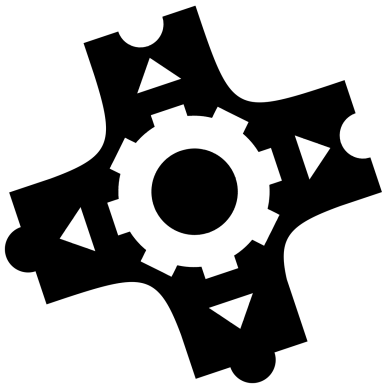


GAMEFLOW

Manual de Usuario

Versión 0.8 Beta

evasiongames

Introducción

GameFlow es un plugin o extensión para el motor Unity que permite simplificar y acelerar el desarrollo de videojuegos al reducir en gran medida la barrera que para muchos diseñadores de juegos y artistas supone el scripting o programación de la lógica y efectos especiales.

GameFlow se integra perfectamente en el editor de Unity añadiendo un soporte de programación visual basado en eventos y bloques de acciones que permite a diseñadores/as de juegos con nociones básicas de programación construir programas perfectamente funcionales en una fracción del tiempo que les llevaría hacerlo mediante lenguajes de scripting como C# o Javascript.

GameFlow incorpora además un conjunto de herramientas destinadas a facilitar las tareas de montaje de niveles de juego, como por ejemplo un editor visual de trayectorias, así como un completo repertorio de prefabs (partes de juego prefabricadas) configurables y listos para usar.

GameFlow está diseñado para la reutilización y dispone de una API propia de extensibilidad para aquellos usuarios avanzados que deseen crear sus propios bloques de acción o eventos a medida.

Programación visual

Llamamos programación visual o “visual scripting” a aquella en la que para crear un programa no es necesario escribir líneas de código en un editor de texto, sino tan sólo manipular elementos visuales. En el caso de GameFlow, estos elementos visuales son bloques de distinto tipo (acciones, condiciones, etc.) que pueden asociarse a los distintos objetos de juego (en adelante *GameObjects*) que haya en la escena.

Características

Programación

GameFlow permite diseñar la lógica o los efectos de un juego mediante programas (secuencias de acciones) que podemos asociar a los *GameObjects*, todo ello sin salir del propio editor de Unity.

Acciones

GameFlow incorpora de serie un repertorio de 150 acciones que cubren gran parte de las necesidades básicas requeridas en el desarrollo de un juego. Una lista completa está disponible en el capítulo Acciones.

Variables

Todas las acciones y herramientas de GameFlow aceptan no sólo valores concretos para sus propiedades sino también variables, lo cual permite construir programas flexibles y versátiles. Las variables, por supuesto, se definen también desde el propio editor.

Eventos

GameFlow facilita la programación basada en eventos (véase una explicación al respecto en el capítulo Eventos) mediante toda una serie de programas especiales (*Event Programs*) que se ejecutan automáticamente cuando ocurren determinados sucesos.

Listas

GameFlow soporta listas dinámicas de elementos en el propio editor. Estas listas pueden utilizarse para muchos propósitos durante el juego, pero son especialmente útiles para aplicar una misma acción a varios objetos en un solo paso.

Trayectorias

GameFlow incorpora un editor de trayectorias tanto lineales como basadas en curvas que pueden combinarse con algunas acciones para conseguir que los objetos del juego las sigan a lo largo del tiempo y también para otros efectos como la generación automatizada de escenarios con curvas (una carretera, por ejemplo).

Fuerzas

GameFlow permite definir fuerzas (vectores con una dirección y una magnitud representados como flechas) que pueden editarse de manera visual en el editor y que pueden aplicarse bajo demanda sobre los objetos que se desee. Es una manera fácil de comunicarse con el motor de Física de Unity para añadir efectos de movimiento a nuestro juego.

Temporizadores

Los temporizadores (*Timers*) son componentes que permiten la ejecución de programas en intervalos regulares o bien al agotarse un tiempo de espera. También pueden emplearse para construir marcadores de tiempo tipo cronómetro o tipo cuenta atrás.

Pools

Los pools son almacenes de objetos de un mismo tipo que permiten mejorar el rendimiento de un juego en aquellas situaciones en las que sea necesario mostrar continuamente una multitud de objetos en pantalla. GameFlow ofrece un soporte de Pools integrado y fácil de usar.

GameFabs

GameFlow incorpora un repertorio de plantillas de objetos (*Prefabs* en la jerga de Unity) que denominamos *Gamefabs* porque han sido diseñados como partes listas para usar y que pueden acelerar considerablemente el montaje de un videojuego. En este repertorio podremos encontrar desde controladores de teclado y de ratón, hasta generadores de objetos y marcadores listos para usar.

Persistencia

Con sólo un click, GameFlow se ocupa de la persistencia de los datos de tu juego de manera que sea muy fácil programar videojuegos que recuerden el estado de una partida.

Parametrización

GameFlow permite la construcción de *Prefabs* parametrizables desde el propio editor y sin tener que escribir código. De esta manera se pueden crear piezas listas para reutilizar en otros proyectos.

Automatización

Quizás la característica “oculta” más potente de GameFlow es que permite la ejecución de programas no sólo en tiempo de ejecución sino también en tiempo de edición. Es decir, se pueden construir fácilmente programas que generen o manipulen elementos de la escena desde el propio editor, facilitando así la construcción de niveles de juego.

Primeros Pasos

Instalación

Si GameFlow fue adquirido en la Unity Asset Store su paquete (archivo .unityPackage) ya estará instalado en el sistema y listo para usar en nuestros proyectos. El único paso requerido es el de importar el paquete dentro de nuestro proyecto de una de estas dos formas:

1. En caso de crear un nuevo proyecto marcar el check correspondiente al paquete *GameFlow.unityPackage* en la ventana de creación de nuevo proyecto para que aparezca directamente importado.
2. En caso de querer incorporar GameFlow a un proyecto ya existente, buscaremos la opción de menú *Assets* > *Import Package* > *GameFlow* y pulsaremos el botón "Import" en la ventana de importación que aparecerá a continuación.

Si por el contrario GameFlow no fue adquirido en la Unity Asset Store la instalación del paquete es también muy simple, pues basta con abrir o crear el proyecto donde queramos importar GameFlow y luego hacer doble click sobre el archivo *GameFlow.unityPackage* para que inmediatamente comience su importación.

Si lo deseamos también podemos copiar el archivo *GameFlow.unityPackage* a la carpeta *Standard Packages* que debe aparecer en la ruta de instalación de Unity para que de ese modo esté disponible como un paquete comprado en la Unity Asset Store.

Después de la importación y con independencia del modo en que la hayamos hecho, deberíamos observar que en la ventana *Project* de Unity aparece una nueva carpeta GameFlow conteniendo una carpeta de documentación, unos archivos .DLL y un archivo *Readme* conteniendo las notas de versión.

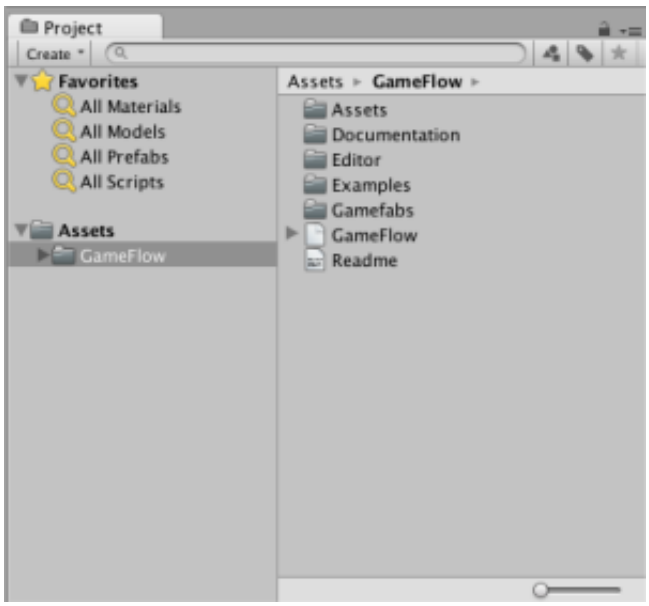
Actualización

El proceso de actualización es exactamente igual que el de instalación, pues se podría considerar una re-importación del paquete. Conviene, no obstante, leer siempre las notas de lanzamiento de la nueva versión antes de proceder a la actualización para asegurarnos de que la misma no incluyen cambios que puedan alterar el correcto funcionamiento de nuestro juego o aplicación.

Ayuda

Documentación

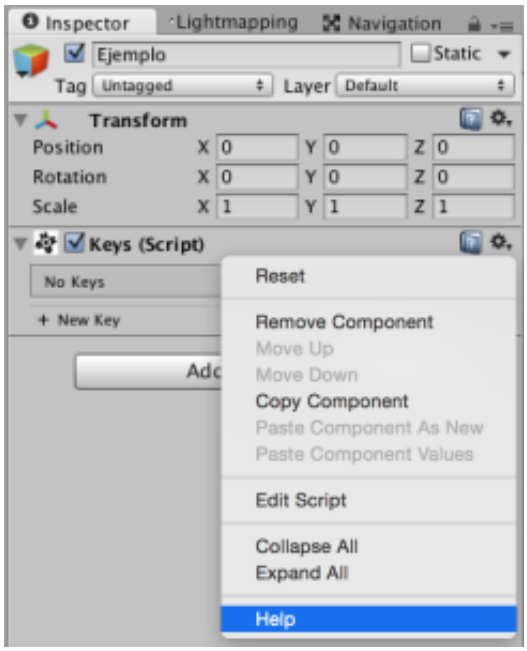
La documentación conteniendo el manual de usuario y la referencia de programación de la API puede encontrarse en la sección correspondiente de la web de GameFlow. Esta misma documentación está disponible también para consulta offline bajo la carpeta *GameFlow/Documentation* del proyecto Unity donde hayamos importando el paquete de GameFlow.



El manual de usuario está en formato PDF mientras que la documentación de referencia está en formato HTML, por el momento sólo en inglés y en español.

Ayuda contextual

En cuanto a ayuda en el editor de Unity, estará siempre disponible como última opción del menú contextual de cada elemento. El menú contextual es el que se despliega al hacer click con cualquiera de los botones del ratón en el icono de rueda dentada que aparece en la esquina superior derecha de cualquier componente o bloque de GameFlow en la ventana Inspector.



También se mostrará una ayuda rápida resumida en las ventanas de selección de acciones y condiciones, incluyendo un icono que nos da acceso a la página del manual del elemento seleccionado.

Tutoriales

En la sección de tutoriales (http://evasiongames.com/gameflow/es/tutorials) de la web de GameFlow se publican periódicamente nuevos tutoriales en vídeo que explican mediante ejemplos cortos desarrollados paso a paso técnicas concretas que pueden ser de utilidad a la hora de desarrollar videojuegos.

Foro

Los usuarios que lo necesiten pueden encontrar más ayuda en el foro oficial de GameFlow, un sitio destinado a compartir experiencias, dudas y conocimientos con el resto de la comunidad.

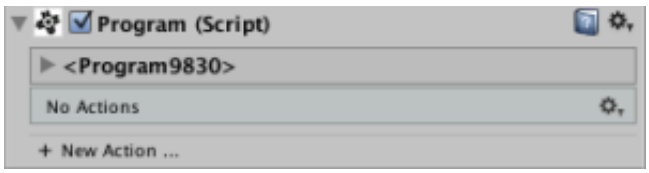
Conceptos Básicos

Componentes básicos

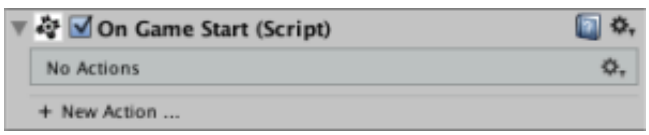
Contenedores

Son componentes de primer nivel que se muestran en la ventana del Inspector como otros componentes de Unity, y cuya característica básica es que contienen a otros componentes menores llamados **bloques**. En GameFlow se distinguen los siguientes tipos de contenedores:

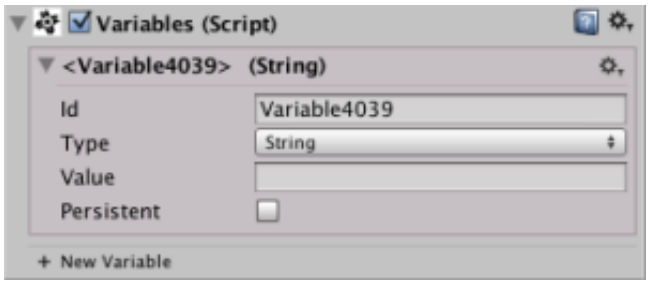
- **Programa (Program):** Una secuencia de acciones que se mantiene inactiva (es decir, no se ejecuta) hasta que se ordena su ejecución explícitamente.



- **Programa de evento (On ...):** Una secuencia de acciones que comenzará a ejecutarse automáticamente al detectarse un evento del tipo reconocido por el programa.



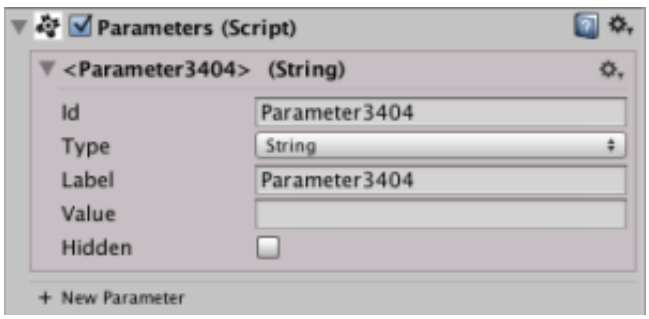
- **Variables:**Una lista que muestra todas las variables definidas dentro del objeto.



- **Teclas (Keys):** Una lista que muestra todas las teclas definidas dentro de ese objeto.

- **Fuerzas (Forces):** Muestra todas las fuerzas definidas dentro del objeto.

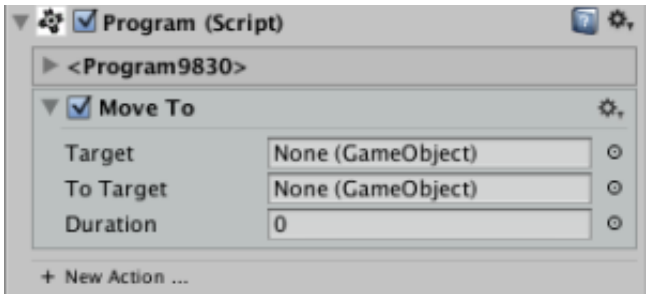
- **Parámetros (Parameters):** Muestra todos los parámetros definidos en ese objeto. Los parámetros son un tipo especial de variables como veremos en el capítulo correspondiente.



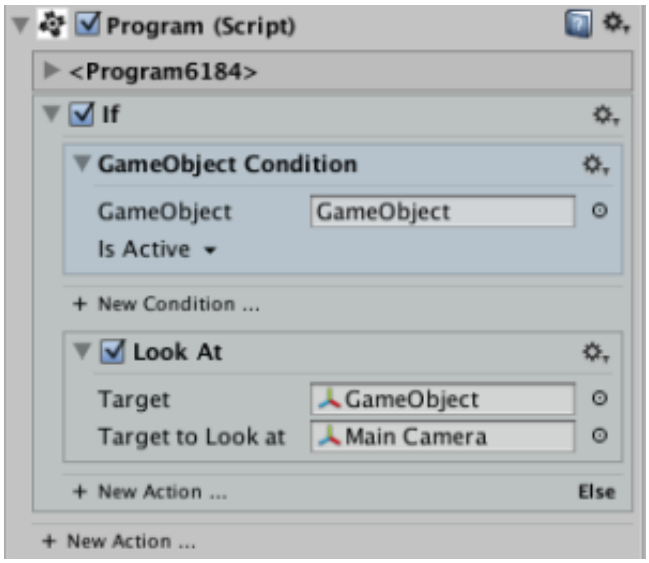
Bloques

Son elementos que no pueden vivir de manera independiente fuera de un contenedor y que juegan un papel concreto dentro de la funcionalidad de dicho contenedor. En GameFlow encontramos los siguientes tipos:

- **Acción (Action):** Un elemento del programa que realiza (si está habilitada) una tarea muy concreta en función de unos parámetros o propiedades. La mayoría de las acciones se ejecutan de manera instantánea, pero otras por el contrario se ejecutarán durante el intervalo de tiempo que se le especifique.



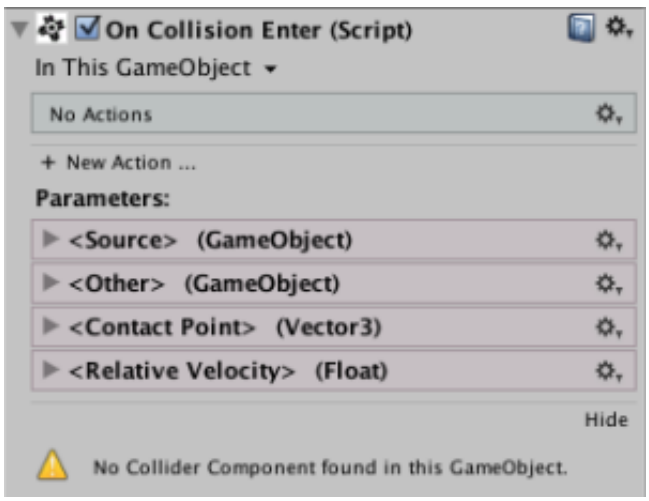
- **Condición (Condition):** Un elemento del programa que realiza una determinada evaluación para devolver un resultado “Verdadero” o “Falso” que podrá ser utilizado por acciones condicionales como *If* o *While* para controlar el flujo de un programa.



- **Variable:** Una unidad dinámica de almacenamiento de información que puede guardar un dato de un tipo determinado. La gran potencia de las

variables es que pueden utilizarse como valor para las propiedades de prácticamente cualquiera de los elementos de GameFlow: acciones, condiciones, herramientas o incluso para otras variables.

- **Variable predefinida (Built-in Variable):** Un tipo especial de variable que no puede ser editada porque su tipo y valor ya vienen predefinidos. Suelen servir para consultar información del sistema como hora y fecha, resolución, plataforma, etc.
- **Parámetro (Parameter):** Un tipo especial de variable pensado que sirve tanto para mostrar los parámetros de los programas de evento (en la imagen de la derecha) como para permitir la construcción de Prefabs parametrizables.



- **Tecla (Key):** Una definición de tecla o manera de indicar a GameFlow que queremos que controle el estado de esa tecla en concreto.
- **Fuerza (Force):** Una definición de una fuerza que puede visualizarse en el editor.

Herramientas

Son componentes independientes básicos que tienen su propia lógica y su propia forma de edición visual, pero que están integrados con GameFlow al soportar variables y ser manipulables por medio de una serie de acciones que trabajan específicamente con ellos. Podemos acceder a ellas a través del navegador de componentes y seleccionando GameFlow > Tools. En GameFlow se distinguen las siguientes herramientas.

- **Lista (List):** Un componente que permite almacenar un conjunto secuencial de valores con un mismo tipo base. Al igual que las variables, las listas en GameFlow son dinámicas y pueden ser manipuladas para añadir, insertar, reemplazar o eliminar elementos tanto en tiempo de ejecución como en tiempo de edición.
- **Temporizador (Timer):** Un componente que permite enviar eventos de temporización en intervalos regulares de tiempo lo cual, en combinación con un programa de evento como *On Timer Expire* permite la ejecución de acciones cada cierto tiempo o bien al agotarse un tiempo de espera.
- **Trayectoria (Path):** Un componente que define una trayectoria en el espacio de la escena a partir de una serie de puntos, y que puede manipularse visualmente y ser recorrida por los objetos de nuestro juego mediante la acción *Follow Path*.
- **Pool:** Un componente que define una colección de objetos pre-instanciados que pueden usarse en tiempo de ejecución. El uso de pools mejora el rendimiento en situaciones en las que es necesario mostrar continuamente muchos objetos en pantalla, un ejemplo podría ser un juego de naves con muchos disparos en pantalla.
- **Área (Area):** Un componente que permite definir un área cúbica en la escena y que mientras esté activado y desplegado en el Inspector se dibujará en todo momento. Se utiliza para definir áreas del juego sin tener que recurrir al uso de Box Colliders.
- **Controlador de eventos (Event Controller):** Un componente que permite controlar qué eventos serán detectados y cuáles serán ignorados por el objeto.
- **Nota (Note):** Un componente simple que permite añadir una nota de texto al objeto, lo cual puede ser útil, por ejemplo, para recordar el motivo por el que configuramos ciertos parámetros de una manera concreta en su momento.

Complementos

GameFabs

Los *GameFabs* (contracción de Game Prefabs) son *GameObjects* o jerarquías de *GameObjects* configurables que ofrecen al usuario componentes y/o lógicas frecuentemente utilizados en videojuegos y cuya incorporación a un juevo es trivial, por lo que pueden acelerar muchísimo el montaje de prototipos de juegos. Una lista de GameFabs está disponible en el capítulo GameFabs.

Utilidades

Son herramientas integradas que no pueden considerarse componentes y que normalmente se ocupan de realizar operaciones especiales que pueden ir más allá de la edición. Un ejemplo de utilidad sería el depurador integrado que GameFlow incorporará en una futura versión.

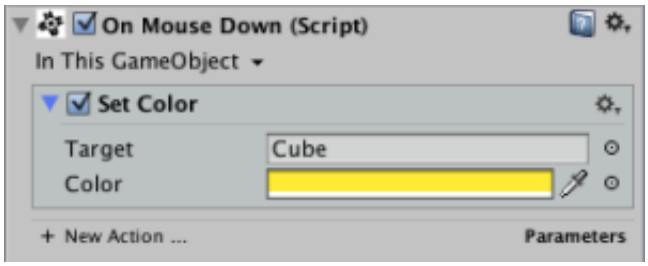
Plantillas

Las plantillas son proyectos prefabricados que incorporan un sistema de configuración guiado y de un nivel superior al de los *GameFabs*, pues se ocupa de definir la configuración e interrelación de estos. El soporte de plantillas se incorporará en futuras versiones de GameFlow.

Programas

Un programa es un componente que contiene una lista ordenada de acciones que el motor de GameFlow ejecutará de manera secuencial permitiéndonos tanto dotar de vida a nuestros objetos de juego como controlar el flujo del propio juego, de ahí el nombre de la herramienta.

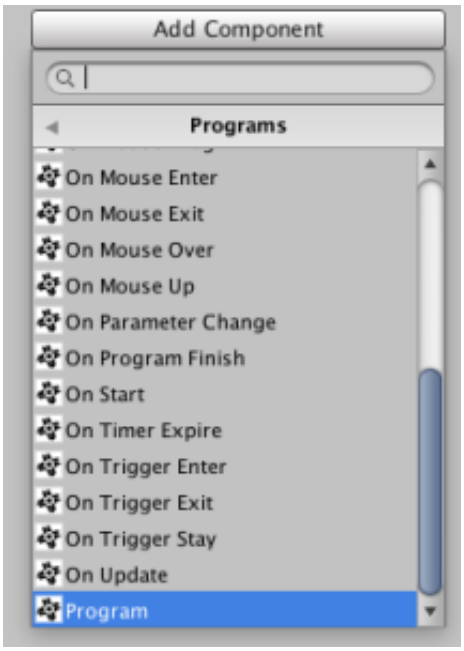
Un programa puede, por ejemplo, encargarse de controlar el movimiento de nuestro personaje principal, mientras que otra serie de programas puede ocuparse de manejar los enemigos, controlar los marcadores de juego e ir cambiando la música al llegar a ciertas partes del escenario. Básicamente los programas dan vida a un juego.



Creando programas

Como todo componente, un programa comienza a existir cuando es añadido un objeto de juego determinado. Esto lo podemos hacer usando el botón *Add Component* que aparece al final de la lista de componentes en la

ventana Inspector, buscando a continuación el menú *GameFlow* > *Programs* y eligiendo a continuación uno de los tipos de programas disponibles.



Como veremos más adelante, atendiendo al modo en que un programa comienza a ejecutarse, en GameFlow se distinguen dos tipos básicos de programas:

- **Programa inactivo:** programa se mantiene inactivo hasta que se ordena su ejecución mediante bien una acción *Execute Program* o bien la opción *Execute* del menú contextual.
- **Programa de Evento:** programa que, si está habilitado, comienza a ejecutarse automáticamente al detectarse un evento del tipo reconocido en el rango de escucha.

Program es el único programa inactivo, siendo todos los demás programas de evento. Además todos los programas de evento comienza con “On” por lo que es fácil distinguirlos.

Añadiendo acciones

Las acciones son los bloques a partir de los que se construyen los programas. Pueden considerarse pequeños componentes especializados que realizan tareas muy concretas en función de unas propiedades que se le especifican dentro de su propia área de edición en la ventana Inspector.

Para añadir acciones a un programa que hayamos añadido podemos hacerlo también usando el botón *Add Component* del Inspector y buscando a continuación el menú *GameFlow* > *Actions*, pero tendremos más control si utilizamos el botón + *New Action* que aparece en la parte inferior del área del programa en el Inspector.

Dado que este botón aparece en el pie de cada contenedor de acciones, sea el propio programa o sea una acción capaz de contener a otras acciones como *While*, usarlo nos permitirá decidir en qué punto exacto del programa queremos añadir la acción, lo cual será más cómodo que andar reordenando luego las acciones.

Además, usar este botón nos permitirá buscar acciones de una forma más fácil pues en este caso en vez de usar el menú de componentes estándar de Unity, se abrirá una ventana especial de selección de acciones preparada especialmente para ese cometido.

Esta nueva ventana nos mostrará en una única lista todas las acciones disponibles, tanto las que trae la herramienta de serie como aquellas acciones creadas a medida por el usuario mediante la API de GameFlow, junto con una ayuda rápida integrada que nos permitirá hacernos una mejor idea del propósito de cada acción.

Para elegir una acción sólo tenemos que movernos con las flechas arriba y abajo y pulsar *Enter*, o bien hacer doble click directamente sobre la acción que deseemos incorporar al programa.

La ventana cuenta además con un filtro de búsqueda que nos permitirá filtrar rápidamente las acciones para que sólo queden aquellas que contengan en su nombre el texto introducido. En un futuro el mecanismo de filtrado de esta ventana se irá mejorando para incorporar también búsqueda mediante etiquetas, de manera que el usuario tenga más posibilidades de encontrar una acción adecuada a sus propósitos.

Orden de ejecución de acciones

Como indicamos al comienzo del capítulo, las acciones contenidas en un programa se ejecutan en orden secuencial, es decir, se comienza con la primera acción del programa y se continúa con su ejecución hasta que finalice, momento en el cual se comienza con la ejecución de la segunda acción de la lista, y así sucesivamente hasta finalizar el programa.

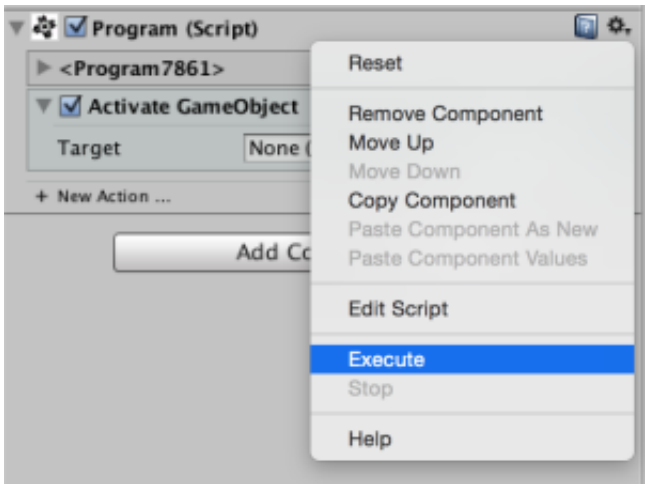
Como veremos en el capítulo dedicado a **Control de flujo**, este es el orden por defecto de ejecución para un programa lineal, pero mediante el uso de acciones especiales como *Group* o *If* es posible variar este orden de ejecución para que no sea totalmente secuencial y se adapta mejor a distintos casos de uso.

Tiempo de ejecución

La mayoría de los programas de evento sólo se ejecutarán en tiempo de ejecución (es decir, cuando le hemos dado al botón de *Play*) y no durante tiempo de edición (es decir, mientras estamos trabajando en el editor de Unity) debido a que la mayoría de los eventos no se generan hasta que el juego no está en marcha.

No obstante, es importante comentar que GameFlow permite que cualquier *Program* (programa inactivo) sea ejecutado también en tiempo de edición mediante la opción *Execute* de su menú contextual, lo cual puede ser muy útil tanto para realizar pruebas rápidas de funcionamiento sin tener que

arrancar el juego como para automatizar tareas de edición mediante programación.



Acciones

Las acciones son los bloques a partir de los que se construyen los programas. Pueden considerarse pequeños componentes especializados que realizan tareas muy concretas en función de unas propiedades que se le especifican dentro de su propia área de edición en la ventana Inspector.

La mayoría de las acciones se ejecutan de manera puntual e instantánea cuando les llega el turno dentro de su programa, pero otras acciones, por el contrario, están diseñadas para ejecutarse durante un intervalo de tiempo, motivo por el cual suelen incluir una propiedad Duración *(Duration)*.

Además de esa, otra de las diferencias de una acción respecto un script típico de Unity (*MonoBehaviour*) es que las acciones están diseñadas para aplicarse sobre un objetivo que no tiene necesariamente que coincidir con el GameObject que contiene al programa al que pertenece la acción.

Las acciones en GameFlow se representan como bloques colapsables que tienen su propia interfaz a modo de subcomponentes. Cuando una acción está colapsada, sólo muestra su barra de título y cuando esté desplegada mostrará además todas las propiedades. El color de estos bloques es intencionadamente ligeramente distinto al color de fondo por defecto de la ventana de Inspector.

Existen acciones que pueden a su vez contener a otras acciones, como por ejemplo acciones como *For* o *Repeat*, que permiten construir bucles de repetición. En estos casos, las acciones contenidas se mostrarán dentro del área de la acción contenedora con un nivel más de indentación.

Operaciones de edición

La mayoría de las operaciones que permiten las acciones en tiempo de edición están disponible en su menú contextual (rueda dentada en la parte derecha del área de título de la acción).

Colapsar / Expandir

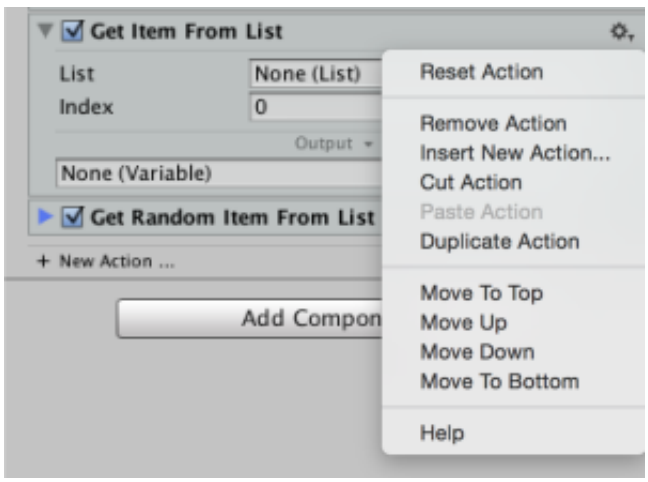
Una acción puede ser expandida o colapsada pulsando bien en la flecha en la esquina superior izquierda de su título, o bien pulsando en cualquier punto del área de título de la acción.

Activación / Desactivación

Para activar o desactivar una acción basta con marcar o desmarcar el checkbox que precede a su título. La activación o desactivación no afecta la edición de las propiedades de la acción, sólo a su ejecución. Una acción desactivada no será ejecutada por el programa, que continuará su ejecución en la siguiente acción activada.

Reordenación

Una acción puede ser movida a otro lugar del programa bien mediante las acciones de movimiento de su menú contextual (*Move to Top, Move Up, Move Down, Move to Bottom*) o bien mediante una operación de arrastrar y soltar *(drag and drop)* iniciada con un click en su barra de título.



Una acción puede ser transferida también a otro programa mediante *drag and drop* siempre y cuando ambos programas estén contenidos dentro del mismo *GameObject*.

Por último es posible también mover una acción de un programa a otro haciendo uso de las acciones *Cut Action* y *Paste Action* del menú contextual, pero de nuevo con la limitación de que ambos programas, tanto el de origen de la acción como el de destino estén contenidos dentro del mismo *GameObject*.

Nota: no existe actualmente una acción *Copy Action*, pero se planea su implementación en futuras versiones.

Reinicio

Una acción puede ser reiniciada para que tome sus valores por defecto mediante la opción *Reset* de su menú contextual.

Eliminación

Una acción puede ser eliminada mediante la opción *Remove Action* de su menú contextual.

Inserción

Es posible insertar una nueva acción delante de otra usando la opción *Insert New Action* del menú contextual de dicha acción.

Duplicación

Es posible duplicar una acción usando la opción *Duplicate Action* de su menú contextual.

Nueva acción

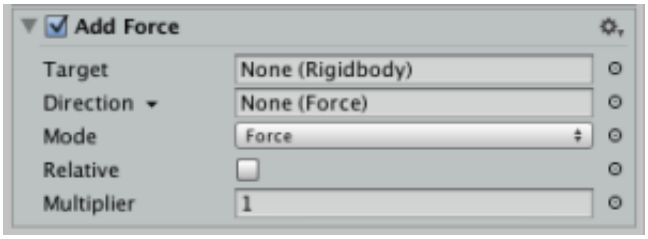
En aquellas acciones que tenga la capacidad de contener a otras acciones como por ejemplo *If* aparecerá un botón + *New Action* en la parte inferior del área de la acción que al ser pulsado mostrará la ventana de selección de acciones para añadir acciones.

Edición de propiedades

La edición de propiedades de una acción se realiza como en cualquier otro componente de Unity, con algunas ligeras diferencias:

- Algunas propiedades pueden tener opciones adicionales en forma de menús contextuales asociados a sus etiquetas, lo cual se indicará en la interfaz con una pequeña flecha apuntando hacia abajo a la derecha del título. Un ejemplo lo encontramos en la propiedad *Direction* de la acción *Add Force*, que permite especificar el tipo de valor de entrada mediante

este menú contextual incorporado a la etiqueta.



- La mayoría de las propiedades, sean del tipo que sean, incorporan en la parte derecha un pequeño círculo mediante el cual es posible asociarle, como veremos en el siguiente capítulo, una variable.
- Algunas acciones (normalmente aquellas cuyo nombre comienza con la palabra *Get*) incorporan un campo especial para especificar una variable de salida, y dicho campo puede ser arrastrado como una variable más para una mayor facilidad a la hora de componer los programas.

Ayuda

Es posible consultar la ayuda de una acción mediante la opción Help de su menú contextual.

Ejecución de acciones

Aunque la mayorías de las acciones disponibles en GameFlow están pensadas para funcionar en tiempo de ejecución, sólo algunas de estas funcionarán también en tiempo de edición. Cuando una acción no pueda ejecutarse en tiempo de edición GameFlow lo indicará enviando un mensaje de advertencia a la consola.

Por otra parte, existen también acciones que sólo realizarán una tarea efectiva al ser ejecutada en tiempo de edición y que serán ignoradas en tiempo de ejecución por no tener sentido en ese contexto.

Acciones disponibles

La siguiente es una lista de las acciones actualmente incorporadas en GameFlow organizadas por funcionalidad.

Activación

- Activate GameObject
- Activate GameObjects In List
- Deactivate GameObject
- Deactivate GameObjects In List
- Disable Behaviour
- Disable Behaviours In List
- Disable Collider
- Disable Program
- Enable Behaviour
- Enable Behaviours In List
- Enable Collider
- Enable Program

Variables

- Decrement Variable Value
- Divide Variable Value
- Increment Variable Value
- Limit Variable Value
- Multiply Variable Value
- Set Variable Value
- Toggle Variable Value

Listas

- Add Item To List
- Clear List
- Get Item From List

- Insert Item In List
- Remove Item
- From List
- Set Item In List

Control de flujo

- Break
- For
- Group
- If
- Loop
- Repeat
- Repeat Until
- Restart Program
- While

Ejecución

- Execute Program
- Start Program

Editor

- Close Progress Window
- Setup Progress
- Show Progress Window
- Update Progress

Cadenas de texto

- Concatenate Strings

- Get String Length
- Get Substring
- Replace In String

Documentación

- Comment

Creación de objetos

- Clone
- Create Empty GameObject
- Instantiate

Destrucción de objetos

- Destroy

Estado de juego

- Game Over
- Start Game
- Pause Game
- Resume Game
- Toggle Pause

Trayectorias

- Follow Path
- Get Path Property
- Set Path Property

Movimiento

- Follow

- Interpolate
- Look At
- Move To
- Rotate
- Rotate To

Animación

- Play Animation
- Set Animator State

Transform

- Get Position
- Get Rotation
- Get Transform
- Get Transform Property
- Set Position
- Set Position From Screen Point
- Set Rotation
- Set Transform Property
- Get World Point From Screen Point

Audio

- Get Audio Property
- Set Audio Property
- Play Music
- Play Sound
- Play Sound At Source
- Stop Music

- Stop Sound At Source
- Toggle Audio Mute

GUI

- Get GUIText Property
- Get GUITexture Property
- Set GUIText Property
- Set GUITexture Property

Temporizadores

- Wait For Timer
- Restart Timer
- Resume Timer
- Stop Timer

Cámaras

- Get Camera Property
- Set Camera Property

Pools

- Get Object From Pool
- Get Pool Capacity
- Reset Pool

Valores aleatorios

- Get Random Color
- Get Random Item From List
- Get Random Number
- Get Random Point In Collider

- Get Random Point In Collider List
- Get Random Vector
- Set Random Seed

Restricciones

- Confine
- Set Distance
- Get Distance

Persistencia

- Save Data

Ratón

- Hide Mouse Cursor
- Show Mouse Cursor

Escenas

- Load Scene
- Get Scene Property

Depuración

- Clear Console
- Log Message
- Pause Editor
- Hello World

Navegación

- Set NavMesh Agent Destination

Física

- Add Force
- Get Rigidbody Property
- Get Velocity
- Limit Velocity
- Sleep
- Sleep List
- Wake Up
- Wake Up List
- Set Rigidbody Property
- Set Velocity

Tiempo

- Set Time Scale
- Wait

GameObjects

- Get GameObject Property
- Set GameObject Property
- Set Parent

Vectores

- Get Vector Component
- Set Vector Component

Efectos visuales

- Set Color

Aplicación

- Set Application Property

- Exit Game

Variables

Una variable es un componente que se comporta como una unidad dinámica de almacenamiento de información. Esto significa que puede contener un dato de un tipo determinado (un texto, un número, un color, una referencia a un objeto, etc.) y además ser utilizada en lugar de un valor de ese tipo allí donde GameFlow admita el uso de variables.

Así, si por ejemplo tenemos una acción como *Set Color*, que sirve para cambiar el color a un objeto, y que requiere como una de sus propiedades el color al que queremos cambiar el objeto, tendríamos dos maneras de indicar cuál debe ser dicho color:

1. Asignar a la propiedad un color concreto -por ejemplo, el rojo- de manera directa mediante el selector de colores, o
2. Asignar a la propiedad una *Variable* de tipo *Color* indicando que queremos usar como color el valor de la variable (que será también un color, por ejemplo el verde).

Lo que ocurrirá cuando ejecutemos el juego es que la acción *Set Color* al ser ejecutada por un programa evaluará a su vez el valor de cada una de las variables que tenga asignadas para determinar su valor actual. En otras palabras y volviendo a nuestro ejemplo, lo que ocurrirá es que la acción preguntará a la variable: "¿Qué color tienes en este momento?" y la variable le responderá con un valor "color verde", que será el color que la acción usará finalmente para su propósito.

Dado que es posible cambiar el valor de las variables tantas veces como se necesite a lo largo de la vida del programa, lo que nos ofrece el soporte integrado de variables de GameFlow es la posibilidad de construir programas o usar herramientas con valores de entrada dinámicos que pueden variar según el estado del juego o según otro tipo de parámetros.

Por su naturaleza dinámica, en un videojuego nos encontraremos a menudo con la necesidad de trabajar con variables. Algunos ejemplos en los que se aconseja su utilización serían: crear un marcador de puntuación,

controlar el estado de daño de nuestro personaje, llevar un control del número de objetos recogidos, etc.

Crear variables

Las variables viven como cualquier otro componente más dentro de un *GameObject*, por tanto para crearlas sólo es necesario añadir un componente de tipo Variable:

- Haciendo uso del botón *Add Component* en la ventana Inspector, o
- Usando la opción del menú *Component* > *Gameflow* > *Data* > *Variable*.

En cualquiera de los dos casos si estamos creando la primera variable del GameObject se creará un componente Variables (en plural) que actuará como contenedor de todos los bloques de tipo Variable que añadamos al objeto.

A partir de ahí tendremos disponible una tercera forma de añadir una nueva variable mediante el botón + *New Variable* que aparece en la parte inferior del área del componente en la ventana Inspector.

También dispondremos de varias opciones adicionales en el menú contextual de este componente Variables como la opción *Reset* o las opciones *Collapse All* | *Expand All* que nos permiten respectivamente colapsar o expandir todas las variables del objeto en un paso.

Definición de variables

Para definir una variable bastará con dotarla de un identificador adecuado al propósito que queramos dar a la variable, indicar el tipo de dato que queremos almacenar en ella y darle un valor inicial. Opcionalmente, también podremos indicar si queremos que la variable sea persistente o no.

Tipos

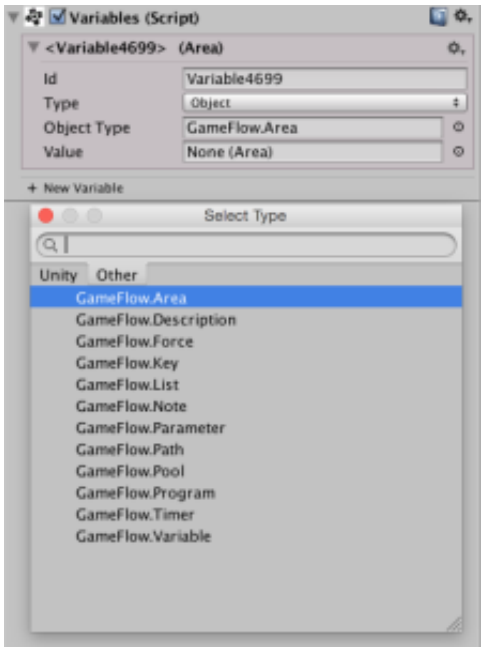
Los tipos soportados por las variables de GameFlow son los siguientes:

- **String:** Texto.
- **Integer:** Números enteros.
- **Float:** Números decimales.
- **Boolean:** Verdadero o Falso.
- **Vector2:** Vector de 2 dimensiones (X, Y).
- **Vector3:** Vector de 3 dimensiones (X, Y, Z).
- **Rect:** Rectángulo (X, Y, Ancho, Alto).
- **Color:** Color.
- **Object:** Referencia a un objeto.
- **Enum:** Valor de enumeración.
- **Toggle:** Marcado o no marcado (equivalente a Boolean).
- **Tag:** Etiqueta.
- **Layer:** Capa.

Especificación de tipo

Los tipos *Object* y *Enum* requerirán una especificación adicional del tipo concreto de objeto o de enumeración que contendrá la variable que puede hacerse mediante una ventana especial de selección de tipos organizada en dos pestañas:

- **Pestaña *Unity*:** contiene sólo los tipos de Unity utilizables en GameFlow.
- **Pestaña *Other*:** contiene sólo los tipos de GameFlow y los tipos que el usuario haya definido por medio de sus propios scripts.



Conversiones de tipo y valor

La conversión de tipo *(Type Casting)* es la modificación automática del tipo de una variable que GameFlow realiza cuando dicha variable es modificada por alguna acción que devuelve valores que son de un tipo distinto al que la variable tuviese en ese momento.

Esto quiere decir que en GameFlow las variables son de tipo dinámico y se adaptan automáticamente cuando se utilizan para guardar un tipo de dato distinto al que tenían inicialmente. Esto tiene sus ventajas, pero también sus inconvenientes por lo que conviene tenerlo en cuenta.

La otra operación de conversión que realiza GameFlow automáticamente es la de tratar de convertir el valor actual al del nuevo tipo, cuando lo único que modificamos es el tipo de la variable, cosa que no será extraño realizar en tiempo de edición.

Así, si en una variable de tipo *String* tenemos el valor alfanumérico “3” y de repente decidimos cambiar el tipo de esa variable a *Integer*, GameFlow lo que hará será intentar hacer la mejor conversión de valores posible para el usuario. En este ejemplo la conversión es posible y el valor se actualiza convenientemente, pero si la conversión no fuese posible el valor de la variable pasaría a ser el valor por defecto para el nuevo tipo elegido.

Operaciones de edición

Las operaciones de edición son esencialmente las mismas enumeradas para las acciones en el capítulo anterior, con la diferencia de que las operaciones Cortar y Pegar no estarán disponibles.

Asignación por arrastre

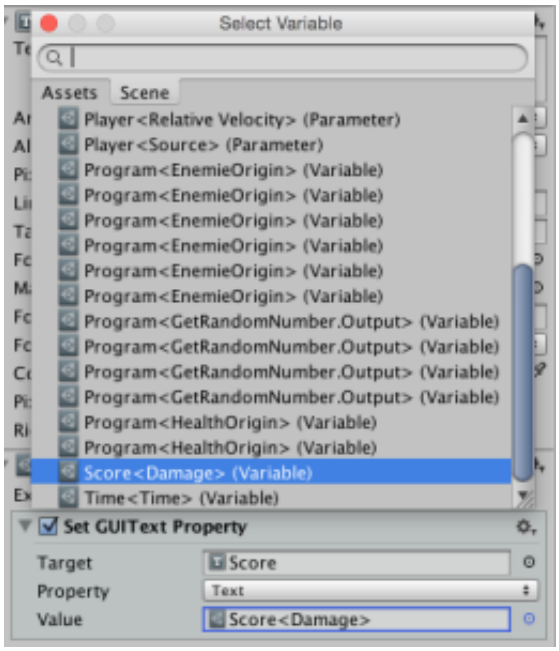
Quizás la operación más importante que debemos conocer respecto a las variables es que podemos asignarlas a prácticamente cualquier propiedad de cualquiera de las acciones, condiciones o herramientas que incluye GameFlow.

Para ello, basta con hacer click en el título del bloque de variable que deseemos para arrastrarlo hasta la propiedad a la que queramos asignar la variable y una vez observaremos que el campo de la propiedad se dibuja con otro fondo, soltar el botón del ratón.

Asignación mediante selector de objetos

La otra manera de asignar una variable a una propiedad es pulsar en el pequeño círculo que aparece a la derecha del campo de valor de la propiedad para que GameFlow nos muestre una ventana de selección de objetos especial, que se parece a la estándar de Unity pero que no es tal.

La primera diferencia que observaremos respecto a la ventana estándar es que el selector de GameFlow no se queda como el estándar en el nivel de los *GameObjects* sino que es capaz de llegar al nivel de los componentes, motivo por el cual podremos ver todas y cada una de las variables que hayamos creado en el proyecto y en la escena, cada una etiquetada convenientemente con el identificador que le hubiésemos puesto.



La segunda diferencia que observaremos es que GameFlow añade una pestaña adicional que nos permitirá ver sólo las variables definidas dentro del *GameObject* actualmente seleccionado o, en caso de que este pertenezca a un prefab, ver únicamente las variables definidas dentro de la jerarquía de dicho prefab. Esto es especialmente cómodo cuando estamos construyendo prefabs, pues nos ayuda a encontrar más rápidamente las variables que buscamos y a asegurarnos de que no elegiremos nunca una variable externa al prefab.

Conviene indicar también que cuando la propiedad que queremos cambiar es de tipo objeto, el selector nos permitirá elegir tanto objetos de dicho tipo (tanto si son componentes como *GameObjects*) como variables, por lo que el modo de trabajo es idéntico al comentado y si lo que deseamos es asignar directamente referencias de objetos a la manera tradicional de Unity lo podremos hacer también sin ningún problema.

Variables Integradas

Las variable integradas *(Built-in Variables)* son variables especiales cuyo valor no puede fijarlo el usuario sino que viene ya predefinido (son variables de solo lectura) y que nos permiten consultar valores dinámicos del sistema, como por ejemplo la fecha actual o la resolución de la pantalla, de una manera simple.

Todas las variables predefinidas las podremos encontrar ya agregadas al proyecto pues forman parte del prefab *Built-in* de la carpeta *GameFlow/Assets* de nuestro proyecto, por lo que podremos encontrarlas en cualquier momento en la pestaña *Assets* de la ventana de selección de objetos.

A continuación se detallan las variables integradas disponibles:

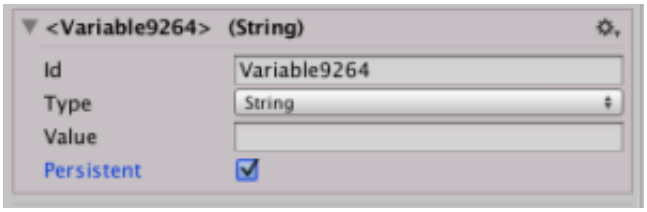
- **Built-in<Day>:** Día del mes de la fecha actual.
- **Built-in<Delta Time>:** Tiempo transcurrido desde el último fotograma.
- **Built-in<Device Name>:** Nombre del dispositivo en tiempo de ejecución.
- **Built-in<Hour>:** Hora actual en formato 24h.
- **Built-in<Minute>:** Minuto actual.
- **Built-in<Month>:** Número de mes de la fecha actual.
- **Built-in<Mouse Position>:** Posición actual del puntero del ratón.
- **Built-in<Native Resolution>:** Resolución nativa de la pantalla.
- **Built-in<Screen Size>:** Resolución pantalla / ventana actual.
- **Built-in<Second>:** Segundo de la hora actual.
- **Built-in<Year>:** Año de la fecha actual en formato largo.

Persistencia

Cuando hablamos de “persistencia” hablamos de la capacidad de que ciertos datos no sean borrados al salir de nuestro juego sino que queden guardados en la memoria no volátil del dispositivo sobre el que corremos nuestro juego para que podamos restaurarlos la siguiente vez que arranquemos el juego.

Esto es muy útil en videojuegos pues nos permite guardar información valiosa sobre los progresos del jugador o el último estado en el que se encontraba el juego, de manera que podemos implementar de una manera relativamente sencilla juegos que permitan “continuar” una partida en el último punto que se dejó o sin ir tan lejos, que simplemente recuerden cuál fue nuestra mejor puntuación.

GameFlow ofrece un soporte de persistencia transparente para el usuario e integrado en las variables, pero que por razones técnicas sólo podremos utilizar en variables que NO sean de tipo *Object*. Todo lo que tenemos que hacer para que el valor de una variable “sobreviva” al cierre del juego es marcar la casilla *Persistent* que aparece en su interfaz en la ventana Inspector.



A partir de ese momento, GameFlow se ocupará de que el valor de esa Variable sea persistente, cosa que podremos verificar incluso mientras estamos trabajando en el Editor, pues cada vez que pulsemos *Play* la variable volverá a tener el último valor que tuviese al término de la anterior ejecución.

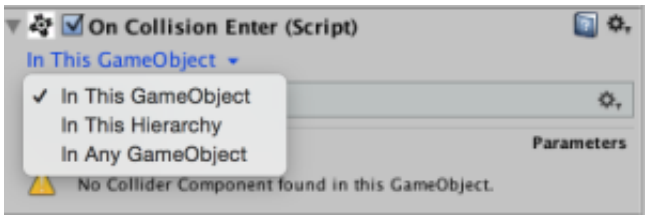
Eventos

Como vimos en capítulos anteriores, los programas de evento son los programas que comienzan a ejecutarse automáticamente cuando estando habilitados detectan en su rango de escucha un evento de un tipo determinado.

En este capítulo veremos con más detalle todos los aspectos relativos a los programas de evento para entender cómo sacar el mayor partido posible al soporte de programación dirigida por eventos que GameFlow nos ofrece.

Rango de escucha

El rango de escucha lo encontraremos en la parte superior de algunos programas de eventos como un menú desplegable de opciones y funciona como un filtro que el programa puede usar para atender sólo determinados eventos en función de su procedencia.



El rango de escucha tendrá normalmente tres posibles valores:

- ***In this GameObject:*** cuando sólo queremos atender eventos que han sido detectados por componentes de ese mismo GameObject en el que está incluído el programa. Este es el valor por defecto y que mejor rendimiento proporciona.
- ***In this Hierarchy:*** cuando queremos atender eventos que procedan también tanto del objeto padre del GameObject en el que está incluido el programa así como de cualquiera de sus GameObjects hijos.
- ***In Any GameObject:*** cuando no queremos hacer ningún tipo de

filtrado y queremos atender el evento proceda de donde proceda.

En la mayoría de los casos, el rango de escucha puede dejarse con el valor por defecto pues es una buena práctica limitar el alcance de un programa al mínimo indispensable. Sólo allí donde necesitemos que nuestro programa sea capaz de actuar sobre elementos de toda la escena deberíamos ampliar el rango de escucha.

Por poner un ejemplo, si hacemos un programa que cambie el color de su GameObject al hacer click con el ratón sobre él bastará con el rango mínimo, pero si lo que buscamos es hacer un programa que cambie el color de cualquier objeto de la escena sobre el que hagamos click, necesitaremos obligatoriamente ampliar el rango al máximo.

Condiciones para la ejecución

Para que un programa de evento pueda comenzar su ejecución automáticamente es necesario:

1. Que el propio programa esté activado o habilitado.
2. Que el evento se haya producido en el rango de escucha especificado en el programa de evento.

Salvo en algunos casos, como On Start, para que un programa de evento se ejecute no es indispensable que el GameObject en el que esté incluido esté también activo, especialmente si el rango de escucha está puesto al máximo.

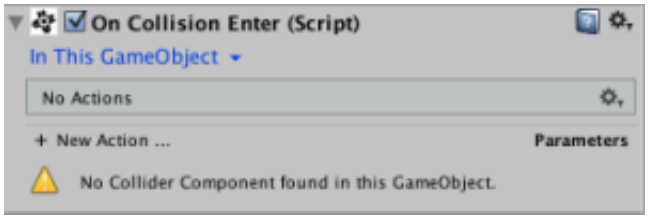
Parámetros de evento

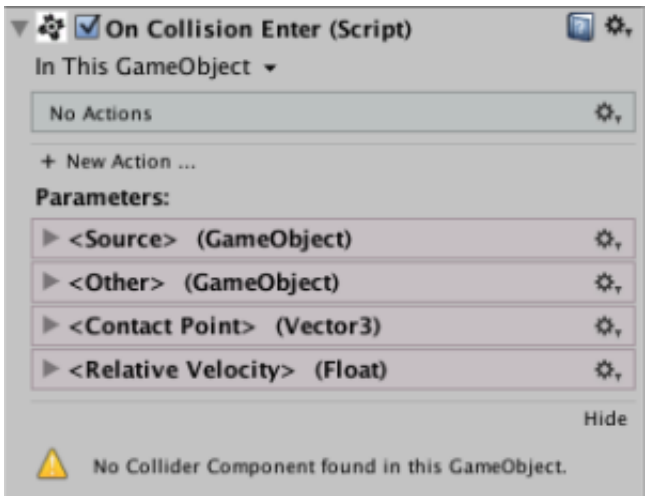
La mayoría de los programas de evento ofrecen una lista de parámetros cuyo valor se completa con determinada información específica del evento producido justo antes de que comience a ejecutarse el programa.

Los parámetros de evento amplían de manera notable las posibilidades de un programa, ya que ahora podemos hacer que el programa sepa, por ejemplo, el GameObject que originó el evento y pasar dicho parámetro a alguna de las acciones del programa. Sería, en otras palabras, como

permitir al programa actuar sobre un GameObject indeterminado, que en algún momento, cuando se produzca el evento, se concretará para ser uno de los GameObjects de nuestro juego.

Cuando el programa de evento soporta parámetros, mostrará una pequeña etiqueta en la esquina inferior derecha de su área en el Inspector con el texto **Parameters** en negrita. Al pulsar en esa etiqueta, se desplegará una lista de parámetros justo debajo de la lista de acciones, con un formato de bloques similar al de las variables, pues los parámetros pueden considerarse como variables de sólo lectura.





Al igual que las variables, los parámetros de evento pueden arrastrarse a los campos de las acciones del programa que admitan variables, pero a diferencia de las variables, los parámetros sólo pueden utilizarse en el propio programa de evento, no siendo posible arrastrarlas a otros programas (ni siquiera a otros del mismo tipo) ni tampoco seleccionarlas en el selector de objetos, pues no aparecen reflejadas en el mismo.

Eventos soportados

La siguiente es una lista de los tipos de eventos reconocidos y soportados por GameFlow junto con una breve descripción de los programas de evento que se ocupan de gestionarlos.

Eventos de inicialización

Son los eventos relativos a procesos de inicialización del juego. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Start:** un programa de este tipo se ejecutará al arrancar un juego si el GameObject está activo o si no estuviese activo, la primera (y sólo la primera) vez que sea activado.

Eventos de activación

Son los eventos relativos a la activación / desactivación de un GameObject. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Activate:** un programa de este tipo se ejecutará cada vez que un GameObject pase de inactivo a activo. No debe confundirse con On Start, aunque pueda ser equivalente para el caso concreto de un GameObject que al arrancar el juego está desactivado y luego es activado en algún momento posterior.
- **On Deactivate:** un programa de este tipo se ejecutará cada vez que un GameObject pase de activo a inactivo.

Eventos de actualización

Son los eventos que se lanzan regularmente coincidiendo con los procesos de actualización propios del motor de Unity. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Update:** un programa de este tipo se ejecutará al comienzo de cada fotograma de juego. Es el tipo de programa ideal cuando queremos ejecutar acciones rápidas y de manera continua.
- **On Late Update:** un programa de este tipo se ejecutará al finalizar todos los programas On Update.
- **On Fixed Update:** programa que se ejecuta a intervalos regulares fijos y que se emplea sobre todo para dar órdenes al motor de Física de

Unity.

Eventos de juego

Son los eventos que se lanzan cuando el estado general del juego es modificado. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Game Start:** un programa de este tipo se ejecutará como consecuencia del cambio de estado que produce la acción *Start Game*. Indica, normalmente, que el juego en sí ha dado comienzo.
- **On Game Over:** un programa de este tipo se ejecutará como consecuencia del cambio de estado que produce la acción *Game Over*. Indica, normalmente, que la partida ha terminado.
- **On Game Pause:** un programa de este tipo se ejecutará cuando el juego entre en pausa, normalmente como consecuencia de una acción *Pause Game* o *Toggle Pause*.
- **On Game Resume:** un programa de este tipo se ejecutará cuando el juego se reanude tras estar anteriormente en pausa, normalmente como consecuencia de una acción *Resume Game* o *Toggle Pause*.

Eventos de ratón

Son los eventos que se lanzan cuando el usuario realizan acciones con el ratón sobre elementos que lo soporten como elementos GUI o *Colliders*. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Mouse Down:** un programa de este tipo se ejecutará cuando se haga click con alguno de los botones del ratón sobre un elemento GUI o un *Collider*.
- **On Mouse Drag:** un programa de este tipo se ejecutará cuando se detecte movimiento de ratón sobre un elemento GUI o un *Collider* mientras se mantiene alguno de los botones del ratón pulsados.
- **On Mouse Enter:** un programa de este tipo se ejecutará cuando el puntero del ratón entre en el área ocupada por un elemento GUI o un *Collider*.
- **On Mouse Exit:** un programa de este tipo se ejecutará cuando el

puntero del ratón salga del área ocupada por un elemento GUI o un *Collider*.

- **On Mouse Over:** un programa de este tipo se ejecutará cuando se detecte movimiento de ratón sobre un elemento GUI o un *Collider* y a la vez no se esté pulsando ninguno de los botones del ratón.
- **On MouseUp:** un programa de este tipo se ejecutará al liberar un botón de ratón que estuviese siendo pulsado sobre un elemento GUI o un *Collider*.

Eventos de colisión

Son los eventos que el motor de Física de Unity lanza para informar de que se han producido colisiones entre objetos y también para indicar el estado de dichas colisiones. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Collision Enter:** un programa de este tipo se ejecutará en cuanto se detecte que ha comenzado una colisión entre dos objetos.
- **On Collision Exit:** un programa de este tipo se ejecutará en cuanto se detecte que una colisión entre dos objetos ha dejado de producirse.
- **On Collision Stay:** un programa de este tipo se ejecutará en cada fotograma de juego mientras se siga produciendo una determinada colisión entre dos objetos.
- **On Trigger Enter:** similar a On Collision Enter pero aplicable cuando al menos uno de los objetos tiene un Collider marcado como *Is Trigger*, es decir, que detecta colisiones pero es atravesable por otros objetos.
- **On Trigger Exit:** similar a On Collision Exit pero aplicable cuando al menos uno de los objetos tiene un Collider marcado como *Is Trigger*.
- **On Trigger Stay:** similar a On Collision Stay pero aplicable cuando al menos uno de los objetos tiene un Collider marcado como *Is Trigger*.

Es importante comentar que al ser estos unos eventos en los que participan siempre dos objetos, su notificación es doble pues se notifica la colisión a ambos objetos por igual. Esto significa que puede asociarse un programa

de este tipo a cualquier de los dos objetos susceptibles de colisionar y en ambos casos el programa se ejecutará. La única diferencia, como veremos, estará en los parámetros enviados a cada programa.

Eventos de tiempo

Son eventos relacionados con el tiempo y componentes que trabajan en base al tiempo, como el Temporizador *(Timer)*. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Timer Expire:** un programa de este tipo se ejecutará cuando un componente *Timer* notifique que ha transcurrido el intervalo de tiempo para el que había sido programado.

Eventos de programa

Son eventos que lanzan los propios programas para informar acerca de cambios en su estado. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Program Finish:** un programa de este tipo se ejecutará cuando un programa en el rango de escucha finalice su ejecución.

Eventos de variables

Son eventos que lanza GameFlow para notificar que una variable (o parámetro) ha sido modificada. Este tipo de eventos es gestionado por los siguientes programas de evento:

- **On Variable Change:** un programa de este tipo se ejecutará tanto en tiempo de ejecución como de edición cuando una variable en el rango de escucha haya sido modificada.
- **On Parameter Change:** un programa de este tipo se ejecutará tanto en tiempo de ejecución como de edición cuando un parámetro en el rango de escucha (que en este tipo de programas está limitado al GameObject padre) haya sido modificado.

Controladores de eventos

Los controladores de eventos son unos componentes especiales que sirven

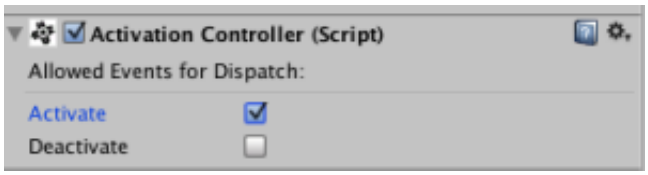
como filtros generales de eventos. Al añadirlos a un GameObject permiten especificar cuáles de los eventos generados por dicho GameObject que correspondan al tipo gestionado por el controlador deben ser notificados y cuáles serán simplemente ignorados.

Este control es especialmente útil para optimización del rendimiento de un juego, pues permite reducir el número de mensajes internos que el sistema debe procesar haciendo que sea necesario menos tiempo de proceso por fotograma.

Los tipos de controladores de eventos soportados son los siguientes:

- **Activation Controller:** controlador de eventos de activación.
- **Collision Controller:** controlador de eventos de colisión.
- **Mouse Controller:** controlador de eventos de ratón.
- **Trigger Controller:** controlador de eventos de colisión con *Triggers*.

Para que un tipo de evento sea notificado su *checkbox* debe estar marcado, mientras que aquellos eventos que lo tengan sin marcar serán ignorados.



Control de flujo

El flujo de programa es el orden en el que un programa ejecuta su lista de acciones. El flujo de un programa por defecto será secuencial, es decir, se ejecutará primero la primera acción activada hasta completarla, a continuación la siguiente acción y así sucesivamente hasta acabar el programa, pero este orden no siempre nos permite obtener los resultados que necesitamos en nuestro juego, motivo por el cual GameFlow nos da la posibilidad de modificar este flujo de una manera simple, por medio de un tipo de acciones especializadas en dicho cometido.

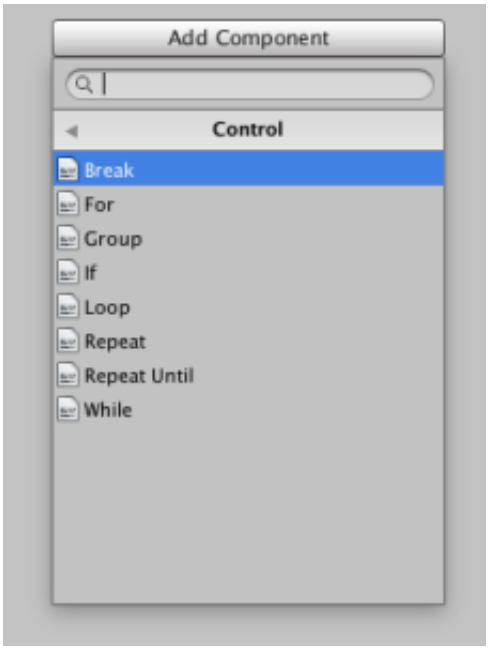
Con estas acciones especializadas, que detallaremos a continuación, será posible implementar lógicas de juego más sofisticadas en las que:

- Ciertas partes de un programa se ejecuten de manera repetida un determinado número de veces. A esto en metodología de la programación se le denomina “bucle”.
- Ciertas partes de un programa se ejecuten únicamente cuando se cumplan (o no se cumplan) unas determinadas condiciones. A esto se le denomina “bifurcación” o “ejecución condicional”.
- Ciertas partes de un programa no se ejecuten secuencialmente, es decir acción por acción, sino que ejecuten todas sus acciones de manera simultánea como un grupo.

El modo en que controlamos el flujo de los programas define en gran parte la calidad de la programación de nuestro juego, motivo por el cual se recomienda tener claras las ideas básicas de la metodología de la programación antes de intentar construir programas con un flujo de control muy complejo.

Acciones de control de flujo

Como hemos mencionado, GameFlow permite controlar el flujo de los programas mediante el uso de un grupo especial de acciones que podemos encontrar bajo el menú *Component* > *GameFlow* > *Control.*



El propósito de cada una de estas acciones es el siguiente:

- ***If:*** permite bifurcar la ejecución en función del resultado de una lista de condiciones. Si dichas condiciones se cumplen (su resultado total es Verdadero) se ejecutará la lista principal de acciones contenidas, mientras que si el resultado es Falso se ejecutará una lista de acciones secundaria (acciones bajo la sección *Else*).
- ***For:*** permite crear un bucle de ejecución para una lista de acciones contenidas (en otras palabras, ejecutar esas acciones repetidamente) un número de veces que viene dado por el valor de una variable que se va modificando en cada iteración o vuelta del bucle y un valor límite contra el que se va comparando.
- ***Repeat:*** una versión simplificada de *For* que sólo requiere especificar el número de veces que queremos repetir las acciones contenidas, sin necesidad de variable ni valor límite.
- ***Loop:*** una versión simplificada de *Repeat* pues por defecto creará un bucle de ejecución que se repetirá de manera indefinida bien hasta que el programa sea detenido o bien hasta que dentro de las acciones contenidas se ejecute una acción *Break* (véase descripción más adelante).
- ***While:*** ejecutará las acciones contenidas si se cumple una determinada

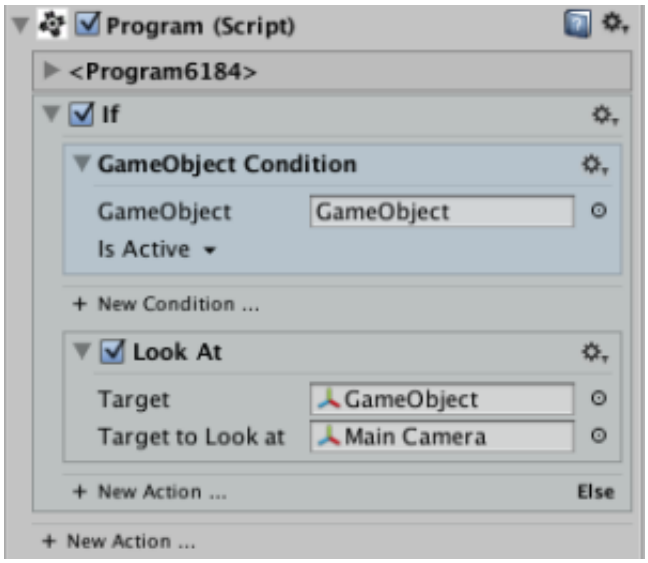
condición y las seguirá ejecutando en bucle mientras se siga cumpliendo la condición.

- ***Repeat Until:*** ejecutará las acciones contenidas en bucle hasta que la lista de condiciones asociada se cumpla.
- ***Group:*** permite ejecutar todas las acciones contenidas de manera paralela, es decir, dentro de este tipo de bloques no se espera a que una acción termine de ejecutarse, sino que todas las acciones se ejecutan a la vez y es únicamente cuando todas las acciones terminan cuando se considera que la acción ha terminado.
- ***Break:*** permite interrumpir un bucle, haciendo que el programa siga ejecutándose en la siguiente acción después de la que generó el bucle que esté activada.

Condiciones

Las condiciones son un tipo de bloques especiales que utilizan algunas acciones para determinar, mediante su evaluación, cómo o cuándo deben modificar el flujo del programa.

Visualmente las condiciones son muy similares a las acciones pues al igual que estas se representan como bloques colapsables que contienen campos de propiedades que nos permiten configurar el funcionamiento de la condición. Sólo varían en su color, que es distinto al de las acciones y en su ubicación, pues sólo pueden vivir dentro de acciones que hayan sido diseñadas para utilizar condiciones.



En cuanto a operaciones de edición, las condiciones soportan las mismas operaciones de edición que las acciones dentro de sus propios contextos y por medio de sus menús contextuales.

Evaluación

Evaluar una condición es comprobar si se cumple o no se cumple. En programación, cuando una condición se cumple decimos que su resultado es Verdadero, siendo Falso cuando no se cumple. El resultado de la evaluación de una condición es lo que denominamos un valor booleano, pues sólo admite dos posibles valores opuestos entre sí.

Este resultado y su interpretación dependerá en gran medida del modo en que la condición haya sido configurada por medio de las propiedades que expone en su interfaz, que normalmente incluirá como mínimo un campo para indicar qué comparación exactamente queremos hacer dentro de la condición.

GameFlow no sólo soporta la evaluación de una condición sino que permite crear una lista de condiciones enlazadas mediante operadores lógicos que serán evaluados siempre usando el último resultado obtenido y el resultado de la condición actualmente siendo evaluada. Estos operadores pueden ser:

- **Y *(And)*:** que devolverá como resultado Verdadero si (y sólo si) el anterior resultado es Verdadero y la condición con la que conecta también se evalúa como Verdadero. En cualquier otro caso, devolverá

como resultado Falso.

- **O *(Or)*:** que devolverá como resultado Verdadero tanto si el resultado anterior era Verdadero como si el resultado de la condición con la que conecta se evalúa como Verdadero. Sólo devuelve Falso si ambos resultados son Falsos.

Como hemos visto, el modo en el que una acción de control interpreta el resultado de la evaluación de su lista de condiciones depende de la lógica interna de cada acción. Para aclarar cualquier duda al respecto siempre podemos consultar la ayuda contextual de cada acción.