Guía de uso de la herramienta de Unity

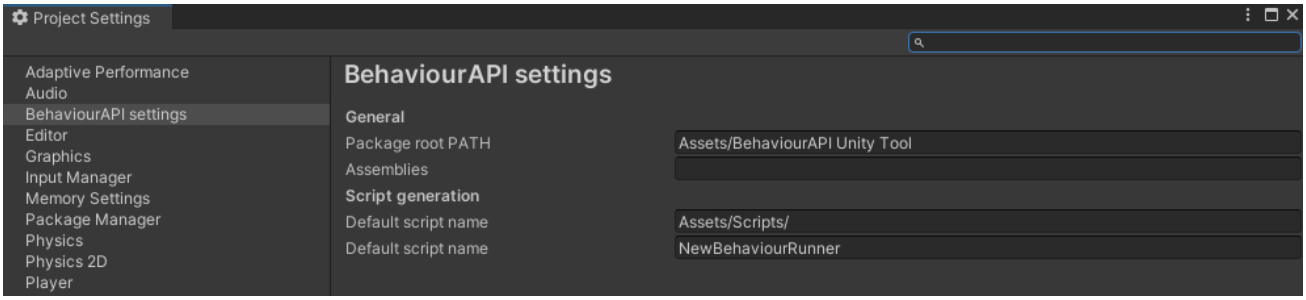
1. Como importar y configurar el paquete de Unity

El paquete "*Behaviour API Unity Tool*" incluye la API de creación de sistemas de comportamiento además de componentes para crear, ejecutar y depurar estos sistemas dentro de Unity.

Para usarlo se debe importar el paquete dentro de la carpeta "*Assets*". Si el paquete se importa en otra carpeta, se debe especificar la ruta en la configuración.

1.1 Configuración

Algunos elementos de la herramienta se pueden configurar en *Proyect Settings* > *BehaviourAPI Settings*.



- **Package root Path**: Indica el directorio raíz donde está el paquete de Unity.
- **Default script path**: El directorio por defecto donde se generan los scripts.
- **Default script name**: El nombre por defecto de los scripts generados.

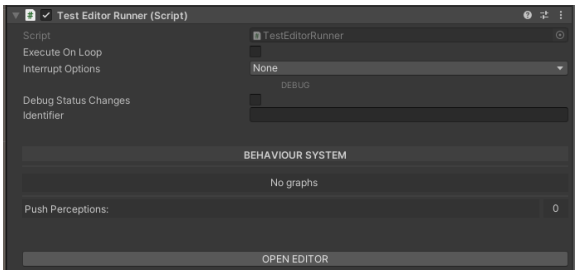
También es posible configurar los colores asociados a cada tipo de nodo en el editor.

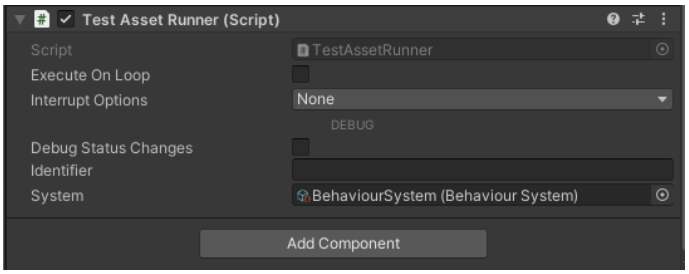
2. Crear sistemas de comportamiento con el editor visual

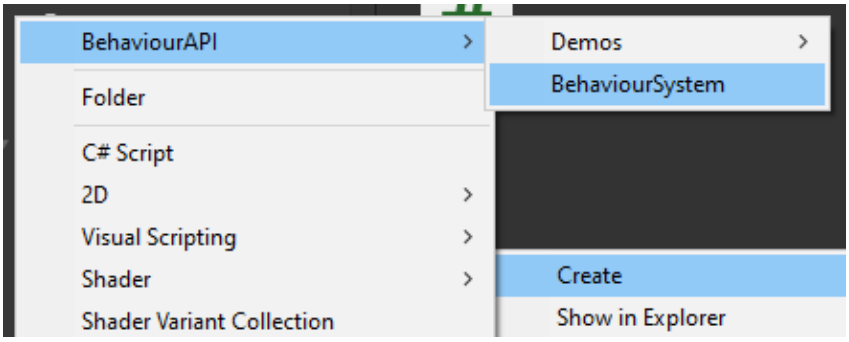
2.1. Scripts principales

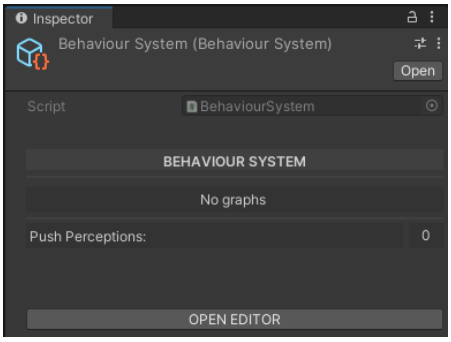
Se proporcionan dos formas de crear sistemas de comportamiento de forma visual.

- Vincular un sistema a un script de la escena: Para crear un sistema de comportamiento dentro de un *Script* usando la ventana del editor, el *Script* debe heredar de la clase *EditorBehaviourRunner*. Una vez creado se pulsa el botón *Open Editor* para abrir el editor visual.
- Vinculando el sistema a un asset del proyecto y referenciando el asset desde un script: Al hacer clic derecho sobre la carpeta de assets del proyecto y seleccionar *Create* > *BehaviourAPI* > *Behaviour*System se crea un archivo que contiene un sistema de comportamiento. Para editarlo se pulsa el botón *Open Editor* y para vincularlo a un *Script*, este debe heredar de la clase *AssetBehaviourRunner*.









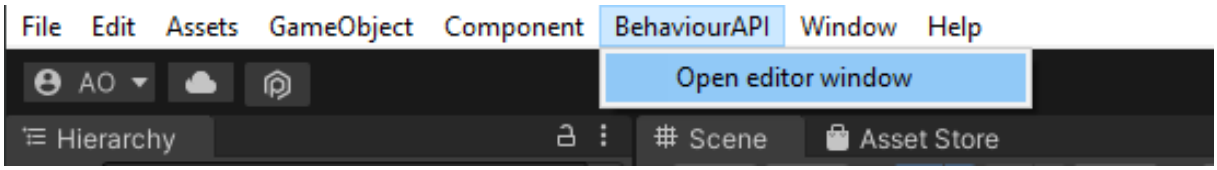
2.2 Modificar en el código el sistema creado

Es posible modificar el sistema de comportamiento creado en el método virtual *ModifyGraphs*, que recibe como parámetros diccionarios para buscar los grafos y percepciones push usando su nombre. Esto puede usarse para asignar acciones, percepciones, guardar referencias a nodos, etc.

```
protected override void ModifyGraphs (
          Dictionary<string, BehaviourGraph> graphMap,
          Dictionary<string, PushPerception> pushPerceptionMap)
{
   Node node = graphMap[“main tree”].FindNode<LeafNode>(“leaf 1);
   node.Action = new FunctionalAction(…);
}
```

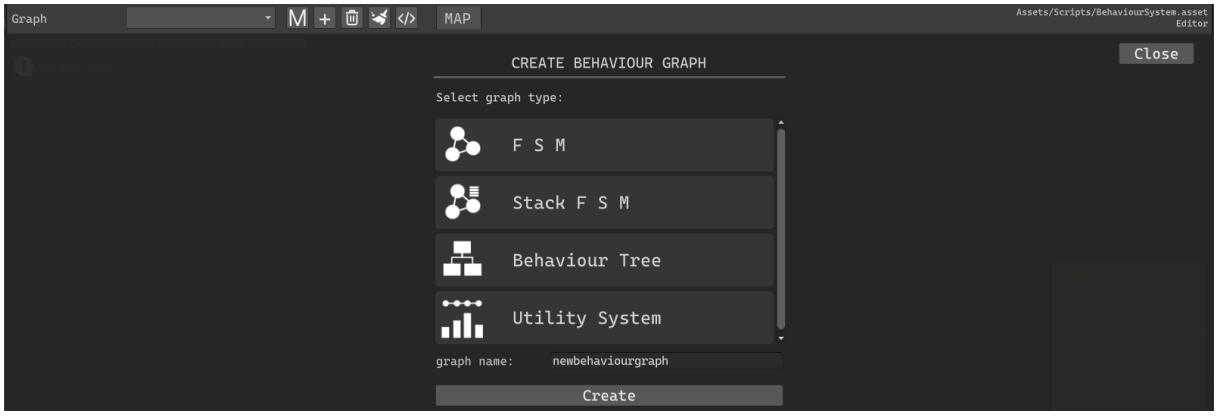
3. Guía para usar la ventana del editor de sistemas de comportamiento

La ventana del editor puede abrirse desde el menú *BehaviourAPI > Open Editor Window*. Para editar un sistema de comportamiento se debe pulsar el botón *Edit* en un componente de tipo `EditorBehaviourRunner` o en un asset de tipo `BehaviourSystem`.

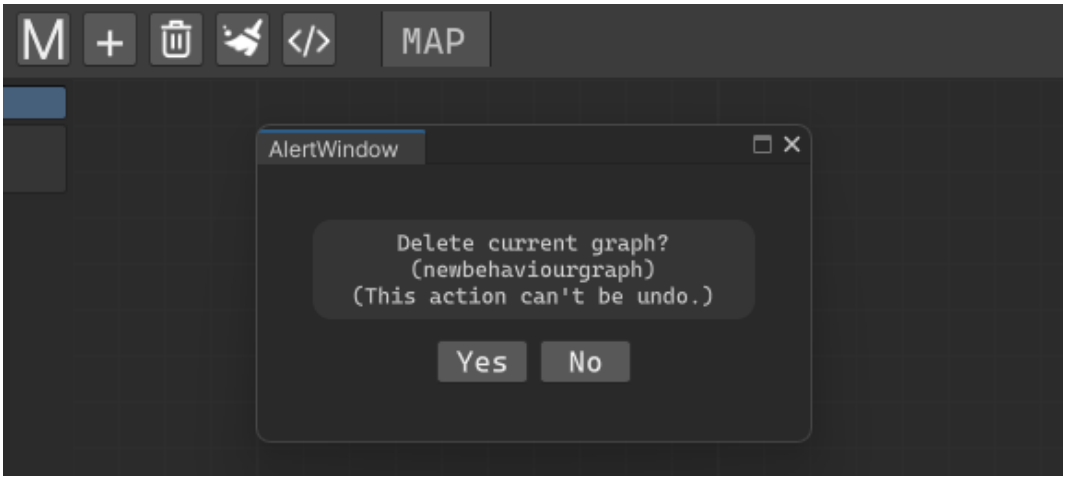


3.1. Añadir y borrar grafos

Si se abre un sistema de comportamiento sin grafos, se mostrará el panel de creación de grafos. Para crear uno hay que seleccionar el tipo y pulsar el botón “*Create*”. Es posible añadir nuevos grafos en cualquier momento pulsando el botón *“+”* de la barra de herramientas.

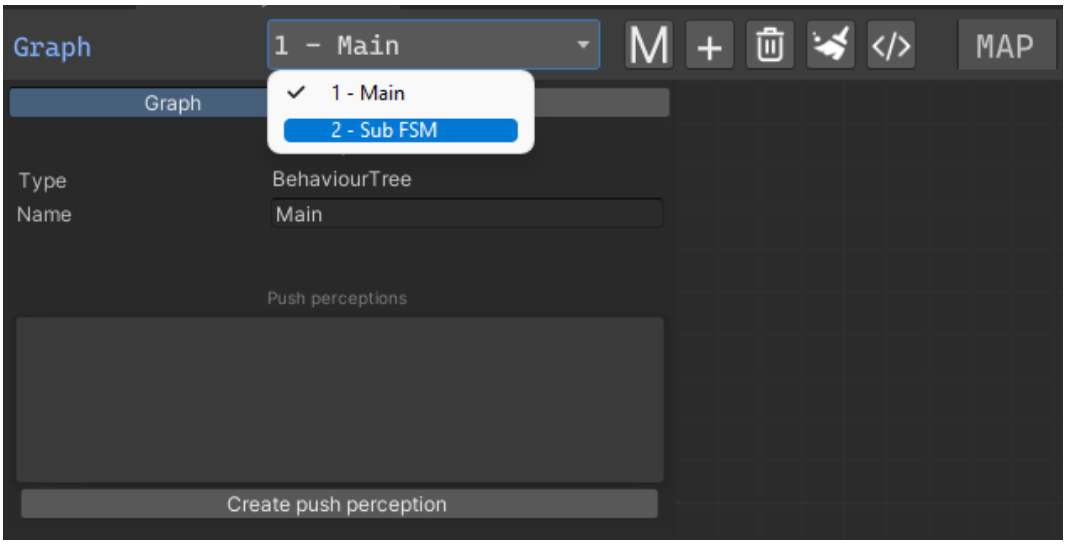


Para borrar el grafo actual se pulsa el **botón con el icono de la papelera** en la barra de herramientas. Aparecerá una ventana de confirmación antes de borrarse.



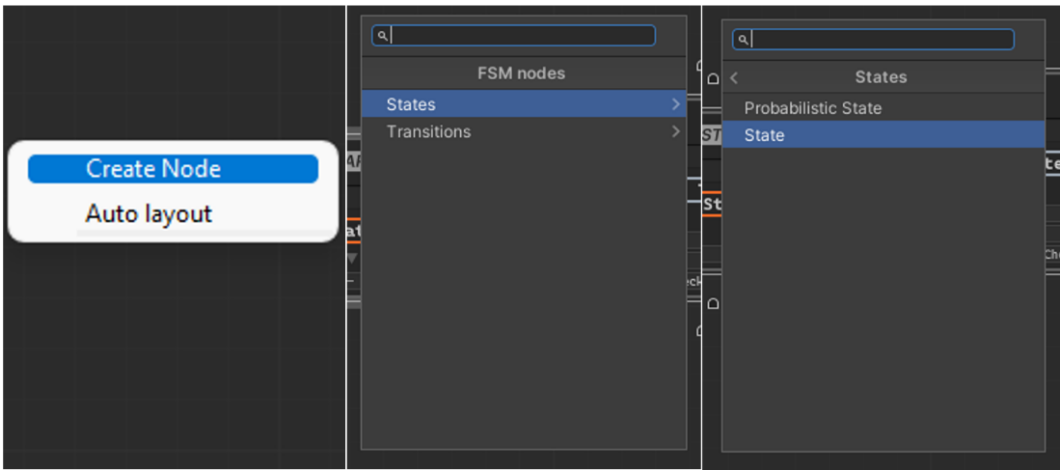
3.2. Seleccionar y editar grafo

Para cambiar el grafo seleccionado se usa el menú de selección de grafo de la barra de herramientas. Una vez seleccionado se cargará su representación y se pueden editar sus variables en el inspector.



3.3. Añadir y borrar nodos

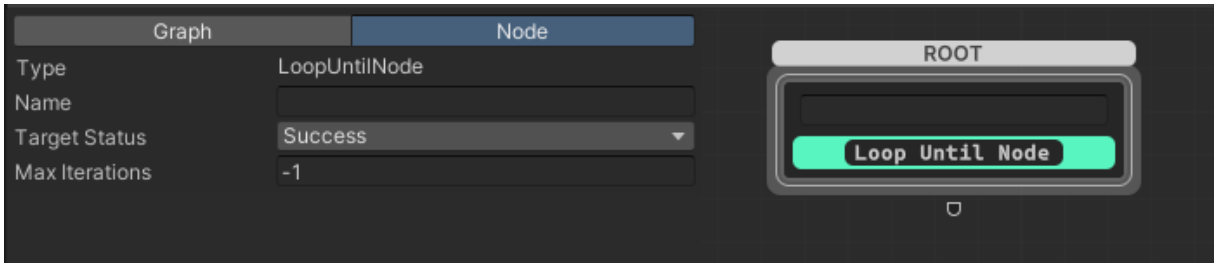
Para crear un nuevo nodo hay que hacer clic derecho para abrir el menú contextual y seleccionar *"Create Node"* o bien pulsar la tecla espacio. Al hacerlo se abre un menú que permite seleccionar un tipo de nodo de los disponibles para el grafo seleccionado.



Para borrar un nodo hay que hacer clic sobre el y pulsar la tecla suprimir.

3.4. Editar nodos

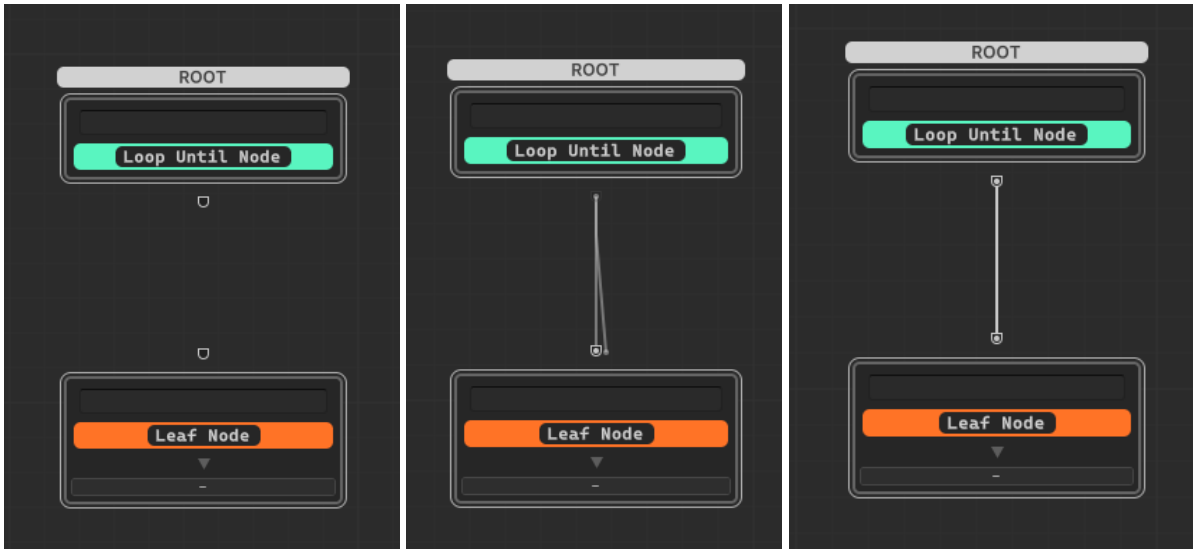
Hacer clic sobre uno de los nodos del editor abre el **editor de nodos**, que permite modificar el nombre del nodo y sus variables.



3.5. Crear y eliminar conexiones entre nodos

Los nodos pueden tener uno o varios “puertos” de entrada y de salida dependiendo de su tipo. Al hacer clic sobre uno de los puertos de un nodo y arrastrar hacia un puerto del tipo contrario en otro nodo se crea una conexión entre ellos, en la que el nodo con el puerto de salida es el “padre” y el otro nodo es el “hijo”.

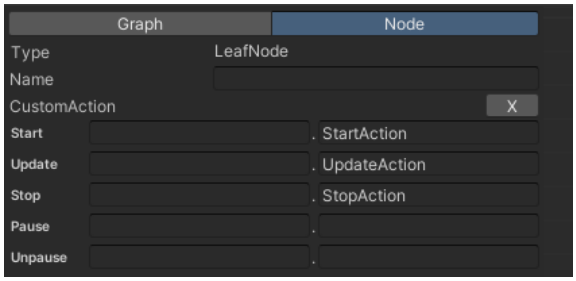
El tipo de puerto se distingue en que los puertos de entrada tienen el lado curvo hacia dentro y los de salida hacia fuera.

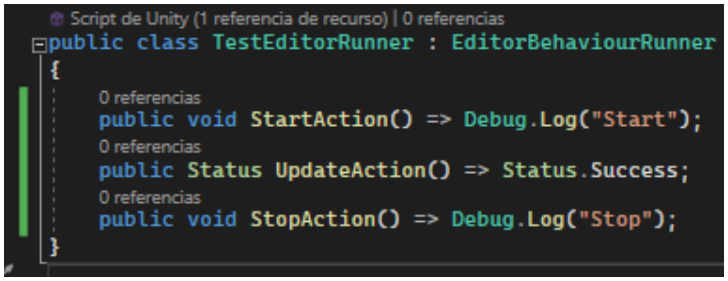


Para borrar una conexión hay que hacer clic sobre la arista y pulsar la tecla suprimir.

3.6. Asignar acciones y percepciones

En los nodos que pueden tener acciones o percepciones asignadas aparece un botón “Assign action”, que permite abrir un menú para elegir el tipo de acción. Las acciones de tipo *FunctionalAction* y percepciones de tipo *ConditionPerception* no están soportadas directamente, pero es posible simularlas usando *CustomActions* y *CustomPerceptions*. Para ello hay que definirse define en cada evento el nombre del componente y del método que se va a usar. Estos métodos no deben tener parámetros y su tipo devuelto debe coincidir con el tipo devuelto del evento.





3.7. Crear y asignar percepciones push

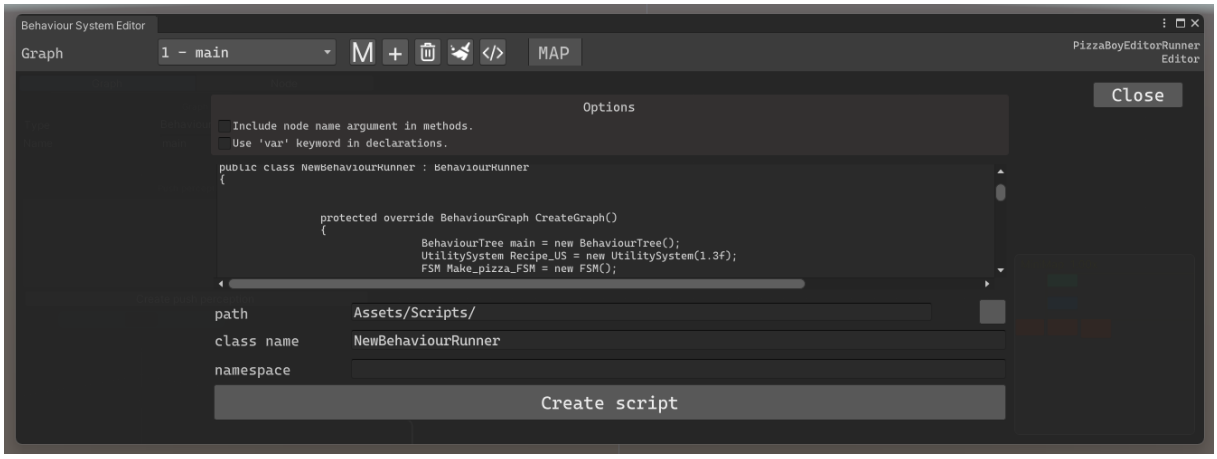
Para crear una percepción push se pulsa el botón *"Create push perception"* en el inspector de grafos. Una vez creada al hacer clic sobre una percepción push se abrirá su editor donde podemos cambiar el nombre y añadir y borrar nodos.

1 Crear percepción push

3.8. Generar un script a partir de un sistema de comportamiento

Al pulsar el botón con el icono "</>" en la barra de herramientas se abrirá un panel para generar un script a partir de los datos del sistema de comportamiento que se está editando. En esta ventana se puede especificar el nombre del script y la carpeta donde se va a guardar, además de otras opciones como el espacio de nombres.

Importante: Si se especifica un nombre y una ruta que ya existe, se sobrescribirá el archivo.

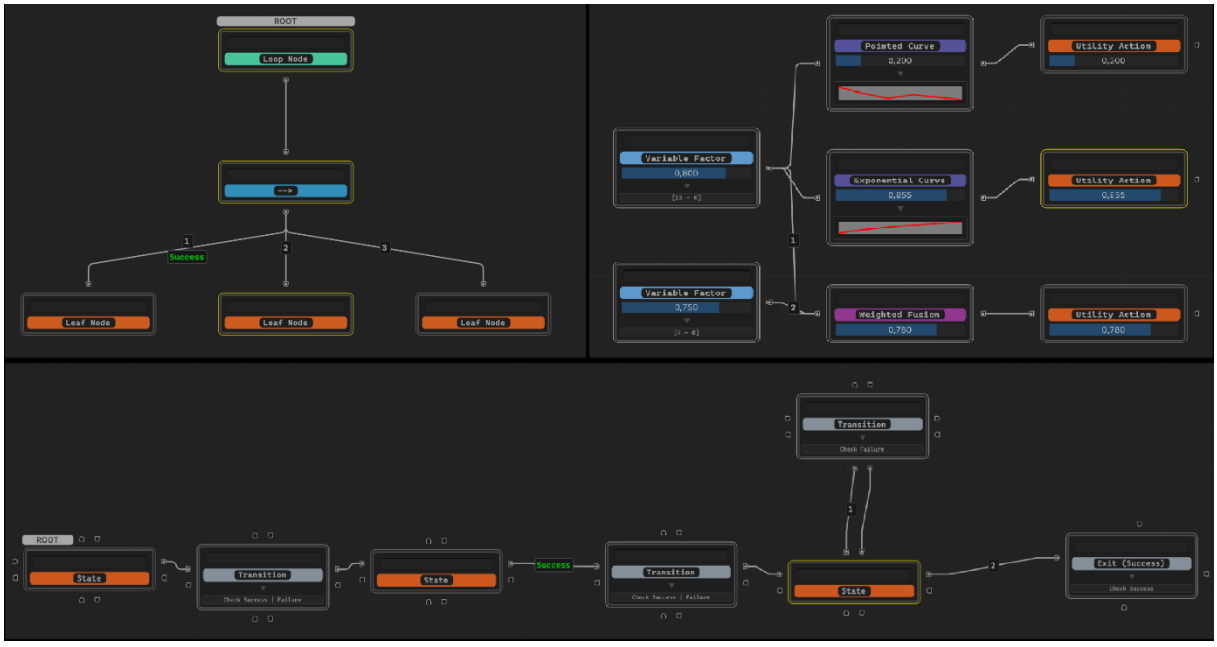


Se recomienda que el código generado se revise siempre antes de usarlo, ya que es posible que tenga errores de compilación, especialmente si se incluye nodos o acciones creadas por el usuario.

3.9. Como usar el depurador en tiempo real

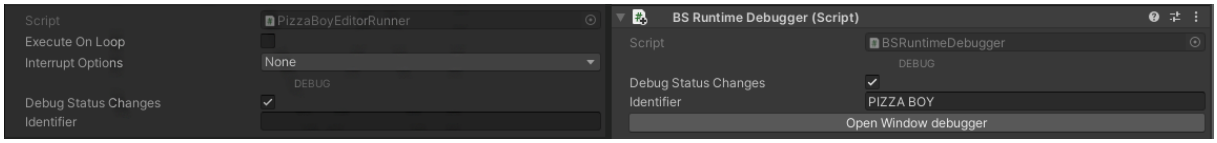
Es posible abrir la ventana del editor de sistemas de comportamiento en modo *"Debug"*. En este modo, todos los controles de edición están desactivados excepto la posibilidad de mover nodos, pero se ofrecen varias funcionalidades:

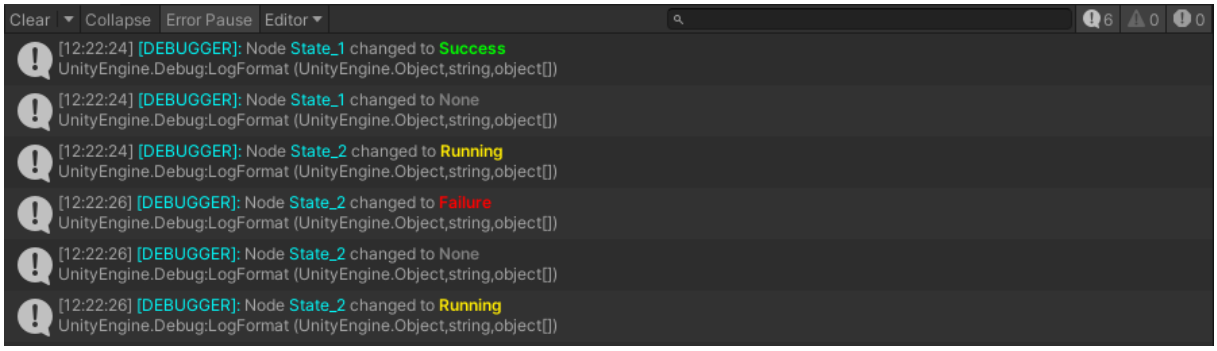
- Ver en tiempo real los nodos activos de cada grafo.
- En árboles de comportamiento, se muestra en las conexiones el valor devuelto por la rama concreta cuando ha terminado su ejecución.
- En máquinas de estados, se muestra el valor con el que ha terminado el último estado ejecutado.
- En sistemas de utilidad, cada nodo muestra su valor de utilidad actual.



Otra funcionalidad incluida en estos *Scripts* es el mostrar por consola los cambios de estado de ejecución de los nodos. Para ello hay que activar el flag *"Debug status changes"*.

También se puede especificar el identificador que aparece en los mensajes con la variable "Identifier".





3.10. Usar el depurador con sistemas generados por código

Para que el depurador tenga acceso a los grafos generados por código debemos incluir el componente *BSRuntimeDebugger*. Para registrar un grafo en el depurador en tiempo real se usa el método *RegisterGraph*, al que se puede pasar también el nombre del grafo para identificarlo en el editor.

```
public class ExampleRunner: BehaviourRunner
{
    Public BSRuntimeDebugger _bsRuntimeDebugger;

    protected override BehaviourGraph CreateGraph()
    {
          var fsm = new FSM();
          var subBt = new BehaviourTree();
          …
          _bsRuntimeDebugger.RegisterGraph(fsm, “Main FSM”);
          _bsRuntimeDebugger.RegisterGraph(subBt, “sub BT”);
          return fsm;
    }
}
```