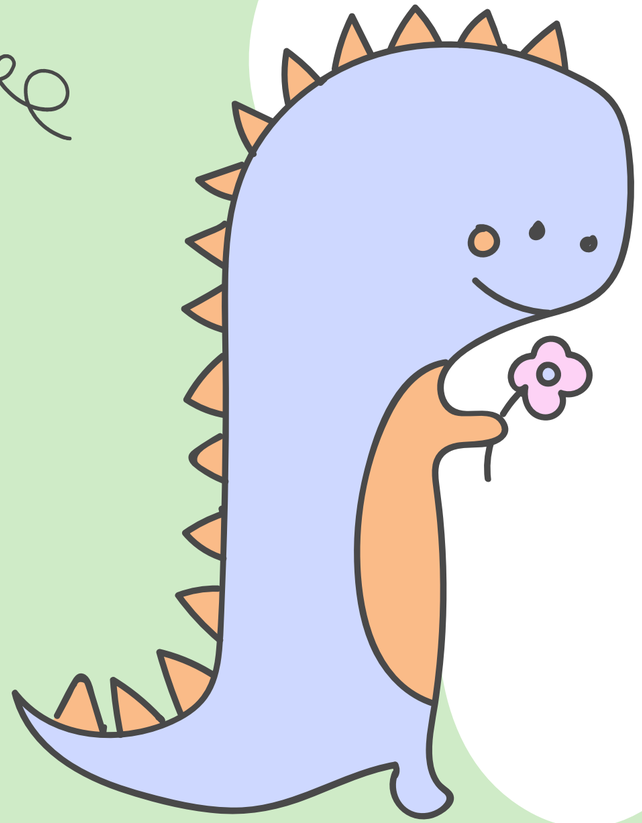
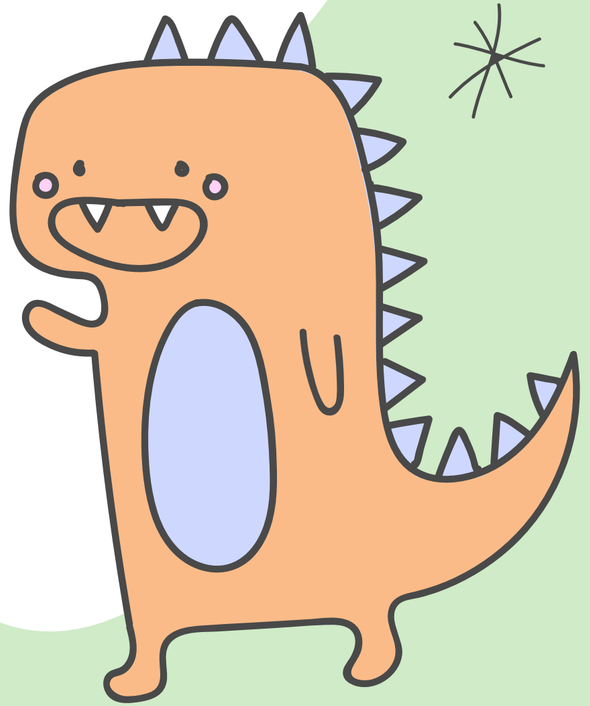


Control de efectos por animación

Sergio Peñaloza



Para comunicar un comportamiento dentro de un videojuego es común que se haga uso de animaciones. Por lo general, Las animaciones suelen estar presentes en la mayoría de los casos en los que se usan VFX de gameplay dentro del contexto de un videojuego.


1:12
100


44
WINS:3

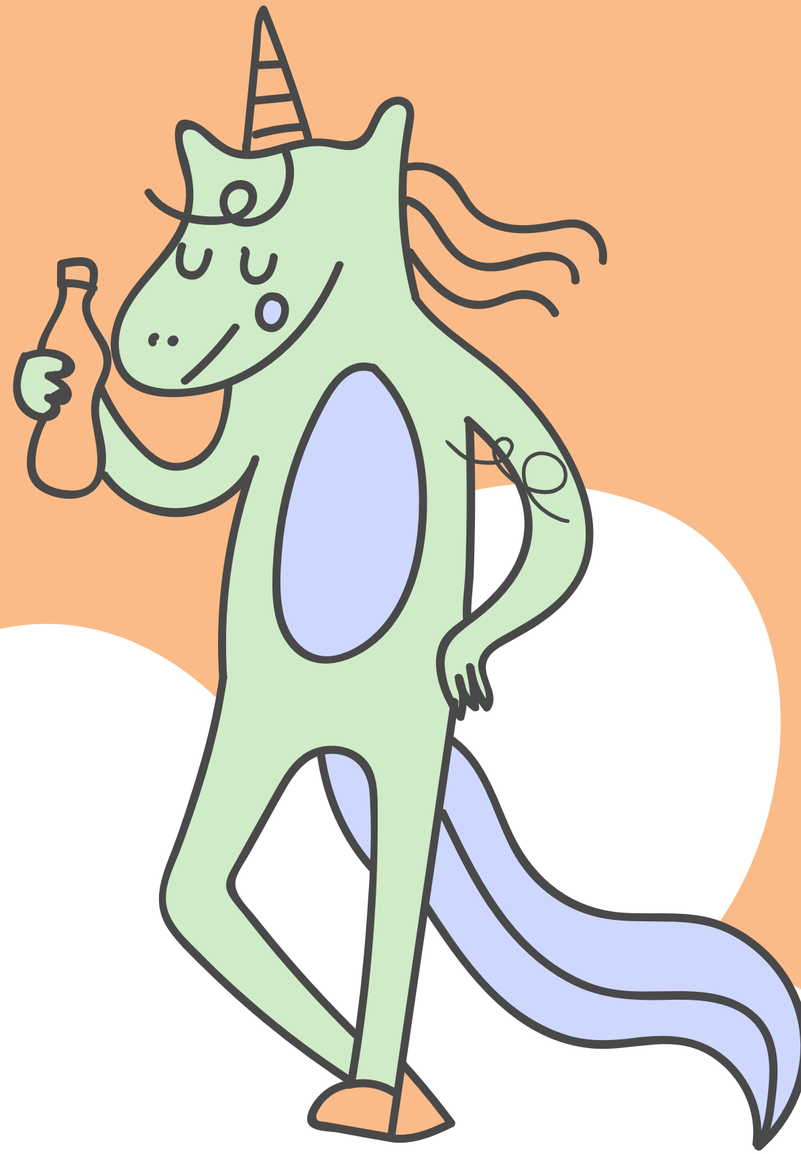
En Unity

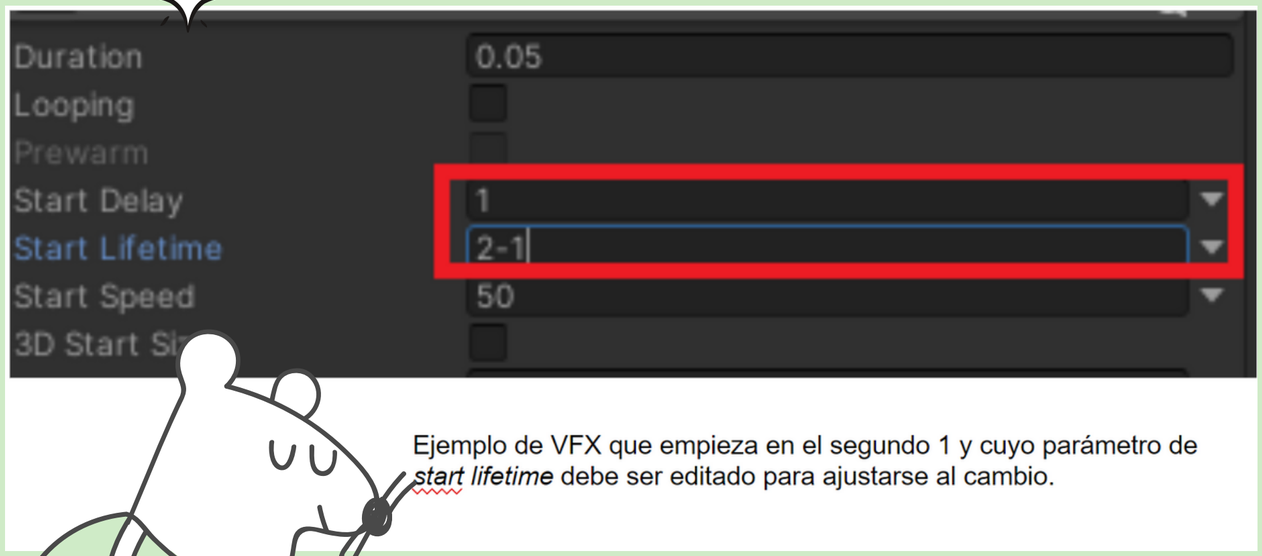
Para lograr la coordinación entre un efecto y una animación, existen distintos métodos con igual validez que podemos implementar.

Controlar start delay del sistema de partículas

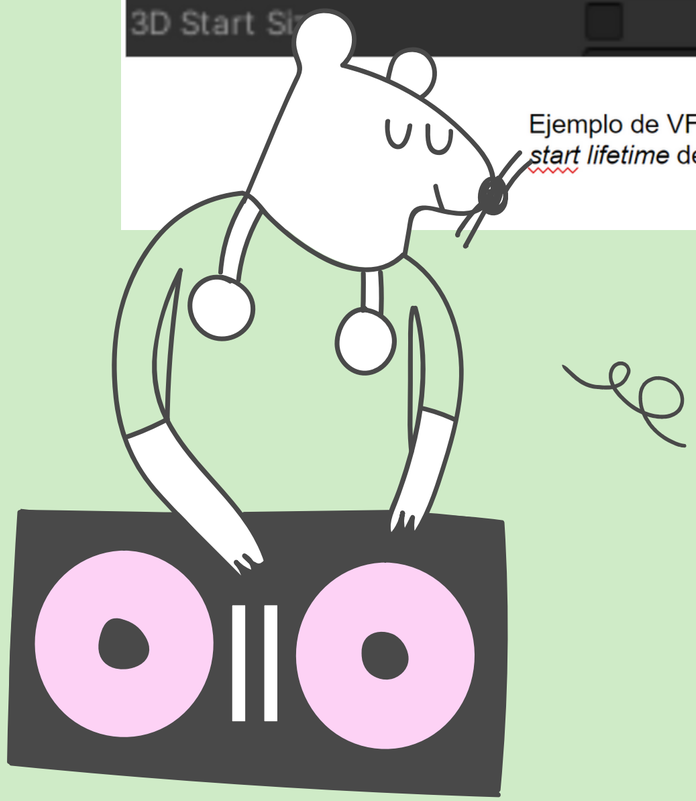
Usar eventos de animación

Medir el tiempo normalizado de un estado de mecanim por script





Ejemplo de VFX que empieza en el segundo 1 y cuyo parámetro de *start lifetime* debe ser editado para ajustarse al cambio.

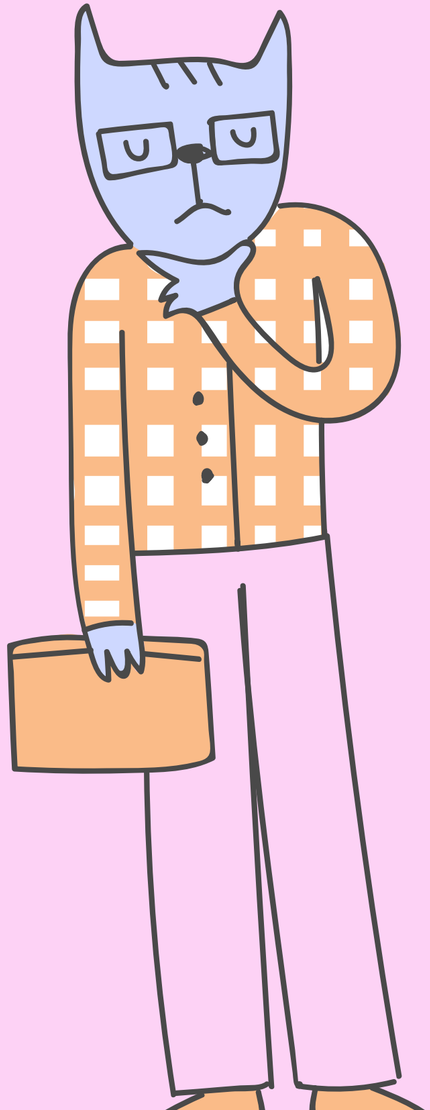


Start delay

Si queremos que un vfx empiece en el segundo 1 de una animación de 2 segundos entonces podemos asignar el valor de **Start Delay** en 1 y ajustar los parámetros necesarios para ajustarse a ese delay

Pros VS Contras

- Facil de configurar.
- Alto rendimiento, no necesita cálculos extra en tiempo de ejecución.
- Alta precisión de evaluación.



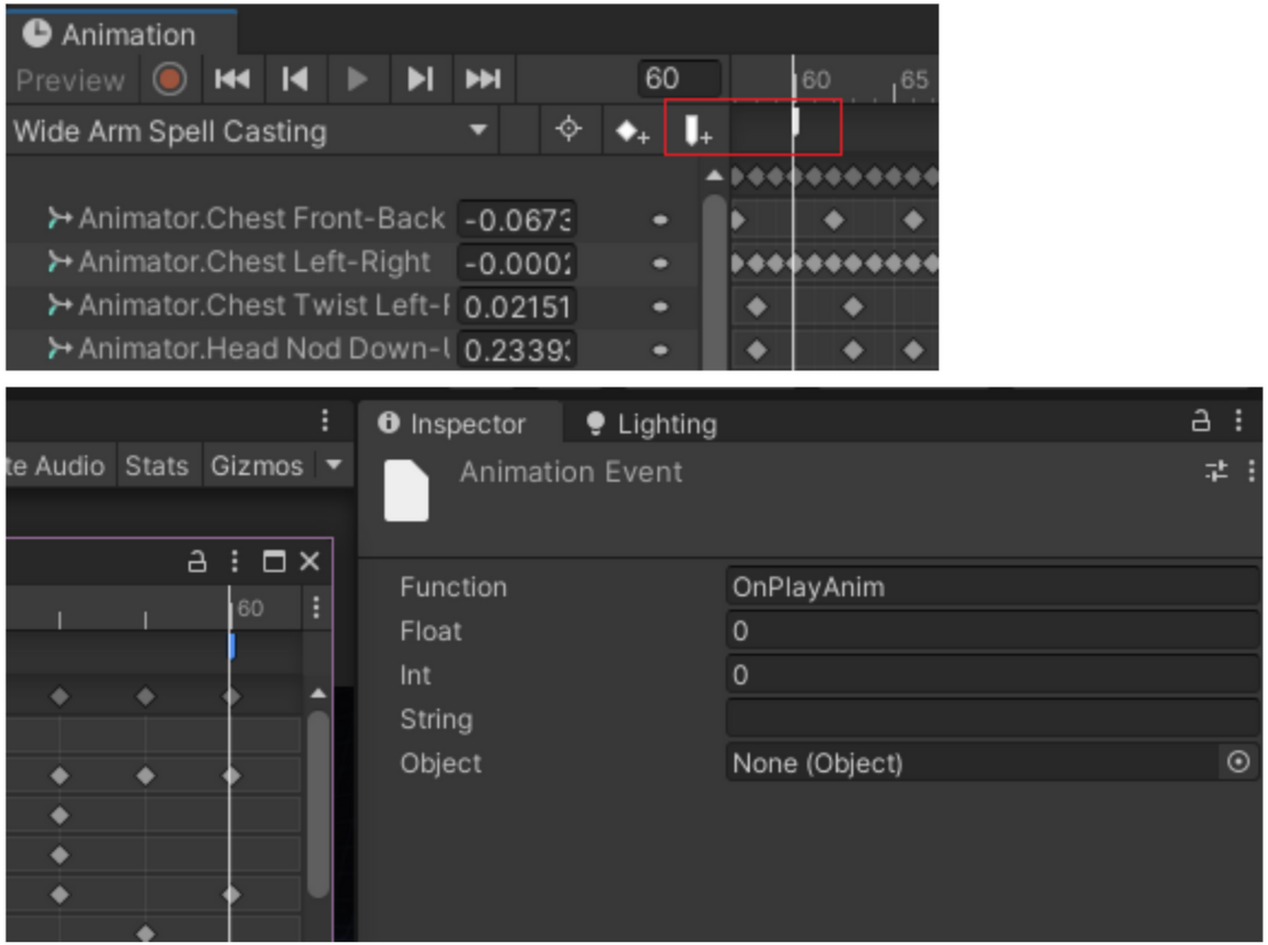
- Poca escalabilidad, hay que realizar ajustes si se cambia de animación

Eventos de animación

- Un evento de animación es un **Callback** que se llamará en un **keyframe** específico en el momento de ejecutar una animación. Estos eventos utilizan el mismo mecanismo de **SendMessage** el cual llama una función con un nombre específico en todos los **componentes** adjuntos al mismo **GameObject** en el que se encuentra el **componente Animator** que ejecuta la animación.

Eventos de animación

Para crear el evento, debemos modificar la animación y presionar el botón de evento con la barra de tiempo posicionada en el frame que lo queremos.
A continuación, seleccionamos el evento creado y le asignamos un nombre y sus parámetros



Eventos de animación

Por ultimo, declaramos una función con un nombre igual al campo **Function** del evento

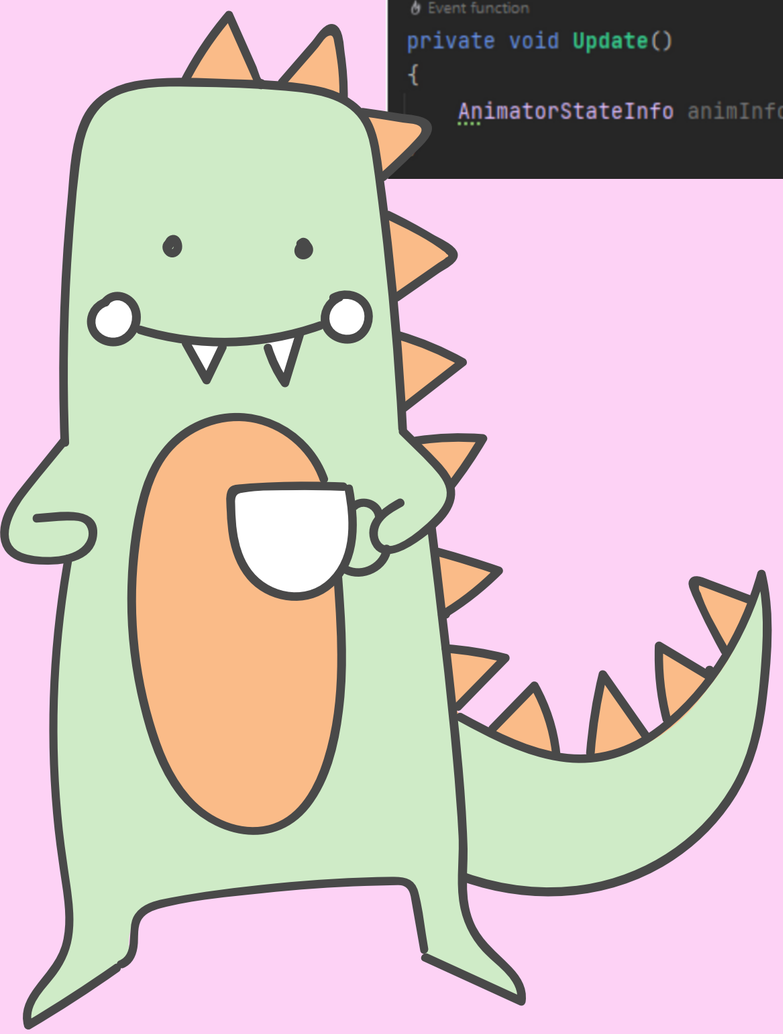
```
public void OnPlayAnim()
{
    Debug.Log( message: "Animation Playing");
}
```

Pros VS Contras

- Facil de configurar.
- Rendimiento medio-alto.
- Alta precisión de evaluación. Frame Perfect.

- Muchos archivos de animación duplicados (al tener un evento unity imprimirá un error si no hay componentes con esa función declarada).
- Puede generar reprocesos si se utiliza el mismo efecto en personajes fundamentalmente diferentes

Evaluación por animator



```
Event function
private void Update()
{
    AnimatorStateInfo animInfo = anim.GetCurrentAnimatorStateInfo( layerIndex: 0);
```

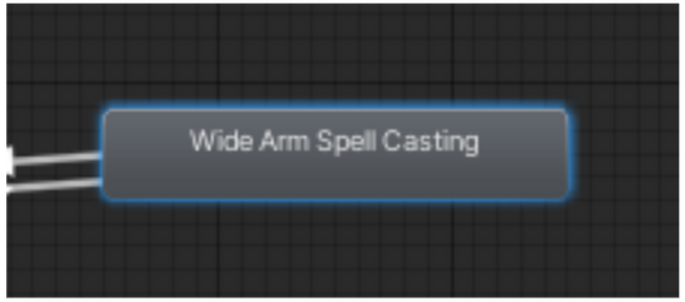
Para este método utilizaremos el estado de animator para ver en qué porcentaje de la animación se encuentra para reproducir nuestros efectos visuales.
Empezamos accediendo al estado de animator actual

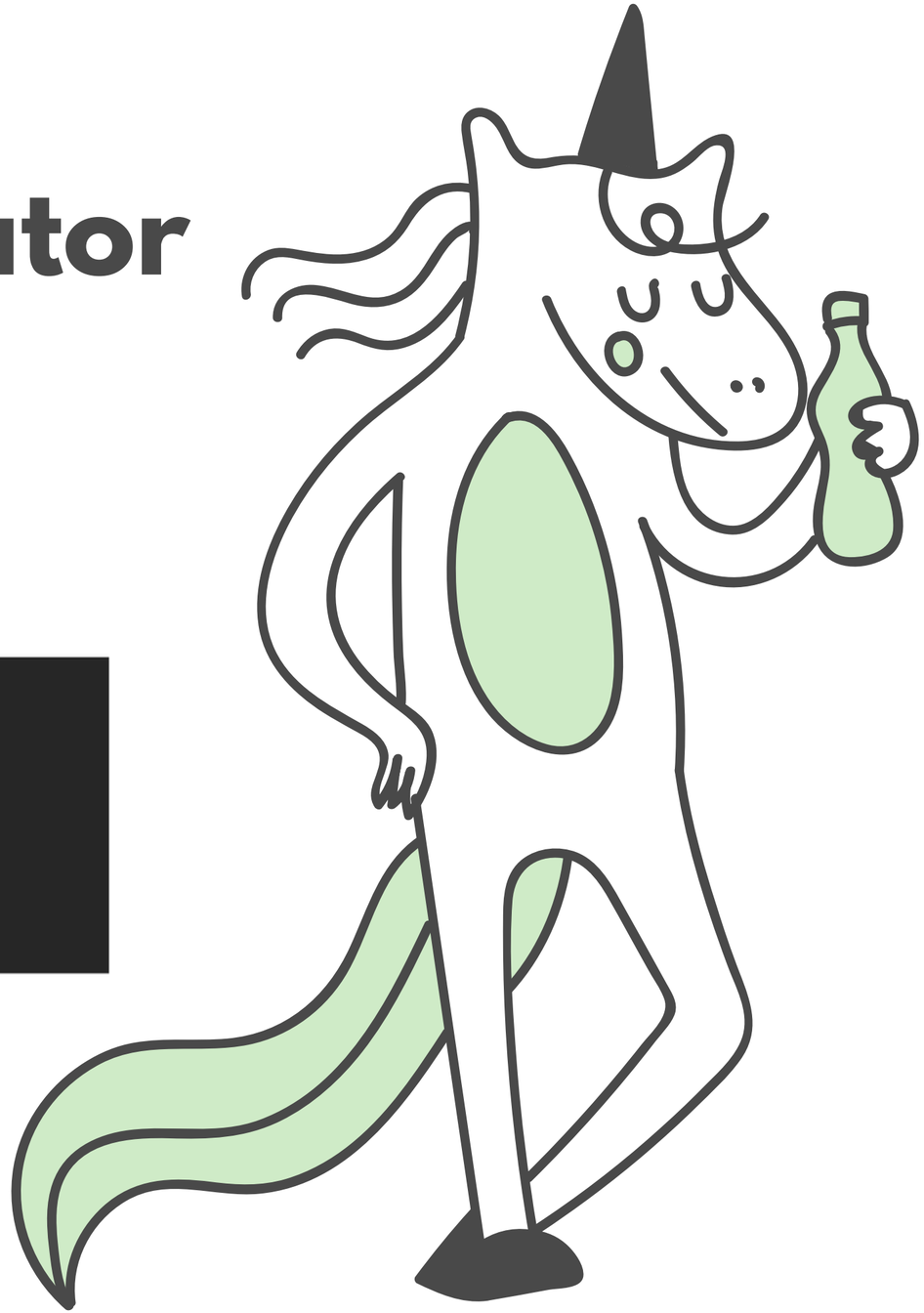
Evaluación por animator

A continuación, revisamos si nos encontramos en un estado de interés, podemos usar el método **IsName**.

```
if (animInfo.IsName("Wide Arm Spell Casting"))
{

}
```



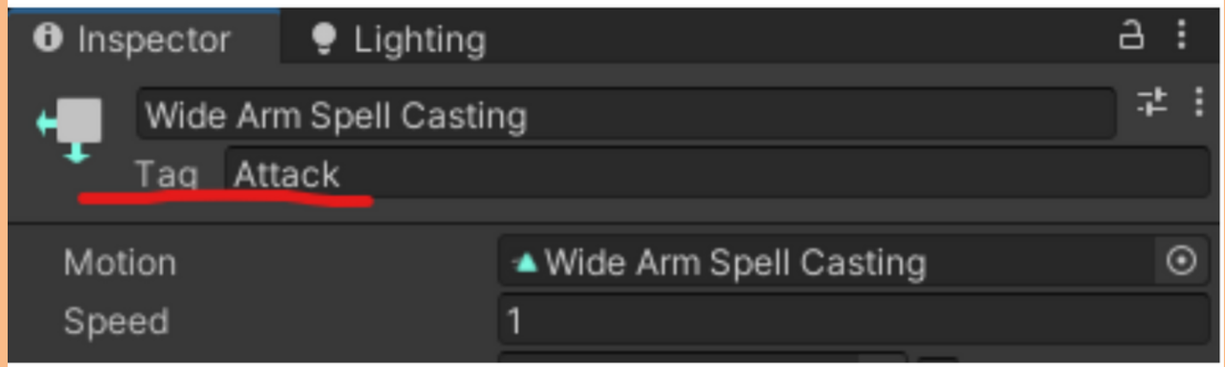


Evaluación por animator

Adicionalmente, podemos usar el método **IsTag** para obtener el stado de interés

```
if (animInfo.IsTag("Attack"))
{


}
```



Inspector del estado de animator mostrado anteriormente

Evaluación por animator

Además, podemos evaluar el **hash** del nombre o del tag

```
Event function
private void Awake()
{
    anim = GetComponent<Animator>();
    interestStateHash =
        Animator.StringToHash(name:"Wide Spell Arm Casting");
}
```

```
if (animInfo.shortNameHash == interestStateHash)
{

}
```

Evaluación por animator

```
// 50% de 2 segundos = 1 segundo
if (animInfo.normalizedTime > 0.5f)
{
    //Reproducir VFX
}
```

Por ultimo, podemos evaluar el tiempo actual de el estado de interés con la propiedad **NormalizedTime**