพที่ 3.7** โฟลเดอร์สำหรับจัดเก็บข้อมูลต่าง ๆ ใน Project Panel

1. คลิกขวา เลือก Import New Asset.. ดังภาพที่ 3.8



**ภาพที่ 3.8** การนำเข้าข้อมูล

1. เลือกโฟลเดอร์ที่ต้องเก็บไฟล์ภาพที่จะใช้ทำฉากในแต่ละด่าน ดังภาพที่ 3.9



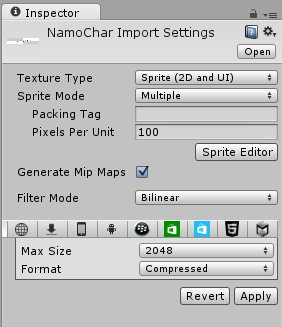
**ภาพที่ 3.9** โฟลเดอร์สำหรับจัดเก็บภาพในแต่ละด่าน

1. เลือกโฟลเดอร์ที่ต้องเก็บไฟล์ภาพหรือตัวละครต่างๆ ดังภาพที่ 3.10



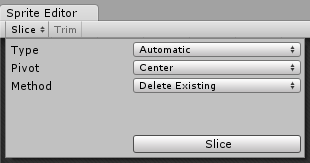
**ภาพที่ 3.10** โฟลเดอร์สำหรับจัดเก็บสัตว์หรือตัวละครต่างๆ

1. ตั้งค่าที่ Inspector Panel ดังภาพ แล้วกด Apply ดังภาพที่ 3.11



**ภาพที่ 3.11** การตั้งค่าภาพที่ Inspector Panel

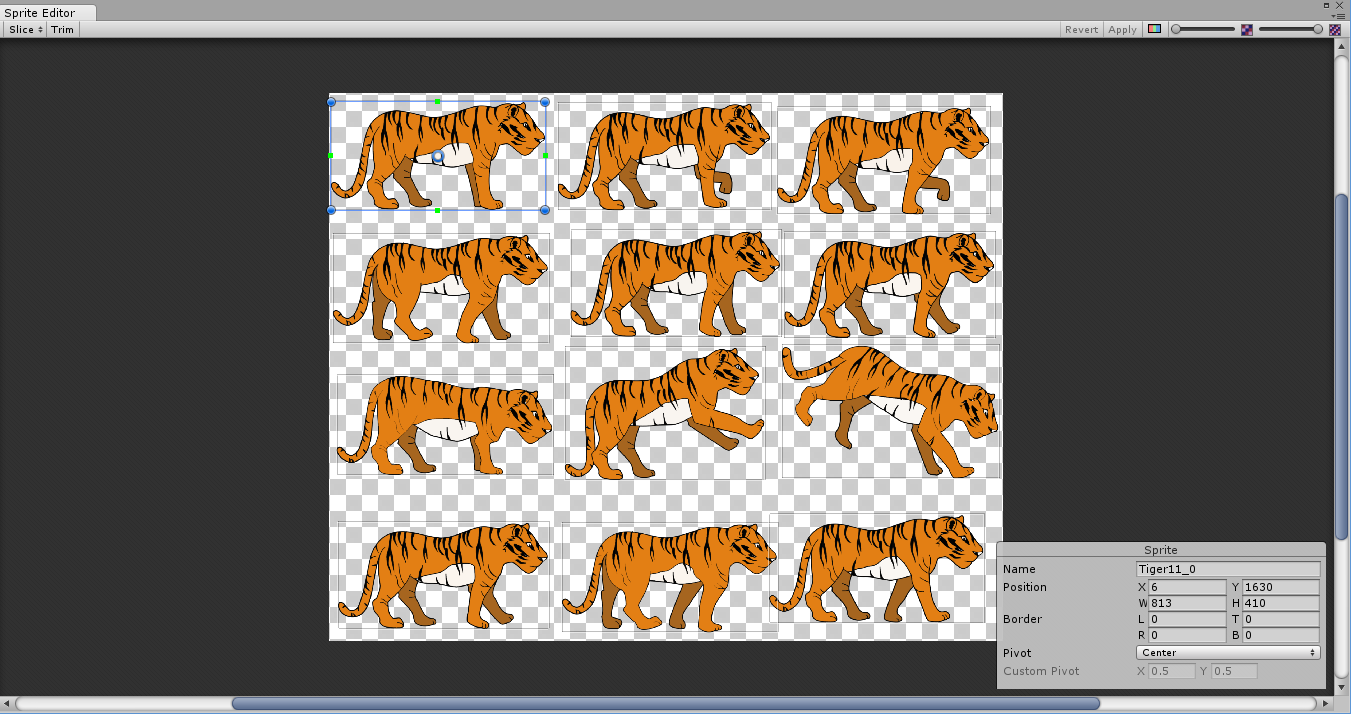
1. คลิกที่ปุ่ม Sprite Editor ใน Inspector Panel แล้วคลิกที่คำว่า Slice ดังภาพที่ 3.12



**ภาพที่ 3.12** การตั้งค่า Sprite Editor

1. คลิกที่ปุ่ม Apply โปรแกรมก็จะทำการตัดแบ่งส่วนของภาพให้อยู่ในรูปแบบของ Sprite

ดังภาพที่ 3.13

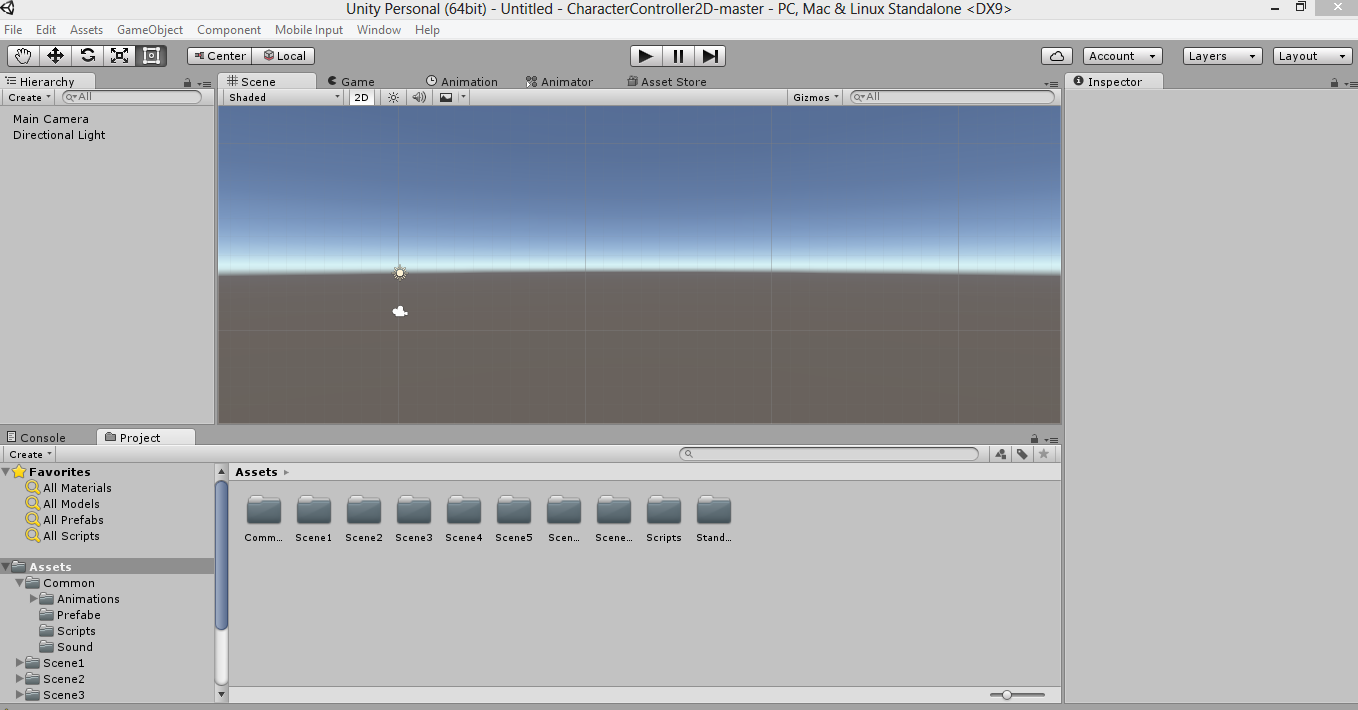


**ภาพที่ 3.13** การแบ่งภาพเป็น Sprite

**3.5.2 เมนูหลัก**

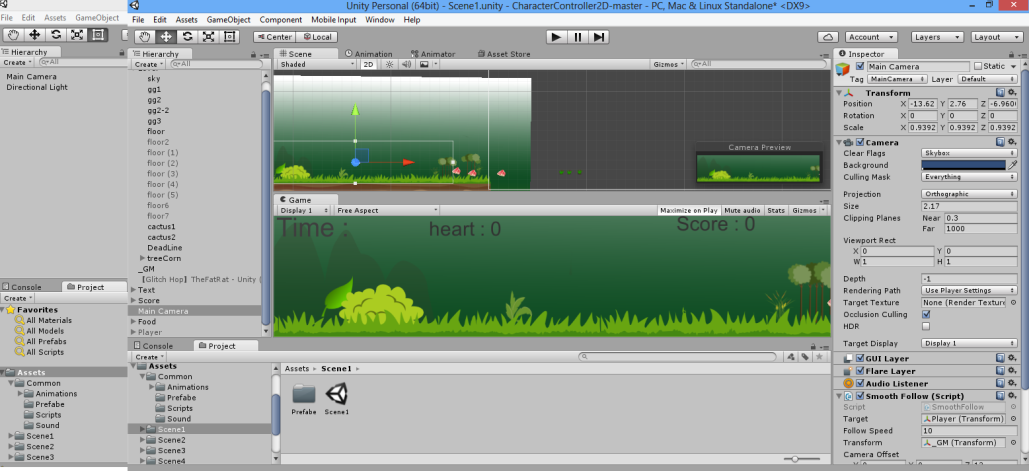
ในส่วนนี้เป็นการพัฒนาหน้าจอเมนูหลักของเกม ซึ่งเมนูหลักประกอบด้วย เมนูจัดการผู้เล่น เมนู วิธีการเล่นเกม เมนูผู้จัดทำ เมนูเลือกด่าน เมนูออกจากเกม

1. เมื่อเปิดโปรแกรมขึ้นมาครั้งแรก ก็จะมีหน้าสีน้ำเงินว่างเปล่า หากต้องการบันทึกให้กดปุ่ม Ctrl และ S หรือกดที่เมนู File แล้วเลือก Save เพื่อทำการบันทึก โดยตั้งชื่อว่า CaracterController2D หรือชื่อที่ต้องการ ดังภาพที่ 3.14



**ภาพที่ 3.14** หน้าจอ CaracterController2D

1. นำรูปภาพที่ import เข้ามาใน project มาแล้วจัดวางองค์ประกอบไว้ใน Scene Panel ดังภาพที่ 3.15

****

**ภาพที่ 3.15** การจัดวางองค์ประกอบของหน้า CaracterController2D ฉากที่1ของเกม

3) ตั้งค่า Main Camera ในส่วนของ Component ที่ Inspector Panel และ คลิกที่ Add Component ใส่source code ScaleWidthCamera.cs เข้าไป เพื่อกำหนดขนาดหน้าจอให้รองรับขนาดหน้าจอโทรศัพท์ที่ต่างกัน ดังภาพที่ 3.16



**ภาพที่ 3.16** 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 3.17** 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

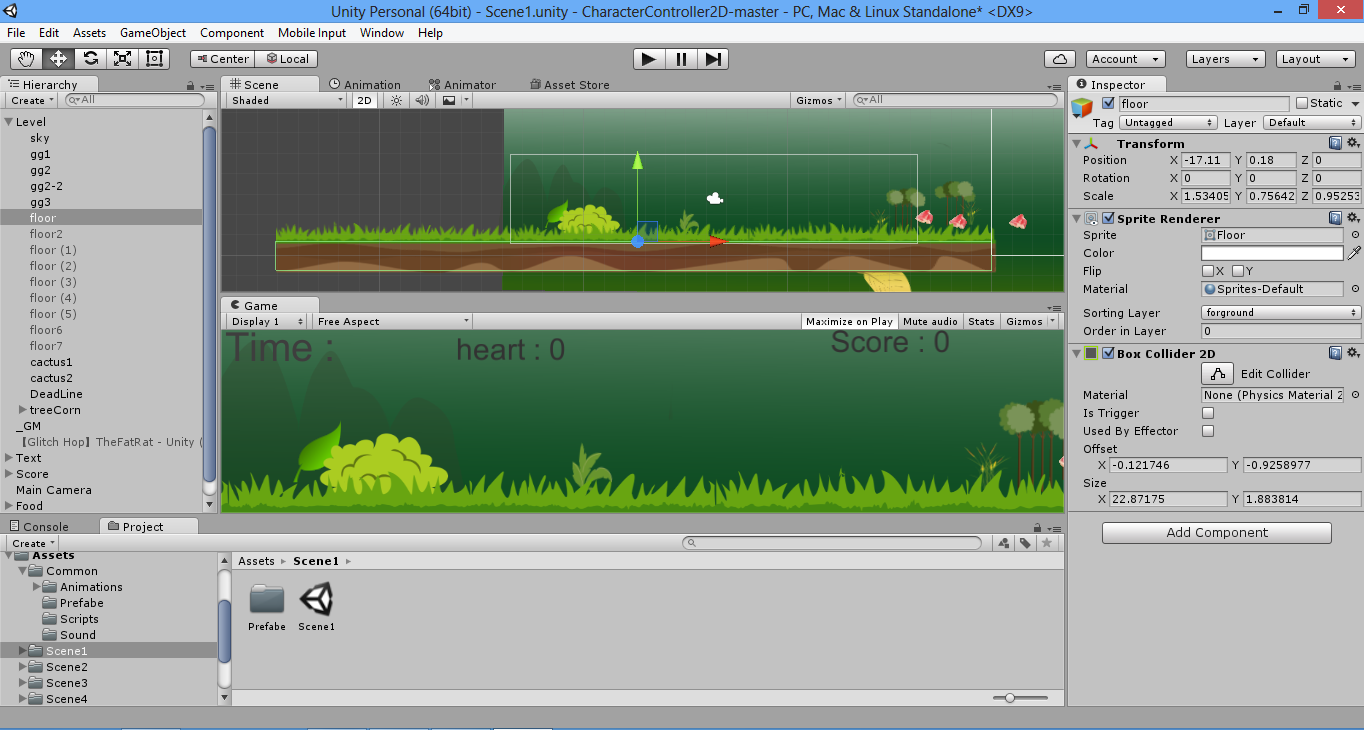
บรรทัดที่ 5 และ 6 เป็นการสร้าง Attribute สำหรับกำหนดขนาดหน้าจอ และอัตราส่วนต่อขนาดหน้าจอ

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

**3.5.3 การพัฒนาการเคลื่อนไหวของตัวละครในเกม**

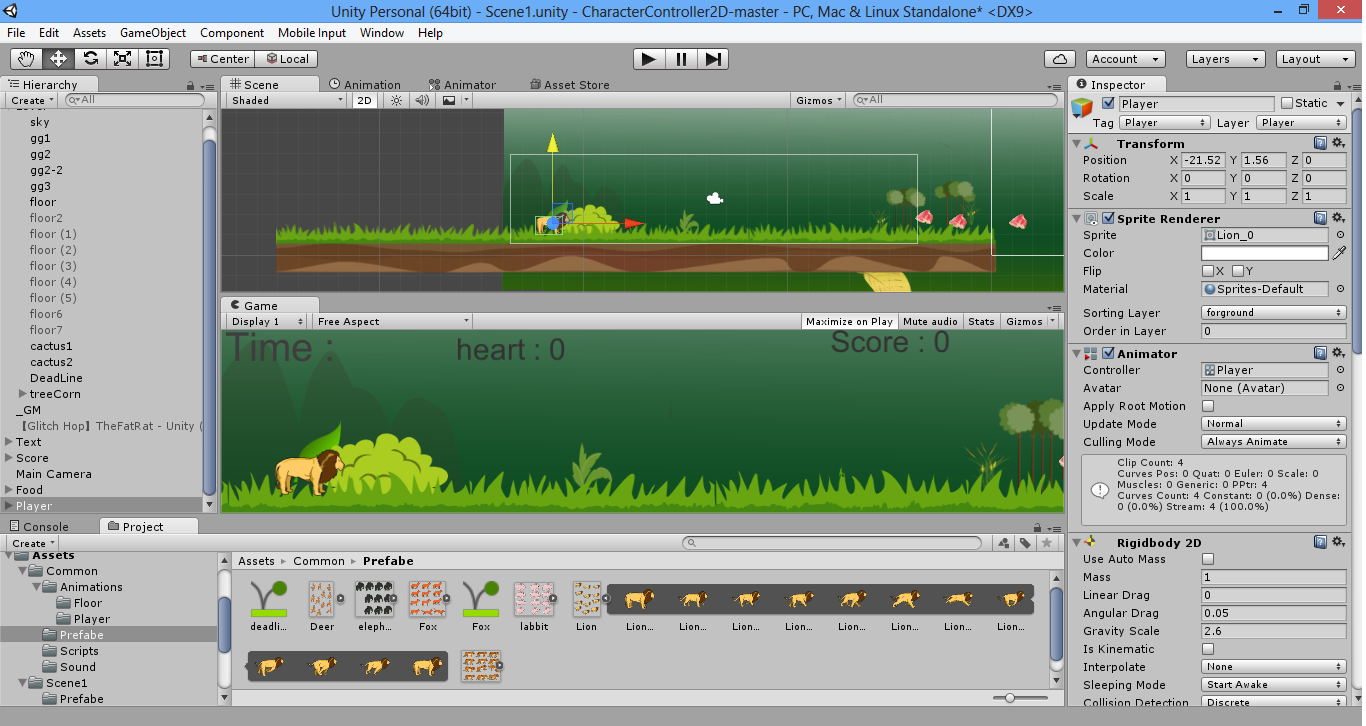
ในหัวข้อนี้จะเป็นการนำรูปภาพตัวละครที่เป็นท่าทางต่าง ๆ ที่ได้ทำไว้มาพัฒนาเป็นแอนนิเมชันสำหรับนำไปใช้ในเกม โดยมีวิธีดังนี้

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 3.18



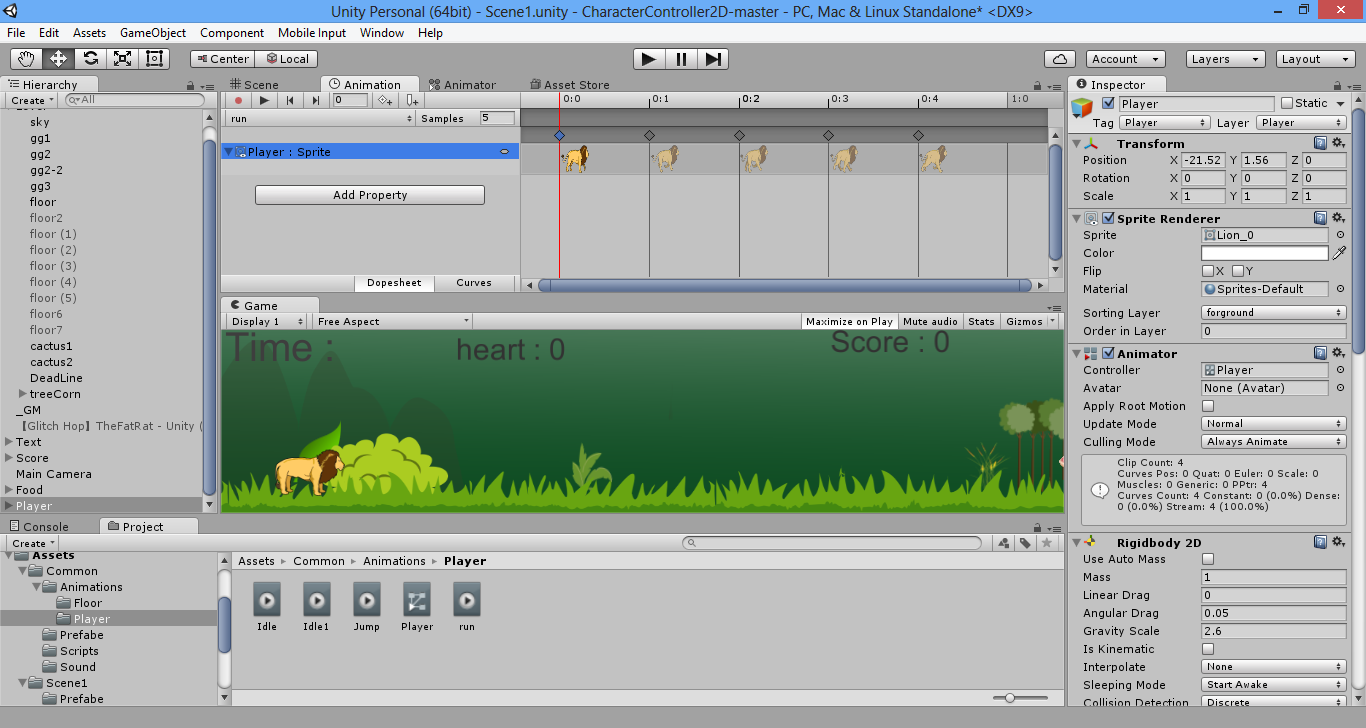
**ภาพที่ 3.18** การใส่ Box Collider 2D

1. นำรูปภาพที่ Import เข้ามาแล้วใน Project โดยนำ Sprite ที่เป็นท่าทางต่าง ๆ รูปแรกเข้ามาแทรกลงที่ Hierarchy Panel แล้วคลิกที่ Sprite แล้วเพิ่ม Component ชื่อ Animator ที่ Inspector ของ Sprite ตัวละคร ดังภาพที่ 3.19



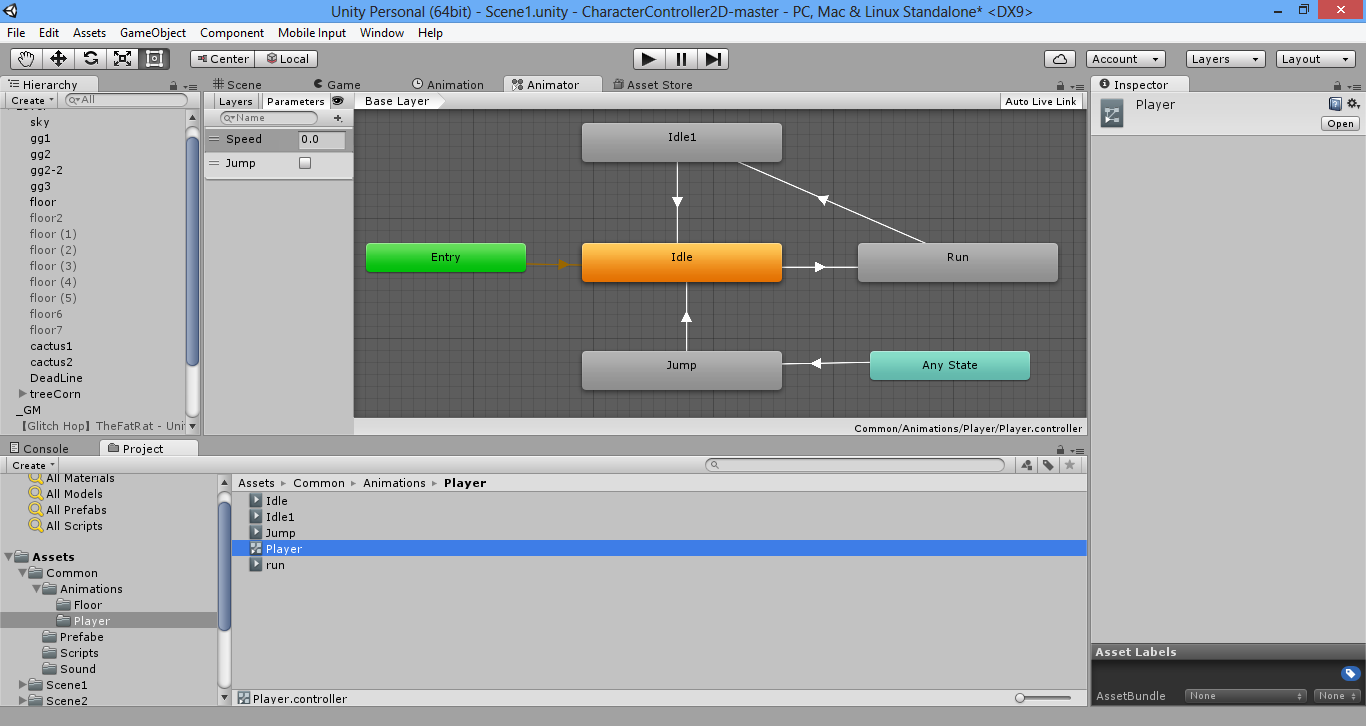
**ภาพที่ 3.19** การทำแอนนิเมชันด้วย Sprite

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 3.20



**ภาพที่ 3.20** 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 3.21



**ภาพที่ 3.21** 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 3.22** 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 3.22** 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. } |

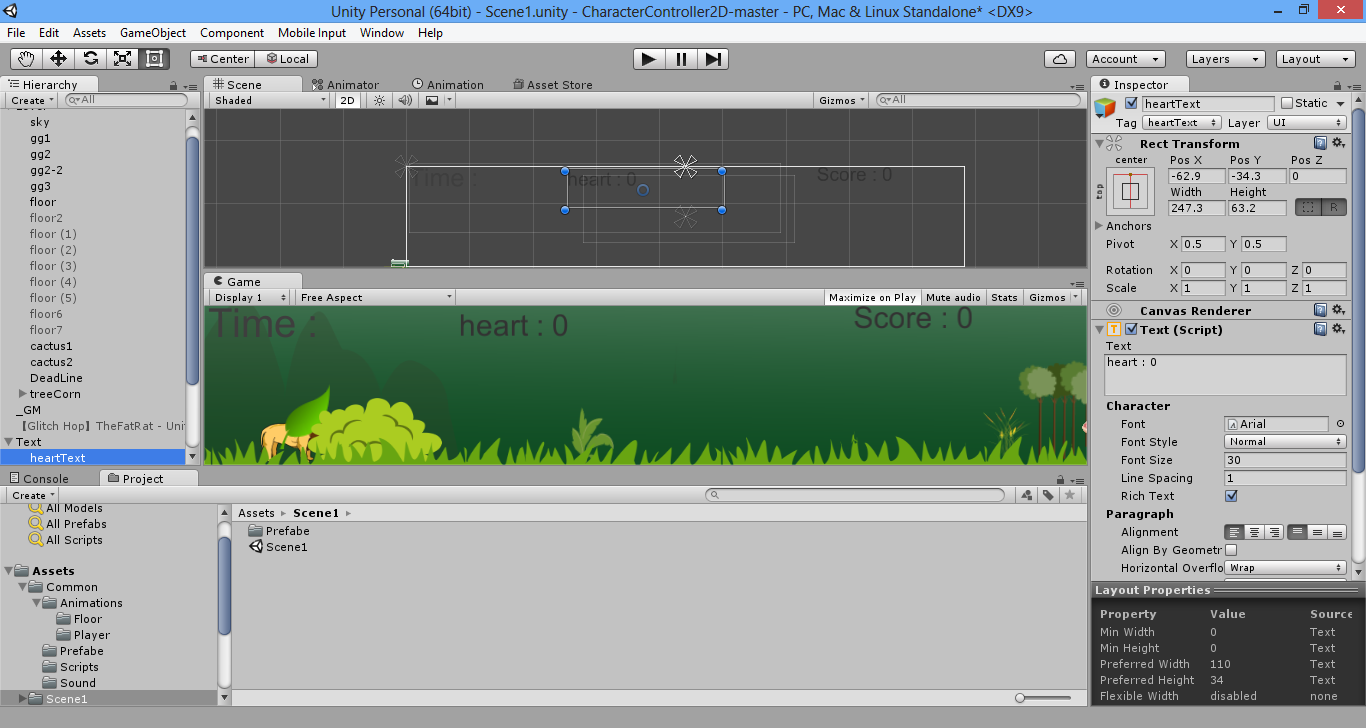
**ภาพที่ 3.22** ตัวอย่าง source code AnimationControl.cs (ต่อ)

จากตารางที่ 3.22 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

**3.5.4 การพัฒนาระบบพลังชีวิตของตัวละครหลัก**

ในขั้นตอนนี้เป็นการพัฒนาในส่วนของพลังชีวิตของผู้เล่น ซึ่งเป็นส่วนสำคัญในการกำหนดเงื่อนไขการเล่น โดยมีขั้นตอนการดำเนินการดังนี้

1. แทรก Script Component โดยเลือกไฟล์ PlayerHealth.cs ลงใน Game Object ชื่อ Player ซึ่งเป็นตัวละครในเกม พร้อมทั้งใส่องค์ประกอบต่าง ๆ ลงใน Inspector ของ Player ดังภาพที่ 3.23



**ภาพที่ 3.23** 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 3.24** 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 3.24** 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 3.24** ตัวอย่าง source code PlayerHealth.cs (ต่อ)

จากภาพที่ 3.24 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

**3.6 การพัฒนาระบบ**

**\*\*\*\*\*ยังไม่เสร็จ\*\*\*\*\*\***

**3.7 การออกแบบการทดสอบระบบและแบบประเมินความพึงพอใจของผู้ใช้งาน**

การออกแบบแบบประเมินความพึงพอใจของผู้ใช้งานเกมผจญภัยสัตว์ป่า ด้วยการเก็บข้อมูลความคิดเห็นด้วยแบบสอบถาม โดยวิธีประเมินของลิเคิร์ท (Likert Scale) [12]

**ตารางที่ 3.7** เกณฑ์คะแนนระดับความพึงพอใจ

|  |  |  |
| --- | --- | --- |
| ระดับเกณฑ์การให้คะแนน | | ความหมาย |
| เชิงคุณภาพ | เชิงปริมาณ |
| มากที่สุด | 5 | เกณฑ์ความน่าสนใจอยู่ในระดับมากที่สุด |
| มาก | 4 | เกณฑ์ความน่าสนใจอยู่ในระดับมาก |
| ปานกลาง | 3 | เกณฑ์ความน่าสนใจอยู่ในระดับปานกลาง |
| น้อย | 2 | เกณฑ์ความน่าสนใจอยู่ใจในระดับน้อย |
| น้อยที่สุด | 1 | เกณฑ์ความน่าสนใจอยู่ในระดับน้อยที่สุด |

เกณฑ์การจัดระดับคะแนนเฉลี่ยของระดับความพึงพอใจ ดังนี้

คะแนนเฉลี่ย 4.21 – 5.00 หมายความว่า มีผลมากที่สุด

คะแนนเฉลี่ย 3.41 – 4.20 หมายความว่า มีผลมาก

คะแนนเฉลี่ย 2.61 – 3.40 หมายความว่า มีผลปานกลาง

คะแนนเฉลี่ย 1.81 – 2.60 หมายความว่า มีผลน้อย

คะแนนเฉลี่ย 1.00 – 1.80 หมายความว่า มีผลน้อยที่สุด

**แบบการประเมินผลความพึงพอใจ**

**ผู้ใช้งานเกมผจญภัยสัตว์ป่า**

**ตอนที่ 1 สถานภาพของผู้ตอบ** (กรุณาทำเครื่องหมาย ✓ หน้าข้อความต่อไปนี้)

บุคลทั่วไป

ชั้นประถมศึกษาชั้นปีที่......

เพศ ชาย หญิง

**ตอนที่ 2 ความพึงพอใจต่อการใช้งาน**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **เกณฑ์พิจารณา** | **มากที่สุด(5)** | **มาก**  **(4)** | **ปานกลาง**  **(3)** | **น้อย**  **(2)** | **น้อยมาก**  **(1)** |
| 1. ความสวยงามของการออกแบบตัวละครและฉากภายในเกมมีความเหมาะสม |  |  |  |  |  |
| 2.สามารถเรียนรู้ความรู้เกี่ยวกับสัตว์ป่าได้ในขณะที่เล่นเกม |  |  |  |  |  |
| 3.ขนาดของตัวอักษรเหมาะสม |  |  |  |  |  |
| 4.ช่วยให้เกิดความเข้าใจในการแบ่งประเภทของสัตว์ป่ามากขึ้น |  |  |  |  |  |
| 5.เนื้อหาสาระมีความเหมาะสม |  |  |  |  |  |
| 6.รูปแบบของเกมมีความน่าสนใจ |  |  |  |  |  |

ข้อเสนอแนะ……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….………………………………..