程式介紹

程式碼也都附加了註解，也就是該段或該區程式的用途

playerControls玩家控制

1. using UnityEngine;

2. using System.Collections;

3.

4. public class playerControls : MonoBehaviour {

首先先是宣告一些玩家會用到的參數，像是走路速度、跳躍的高度等等，輸入的數值在unity介面內也能直接做調整而不需要每次都開啟程式來修改參數數值

5. //走路速度

6. public float runSpeed = 12.0f;

7. //跳躍高度

8. public float jumpHeight = 12.0f;

9. //墜落死亡

10. public float fall = -12;

11. //音效.跳躍聲音

12. public AudioClip jumpSound;

13.

接著是程式中會用到的判斷方向還有是否可以操作以及向量速度等等的宣告

14. //撞

15. private RaycastHit hit;

16. //跳躍計時

17. private float jumpCounter = 0.0f;

18. //角色控制

19. private CharacterController controller;

20. //方向速度

21. private Vector3 vel;

22. //判斷正負用

23. private float lookX;

24. //是否可以控制

25. private bool canControl = true;

26. //是否可以撞到天花板

27. private bool canCeiling = true;

28.

首先抓取玩家組建中的CharacterController並宣告面向Ｘ軸的變量用於判斷人物是面向左邊還是右邊，朝右移動是＋朝左則是－

29. void Start () {

30. controller = GetComponent<CharacterController>();

31. lookX = transform.localScale.x;

32. }

33.

34. void Update () {

35.

36. //重力應用

37. //如果人物不在地面將給予Y軸重力40

38. //如果人物沒有重力就像在外太空,一躍起將會突破天際

39. //否則給予人物Y軸-1的數值來穩定腳色站在地面上

40. //如果今天想製作彈力球不仿輸入5左右的正數,存在重力的腳色將會不停的跳動

C#中的驚嘆號！指的是不等於，所以if(!controller.isGrounded)就是＂如果（不是在控制的狀態是在地面）＂這樣

41. if(!controller.isGrounded){

42. jumpCounter += Time.deltaTime;

43. vel.y -= Time.deltaTime\*40;

44. }else{

45. jumpCounter = 0.0f;

46. vel.y = -1;

47. }

48.

49. //翻轉人物

50. //X軸速度大於0人物要看向右方,小於0則-lookX多個負號改為左邊

如果人物在操作的狀態下Ｘ軸的數值是正數代表現在正在朝右移動，那麼人物的方向就是朝右，另外朝左移動只要改為－lookX就行了

51. if(controller.velocity.x > 0){

52. transform.localScale = new Vector3(lookX,transform.localScale.y,transform.localScale.z);

53. }

54. if(controller.velocity.x < 0){

55. transform.localScale = new Vector3(-lookX,transform.localScale.y,transform.localScale.z);

56. }

57.

58. //設定移動按鍵

59. //鍵盤上下左右作為移動按鍵

如果習慣用左手操作也可以加入ＷＡＳＤ作為方向鍵來控制人物，操作的案件也可以在unity Edit-Project Settings-Input內做修改

60. if(canControl){

61. if(Input.GetKey("left") || Input.GetKey("right")){

62. //如果按下左鍵往左走

63. if(Input.GetKey(KeyCode.LeftArrow)){

64. vel.x = -runSpeed;

65. }

66. //如果按下右鍵往右走

67. if(Input.GetKey(KeyCode.RightArrow)){

68. vel.x = runSpeed;

69. }

70. //如果沒有按的話就停止不動

如果沒有設置不按時Ｘ軸速度不為０的話玩家就會一直走下去，因為按下按鍵輸入的命令會一直執行，所以也要告訴它停止

71. }else{

72. vel.x = 0;

73. }

74. //跳躍的按鈕是上鍵

75. //如果人物離地<0.1的話可以跳躍,這樣人物就算稍微離地的瞬間也能跳躍才不會玩起來很卡很刁難操作

76. //跳躍則為Y軸所設定的跳躍高度,可以在unity內直接做編輯

77. //最後是撥放跳躍音效

78. if(Input.GetKey(KeyCode.UpArrow)){

79. if(jumpCounter < 0.1f){

80. vel.y = jumpHeight;

81. jumpCounter = 0.1f;

82. GetComponent<AudioSource>().PlayOneShot(jumpSound);

83. }

84. }

85. }

86.

87. //如果跳躍撞到東西時會停止跳躍的加速度並且重製碰撞的處理

88. if ((controller.collisionFlags & CollisionFlags.Above) != 0 && canCeiling){

89. canCeiling = false;

90. vel.y = 0;

91. StartCoroutine(resetCeiling());

92. }

93.

94. //應用動作向量加速度到玩家

95. controller.Move(vel\*Time.deltaTime);

96.

97. //如果玩家落到限制高度以下將恢復原本位置

98. if(transform.position.y < fall){

99. string lvlName = Application.loadedLevelName;

100. Application.LoadLevel(lvlName);

101. }

102. }

103. //角色跳躍撞到東西時的處理時間,如果角色跳躍時間較長就需要注意角色是否會黏在天花板下不來

104. public IEnumerator resetCeiling () {

105. yield return new WaitForSeconds (0.25f);

106. canCeiling = true;

107. }

108. //如果角色死亡將不能操作

109. void died () {

110. canControl = false;

111. vel.x = 0;

112. }

113. }

playerHealth玩家血量

1. using UnityEngine;

2. using System.Collections;

3.

4. public class playerHealth : MonoBehaviour {

5.

6. //主角的血量

7. public int hearts = 6;

8. //受到撞擊的聲音

9. public AudioClip hitSound;

10. //死亡動畫

11. public GameObject deathAnim;

12. //設置三種血格代表血量的變化

當然也可以製作成一顆愛心為一格血量，或是四格血量，或是一顆大愛心作為多格血量

13. public Texture full;

14. public Texture half;

15. public Texture empty;

16. //碰到補血道具的音效

17. public AudioClip heartSound;

18.

19. //設置死亡為否

20. private bool dead = false;

21. //設置可接受傷害為是

22. private bool canGetHurt = true;

23. //渲染2D圖形

24. private SpriteRenderer rend;

25. //血量介面

ＧＵＩ指的是遊戲中的介面，像是玩家的血量關卡的分數等等

26. private GUITexture[] heartsGUI;

27. private int health;

28.

29. void Start () {

30. //一個愛心為兩格血量

31. health = hearts\*2;

32. //將血量與GUI介面上的血量做連結

33. GameObject getHearts = GameObject.Find("GUI/hearts");

34. //得到血量並增加到血量並撥放更新於GUI介面

35. getHearts.SendMessage("addHearts", hearts, SendMessageOptions.DontRequireReceiver);

36. GUITexture[] allChildren = getHearts.GetComponentsInChildren<GUITexture>();

37. heartsGUI = new GUITexture[allChildren.Length];

38. health = allChildren.Length\*2;

39. for(int i = 0;i < allChildren.Length;i++){

40. heartsGUI[i] = allChildren[i] as GUITexture;

41. }

42. rend = GetComponent<SpriteRenderer>();

43. }

44. //如果在可以接受傷害與非死亡的狀態下受到攻擊

45. //播放受傷音效

46. //減少血量

47. //確認血量並重製可受到傷害

CanHurt就是所謂的無敵時間，如果沒有它的畫碰到怪物會一直連續的接受傷害判定，所以需要一段緩衝時間不會受到傷害

48. void takeDamage (int amount) {

49. if(canGetHurt && !dead){

50. canGetHurt = false;

51. GetComponent<AudioSource>().PlayOneShot(hitSound);

52. health -= amount;

53. StartCoroutine(checkHealth());

54. StartCoroutine(resetCanHurt());

55. }

56. }

57.

58. //待在怪物或是陷阱上受到傷害

59. //這裡設定受到的傷害為1,調到100或許就是所謂的超困難遊戲

60. void OnTriggerStay (Collider other){

61. if(other.tag == "enemy" || other.tag == "trap"){

62. if(canGetHurt && !dead){

63. canGetHurt = false;

64. GetComponent<AudioSource>().PlayOneShot(hitSound);

65. health -= 1;

66. StartCoroutine(checkHealth());

67. StartCoroutine(resetCanHurt());

68. }

69. }

70. //碰到補血道具要增加血量

71. if(other.GetComponent<Collider>().tag == "heart"){

72. Destroy(other.gameObject);

73. addHealth();

74. }

75. }

76.

77. //碰撞到怪物或是陷阱時受到傷害,若只有碰撞傷害若玩家不動就只會觸發一次

OnTriggerStay與OnCollisionStay兩段程式長得很像，一個是待在目標上另一個則是與該目標碰撞，如果只有碰撞的話那人物碰撞後若不動的話就只會觸發那一次而已，所以除了物件與物件間碰撞以外還會有stay停留的處理

78. void OnCollisionStay (Collision other){

79. if(other.collider.tag == "enemy" || other.collider.tag == "trap"){

80. if(canGetHurt && !dead){

81. canGetHurt = false;

82. GetComponent<AudioSource>().PlayOneShot(hitSound);

83. health -= 1;

84. StartCoroutine(checkHealth());

85. StartCoroutine(resetCanHurt());

86. }

87. }

88. }

89.

90. //重製可以受到傷害的各種數值與設定

91. //受到傷害時變為紅色(1.0f,0.5f,0.5f,1.0f)就是CMYK

92. //受到傷害後的無敵時間設為1秒,調到100或許就是所謂的超簡單遊戲

93. //結束無敵時間後記得將顏色條回全白,不然就會保持在紅色的狀態

人物受到傷害時給它顏色的改變可以讓玩家判斷自己現在受傷並且是否處於無敵的狀態，無敵時間結束時記得要將顏色改回來

94. public IEnumerator resetCanHurt () {

95. rend.color = new Vector4(1.0f,0.5f,0.5f,1.0f);

96. yield return new WaitForSeconds(1f);

97. rend.color = new Vector4(1.0f,1.0f,1.0f,1.0f);

98. canGetHurt = true;

99. }

100.

受到傷害更新血量並確認腳色是否死亡，角色死亡時要重新讀取關卡

101. //當受到傷害時確認血量

102. public IEnumerator checkHealth () {

103. //更新血量

104. updateHearts();

105. // 如果血量少於等於0並且不是死亡的狀態下宣告死亡成立

106. if(health <= 0 && dead == false){

107. dead = true;

108. //死亡動畫的位置

109. Instantiate(deathAnim, transform.position, Quaternion.Euler(0,180,0));

110. BroadcastMessage("died", SendMessageOptions.DontRequireReceiver);

111. //將2D渲染設為關閉

112. var rend = GetComponent<SpriteRenderer>();

113. rend.enabled = false;

114. //重新復活時間為2秒

115. yield return new WaitForSeconds(2);

116. //重新讀取關卡

117. string lvlName = Application.loadedLevelName;

118. Application.LoadLevel(lvlName);

119. }

120. }

121.

怪物會掉落愛心，碰撞時會吃下去並且＋２滴血也就是一格愛心並更新血量

122. //添加血量

123. void addHealth () {

124. GetComponent<AudioSource>().PlayOneShot(heartSound);

125. //吃到愛心所增加的血量

126. health += 2;

127. //如果吃的血量超過設定值就不要再增加了

128. if(health > 6){

129. health = 6;

130. }

131. //更新血量

132. updateHearts();

133. }

134.

135. //更新血量時要更新畫面上玩家的血量

136. void updateHearts () {

137. //將血量更新於GUI的血量條上

138. for(int i = 0;i < heartsGUI.Length;i++){

139. int check = (i+1)\*2;

140. //如果血格是滿的

141. if(check < health+1){

142. heartsGUI[i].texture = full;

143. }

144. //如果血格是半滿

145. if(check == health+1){

146. heartsGUI[i].texture = half;

147. }

148. //如果血格是空的

149. if(check > health+1){

150. heartsGUI[i].texture = empty;

151. }

152. }

153. }

154. }

155.

playerWeapons玩家武器

1. using UnityEngine;

2. using System.Collections;

3.

4. public class playerWeapons : MonoBehaviour {

5.

武器也就是玩家武器發射出來的子彈，子彈可在unity中調整的參數有它的傷害、速度、射擊速度等等

6. //找到子彈物件

7. public GameObject bullets;

8. //子彈的傷害

9. public float bulletsDamage = 1.0f;

10. //子彈速度

11. public float bulletsSpeed = 20.0f;

12. //子彈射速

13. public float bulletsFirerate = 0.3f;

14. //子彈重製位置

15. public Transform spawnPosition;

16. //射擊聲音

17. public AudioClip bulletSound;

18.

19. //各種子彈的變量

20. //計時

21. private float bulletCounter = 0.0f;

22. //位置

23. private float bulletPos = 0.0f;

24. //目前武器.速度.攻擊力

25. private GameObject currentBullet;

26. private float currentSpeed;

27. private float currentDamage;

28. //射速

29. private float fireRate = 0.25f;

30. private bool dead = false;

31.

32. void Start () {

程式開始時先更新子彈的參數設定值，抓取目前設定的攻擊力、速度、射速等等

33. updateBulletType();

34. }

35.

36. void Update () {

37. //計時了解甚麼時候可以發射

38. bulletCounter += Time.deltaTime;

39. //在未死亡的狀態下按下Z鍵是射擊子彈

40. if(!dead){

41. if(Input.GetKey(KeyCode.Z)){

42. if(bulletCounter > fireRate){

43. shootBullet();

44. }

45. }

46. }

47. }

48.

49. void shootBullet () {

50. //子彈發射的位置

51. Vector3 pos = new Vector3(bulletPos+transform.position.x,-0.40f+transform.position.y,0.01f+transform.position.z);

52. GameObject bulletPrefab = Instantiate(currentBullet, pos, Quaternion.Euler(0,180,0)) as GameObject;

53. bulletPrefab.SendMessage("getDamageAmount", currentDamage, SendMessageOptions.DontRequireReceiver);

54. //播放射擊音效

55. GetComponent<AudioSource>().PlayOneShot(bulletSound);

56. //分辨是朝左還是朝右射擊

57. if(spawnPosition.position.x > transform.position.x){

58. bulletPrefab.transform.GetComponent<Rigidbody>().velocity = new Vector3(currentSpeed,0,0);

59. }else{

60. bulletPrefab.transform.GetComponent<Rigidbody>().velocity = new Vector3(-currentSpeed,0,0);

61. }

62. bulletCounter = 0.0f;

63. }

64. //取得武器參數數值

65. void updateBulletType () {

66. currentBullet = bullets;

67. fireRate = bulletsFirerate;

68. currentSpeed = bulletsSpeed;

69. currentDamage = bulletsDamage;

70. }

71.

人物死亡時不能攻擊所以在武器的程式中也要宣告死亡

72. //如果死掉要將死亡調為成立

73. void died () {

74. dead = true;

75. }

76. }

playerAnimator玩家動畫

1. using UnityEngine;

2. using System.Collections;

3.

4. public class playerAnimator : MonoBehaviour {

5.

因為這是一款點陣圖遊戲所以直接用簡單的貼圖就能完成動畫，甚至許多動作都是一張貼圖，玩家的動畫一共有三種，Idle靜止不動、run走路、jump跳躍三種

6. //在這裡設置人物所會用到的動畫

7. public Sprite[] idle;

8. public float idleFrameRate = 8.0f;

9. public Sprite[] run;

10. public float runFrameRate = 8.0f;

11. public Sprite[] jump;

12. public float jumpFrameRate = 8.0f;

13. public float frameRate = 8;

14.

15. //各種人物動畫的變量

16. private CharacterController controller;

17. private float counter = 0.0f;

18. private int i = 0;

19. private SpriteRenderer rend;

20. private bool isJumping = false;

21.

22. void Start () {

23. controller = GetComponent<CharacterController>();

24. rend = GetComponent<SpriteRenderer>();

25. }

26.

利用X軸與Y軸的速度來判斷撥放的動畫，除了平面Ｘ軸以外也需要Ｙ軸的速度來判斷跳躍的動畫，如果Ｘ軸的速度接近０時撥放靜止不動，不是的話則是撥放走路的動畫，人物不是在地面就撥放跳躍的動畫

27. void Update () {

28. //檢查速度來判定腳色的動作

29. //因為需要平面移動與跳躍的判定所以需要X軸與Y軸的速度

30. float xVelocity = controller.velocity.x;

31. float yVelocity = controller.velocity.y;

32. if(xVelocity < 0){

33. xVelocity \*= -1;

34. }

35. //如果X軸速度趨近0並且是在地面時撥放idle

36. if(xVelocity < 0.25f){

37. if(controller.isGrounded){

38. counter += Time.deltaTime\*idleFrameRate;

39. if(counter > i && i < idle.Length){

40. rend.sprite = idle[i];

41. i += 1;

42. }

43. //讓動畫無限輪播

44. if(counter > idle.Length){

45. counter = 0.0f;

46. i = 0;

47. }

48. }

49. }else{

50. //同樣在地面速度不是趨近於0的話就是走動並撥放run

51. if(controller.isGrounded){

52. counter += Time.deltaTime\*runFrameRate;

53. if(counter > i && i < run.Length){

54. rend.sprite = run[i];

55. i += 1;

56. }

57. //讓動畫無限輪播

58. if(counter > run.Length){

59. counter = 0.0f;

60. i = 0;

61. }

62. }

63. }

64. //如果人物不是在地面

65. if(!controller.isGrounded){

66. //如果人物還不是jump的動畫以及Y軸速度大於設定值時播放jump動畫

67. if(!isJumping && yVelocity > 0.5f){

68. isJumping = true;

69. counter = 0.0f;

70. i = 0;

71. }

72. if(isJumping){

73. counter += Time.deltaTime\*jumpFrameRate;

74. if(counter > i && i < jump.Length){

75. rend.sprite = jump[i];

76. i += 1;

77. }

78. }

79. }

80. //如果人物回到了地面將停止jump動畫

81. if(controller.isGrounded){

82. isJumping = false;

83. }

84. }

85. }

86.

Bullet子彈

1. using UnityEngine;

2. using System.Collections;

3.

首先設置一個倒數的時間作為子彈飛行的時間，飛行時間到了以後要消除這個子彈物件

4. public class bullet : MonoBehaviour {

5. //子彈的飛行時間為一秒，如果時間太長或不消失的話子彈會一直線的飛行下去

6. public float bulletLife = 1.0f;

7. //計數器與變量

8. private float lifeCounter = 0.0f;

9. private float damage;

10.

11.

12. void Update () {

13. //加上計數器的變量

14. lifeCounter += Time.deltaTime;

15.

16. //如果計數器大於設定的秒速(目前設定為1秒)就要毀掉這個物件

17. if(lifeCounter > bulletLife){

18. Destroy(gameObject);

19. }

20. }

21.

22. void getBulletDamage (float amount) {

23. damage = amount;

24. }

25. //攻擊標記tag為enemy的怪物並且摧毀物件，如果把Destroy(gameObject);拿掉的話也可以做成貫通彈喔

26. void OnTriggerEnter (Collider other){

27. if(other.tag == "enemy"){

28. other.SendMessage("takeDamage", damage, SendMessageOptions.DontRequireReceiver);

29. Destroy(gameObject);

30. }

31. }

32. }

33.

cameraFollow攝影機追蹤

1. using UnityEngine;

2. using System.Collections;

3.

4. public class cameraFollow : MonoBehaviour {

攝影機追蹤程式會放到攝影機內，除了追蹤玩家的Ｘ軸與Ｙ軸外還有一個deadZone的設置，它是一個緩衝區域不讓攝影機跟得太緊，攝影機在追蹤時會先等玩家經過這一段距離後再開始追蹤

5. // 這個程式會附加到撥放器main Camera內，這裡死區deadZone設置為0，可以在unity內調到想要的效果

6. public GameObject target;

7.

8. public float deadZone = 0;

9. public bool followVertical = true;

10. public bool followHorizontal = true;

11. public float minimumHeight = 0;

12.

13. void Start () {

14. //讓攝影機追蹤玩家

15. if(target == null){

16. target = GameObject.Find("Player");

17. }

18. }

19.

20. void Update () {

21. if(target != null){

22. //檢查移動的水平方向與死區，移動的水平Ｘ軸速度也要將死區數值加減近來

23. if(followHorizontal == true){

24. if (transform.position.x >= target.transform.position.x + deadZone){

25. transform.position = new Vector3(target.transform.position.x+deadZone,transform.position.y,transform.position.z);

26. }

27. if (transform.position.x <= target.transform.position.x - deadZone){

28. transform.position = new Vector3(target.transform.position.x-deadZone,transform.position.y,transform.position.z);

29. }

30. }

31.

32. //檢查移動的垂直方向與死區，移動的垂直Ｙ軸速度也要將死區數值加減近來

33. if(followVertical == true){

34. if (transform.position.y >= target.transform.position.y + deadZone){

35. transform.position = new Vector3(transform.position.x, target.transform.position.y+deadZone, transform.position.z);

36. }

37. if (transform.position.y <= target.transform.position.y - deadZone){

38. transform.position = new Vector3(transform.position.x, target.transform.position.y-deadZone, transform.position.z);

39. }

40. }

41.

設定一個高度讓攝影機不會一直追到太下面（玩家掉落懸崖攝影機不會追下去）

42. //如果照相機碰到最低高度將不會再降低位置

43. if(target.transform.position.y < minimumHeight){

44. transform.position = new Vector3(transform.position.x, minimumHeight, transform.position.z);

45. }

46. }

47. }

48. }

deathAnimation死亡動畫

1. using UnityEngine;

2. using System.Collections;

3.

4. public class deathAnimation : MonoBehaviour {

5.

人物與怪物死亡的動畫都是用這個程式就能完成，只要將撥放的動畫以及音效使用不同素材的話就會有區別了，當然也可以放一樣的

6. //死亡動畫的程式，用來將角色及怪物死亡時撥放的動畫

7. public Sprite[] deathSprites;

8. //動畫撥放的速度

9. public float frameRate = 12.0f;

10. //音效

11. public AudioClip deathSound;

12. //設置計數器

13. private float counter = 0.0f;

14. private int i = 0;

15. private SpriteRenderer rend;

16.

17. void Start () {

18. rend = GetComponent<SpriteRenderer>();

19. //死亡時撥放死亡音效

20. GetComponent<AudioSource>().PlayOneShot(deathSound);

21. }

22.

23. void Update () {

24. //計數器來控制撥放的動畫與它的撥放速度

25. counter += Time.deltaTime\*frameRate;

26. if(counter > i && i < deathSprites.Length){

27. rend.sprite = deathSprites[i];

28. i += 1;

29. }

30. //當動畫撥放完後要消去這個物件

31. if(counter > deathSprites.Length){

32. Destroy(gameObject);

33. }

34. }

35. }

36.

enemyHealth怪物血量

1. using UnityEngine;

2. using System.Collections;

3.

4. public class enemyHealth : MonoBehaviour {

5.

怪物與玩家一樣需要血量，怪物死亡時除了會出現死亡動畫以外還要掉落愛心，怪物的血量也能夠在unity內做更改

6. //如果怪物死了要出現死亡動畫

7. public GameObject deathAnim;

8. //將怪物打死會掉落愛心

9. public GameObject heartDrop;

10. //怪物血量與打擊音效

11. public int health = 6;

12. public AudioClip hurtSound;

13.

14. private SpriteRenderer rend;

15. private bool isDead = false;

16.

17. void Start () {

18. rend = GetComponent<SpriteRenderer>();

19. }

20.

如果怪物不是在死亡的狀態下碰撞到子彈的話會扣一滴血，如果血量到零或小於零就要讓怪物死亡並且以一定機率掉落愛心

21. //當子彈擊中怪物時會受到傷害並撥放音效，當血量為０時將死亡設為成立撥放死亡動畫並設定一定範圍作為機率生產愛心，最後摧毀物件不然就是重置受傷顏色

22. void takeDamage (float amount) {

23. if(!isDead){

24. GetComponent<AudioSource>().PlayOneShot(hurtSound);

25. health -= 1;

26. if(health <= 0){

27. isDead = true;

28. Instantiate(deathAnim, transform.position, Quaternion.Euler(0,180,0));

29. int randNum = Random.Range(1,4);

30. if(randNum == 2){

31. Instantiate(heartDrop, transform.position, Quaternion.Euler(0,180,0));

32. }

33. Destroy(gameObject);

34. }else{

35. StartCoroutine(resetColor());

36. }

37. }

38. }

39.

40. //與玩家相同受到傷害時也會加上紅色來表示現在是受到傷害的狀態

41. public IEnumerator resetColor () {

42. rend.color = new Vector4(1.0f,0.25f,0.25f,1.0f);

43. yield return new WaitForSeconds (0.125f);

44. rend.color = new Vector4(1.0f,1.0f,1.0f,1.0f);

45. }

46. }

Facecrawl爬行顏文字怪物

1. using UnityEngine;

2. using System.Collections;

3.

4. public class facecrawl : MonoBehaviour {

5.

爬行顏文字怪物只有兩個動作，一是靜止不動二是爬行，在unity內可以變動的兩個變量也只有速度以及動畫

6. //爬行速度

7. public float crawlSpeed = 4.0f;

8. //貼圖動畫

9. public Sprite[] crawl;

10. //各種變量

11. private CharacterController controller;

12. private float counter = 0.0f;

13. private int i = 0;

14. private GameObject target;

15. private float frameRate = 8.0f;

16. private SpriteRenderer rend;

17. private float origX;

18. private Vector3 vel;

19.

20. //怪物的AI

21. void Start () {

22. //找到玩家並朝著玩家前進.移動為水平X軸

23. controller = GetComponent<CharacterController>();

24. rend = GetComponent<SpriteRenderer>();

25. target = GameObject.Find("Player");

26. Physics.IgnoreCollision(target.GetComponent<Collider>(), GetComponent<Collider>());

27. origX = transform.localScale.x;

為防讓每隻怪物都因箱型碰撞而卡住彼此所以要用標籤來讓怪物判別與其他怪物間的碰撞處理方式，也就是改成不會受到其他怪物的影響，這樣一來一快一慢的怪物就不會因為慢的走前面快的就被那隻怪物卡死了，當然也可以拿掉碰撞的處理玩玩看擠在一起的感覺

28. //忽略與其他怪物物件的碰撞.加入標籤"enemy"讓怪物來辨識

29. GameObject[] enemies = GameObject.FindGameObjectsWithTag("enemy");

30. foreach(GameObject en in enemies) {

31. if (en.GetComponent<Collider>() != GetComponent<Collider>()) {

32. Physics.IgnoreCollision(GetComponent<Collider>(), en.GetComponent<Collider>());

33. }

34. }

35. }

36.

怪物也需要重力，不然會橫向的飄在半空中

37. void Update () {

38. //重力

39. if(!controller.isGrounded){

40. vel.y -= Time.deltaTime\*80;

41. }else{

42. vel.y = -1;

43. }

44. //檢查玩家接近怪物之間的距離

45. float distance = target.transform.position.x - transform.position.x;

46. float ydistance = target.transform.position.y - transform.position.y;

47. if(distance < 0){

48. distance \*= -1;

49. }

50. if(ydistance < 0){

51. ydistance \*= -1;

52. }

53. if(target.transform.position.x > transform.position.x){

54. transform.localScale = new Vector3(origX,transform.localScale.y,transform.localScale.z);

55. }

56. if(target.transform.position.x < transform.position.x){

57. transform.localScale = new Vector3(-origX,transform.localScale.y,transform.localScale.z);

58. }

59. //如果距離在10~15之間.目標在左邊就設為負值向左走.右邊為正值向右走

60. if (distance < 15 && ydistance < 5) {

61. counter += Time.deltaTime \* frameRate;

62. if (target.transform.position.x < transform.position.x) {

63. vel.x = -crawlSpeed;

64. }

65. if (target.transform.position.x > transform.position.x) {

66. vel.x = crawlSpeed;

67. }

68. if (counter > i && i < crawl.Length) {

69. rend.sprite = crawl [i];

70. i += 1;

71. }

72. if (counter > crawl.Length) {

73. counter = 0.0f;

74. i = 0;

75. }

76. }

77. //怪物掉落到限制高度要消失掉

78. if(transform.position.y < -10){

79. Destroy(gameObject);

80. }

81. //應用移動

82. controller.Move(vel\*Time.deltaTime);

83. }

84.

85.

86. }

heartAdder血量圖式

1. using UnityEngine;

2. using System.Collections;

3.

4. public class heartAdder : MonoBehaviour {

5. //在遊戲介面上顯示玩家血量，也可以在unity內調整位置

6. public GameObject heartGUI;

7. public Vector2 startPoint = new Vector2(0.03f,0.95f);

8. public float distance = 0.03f;

9.

10. void addHearts (int amount){

11. for(int i = 0;i < amount;i++){

12. Vector3 pos = new Vector3(startPoint.x+(distance\*i), startPoint.y,0);

13. GameObject heartPrefab = Instantiate(heartGUI, pos, Quaternion.Euler(0,0,0)) as GameObject;

14. heartPrefab.transform.name = "heart"+(i+1).ToString();

15. heartPrefab.transform.parent = transform;

16. }

17. }

18. }

heartDrop掉落的愛心

1. using UnityEngine;

2. using System.Collections;

3.

只是普通出現一顆愛心感覺不太好所以就以亂數的方式拋出一顆愛心

4. public class heartDrop : MonoBehaviour {

5. //給它一點力量並且以亂數的方式掉落

6. void Start () {

7. GetComponent<Rigidbody>().velocity = new Vector3(Random.Range(-6,6),Random.Range(4,8),0);

8. }

9. }

Menu選單

這隻程式是放在選單menu的GUI內，在ＵＩ介面內放入按鈕可以在這裡指定它們的功用，按下按鈕發送訊息給GUI做出反應

1. using UnityEngine;

2. using System.Collections;

3.

4. public class menu : MonoBehaviour {

5.

6. private bool canContinue = true;

7.

8. void Start () {

9. string checkLevelName = PlayerPrefs.GetString("savedLevel");

10. if(checkLevelName == null || checkLevelName == ""){

11. canContinue = false;

12. }

13. }

選單內有兩個按鈕，一個是開始遊戲一個是繼續遊戲，按下開始遊戲將訊息startGame傳入menu中做出進入場景level1的動作，另外繼續遊戲則是去向目前遊戲儲存玩家所在的場景名稱

14. //按鈕內sendMessageUp輸入startGame後按下按鈕會到Level1

15. void startGame () {

16. PlayerPrefs.DeleteAll();

17. Application.LoadLevel("level1");

18. }

19.

20. //按鈕內sendMessageUp輸入continueGame後按下按鈕會到儲存的關卡

21. void continueGame () {

22. if(canContinue){

23. string levelName = PlayerPrefs.GetString("savedLevel");

24. Application.LoadLevel(levelName);

25. }

26. }

27. }

mouseButton滑鼠按鈕

1. using UnityEngine;

2. using System.Collections;

3.

4. public class mouseButton : MonoBehaviour {

5.

將物件轉換為對滑鼠點子有反應的程式，並且按下按鈕時傳出訊息交給menu做反應，所以除了要讓物件能夠用滑鼠點擊外也要傳出訊息

6. //設定一個輸入欄位告知按鈕的作用

7. public string sendMessageUp = "";

8. public bool standalone = true;

9.

10.

11. private bool over = false;

12.

13. void Start () {

14. if(!standalone){

15. gameObject.SetActive(false);

16. }

17.

18. }

一般在操作滑鼠時的反應是在圖示上點擊並在圖示上放開才是選擇該選項，如果不小心按錯的話則會將點擊中的滑鼠移出圖示放開，所以分為OnMouseEnter以及OnMouseExit兩種判斷

19. //點擊按鈕在按鈕上放開滑鼠代表是點擊

20. void OnMouseEnter () {

21. over = true;

22. }

23. //點擊按鈕在按鈕外放開滑鼠代表不是點擊

24. void OnMouseExit () {

25. over = false;

26. }

27.

28. void Update () {

29. if(Input.GetMouseButtonUp(0)){

30. if(over){

31. SendMessageUpwards(sendMessageUp, SendMessageOptions.DontRequireReceiver);

32. }

33. }

34. }

35. }

levelManager關卡管理器

1. using UnityEngine;

2. using System.Collections;

3.

4. public class levelManager : MonoBehaviour {

5.

6. public string nextLevel;

7.

除了碰到圖案進入下一個設定的關卡以外也要儲存遊戲進度，讓玩家可以在下次遊玩時繼續遊玩上次到達的關卡

8. void Start () {

9. //保存進度讓玩家在下次遊玩時可以選取繼續來遊玩上次到達的關卡

10. PlayerPrefs.SetString("savedLevel", Application.loadedLevelName);

11. }

12.

13. //檢查是否碰到玩家，如果碰到標籤為Player的物件時要前進至下一個設定的關卡

14. void OnTriggerEnter (Collider other){

15. if(other.tag == "Player"){

16. Application.LoadLevel(nextLevel);

17. }

18. }

19. }

Redguy紅色的傢伙

1. using UnityEngine;

2. using System.Collections;

3.

紅色怪物與爬行臉怪物的程式很像，同樣是會發現玩家並且以水平的方式朝著玩家前進，但是它多了武器的部分所以會多一些設定

4. public class redguy : MonoBehaviour {

5.

6. //走速

7. public float runSpeed = 4.0f;

8. //射擊音效

9. public AudioClip shootSound;

10. //貼圖動畫

11. public Sprite[] run;

12. public Sprite[] shoot;

13. //射擊子彈

14. public GameObject enemyBullet;

15. public float bulletSpeed = 16.0f;

16. public Transform bulletPosition;

17.

18. //各種變量

19. private CharacterController controller;

20. private float counter = 0.0f;

21. private int i = 0;

22. private GameObject target;

23. private float frameRate = 8.0f;

24. private bool shooting = false;

25. private SpriteRenderer rend;

26. private float origX;

27. private Vector3 vel;

28.

29. //怪物的AI

30. void Start () {

31. //找到玩家並朝著玩家前進.移動為水平X軸

32. controller = GetComponent<CharacterController>();

33. rend = GetComponent<SpriteRenderer>();

34. target = GameObject.Find("Player");

35. Physics.IgnoreCollision(target.GetComponent<Collider>(), GetComponent<Collider>());

36. origX = transform.localScale.x;

37. //忽略與其他怪物物件的碰撞

38. GameObject[] enemies = GameObject.FindGameObjectsWithTag("enemy");

39. foreach(GameObject en in enemies) {

40. if (en.GetComponent<Collider>() != GetComponent<Collider>()) {

41. Physics.IgnoreCollision(GetComponent<Collider>(), en.GetComponent<Collider>());

42. }

43. }

44. }

45.

46. void Update () {

47. //重力

48. if(!controller.isGrounded){

49. vel.y -= Time.deltaTime\*80;

50. }else{

51. vel.y = -1;

52. }

53. //檢查玩家接近怪物之間的距離

54. float distance = target.transform.position.x - transform.position.x;

55. float ydistance = target.transform.position.y - transform.position.y;

56. if(distance < 0){

57. distance \*= -1;

58. }

59. if(ydistance < 0){

60. ydistance \*= -1;

61. }

62. if(target.transform.position.x > transform.position.x){

63. transform.localScale = new Vector3(origX,transform.localScale.y,transform.localScale.z);

64. }

65. if(target.transform.position.x < transform.position.x){

66. transform.localScale = new Vector3(-origX,transform.localScale.y,transform.localScale.z);

67. }

68.

首先Ｘ軸的距離小於15與Ｙ軸距離小於8時會開始動作，第一段距離是Ｘ軸的距離再10~15之間尚未射擊的狀況下要步行移動，如果距離到了10以下才會開始射擊

69. //如果接近至設定距離將開始攻擊並撥放攻擊動畫

70. if(distance < 15 && ydistance < 8){

71. counter += Time.deltaTime\*frameRate;

72. //如果距離在10~15之間又沒有在射擊的狀態下.目標在左邊就設為負值向左走.右邊為正值向右走

73. if(distance > 10 && distance < 15 && !shooting){

74. if(target.transform.position.x < transform.position.x){

75. vel.x = -runSpeed;

76. }

77. if(target.transform.position.x > transform.position.x){

78. vel.x = runSpeed;

79. }

80. if(counter > i && i < run.Length){

81. rend.sprite = run[i];

82. i += 1;

83. }

84. if(counter > run.Length){

85. counter = 0.0f;

86. i = 0;

87. }

88. }

89. if(shooting){

90. vel.x = 0;

91. if(counter > i && i < shoot.Length){

92. rend.sprite = shoot[i];

93. i += 1;

94. }

95. }

96. //當距離足夠時才會發射子彈

97. if(distance < 10 && shooting == false){

98. StartCoroutine(shootBullet());

99. }

100. }

101.

102. //怪物掉落到限制高度要消失掉

103. if(transform.position.y < -10){

104. Destroy(gameObject);

105. }

106.

107. //應用移動

108. controller.Move(vel\*Time.deltaTime);

109. }

110.

111. //怪物發射子彈將會應用這個函數

112. public IEnumerator shootBullet () {

113. vel.x = 0;

114. shooting = true;

115. counter = 0.0f;

116. i = 0;

117. //發射前會稍微等待一下下讓玩家注意到

118. yield return new WaitForSeconds(0.5f);

119. //撥放射擊音效

120. GetComponent<AudioSource>().PlayOneShot(shootSound);

121. //產生子彈

122. GameObject bullet = Instantiate(enemyBullet, bulletPosition.position, Quaternion.Euler(0,0,0)) as GameObject;

123. //設置速度與子彈.如果是左邊就設為負值.右邊為正

124. if(bulletPosition.position.x < transform.position.x){

125. bullet.GetComponent<Rigidbody>().velocity = new Vector3(-bulletSpeed,0,0);

126. }else{

127. bullet.GetComponent<Rigidbody>().velocity = new Vector3(bulletSpeed,0,0);

128. }

129. //發射完後也要有空檔準備再次觸發射擊條件

130. yield return new WaitForSeconds(0.5f);

131. shooting = false;

132. }

133. }

enemyBullet怪物專用子彈

1. using UnityEngine;

2. using System.Collections;

3.

4. public class enemyBullet : MonoBehaviour {

5.

與玩家的子彈很像但是它可以直接設定攻擊力以及它的目標是Player

6. //設定子彈的威力

7. public int damage = 1;

8. //子彈的飛行時間為一秒，如果時間太長或不消失的話子彈會一直線的飛行下去

9. public float bulletLife = 1.0f;

10. //計數器與變量

11. private float lifeCounter = 0.0f;

12.

13.

14. void Update () {

15. //加上計數器的變量

16. lifeCounter += Time.deltaTime;

17.

18. //如果計數器大於設定的秒速(目前設定為1秒)就要毀掉這個物件

19. if(lifeCounter > bulletLife){

20. Destroy(gameObject);

21. }

22. }

23.

24. //攻擊標記tag為Player的玩家並且摧毀物件

25. void OnTriggerEnter (Collider other){

26. if(other.tag == "Player"){

27. other.SendMessage("takeDamage", damage, SendMessageOptions.DontRequireReceiver);

28. Destroy(gameObject);

29. }

30. }

31. }

32.