«Talento Tech»

Desarrollo de Videojuegos

Unity 2D

Clase 07

«Talento Tech»

Clase N° 07 | Animaciones

Temario:

- Animation
- Animator
- Parameters para el Animator
- Llamar animaciones en código
- Flip Sprite

Animación.

Una de las cosas que más nos llaman la atención a la hora de jugar, son las animaciones. Muy pocos juegos usan personajes estáticos. Entonces, es vital, para que el jugador se involucre más con nuestra idea, que hagamos animaciones, siendo parte FUNDAMENTAL del **feedback**.

Pero ¿Qué es la animación?

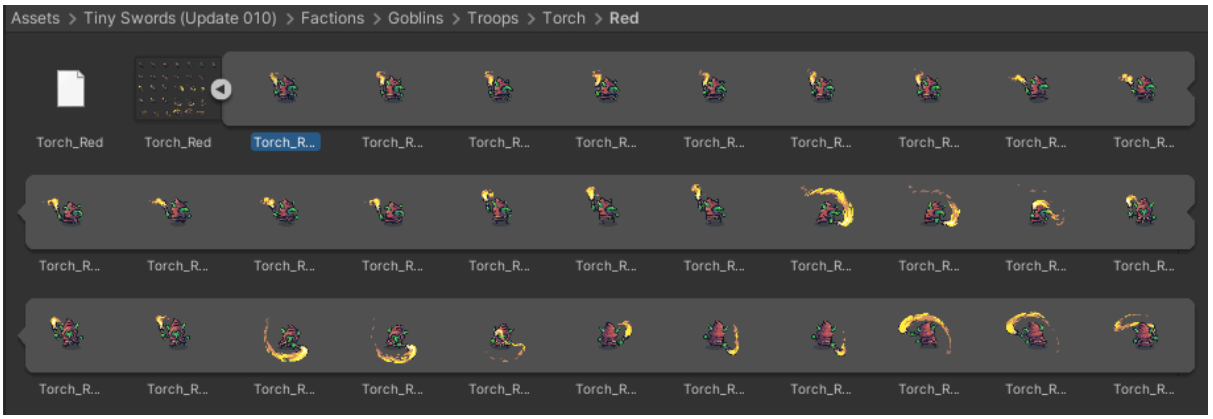
La animación es el proceso de crear la ilusión de movimiento a partir de imágenes estáticas o de manipular objetos en un espacio tridimensional. Se logra mediante la secuenciación de fotogramas o imágenes en una línea de tiempo, donde cada fotograma representa un pequeño cambio en la posición, forma o apariencia de un objeto. Al reproducir estos fotogramas rápidamente, se percibe un movimiento continuo.

En nuestro caso, la haremos usando Sprites. Colocaremos varias imágenes en una línea de tiempo y al ejecutarse veremos como nuestro personaje u objeto se “mueve”.

Seleccionar Sprites.

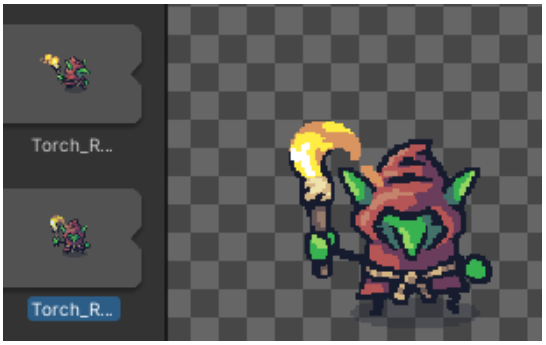
Para empezar, iremos a los Sprites que ya hemos dividido en la **clase 2** y podremos ver una gran cantidad de variaciones del mismo personaje.

I



En este caso, estamos usando al “Goblin”.

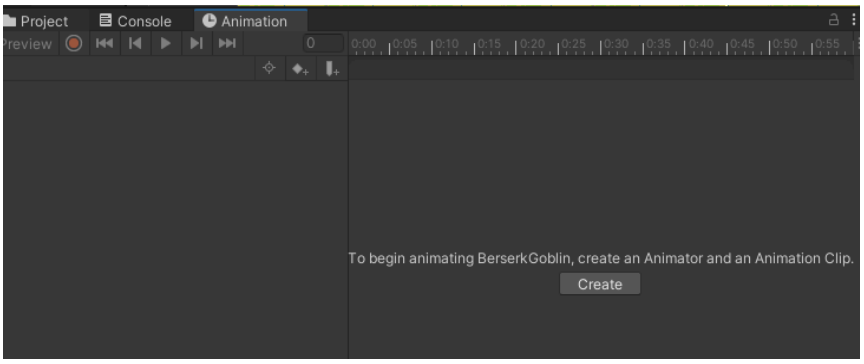
Fijense, que la mayoría de artistas, al hacer un SpriteSheet, ya incluyen varias de las animaciones posibles. Si vemos de cerca podremos identificarlas para clasificarlas y usarlas. Al hacer click en cada una de ellas, a la derecha, en el inspector, verán una miniatura más “apreciable”.



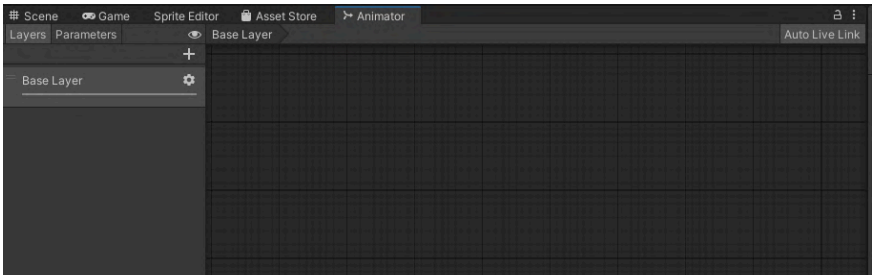
Animation y Animator

El siguiente paso será abrir nuestras ventanas de animación. Iremos a la solapa de Windows -> Animation y ahí haremos click en “Animation” y “Animator”.

La ventana **Animation** nos permite colocar los **KeyFrames** deseados.
¿Qué son? Se refiere a un fotograma específico que marca el inicio o el final de un cambio significativo en la animación, como la posición, la rotación o la escala de un objeto.



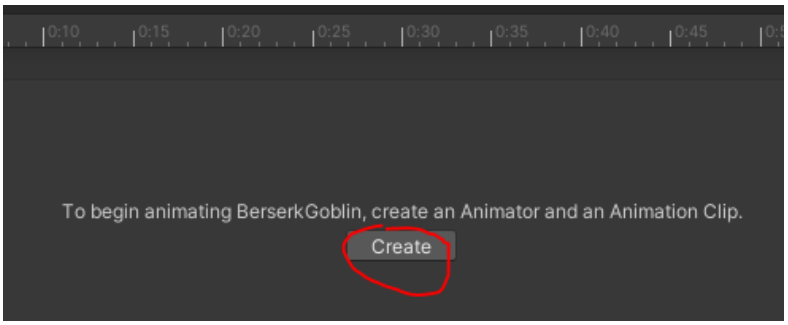
La ventana **Animator** es una herramienta que permite crear y gestionar animaciones para los personajes y objetos en nuestro juego.



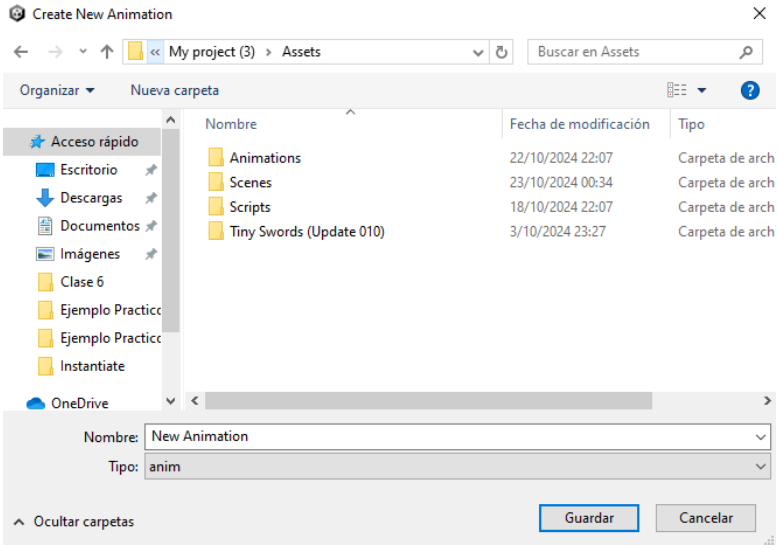
De esta manera en la primera crearemos las animaciones y en la segunda gestionaremos su uso.

Seleccionando Sprites para animar.

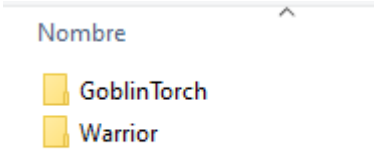
Crearemos las animaciones “**Idle**” (Cuando el personaje está quieto) y “**Walk**” (caminar). Empezaremos por seleccionar el objeto de la escena al que queremos animar. Iremos a la ventana **Animation** y haremos click en “Create”



Nos abrirá una ventana para designar en qué lugar queremos crearla. En estos casos podemos crear una carpeta nueva dentro de “Assets”, llamada “Animaciones” y ubicar todas nuestras animaciones bien organizadas.

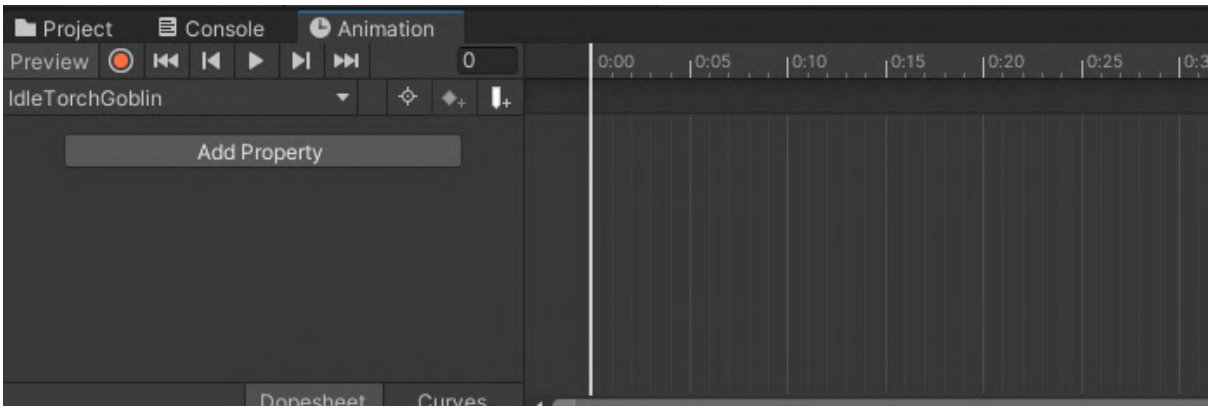


Es recomendable, crear carpetas con el nombre de los objetos que animamos, ya que cada objeto podrá llegar a tener varias de estas.



Y recuerden darle un nombre apropiado a sus animaciones. En ese caso, la primera será “Idle” o “IdleTorchGoblin” o el nombre que les parezca apropiado para identificarla.

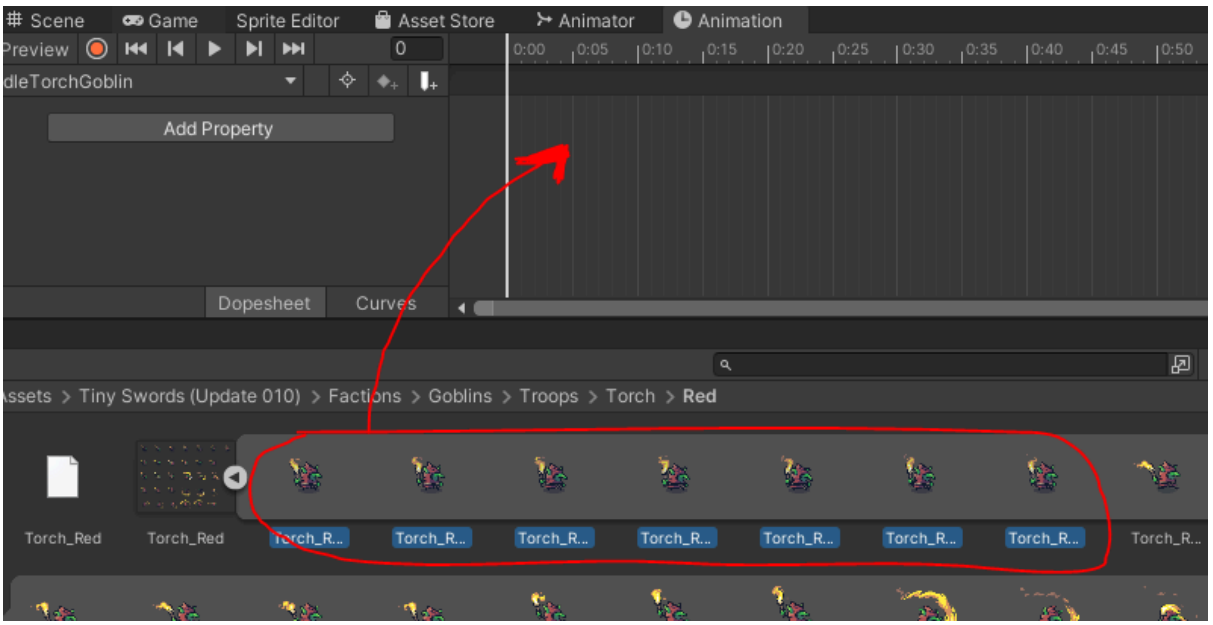
Una vez creada, la ventana de Animation nos aparecerá de esta manera:



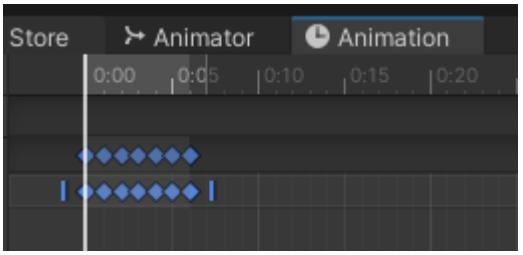
Podrán ver que a la **izquierda figura el nombre** de nuestra animación y a **la derecha la línea de tiempo**.

Agregando los Sprites.

Avancemos un poco más… vamos a seleccionar los **Sprites deseados** y agregarlos a mi **línea de tiempo**. Si tienen problemas con esto, pueden cambiar de lugar las ventanas para poder manejarse mejor.

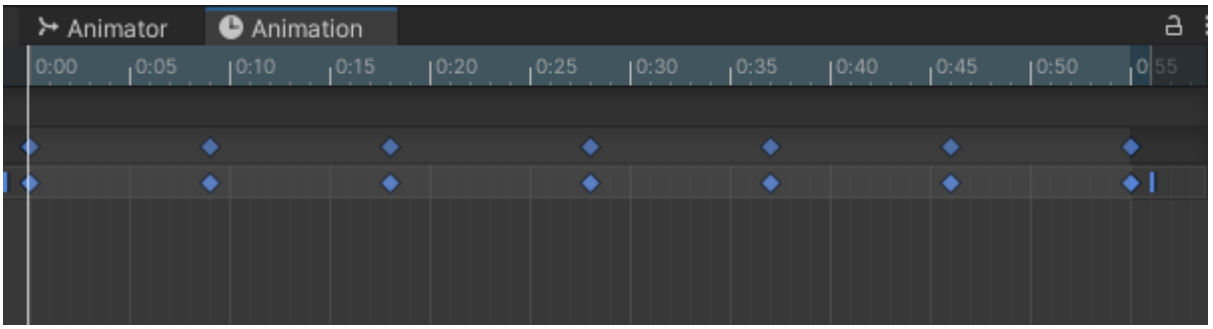


Al hacerlo, los frames aparecerán todos pegados, uno al lado del otro.



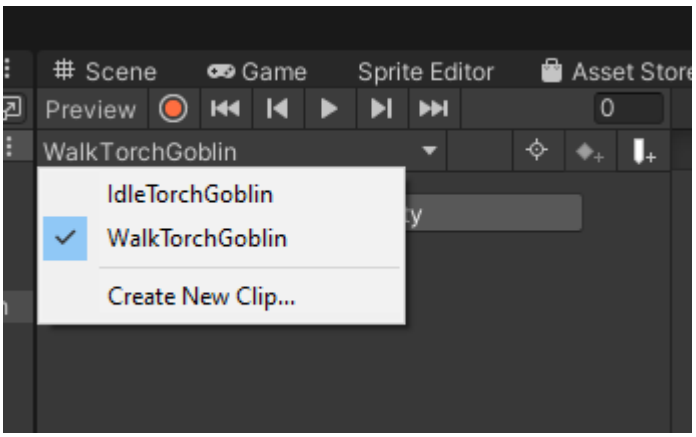
Y si le dan al botón de “Play”, se reproducirá super rapido, sin dar tiempo a entender qué sucede. Para solucionar esto, seleccionaremos todos los frames y veremos que al **mantener el click en la rayita azul**, los podremos ir **dispersando por la línea de tiempo.**

Quedando de esta forma:



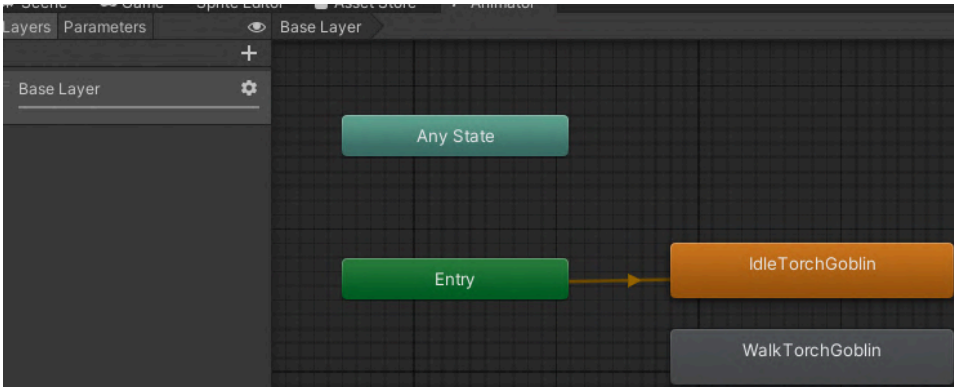
Si ven arriba, notarán que la animación pasó a durar casi 1 segundo.

Realizaremos el mismo proceso creando una animación nueva para la caminata llamada “Walk”



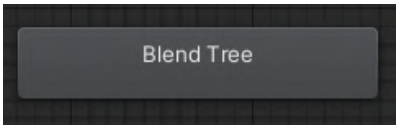
Animator: Seteo.

Si se dirigen a la ventana “Animator”, se debería de ver similar a esto:



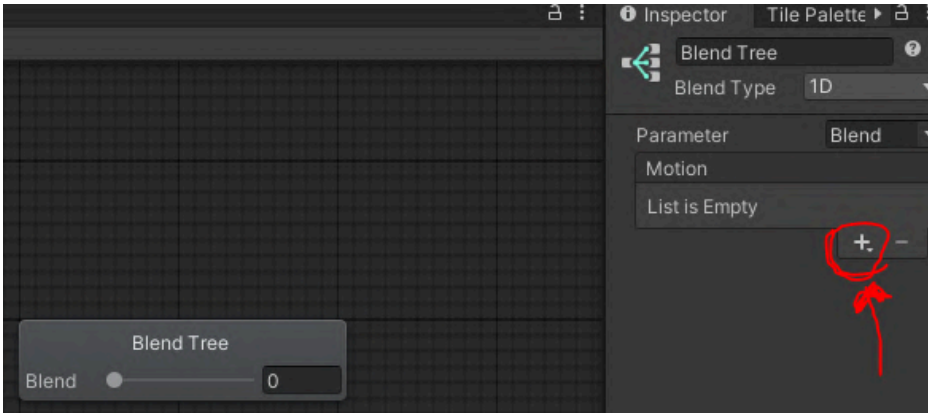
Cada animación que hagamos es un “State”, serán los estados del objeto que se irán alternando. Cada estado tendrá una animación distinta, como por ejemplo “caer”, “saltar”, “caminar”, “atacar”, etc.
La línea que los une con su flecha, indica el traslado de un estado a otro. Más adelante indagaremos más sobre esto.

Ahora mismo, haremos click derecho en un espacio vacío dentro del Animator “**Create State -> From New Blend Tree**”, Este es un Estado que mezcla distintas animaciones para que sea más fluida su transición.

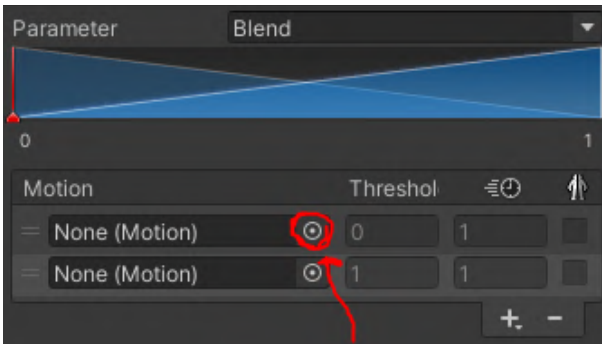


Le haremos Doble Click, nos aparecera esto:

Aquí haremos Click nuevamente e iremos al **Inspector**. En el mismo, iremos a la sección de "**motion**", notaran que dice **"List is Empty**", es decir que no tiene ninguna animacion agregada, y le haremos click al "**+**":



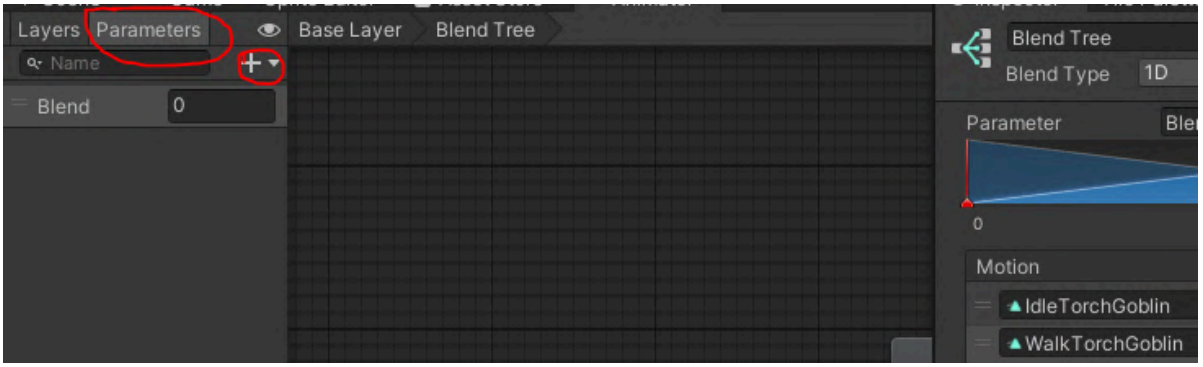
Agregaremos dos "**Motion Field**"



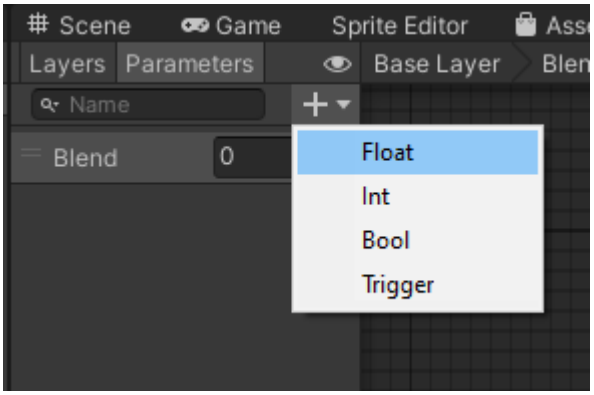
Y haciendo click en los círculos al lado de "None(Motion)", buscaremos las animaciones que queremos agregarle. En este caso serán el **Idle** y el **Walk**

Lo que hace el Blend Tree, es pasar de una animación a otra dependiendo de los parámetros dados. Entonces vamos a crear esos parámetros para que pueda "decidir".

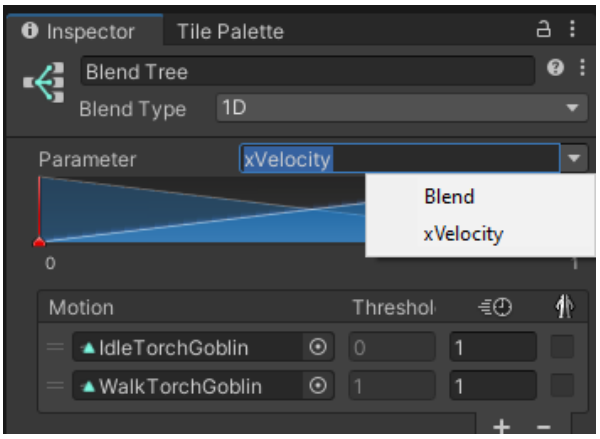
Primero iremos a la parte izquierda y seleccionamos “Parameter” para luego apretar en el “+”



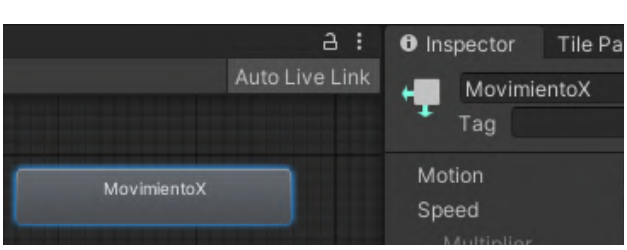
Lo que nos llevará a elegir, **qué tipo de parámetro** deseamos usar. En este caso seleccionaremos un “float” que representará si nos estamos moviendo o no.

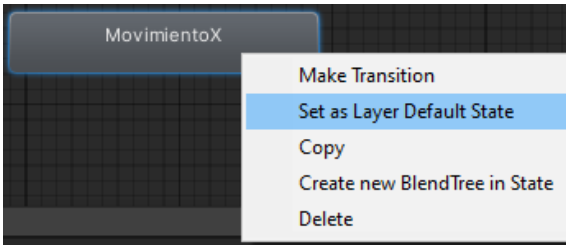


También cambiaremos el “Parameter” de Blend a **xVelocity**



Volveremos a la “Base Layer”, en donde figuraban todos los estados, le cambiaremos el **nombre** a nuestro Blend Tree a algo como “**MovimientoX**” y le haremos click derecho para setearlo como “**Layer Default State**”





Esto hará que la “flecha” Principal, la Transmisión principal, se conecte del Entry a nuestro Blend Tree llamado MovimientoX.
¿Por qué queremos esto?
Por que de esta manera, si esta bien asignado, apenas nuestro objeto se **cargue en la escena** empezará a reproducir los estados de **MovimientoX**, esto sería el **Idle.**

Código: Llamando a la animación.

Hemos creado un **parámetro float** para que, si este, es mayor a 0, nuestra animación pase de **Idle** a **Walk**. Lo que nos falta es poder modificar ese parámetro, y eso lo haremos en el código.

Primero: crearemos una variable del tipo de dato “Animator” dentro del código de movimiento de nuestro personaje:

```
Animator animator;
```

Segundo: le asignaremos un valor a nuestra variable con el **getComponent**, dentro del **Start()**:

```
animator = GetComponent<Animator>();
```

Tercero: usaremos la función **Math.Abs(),** que devuelve un valor absoluto (siempre positivo) de cualquier dato numérico enviado como Argumento, para crear una condición que pregunte si mi personaje “**se está moviendo**”:

```
    //y usando Mathf.Abs devolvemos un valor absoluto (positivo)
```

```
if (Mathf.Abs(rb.velocity.x) > 0 || Mathf.Abs(rb.velocity.y) > 0)
{
}
else
{
}
```

Cuarto: Utilizaremos la función “SetFloat()” de Animator para cambiar el parámetro deseado:

```
 //Usando Mathf.Abs devolvemos un valor absoluto (positivo)
if (Mathf.Abs(rb.velocity.x) > 0 || Mathf.Abs(rb.velocity.y) > 0)
{//Seteamos el float
    animator.SetFloat("xVelocity", 1);
}
else
{  //Seteamos el float
    animator.SetFloat("xVelocity", 0);
}
```

Con esto hecho, CADA VEZ que nuestro personaje se mueve en algún eje, se reproducirá la animación de caminar/**Walk**.

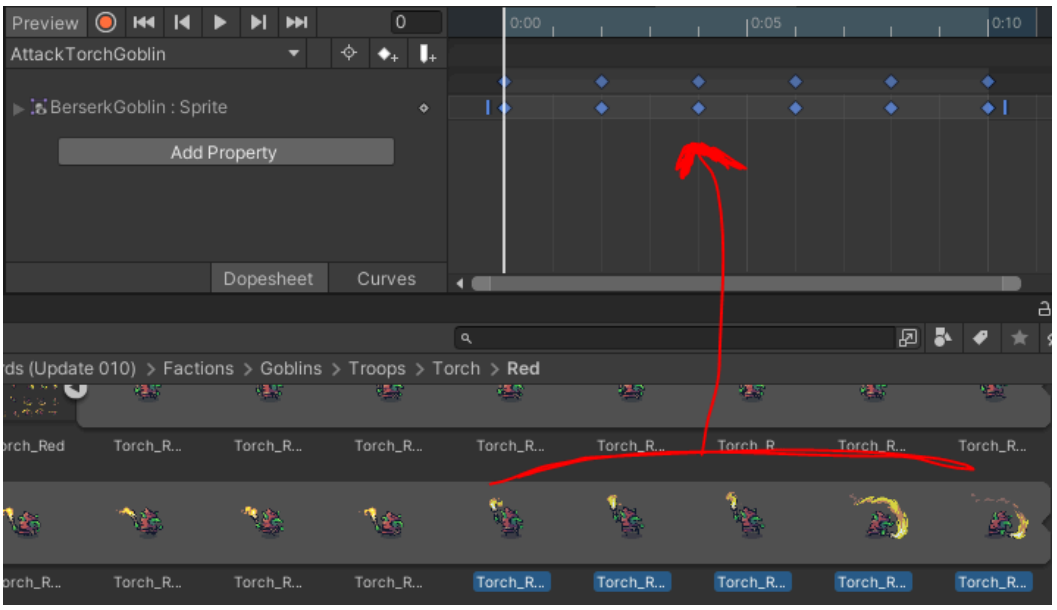
Animación de ataque.

Ahora haremos la animación de ataque de nuestro personaje!

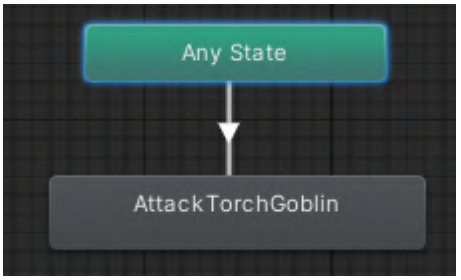
Seteo

Realizaremos el mismo proceso que con las 2 anteriores:

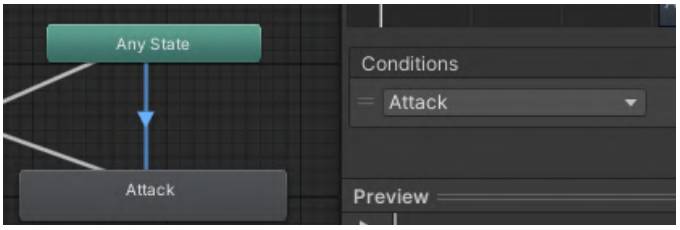
Primero: crearemos la animación y pasaremos los Sprites necesarios y los acomodaremos en su línea de tiempo.



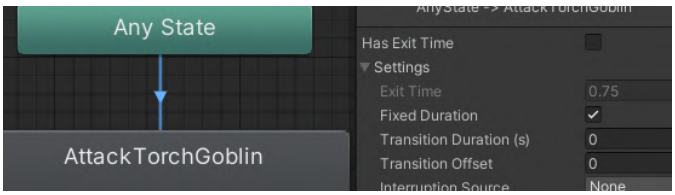
Segundo: Iremos al animator y **esta vez**, desde “Any state”, haremos **Click Derecho -> Make Transition** y lo uniremos a nuestra animación de ataque:



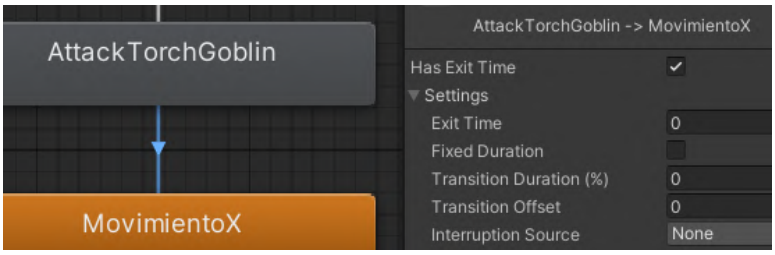
Tercero: iremos al sector de parámetros o “Parameters” y crearemos un Trigger llamado “Attack”. Si seleccionamos la “Transition” creada de **Any State** a nuestro **State de ataque**, podremos asignarle como “Condition” el parameter **Attack** que creamos .



Para terminar de setearlo, vayan a la transición de “**Any State -> AttackTorchGoblin**” y asegurense que el checkbox de “**Has Exit Time**” NO esté marcado y el “**Transition Duration**” este en 0



A la vez que en la siguiente transición del “**AtackTorchGoblin -> MovimientoX**”, SI debe estar marcado el **CheckBox**, manteniendo el “**Transition Duration**” en 0



Esto es para que desde cualquier estado, pase a realizar el ataque sin esperar y termine de realizar la animación del mismo antes de empezar el idle.

Por último iremos al código del personaje y crearemos un if sencillo para que, cuando yo apriete la tecla indicada, mi personaje se anime.

```
private void Attack() {
    if (Input.GetMouseButtonDown(0)){
        animator.SetTrigger("Attack");
    }
}
```

Flip Sprite.

Llegando al final de la clase de hoy, vemos que al mover nuestro personaje para la izquierda o derecha, este no se gira, mira siempre al mismo lado. Para arreglar esto, tendremos que jugar un poco con la escala del GameObject e invertirla a gusto.

Cabe destacar que siempre hay varias formas de resolver el mismo problema. En este caso, usaremos esta función:

```
private void FlipSprite()  {
 if (rb.velocity.x < 0 && !girolzq) {
    girolzq = true;
    Vector3 ls = transform.localScale;
    ls.x *= -1;
    transform.localScale = ls;
```

```
    }
    else if (rb.velocity.x > 0 && giroIzq) {
        giroIzq = false;
        Vector3 ls = transform.localScale;
        ls.x *= -1;
        transform.localScale = ls;
    }
}
```

Explicación y paso a paso:

1) Primero deberemos de crear un **bool** que hará de “check” para saber si nuestro personaje está girado o no:

```
bool giroIzq;
```

2) Seguiremos creando una función con un if, que tendrá como condición “*Si me muevo para la izquierda y no gire a la izquierda, entonces girar a la izquierda*”

```
    private void FlipSprite()
{
    if (rb.velocity.x < 0 && !giroIzq)
    {
```

Si la velocidad en x de mi RigidBody2D es menor a 0, significa que me estoy moviendo a la izquierda.

3) Para hacer invertir la escala, primero nos aseguraremos de que nuestros bool pase a ser true, para evitar problemas,

```
giroIzq = true;
```

Luego accederemos a la información de nuestra escala, creando una variable de tipo Vector3 y asignando transform.localScale, es decir, el dato de la escala del personaje

```
        Vector3 ls = transform.localScale;
```

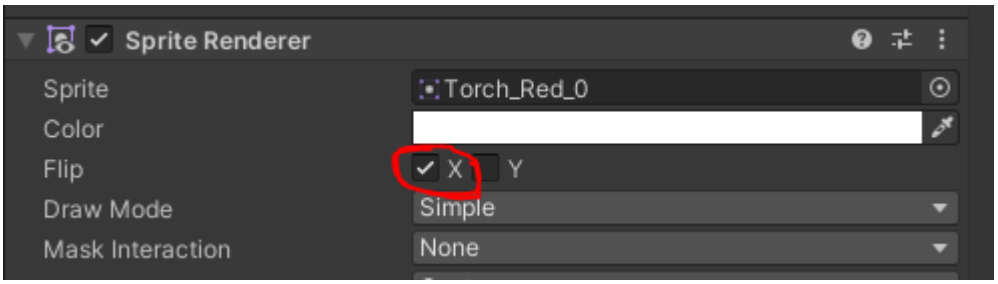
Siguiendo con esto, invertiremos su valor en x y reescribimos la escala de mi personaje con el nuevo dato.

```
    ls.x *= -1;
    transform.localScale = ls;
```

4) Realizaremos exactamente el mismo procedimiento.

```
else if (rb.velocity.x > 0 && girolzq){
        girolzq = false;
        Vector3 ls = transform.localScale;
           ls.x *= -1;
           transform.localScale = ls;
    }
```

Tengan en cuenta que si poseemos un personaje que por diseño está observando al otro lado, podemos girarlos desde el inspector yendo al componente del Sprite:



Ejercicios prácticos:

1) Es momento de crear varias de las animaciones que necesita tu juego. Te sugerimos empezar por tu personaje, los enemigos y, si te quedas con ganas de más, animar los efectos del juego como: habilidades, pickups, fuego, agua, plantas, etc.


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
Buenos
Aires
Ciudad