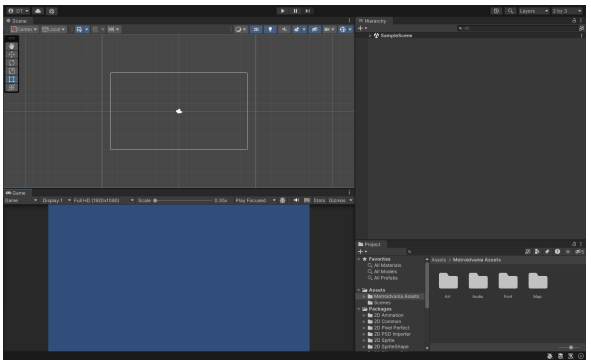
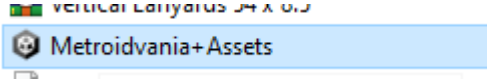
Project MV: Unit 1 IMAGES

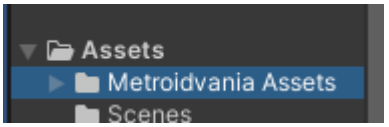
*This doc will include steps and helpful screenshots

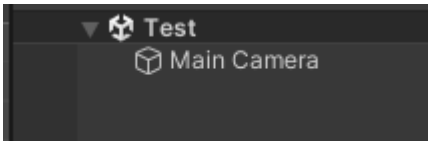
Lesson 1: Intro & SetUp

;



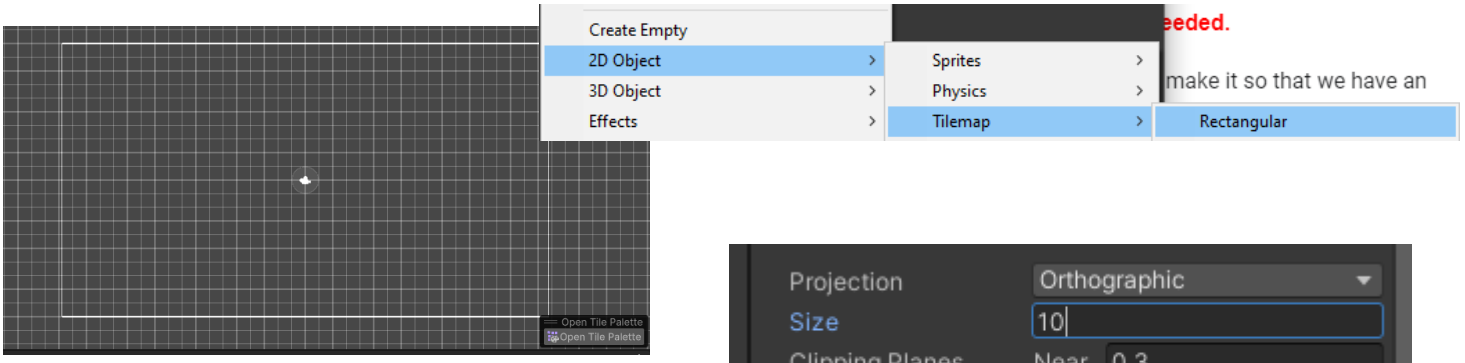


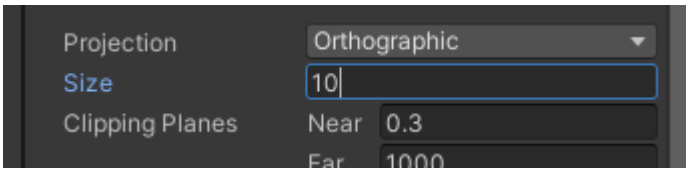


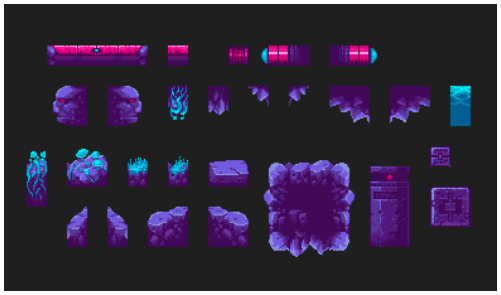


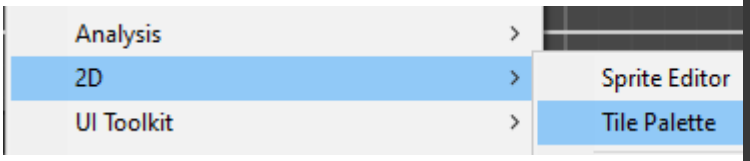
;;;

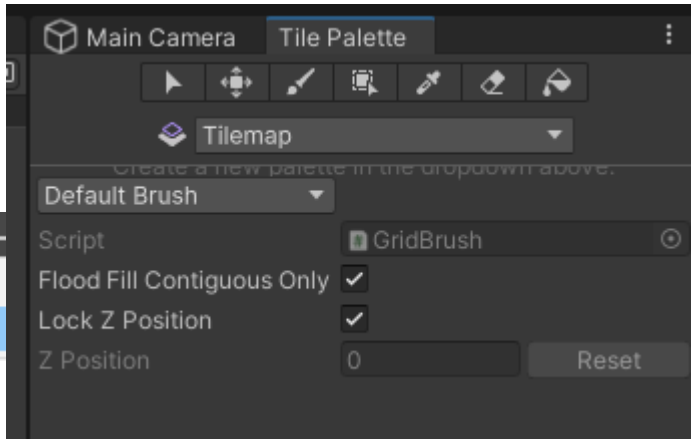
TILEMAP

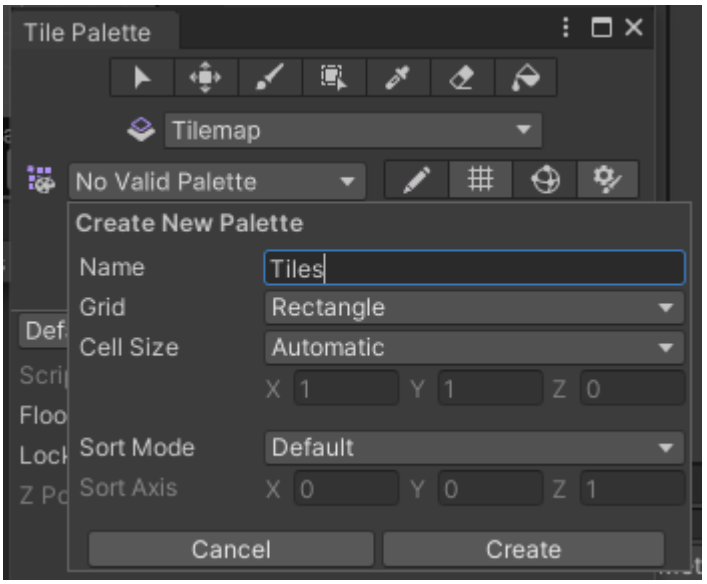


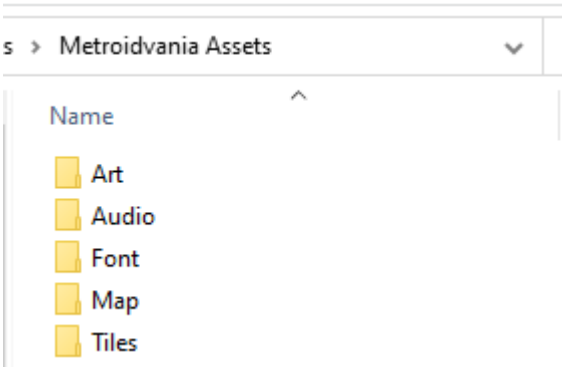


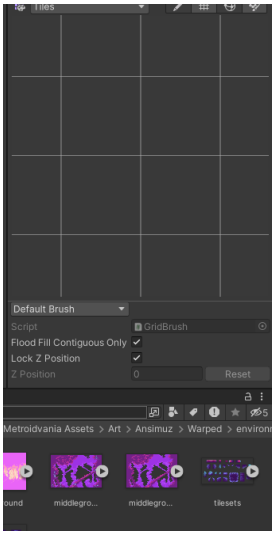


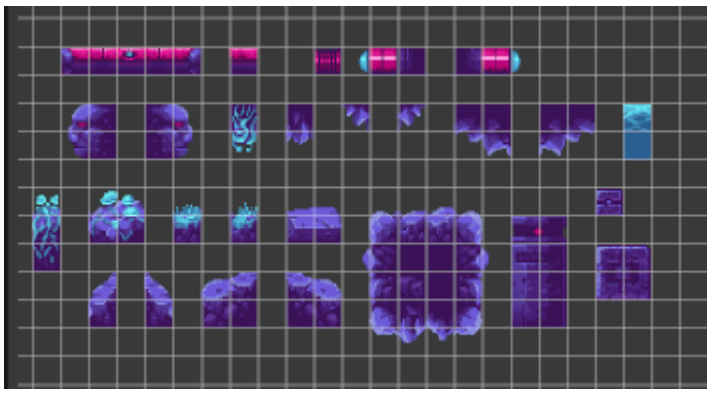


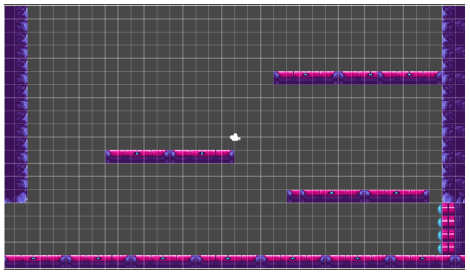


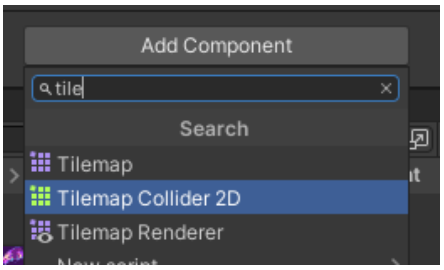

Tile Palette
Tilemap
No Valid Palette
Create New Palette
Name
Tiles
Grid
Rectangle
Cell Size
Automatic
X 1
Y 1
Z 0
Sort Mode
Default
Sort Axis
X 0
Y 0
Z 1
Cancel
Create

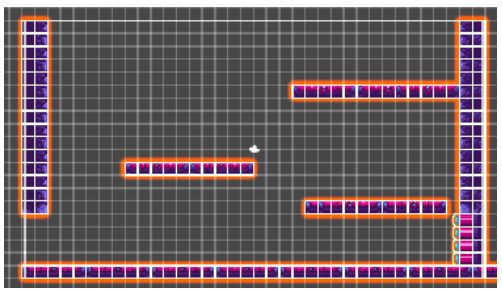

Metroidvania Assets
Name
Art
Audio
Font
Map
Tiles

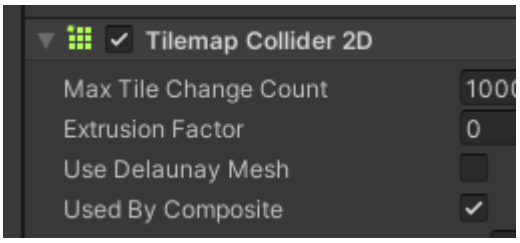

Default Brush
Script
GridBrush
Flood Fill Contiguous Only
Lock Z Position
Z Position
Reset

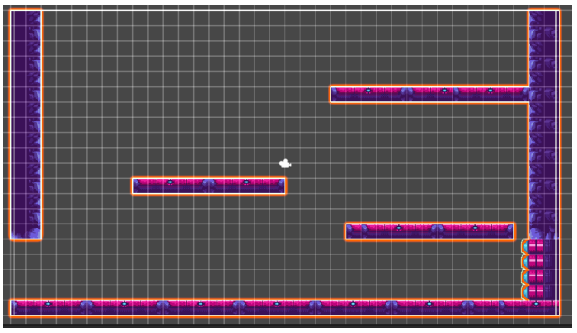


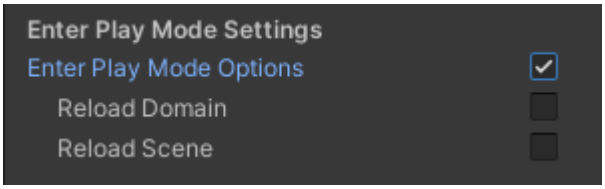



Add Component
tile
Search
Tilemap
Tilemap Collider 2D
Tilemap Renderer

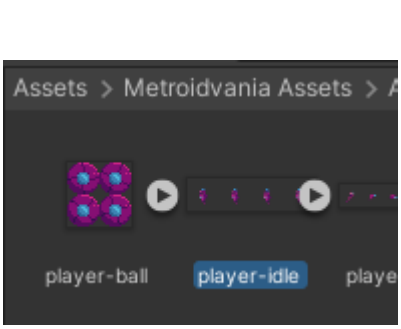


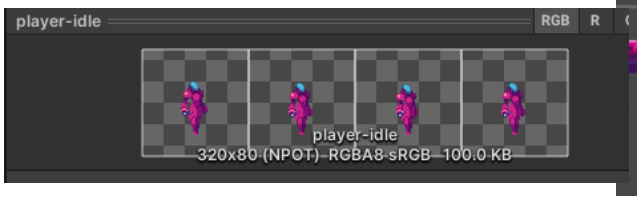


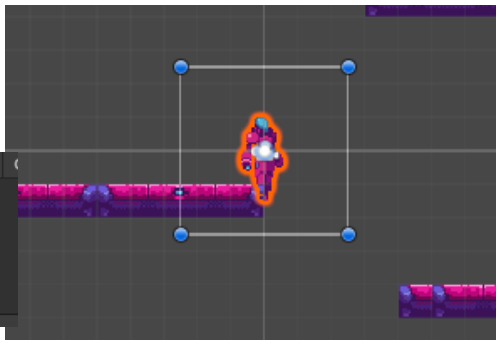


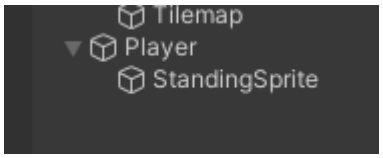


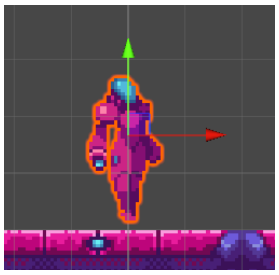
Lesson 2: The Player

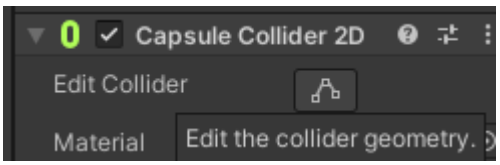


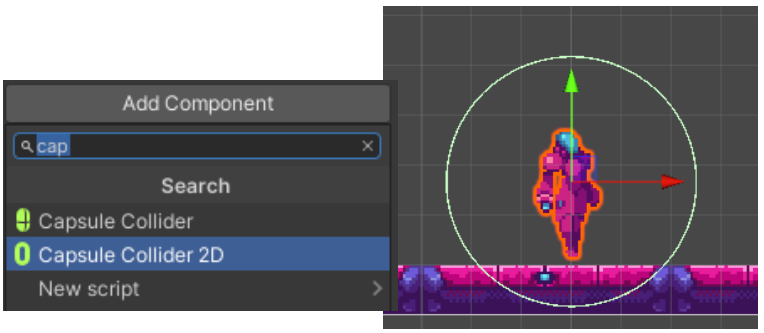


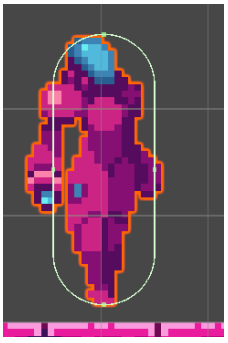


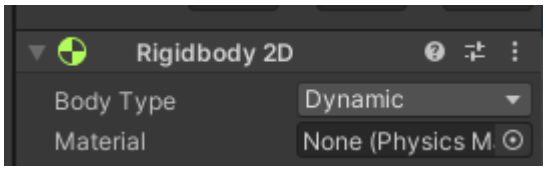


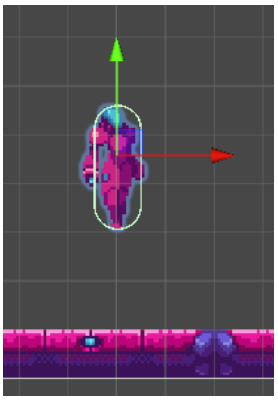


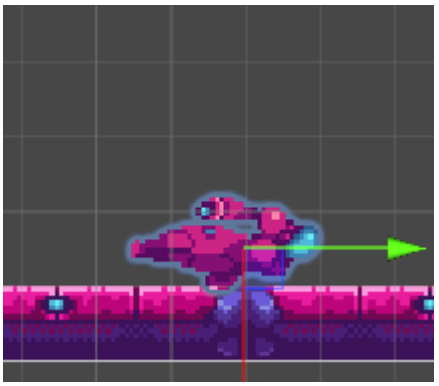


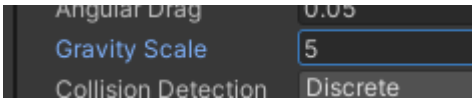




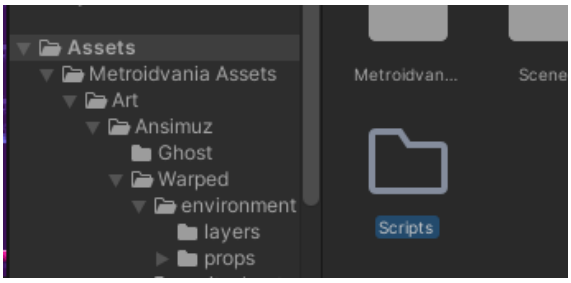


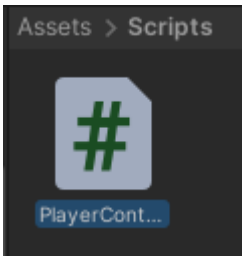


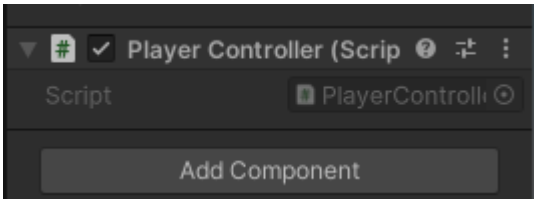




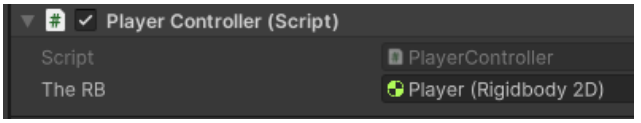
MOVING THE PLAYER








```
5   public class PlayerController : MonoBehaviour
6   {
7       public Rigidbody2D theRB;
```

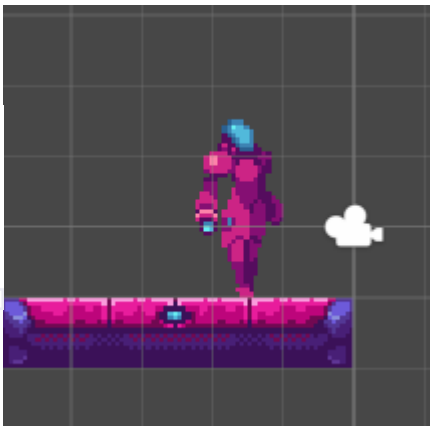


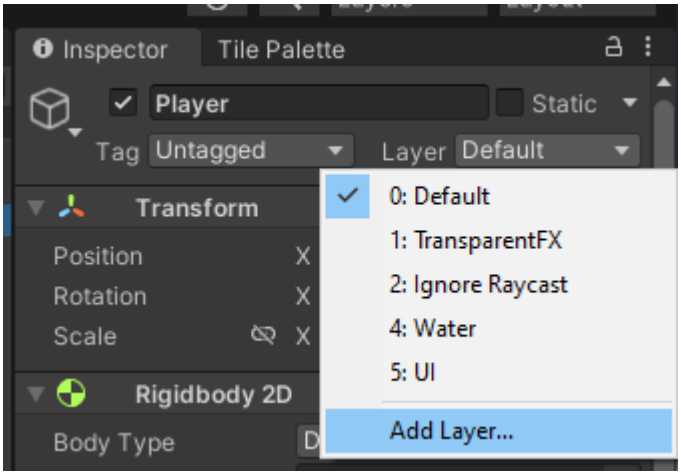
```
8
9       public float moveSpeed;
10      public float jumpForce;
```

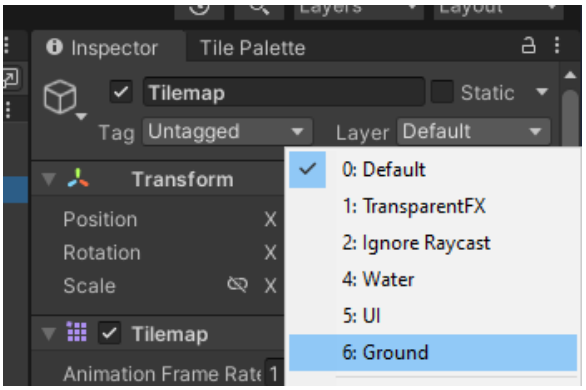
```
17      // Update is called once per frame
18      void Update()
19      {
20          theRB.velocity = new Vector2(Input.GetAxisRaw("Horizontal") * moveSpeed, theRB.velocity.y);
21
22      }
23  }
```

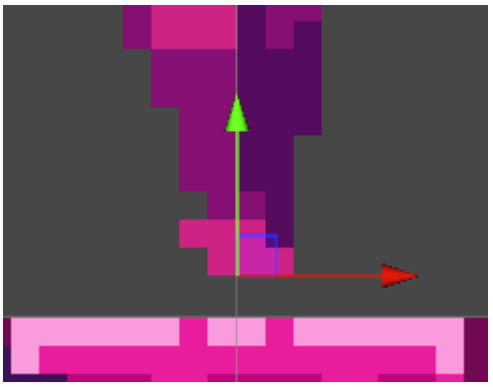
JUMPING -

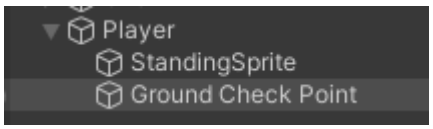
```
        theRB.velocity = new Vector2(Input.GetAxisRaw("Horizontal") * moveSpeed, theRB.velocity.y);

        if(Input.GetButtonDown("Jump"))
        {
            theRB.velocity = new Vector2(theRB.velocity.x, jumpForce);
        }
    }
}
```











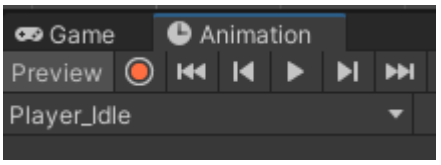

```
    public Transform groundPoint;
    private bool isOnGround;
    public LayerMask whatIsGround;
```

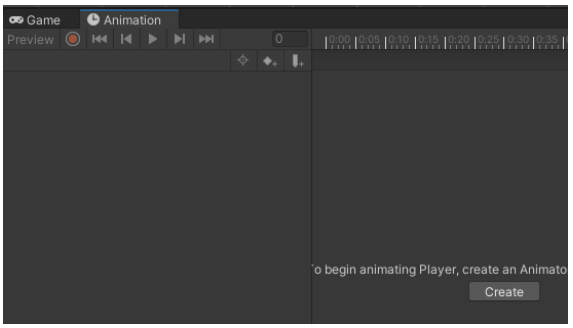
*changed to dark mode here so kids can see easier

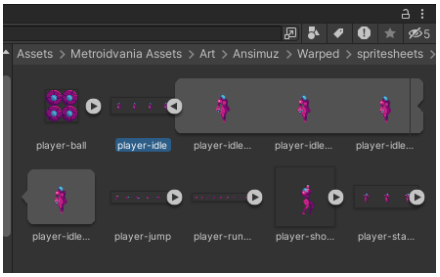
```
23      void Update()
24      {
25          theRB.velocity = new Vector2(Input.GetAxisRaw("Horizontal") * moveSpeed, theRB.velocity.y);
26
27          isOnGround = Physics2D.OverlapCircle(groundPoint.position, .2f, whatIsGround);
```

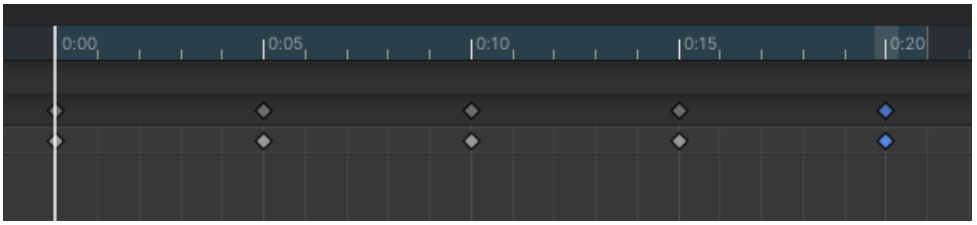
```
27          isOnGround = Physics2D.OverlapCircle(groundPoint.position, .2f, whatIsGround);
28
29          if(Input.GetButtonDown("Jump") && isOnGround)
30          {
31              theRB.velocity = new Vector2(theRB.velocity.x, jumpForce);
32          }
33      }
34  }
```

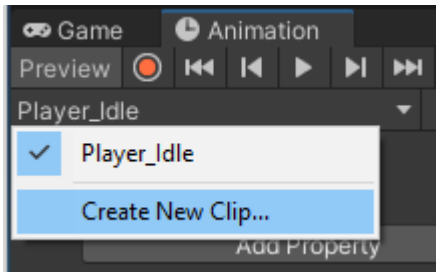
Animations -

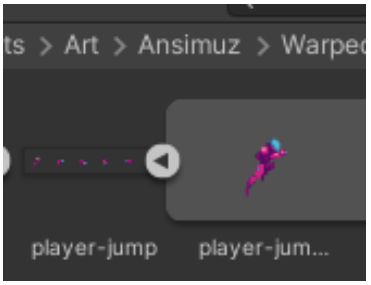


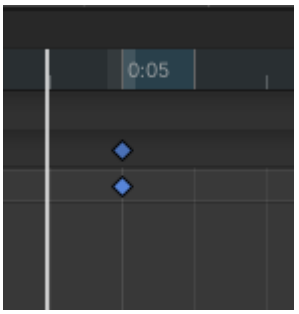


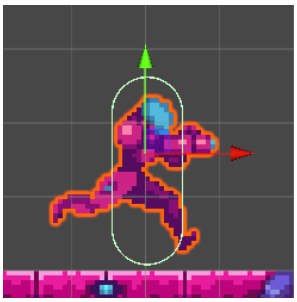


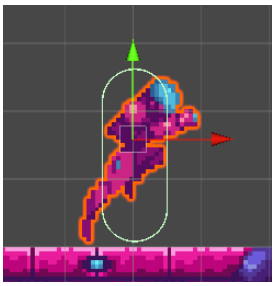


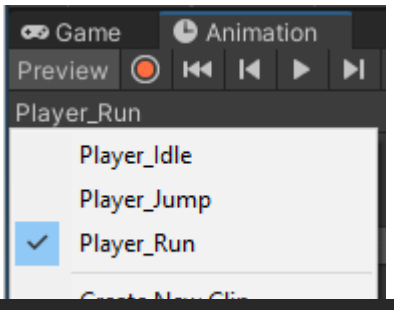


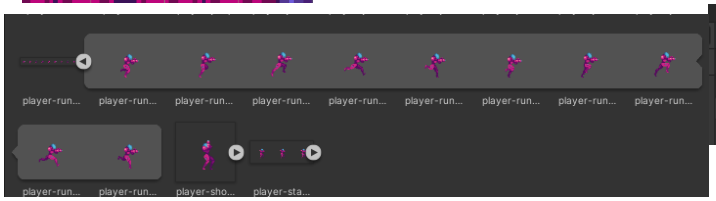


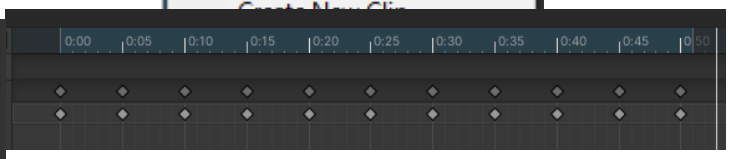




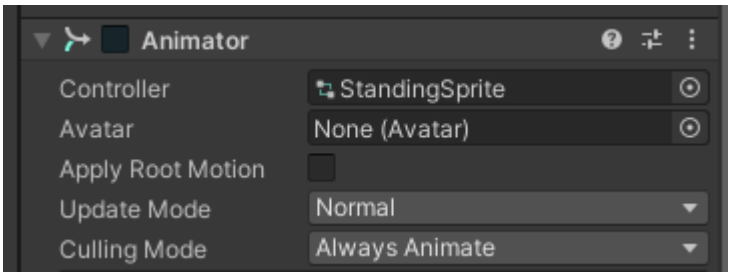


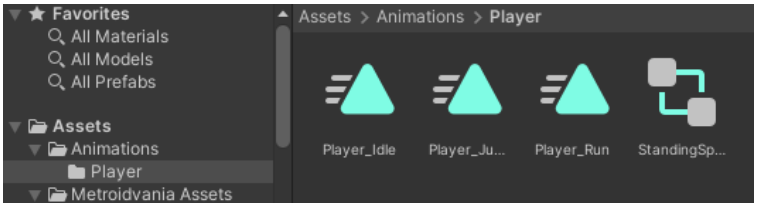


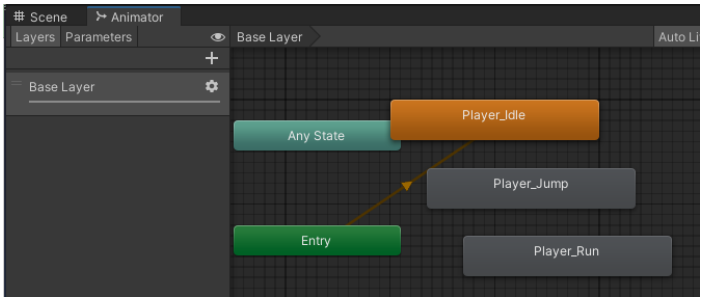


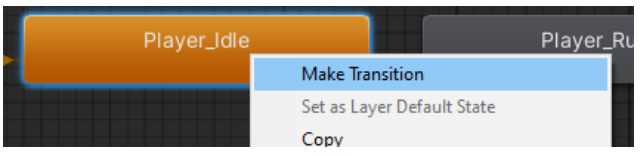


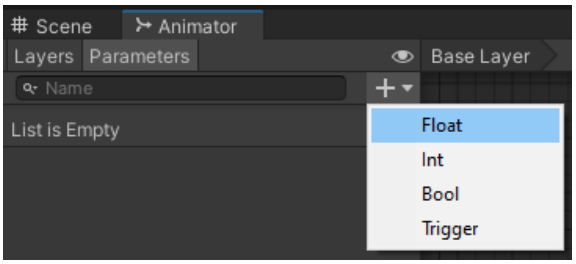
Controlling the animations

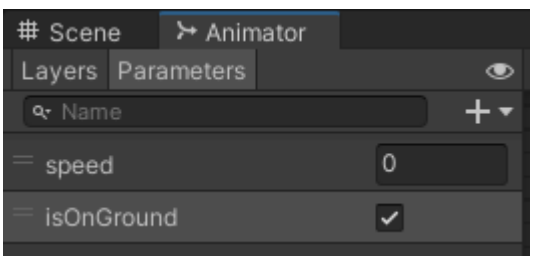


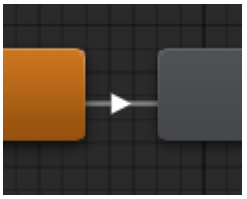


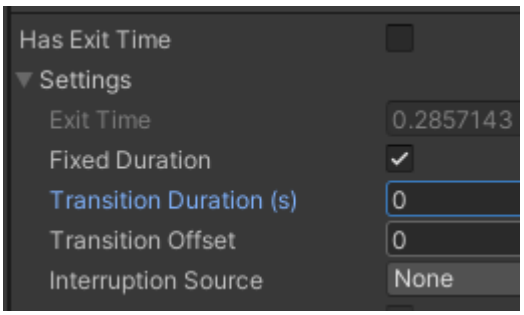


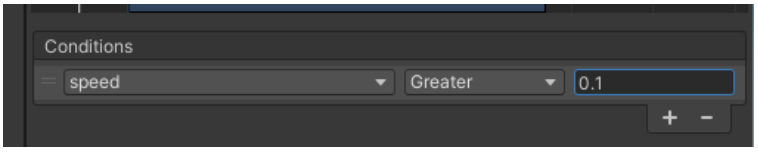


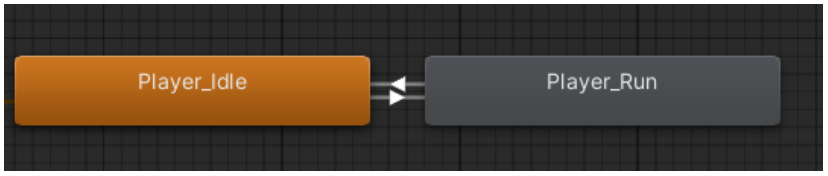


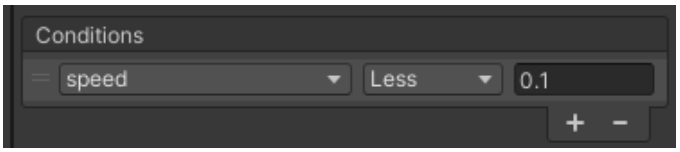


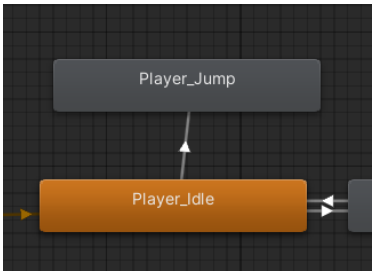


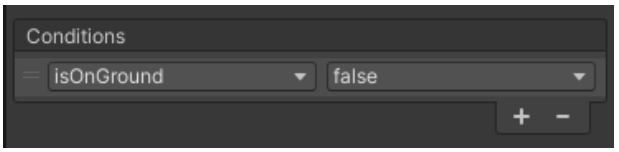


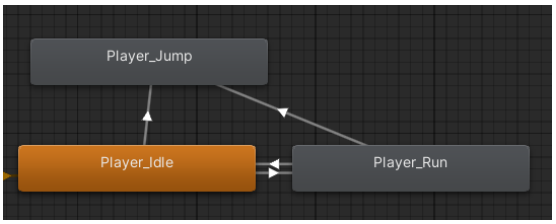


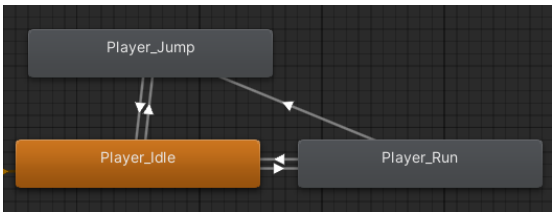


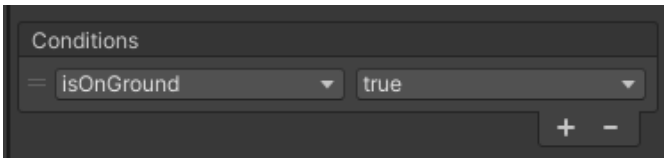












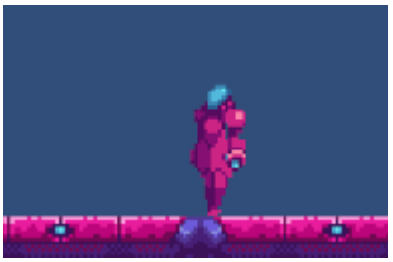

```
12      public Transform groundPoint;
13      private bool isOnGround;
14      public LayerMask whatIsGround;
15
16      public Animator anim;
17
```

```
41
42          anim.SetBool("isOnGround", isOnGround);
43      }
44  }
45
```

```
        anim.SetBool("isOnGround", isOnGround);
        anim.SetFloat("speed", Mathf.Abs(theRB.velocity.x));
    }
}
```

FLIPPING THE PLAYER

```
{
    // move sideways
    theRB.velocity = new Vector2(Input.GetAxisRaw("Horizontal") * moveSpeed, theRB.velocity.y);



    // checking if on ground
    isOnGround = Physics2D.OverlapCircle(groundPoint.position, .2f, whatIsGround);
```

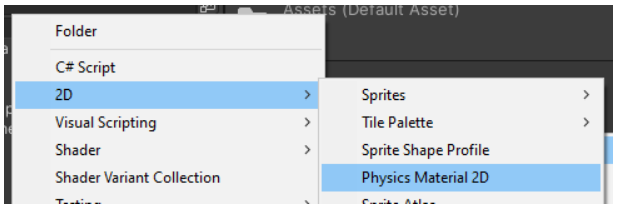


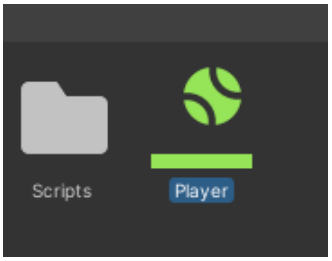
```
    // handle direction change
    if (theRB.velocity.x < 0)
    {
        transform.localScale = new Vector3(-1f, 1f, 1f);
```

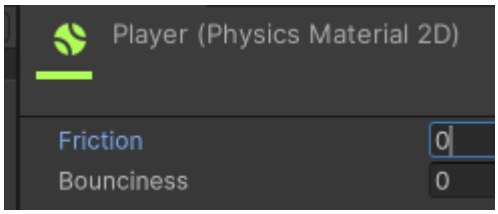
```
    // handle direction change
    if (theRB.velocity.x < 0)
    {
        transform.localScale = new Vector3(-1f, 1f, 1f);
    } else if(theRB.velocity.x > 0)
    {
        transform.localScale = Vector3.one;
    }
```

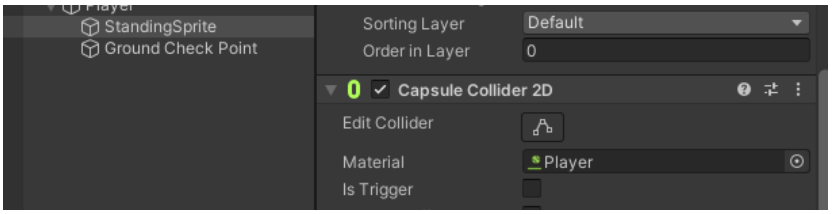
```
    // handle direction change
    if (theRB.velocity.x < 0)
    {
        transform.localScale = new Vector3(-1f, 1f, 1f);
    } else if(theRB.velocity.x > 0)
    {
        transform.localScale = Vector3.one;
    }

    // checking if on ground
    isOnGround = Physics2D.OverlapCircle(groundPoint.position, .2f, whatIsGround);

    // jumping
    if(Input.GetButtonDown("Jump") && isOnGround)
    {
        theRB.velocity = new Vector2(theRB.velocity.x, jumpForce);
    }
```

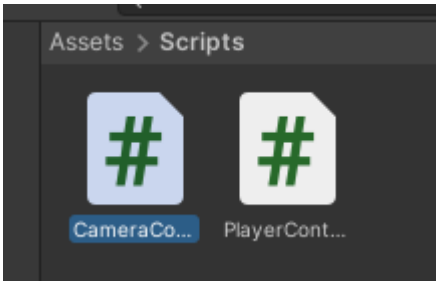
LESSON 3: CAMERA CONTROL

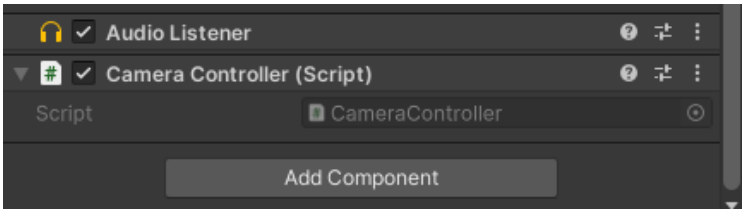






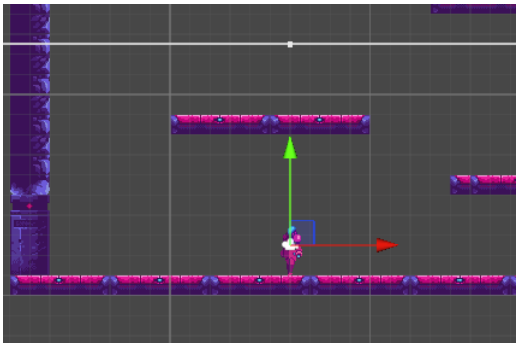






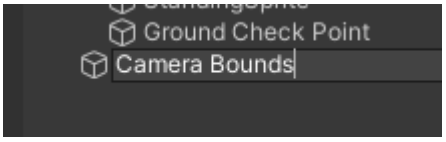

```
5   public class CameraController : MonoBehaviour
6   {
7
8       private PlayerController player;
```

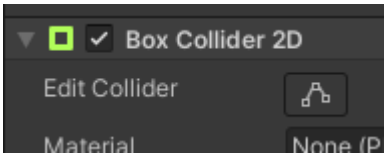
```
10      // Start is called before the first frame update
11      void Start()
12      {
13          player = FindObjectOfType<PlayerController>();
14
15      }
```

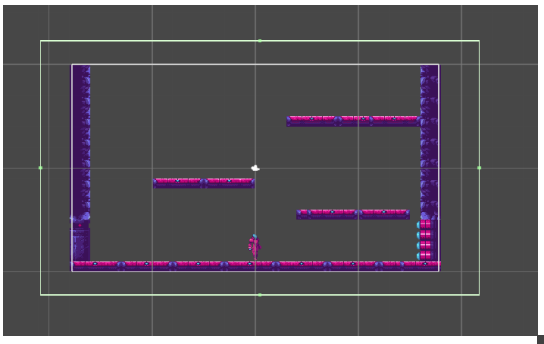


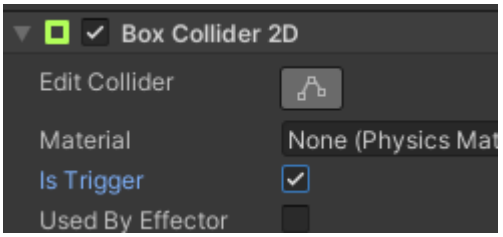
```
17      // Update is called once per frame
18      void Update()
19      {
20          if(player != null)
21          {
22              transform.position = new Vector3(player.transform.position.x, player.transform.position.y, transform.position.z) ;
23          }
24
25      }
26  }
```

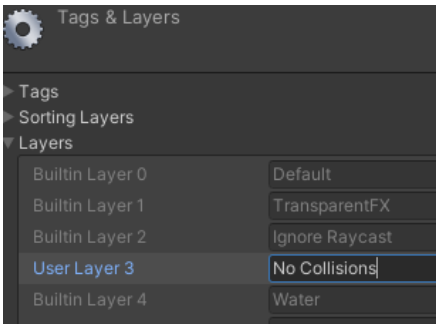
CLAMPING

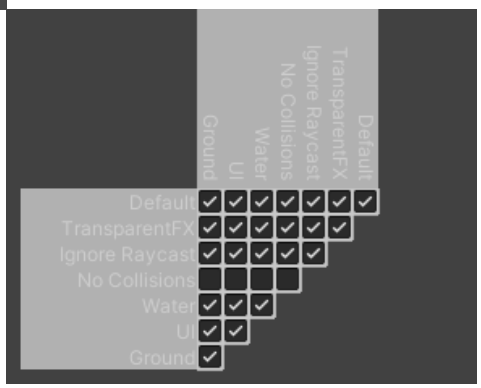






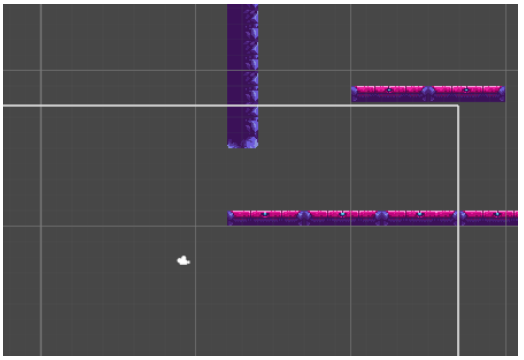







```
8       private PlayerController player;
9       public BoxCollider2D boundsBox;
```

```
19  void Update()
20  {
21      if(player != null)
22      {
23          transform.position = new Vector3(
24              Mathf.Clamp(player.transform.position.x, boundsBox.bounds.min.x, boundsBox.bounds.max.x),
25              Mathf.Clamp(player.transform.position.y, boundsBox.bounds.min.y, boundsBox.bounds.max.y),
26              transform.position.z);
27      }
28
29  }
```

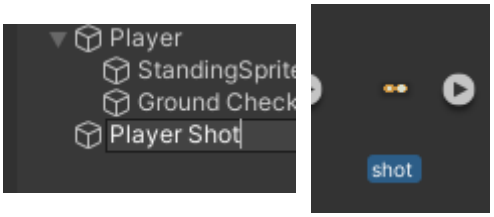


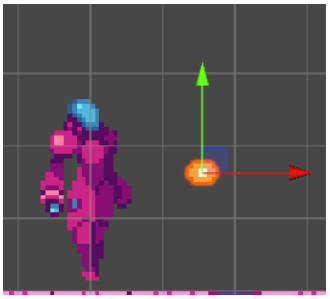
```
8       private PlayerController player;
9       public BoxCollider2D boundsBox;
10
11      private float halfHeight, halfWidth;
```

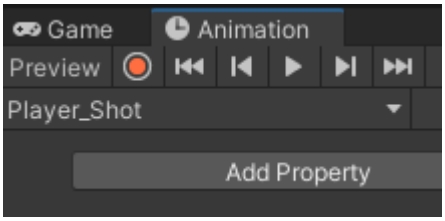
```
14      void Start()
15      {
16          player = FindObjectOfType<PlayerController>();
17
18          halfHeight = Camera.main.orthographicSize;
19          halfWidth = halfHeight * Camera.main.aspect;
20      }
```

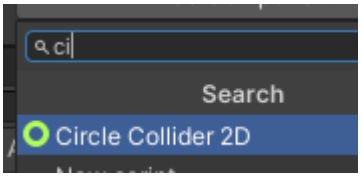
```
    // Update is called once per frame
    void Update()
    {
        if(player != null)
        {
            transform.position = new Vector3(
                Mathf.Clamp(player.transform.position.x, boundsBox.bounds.min.x  + halfWidth, boundsBox.bounds.max.x - halfWidth),
                Mathf.Clamp(player.transform.position.y, boundsBox.bounds.min.y + halfHeight, boundsBox.bounds.max.y - halfHeight),
                transform.position.z);
        }

    }
}
```

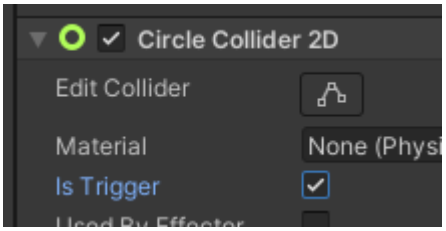
Lesson 4: Firing Mechanics

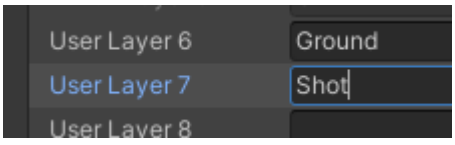


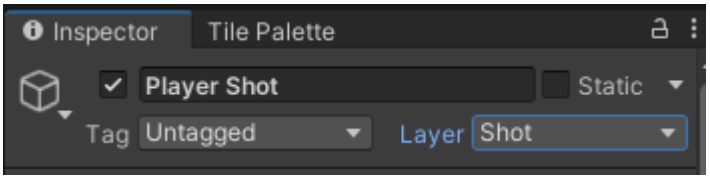


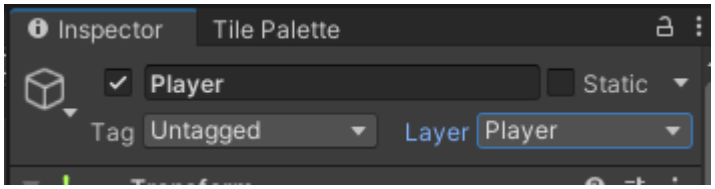


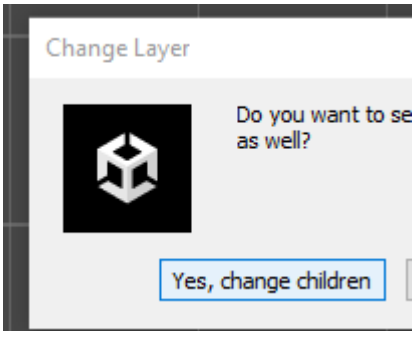


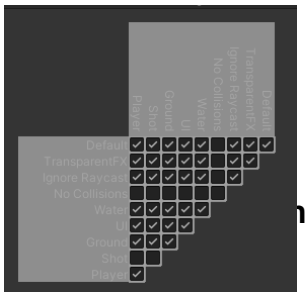


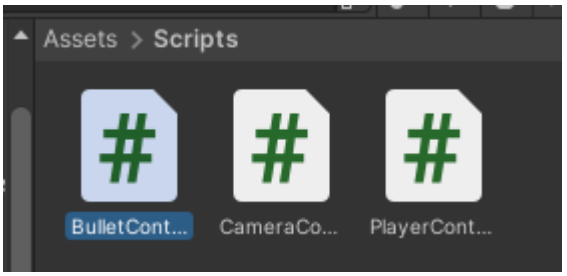


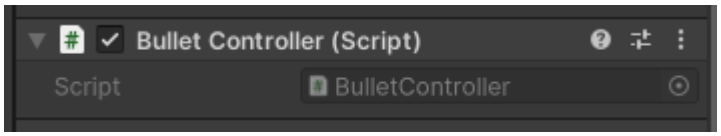




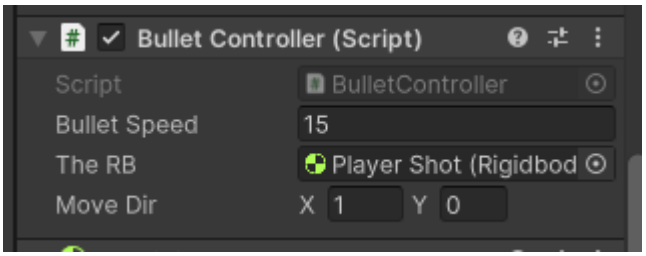









```
6   {
7       public float bulletSpeed;
8       public Rigidbody2D theRB;
9
10      public Vector2 moveDir;
11
```



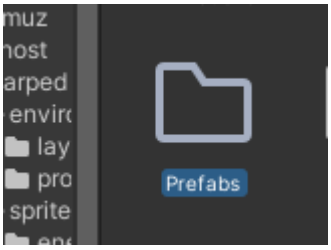

```
19      void Update()
20      {
21          theRB.velocity = moveDir * bulletSpeed;
22      }
23  }
```

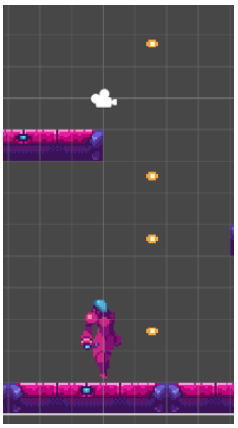
```
19      private void OnTriggerEnter2D(Collider2D other)
20      {
21          Destroy(gameObject);
22      }
```

```
24      private void OnBecameInvisible()
25      {
26          Destroy(gameObject);
27      }
28  }
```

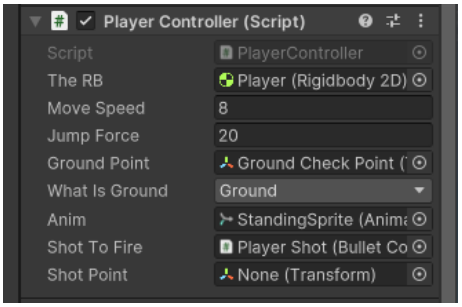
```
51
52          if(Input.GetButtonDown("Fire1"))
53          {
54              Instantiate(shotToFire, shotPoint.position, shotPoint.rotation);
55          }
```

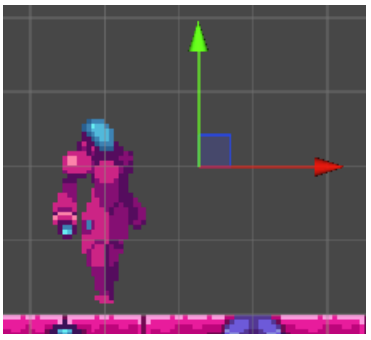
FIRING SHOTS

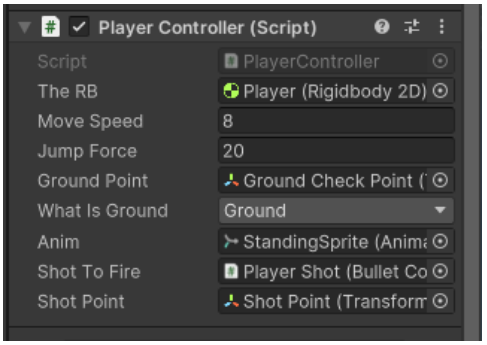




```
17
18      public BulletController shotToFire;
19      public Transform shotPoint;
20
```



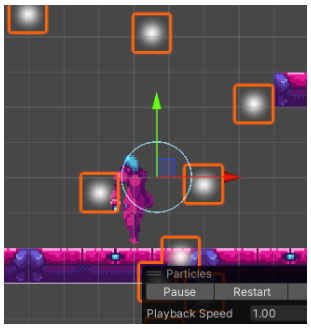


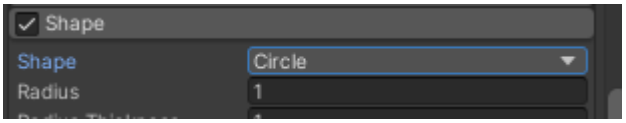


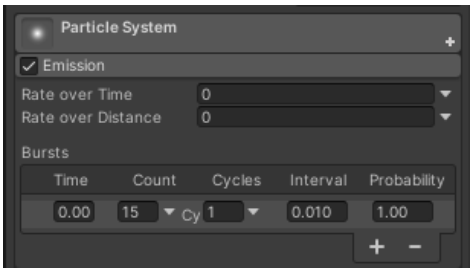

```
51
52      if(Input.GetButtonDown("Fire1"))
53      {
54          Instantiate(shotToFire, shotPoint.position, shotPoint.rotation);
55      }
```

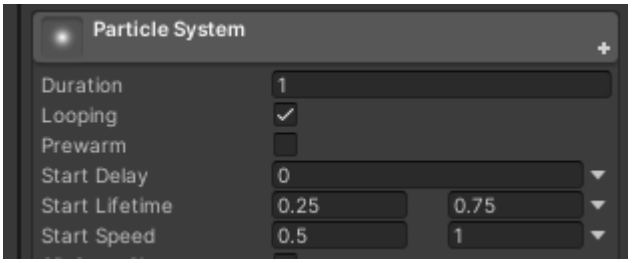
```
51
52      if(Input.GetButtonDown("Fire1"))
53      {
54          Instantiate(shotToFire, shotPoint.position, shotPoint.rotation).moveDir = new Vector2(transform.localScale.x, 0f);
55      }
```

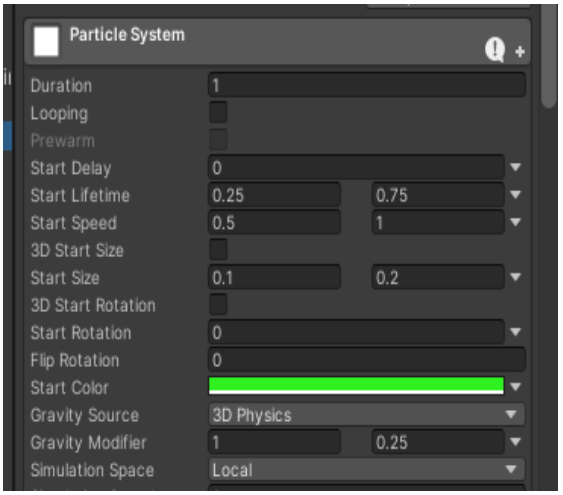
ADDING IMPACT

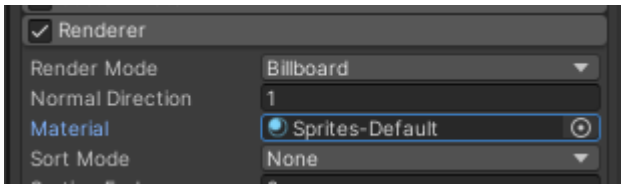


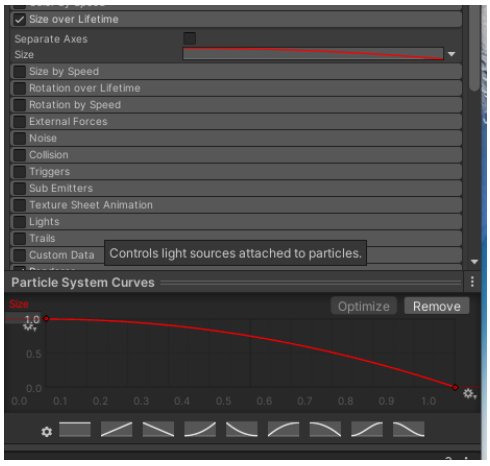


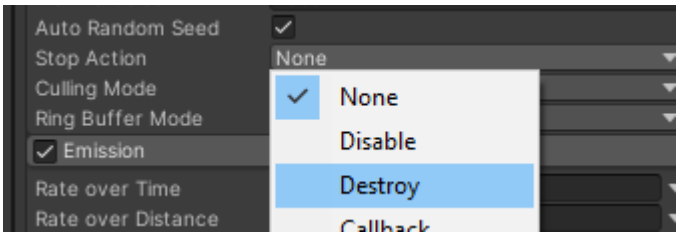


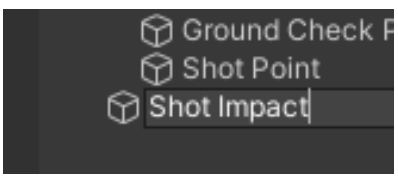






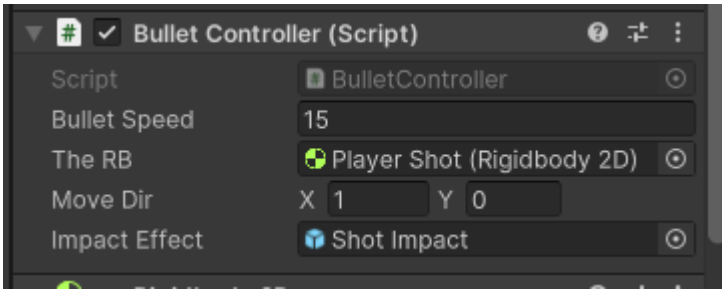




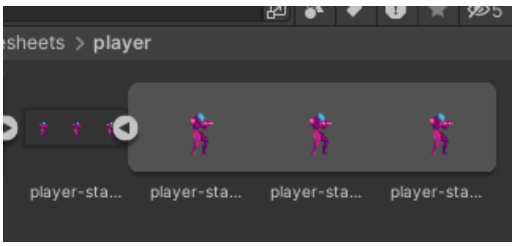


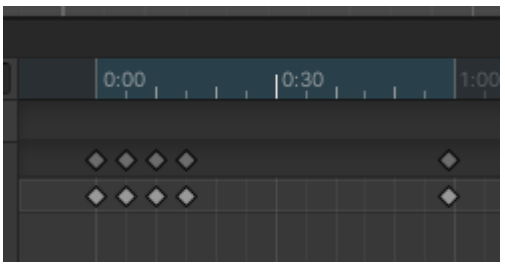
```
9
10      public Vector2 moveDir;
11
12      public GameObject impactEffect;
13
```

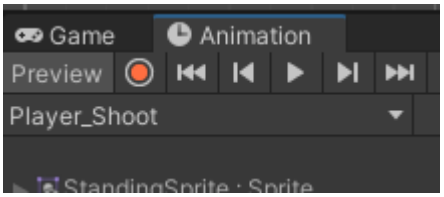
```
20
21      private void OnTriggerEnter2D(Collider2D other)
22      {
23          if (impactEffect != null)
24          {
25              Instantiate(impactEffect, transform.position, Quaternion.identity);
26          }
27
28          Destroy(gameObject);
29      }
```

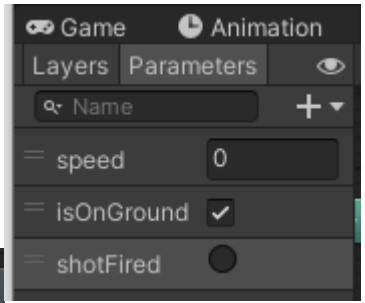


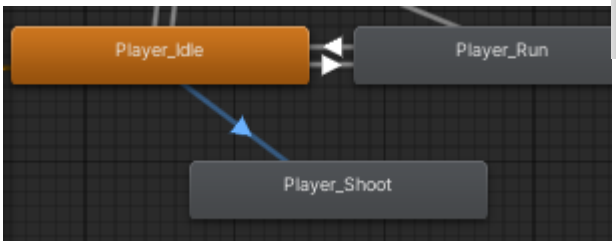
SHOOTING ANIMATION

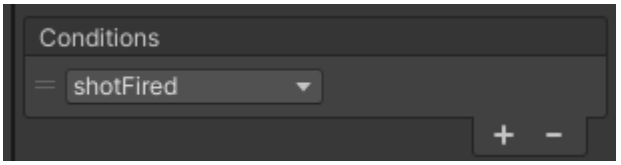


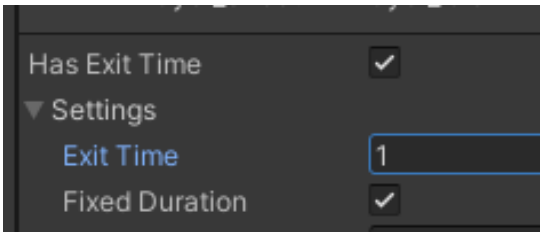


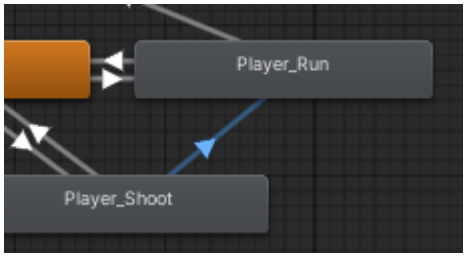


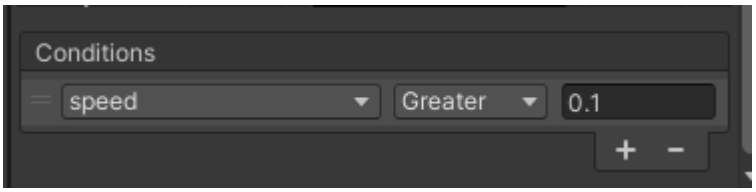


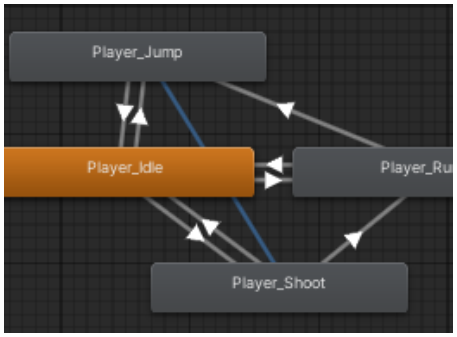


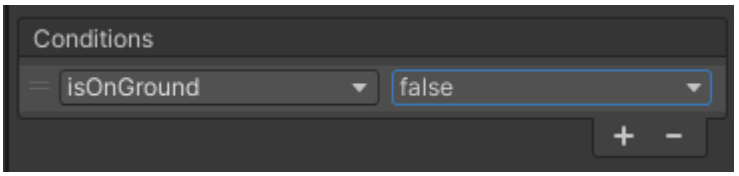




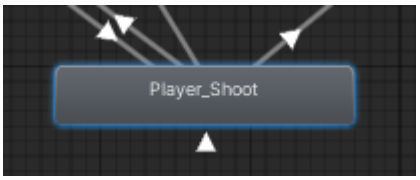


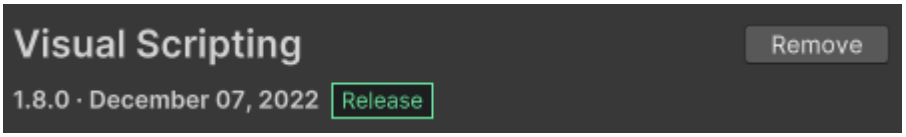


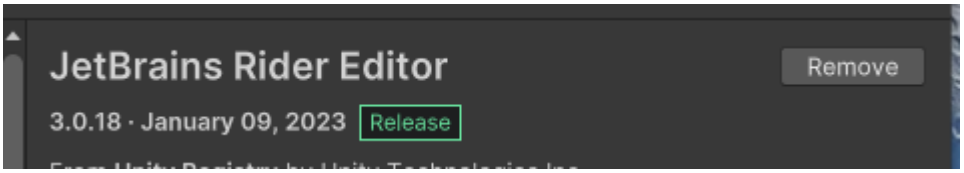





```
50
51          // Shooting
52          if(Input.GetButtonDown("Fire1"))
53          {
54              Instantiate(shotToFire, shotPoint.
55              anim.SetTrigger("shotFired");
56          }
```





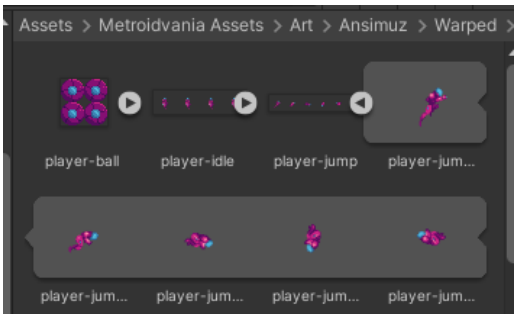


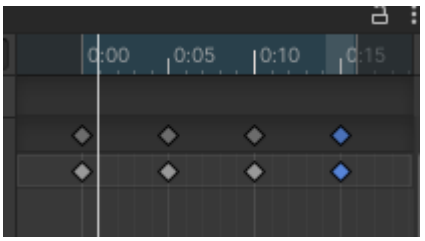
BALH BLAH BLAH

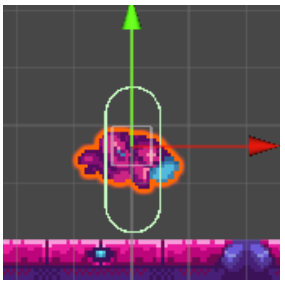
Lesson 5: Creating Abilities

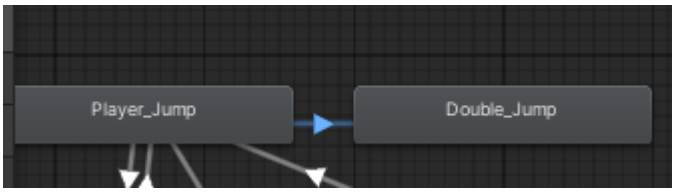
```
18      public BulletController shotToFire;
19      public Transform shotPoint;
20
21      private bool canDoubleJump;
22
```

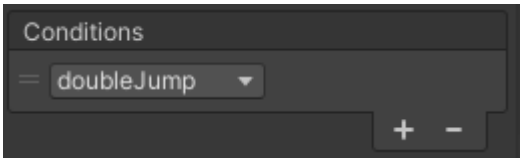
```
46
47          // jumping
48          if (Input.GetButtonDown("Jump") && (isOnGround || canDoubleJump))
49          {
50              if (isOnGround)
51              {
52                  canDoubleJump = true;
53              }
54              else
55              {
56                  canDoubleJump = false;
57              }
58              theRB.velocity = new Vector2(theRB.velocity.x, jumpForce);
59          }
```

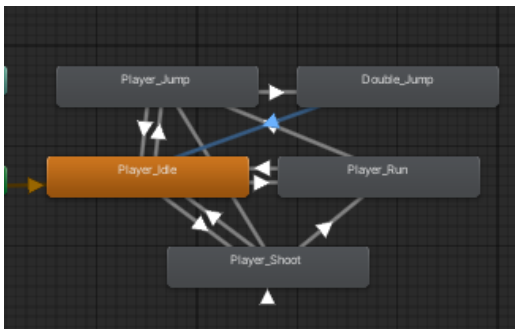


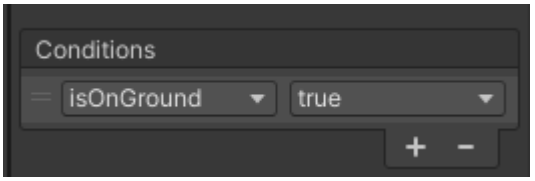






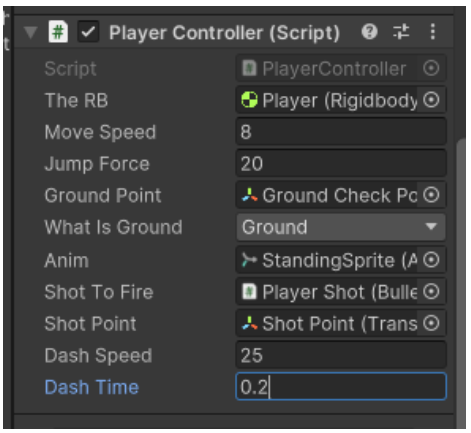






DASHING

```
20
21      private bool canDoubleJump;
22
23      public float dashSpeed, dashTime;
24      private float dashCounter;
25
```

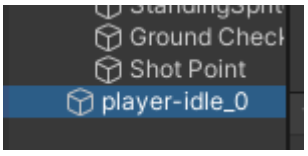


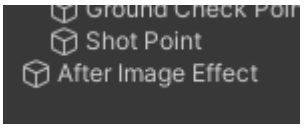
```
// Update is called once per frame
void Update()
{

    if(Input.GetButtonDown("Fire2"))
    {
        dashCounter = dashTime;
    }

    if (dashCounter > 0)
    {

    }
    else
    {
        // move sideways
        theRB.velocity = new Vector2(Input.GetAxisRaw("Horizontal") * moveSpeed, theRB.velocity.y);

        // handle direction change
        if (theRB.velocity.x < 0)
        {
            transform.localScale = new Vector3(-1f, 1f, 1f);
        }
        else if (theRB.velocity.x > 0)
        {
            transform.localScale = Vector3.one;
        }
    }
```

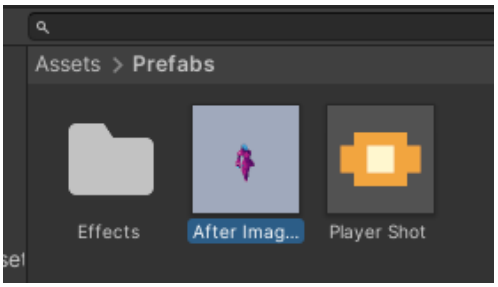
```
    }

    if (dashCounter > 0)
    {
        dashCounter = dashCounter - Time.deltaTime;
    }
    else
    {
```

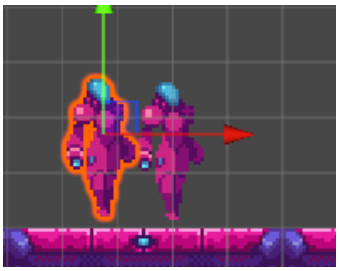
```
    if (dashCounter > 0)
    {
        dashCounter = dashCounter - Time.deltaTime;

        theRB.velocity = new Vector2(dashSpeed * transform.localScale.x, theRB.velocity.y);
    }
    else
```

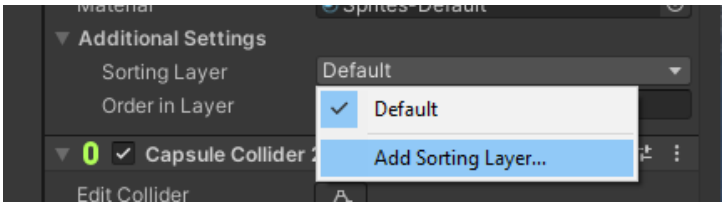
Dash animation

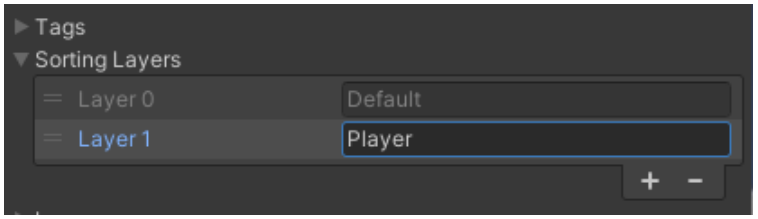


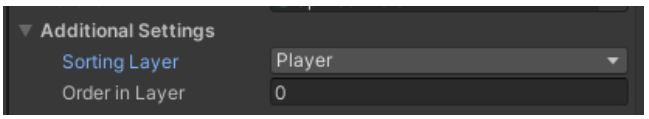


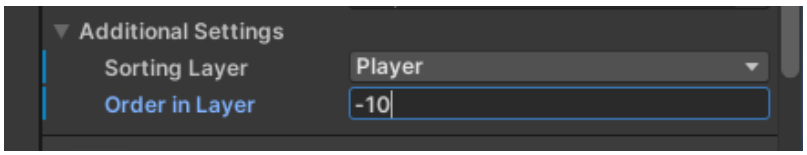


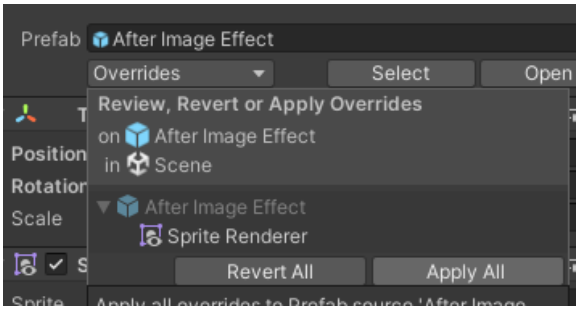








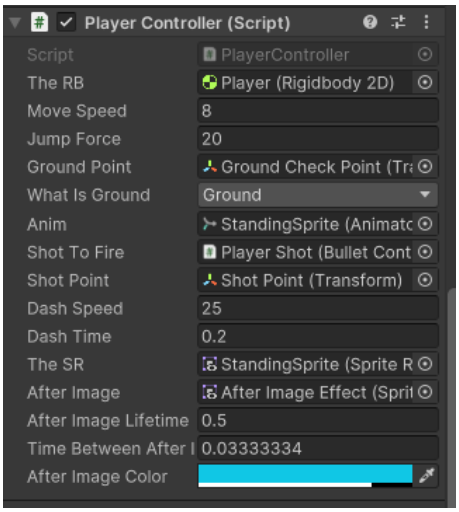


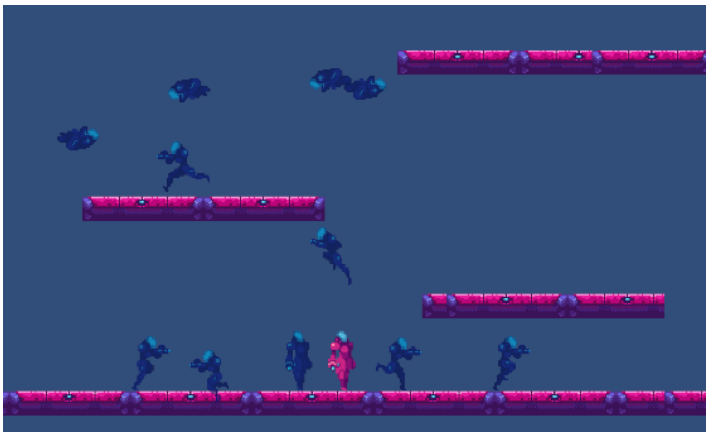


```
24      private float dashCounter;
25
26      public SpriteRenderer theSR, afterImage;
27      public float afterImageLifetime, timeBetweenAfterImages;
28      private float afterImageCounter;
29      public Color afterImageColor;
30
31      // Start is called before the first frame update
```

```
39      {
40          //dashing
41          if(Input.GetButtonDown("Fire2"))
42          {
43              dashCounter = dashTime;
44
45              ShowAfterImage();
46          }
47
48          if (dashCounter > 0)
```

```
103
104     public void ShowAfterImage()
105     {
106       SpriteRenderer image =  Instantiate(afterImage, transform.position, transform.rotation);
107         image.sprite = theSR.sprite;
108         image.transform.localScale = transform.localScale;
109         image.color = afterImage.color;
110     }
111
112 }
```

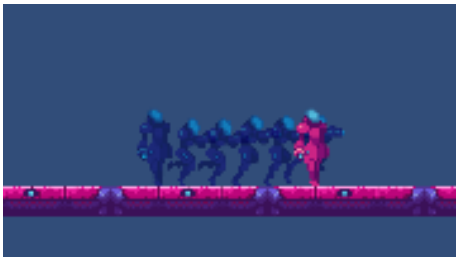




```
109         image.color = afterImageColor;
110
111         Destroy(image.gameObject, afterImageLifetime);
112     }
113
114 }
115
```

```
110
111         Destroy(image.gameObject, afterImageLifetime);
112
113         afterImageCounter = timeBetweenAfterImages;
114     }
115
116 }
```

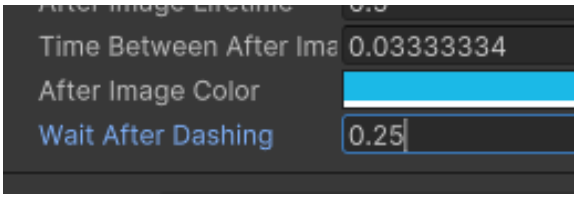
```
47
48      if (dashCounter > 0)
49      {
50          dashCounter = dashCounter - Time.deltaTime;
51
52          theRB.velocity = new Vector2(dashSpeed * transform.localScale.x, theRB.velocity.y);
53
54          afterImageCounter -= Time.deltaTime;
55          if (afterImageCounter <= 0)
56          {
57              ShowAfterImage();
58          }
59
60      }
```



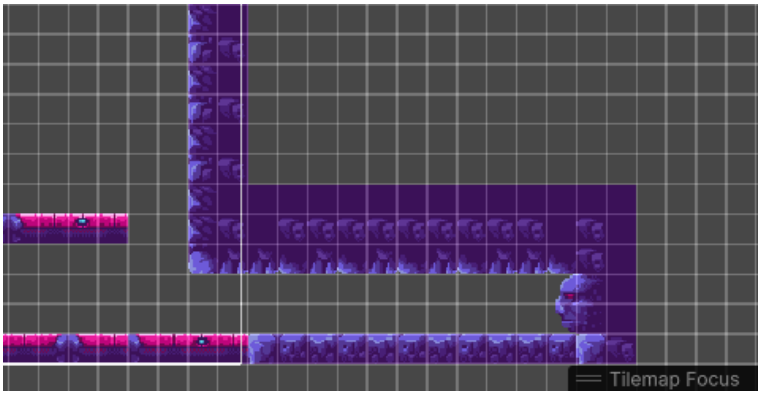
```
29      public Color afterImageColor;
30
31      public float waitAfterDashing;
32      private float dashRechargeCounter;
33
```

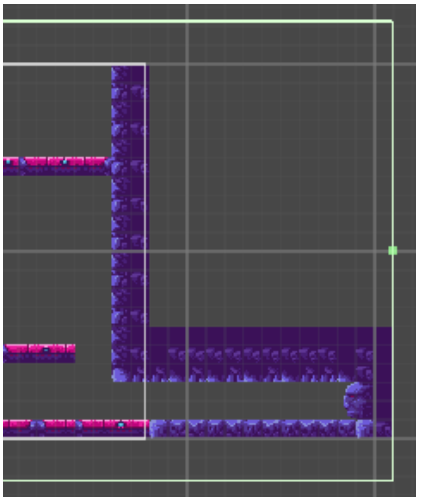
```
39
40      // Update is called once per frame
41      void Update()
42      {
43          if (dashRechargeCounter > 0)
44          {
45              dashRechargeCounter -= Time.deltaTime;
46          }
47          else
48          {
49
50              //dashing
51              if (Input.GetButtonDown("Fire2"))
52              {
53                  dashCounter = dashTime;
54
55                  ShowAfterImage();
56              }
57          }
58
```

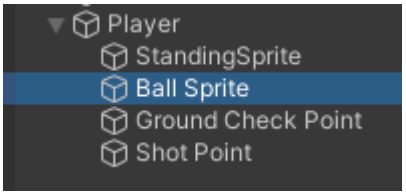
```
64              afterImageCounter -= Time.deltaTime;
65              if (afterImageCounter <= 0)
66              {
67                  ShowAfterImage();
68              }
69              dashRechargeCounter = waitAfterDashing;
70          }
71          else
```

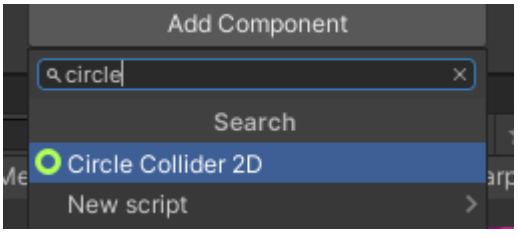


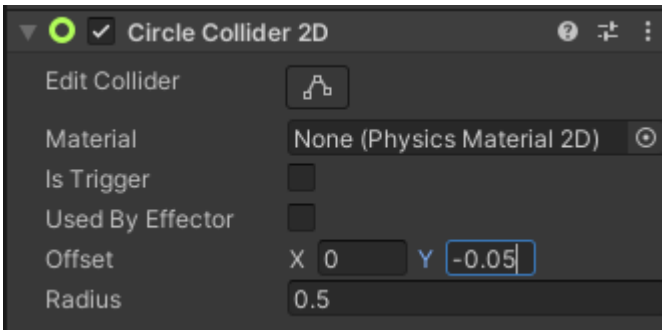
BECOMING A BALL -



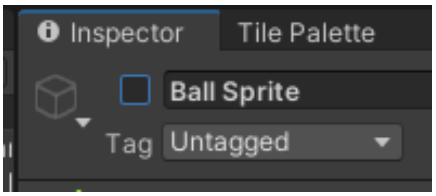


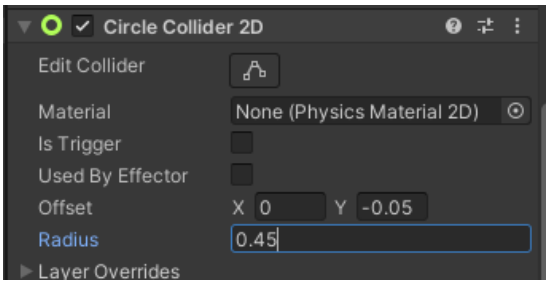






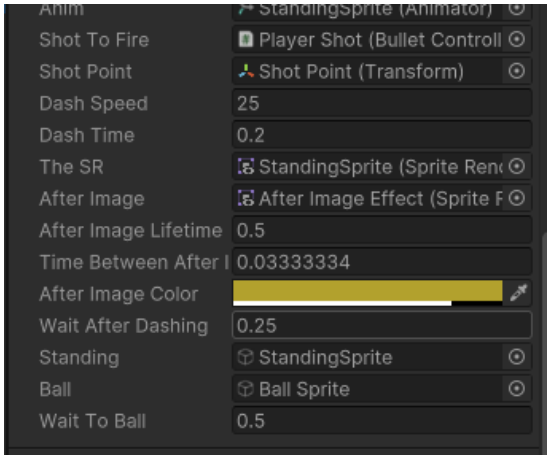
Oi






```
33
34      public GameObject standing, ball;
35      public float waitToBall;
36      private float ballCounter;
37
```

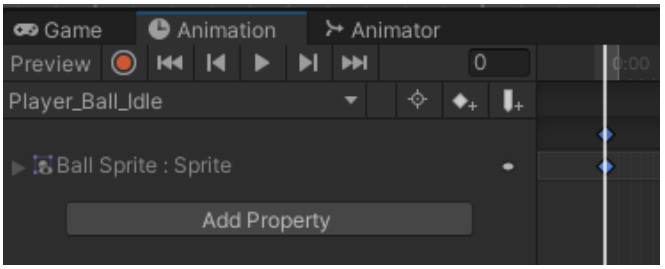
```
53
54          //dashing
55          if (Input.GetButtonDown("Fire2") && standing.activeSelf)
56          {
57              dashCounter = dashTime;
```

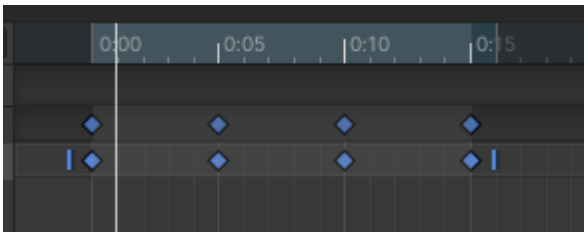


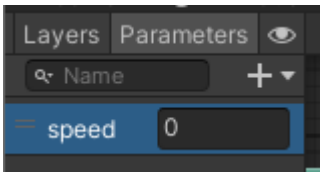

```
117
118         // ball mode
119         if(!ball.activeSelf)
120         {
121             if(Input.GetAxisRaw("Vertical") < -.9f)
122             {
123                 ballCounter -= Time.deltaTime;
124                 if(ballCounter <= 0)
125                 {
126                     ball.SetActive(true);
127                     standing.SetActive(false);
128                 }
129             }
130             else
131             {
132                 ballCounter = waitToBall;
133             }
134         } else
```

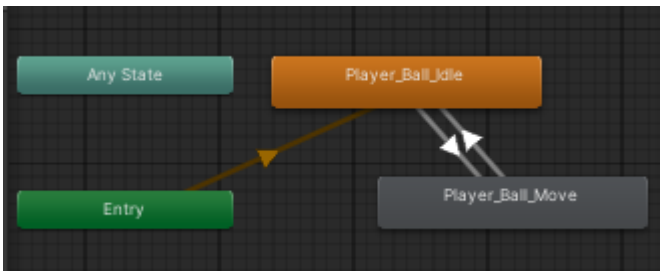
```
            else
            {
                ballCounter = waitToBall;
            }
        } else
        {
            if (Input.GetAxisRaw("Vertical") > -.9f)
            {
                ballCounter -= Time.deltaTime;
                if (ballCounter <= 0)
                {
                    ball.SetActive(false);
                    standing.SetActive(true);
                }
            }
            else
            {
                ballCounter = waitToBall;
            }
        }
```

Oi ANIMATING BALL -



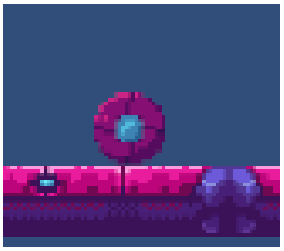




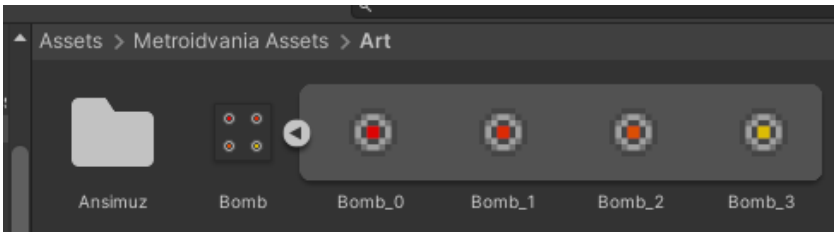



```
35      public float waitToBall;
36      private float ballCounter;
37
38      public Animator ballAnim;
39
```

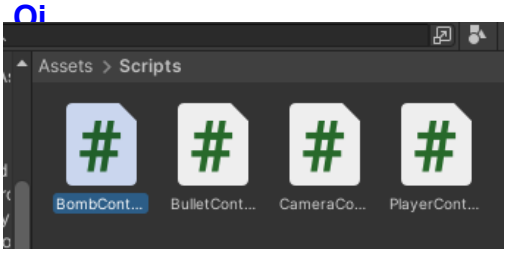
```
154
155         // moving animations
156         if(standing.activeSelf)
157         {
158             anim.SetBool("isOnGround", isOnGround);
159             anim.SetFloat("speed", Mathf.Abs(theRB.velocity.x));
160         }
161
162         if(ball.activeSelf)
163         {
164             ballAnim.SetFloat("speed", Mathf.Abs(theRB.velocity.x));
165         }
166     }
```



Dropping bombs

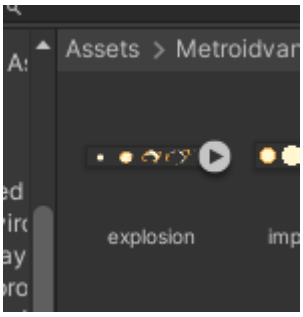


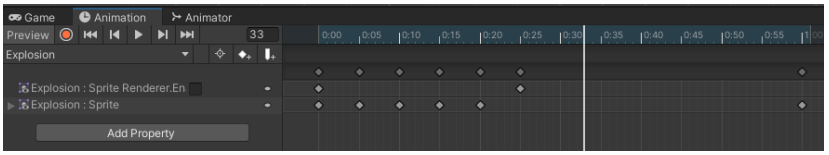
Oi

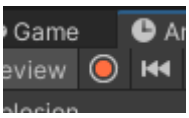


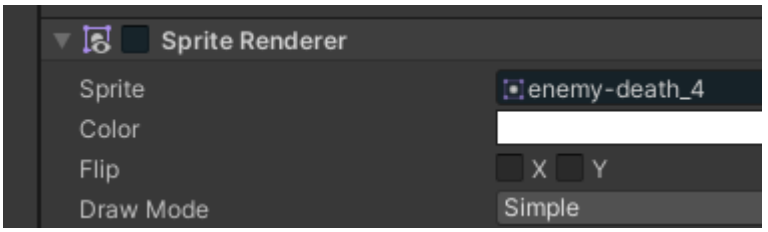

```
6   {
7       public float timeToExplode = .5f;
8       public GameObject explosion;
9
10      // Start is called before the first fra
11      void Start()
```

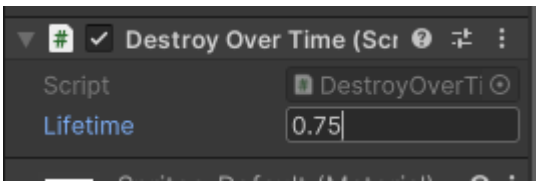
```
16      // Update is called once per frame
17      void Update()
18      {
19          timeToExplode -= Time.deltaTime;
20          if(timeToExplode <= 0)
21          {
22              if(explosion != null)
23              {
24                  Instantiate(explosion,transform.position, transform.rotation);
25              }
26              Destroy(gameObject);
27          }
28      }
29  }
```



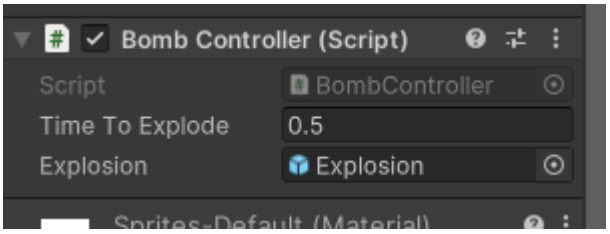




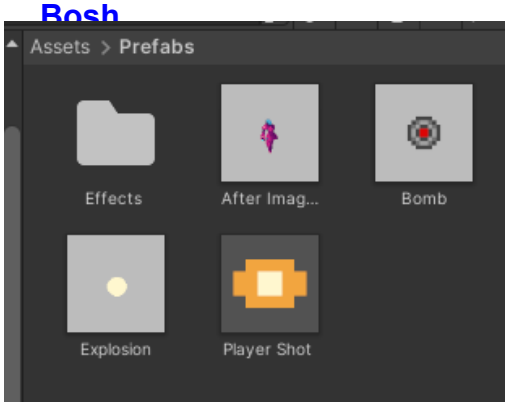





```
Miscellaneous Files                    DestroyOverTime
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class DestroyOverTime : MonoBehaviour
6   {
7
8       public float lifetime;
9       // Start is called before the first frame update
10      void Start()
11      {
12          Destroy(gameObject, lifetime);
13      }
14
15  }
```

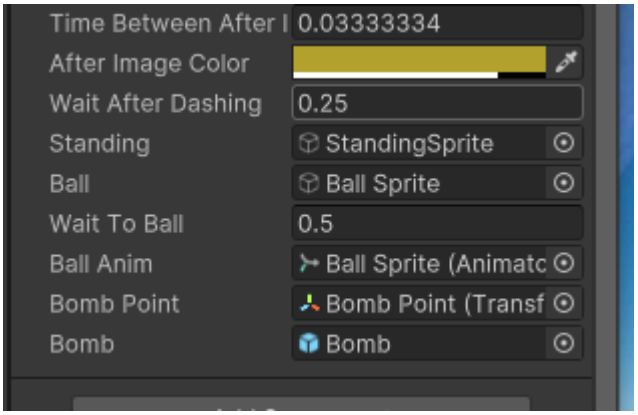


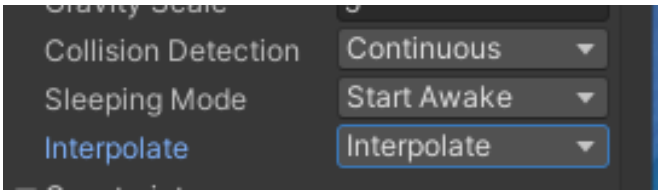
Bosh



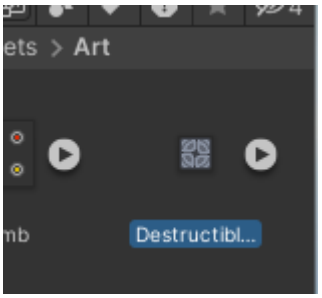

```
37
38      public Animator ballAnim;
39
40      public Transform bombPoint;
41      public GameObject bomb;
42
```

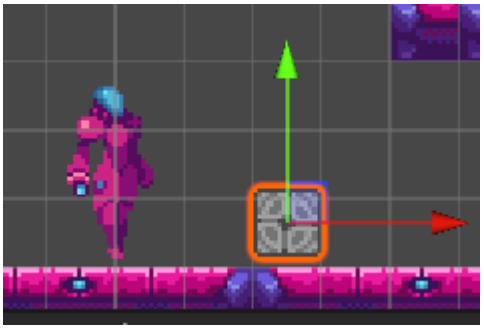
```
116
117     // Shooting
118     if (Input.GetButtonDown("Fire1"))
119     {
120         if (standing.activeSelf)
121         {
122             Instantiate(shotToFire, shotPoint.position, shotPoint.rotation).moveDir = new Vector2(transform.localScale.x, 0f);
123             anim.SetTrigger("shotFired");
124
125         } else if(ball.activeSelf)
126         {
127             Instantiate(bomb, bombPoint.position, bombPoint.rotation);
128         }
129
130     }
```

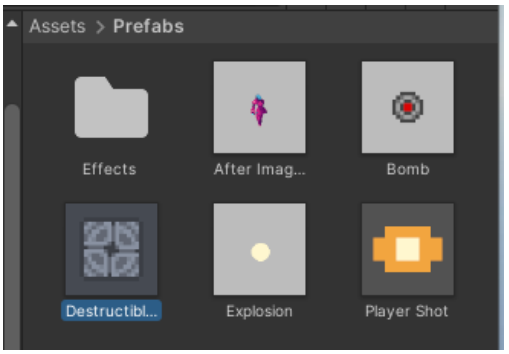


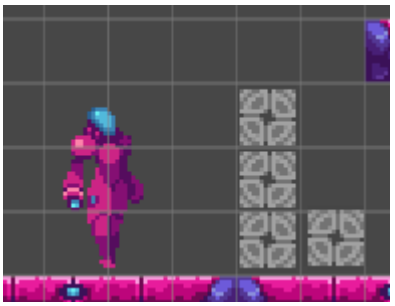


Oi

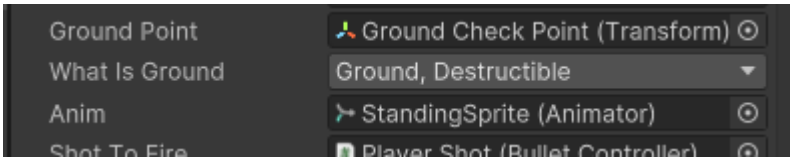




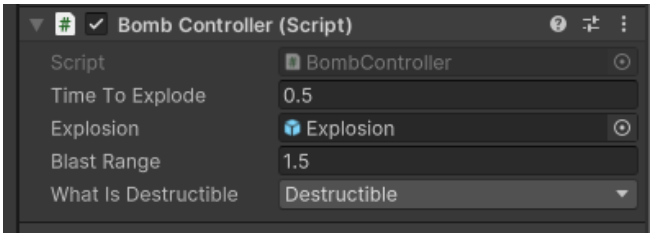


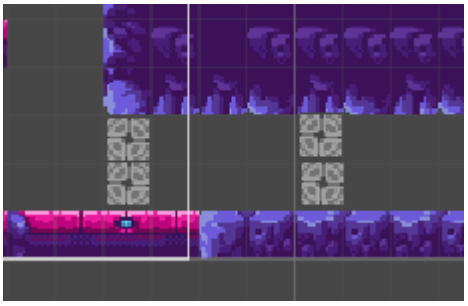


```
7       public float timeToExplode = .5f;
8       public GameObject explosion;
9
10      public float blastRange;
11      public LayerMask whatIsDestructible;
12
```

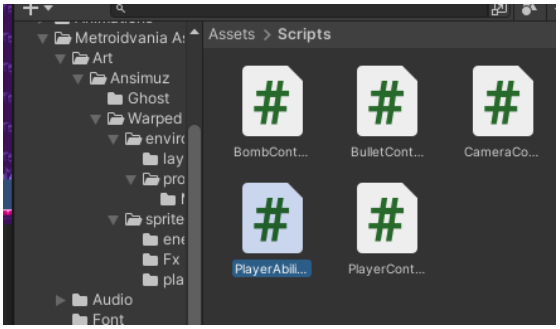


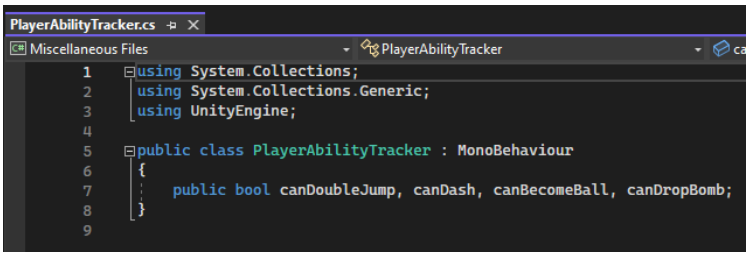

```
26              {
27                  Instantiate(explosion,transform.position, transform.rotation);
28              }
29              Destroy(gameObject);
30
31              Collider2D[] objectsToRemove =  Physics2D.OverlapCircleAll(transform.position, blastRange, whatIsDestructible);
32
33              if(objectsToRemove.Length > 0)
34              {
35                  foreach(Collider2D col in objectsToRemove)
36                  {
37                      Destroy(col.gameObject);
38                  }
39
40              }
41          }
42      }
43  }
```





Lesson 6: Le bosh - unlocking abilities





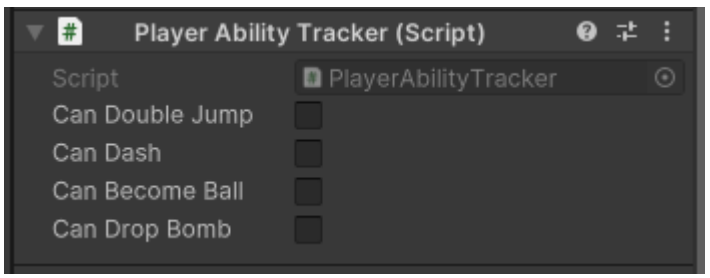

```
public Transform bombPoint;
public GameObject bomb;

private PlayerAbilityTracker abilities;


// Start is called before the first frame update
void Start()
{
    abilities = GetComponent<PlayerAbilityTracker>();
}
```

```
//dashing
if (Input.GetButtonDown("Fire2") && standing.activeSelf && abilities.canDash)
{
    dashCounter = dashTime;
```

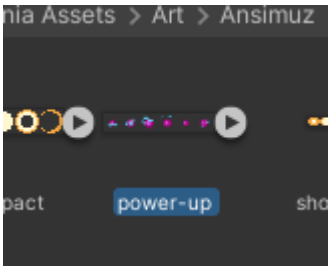
```
isOnGround = Physics2D.OverlapCircle(groundPoint.position, .2f, whatIsGround);

// jumping
if (Input.GetButtonDown("Jump") && (isOnGround || (canDoubleJump && abilities.canDoubleJump)))
{
    if (isOnGround)
```

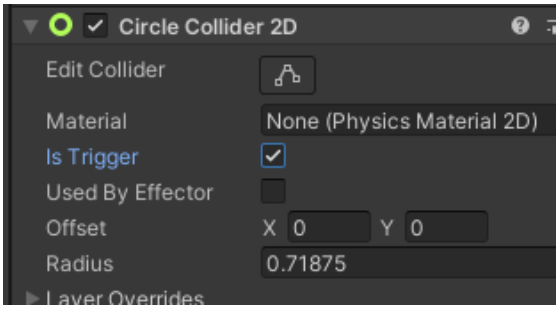
```
// ball mode
if (!ball.activeSelf)
{
    if (Input.GetAxisRaw("Vertical") < -.9f && abilities.canBecomeBall)
    {
        ballCounter -= Time.deltaTime;
```

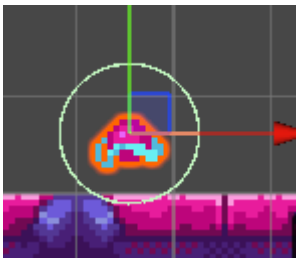
```
        Instantiate(shotToFire, shotPoint.position, shotPoint.rotation
        anim.SetTrigger("shotFired");
        // dropping bombs
    } else if(ball.activeSelf && abilities.canDropBomb)
    {
        Instantiate(bomb, bombPoint.position, bombPoint.rotation);
    }

}
```



Oi unlocking system





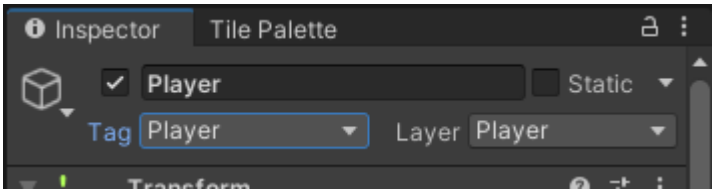


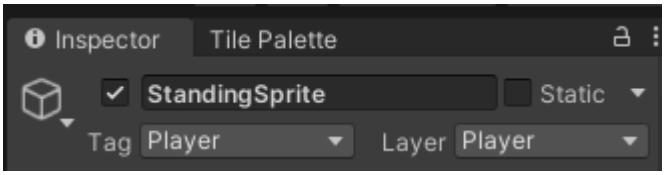
```
public class AbilityUnlock : MonoBehaviour
{

    public bool unlockDoubleJump, unlockDash, unlockBecomeBall, unlockDropBomb;

```

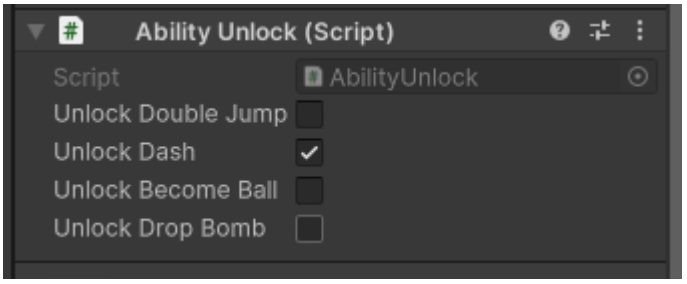
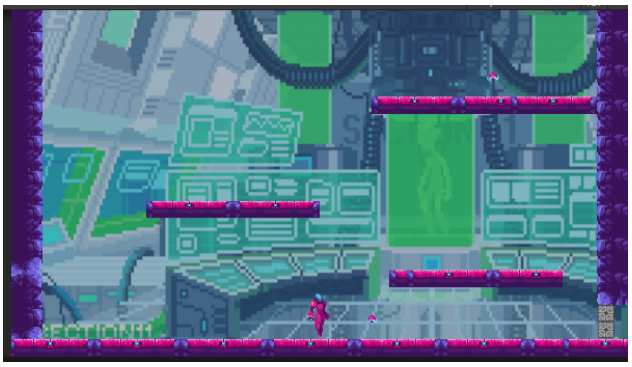
```
    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.tag == "Player")
        {
            PlayerAbilityTracker player = other.GetComponentInParent<PlayerAbilityTracker>();
```

```
    public bool unlockDoubleJump, unlockDash, unlockBecomeBall, unlockDropBomb;

    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.tag == "Player")
        {
            PlayerAbilityTracker player = other.GetComponentInParent<PlayerAbilityTracker>();

            if(unlockDoubleJump)
            {
                player.canDoubleJump = true;

            }
```

```
            }
            if (unlockDash)
            {
                player.canDash = true;

            }
            if (unlockBecomeBall)
            {
                player.canBecomeBall = true;

            }
            if (unlockDropBomb)
            {
                player.canDropBomb = true;

            }

            Destroy(gameObject);
        }
    }
}
```







oioi