

# Taller

## ¡¡Crea tu videojuego con Stencyl!!

### 1 Bienvenid@ al taller 😊

En este taller realizaremos una pequeña introducción a *Stencyl* [1] . *Stencyl* es una aplicación que te permite crear videojuegos sin necesidad de tener conocimientos avanzados de programación. De hecho, con *Stencyl* los videojuegos se programan utilizando bloques de control predefinidos. Estos bloques permiten realizar gran cantidad de acciones, tal y como si estuviésemos utilizando un lenguaje de programación tradicional (pero sin necesidad de tener esos conocimientos).

Nuestro objetivo, en estas horas que pasarás en el taller, es que aprendas los conceptos básicos de uso de esta herramienta y conozcas qué cosas son necesarias programar durante el desarrollo de un videojuego. Si además logramos que te guste un poco más la informática, ¡habremos conseguido nuestra meta!


### 2 Instalando *Stencyl*

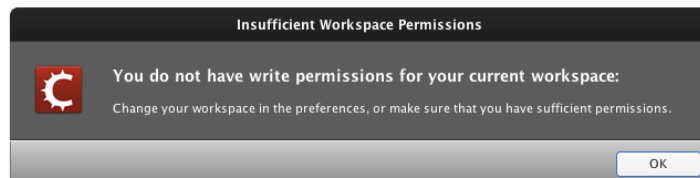
La instalación de *Stencyl* es muy sencilla. En el laboratorio ya lo hemos instalado nosotros, para que puedas comenzar directamente a trabajar con él. Sin embargo, si deseas instalarlo en tu casa (para terminar de construir el videojuego de este taller, o porque quieras hacer otro juego con el que dejar alucinad@s a tus colegas de clase), sólo tienes que seguir estos pasos:

1. *Stencyl* utiliza tecnología JAVA® para funcionar. Por lo que, antes de instalar la aplicación en tu ordenador, deberás instalar la máquina virtual de JAVA®. Ten cuidado porque sólo funciona con la versión 8 (no con versiones superiores). En la referencia [2] tienes una URL que te llevará a la página donde puedes descargar una versión adecuada.
2. Una vez instalada la máquina virtual de JAVA®, puedes ir a la página web de *Stencyl* y descargarte el programa. Para ello, utiliza la URL de la referencia [1] .

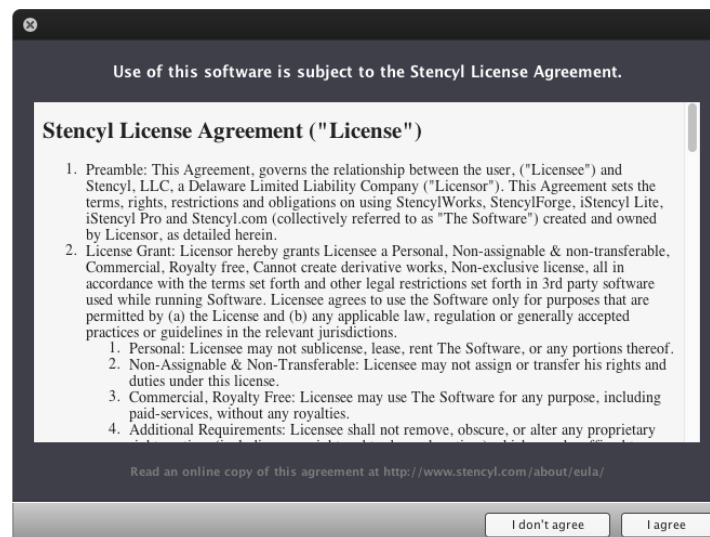
Llegados a este punto, podrás utilizar *Stencyl* en tu ordenador sin ningún problema.

### 3 ¡Arrancando los motores!

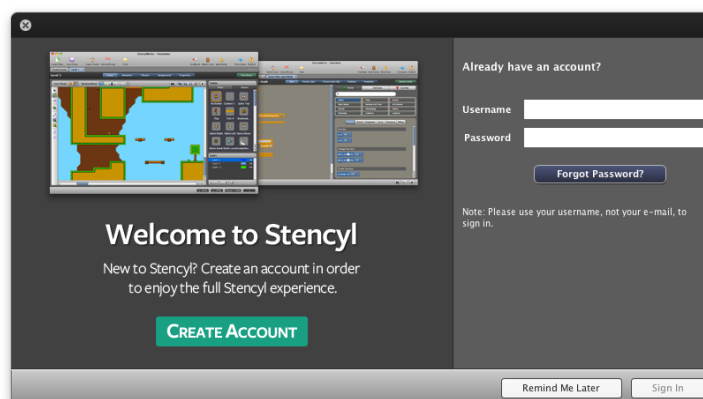
Para comenzar con la aplicación sólo tienes que buscar el icono de *Stencyl*  y pulsar sobre él. Nada más comenzar, quizá te diga que no tienes permisos para escribir en tu espacio de trabajo (¡no te preocupes!, dale a “OK” y luego nos preocuparemos de configurarlo):



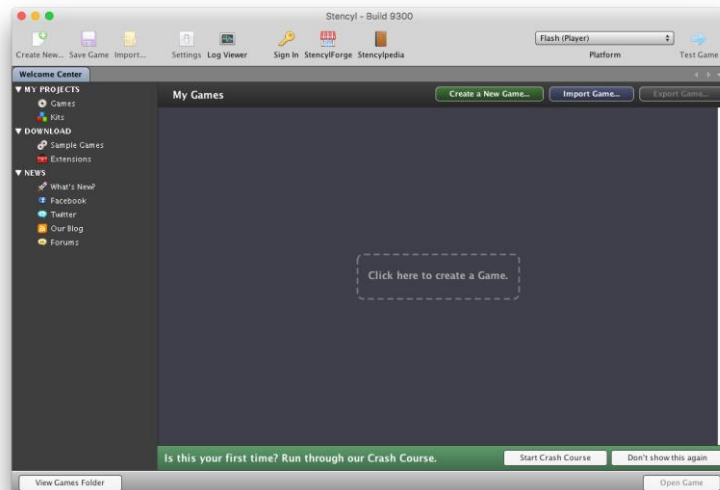
Quizá también te solicite que aceptes la licencia, léela atentamente y, si estás de acuerdo, acéptala:



Por último, te dará la oportunidad de que te crees una cuenta. No es necesario, puedes hacerlo si, en algún momento, quieres publicar tus juegos en la comunidad de *Stencyl*.



Una vez cerradas todas esas ventanas, podrás comenzar a trabajar con el programa. Si te das cuenta, en la parte inferior, *Stencyl* te sugiere comenzar con un curso de introducción. Eso será lo que hagamos en este taller. No obstante, *Stencyl* te ofrece algunos otros cursos para seguir aprendiendo el desarrollo de juegos. Si te interesan, los puedes encontrar en la referencia [3] .

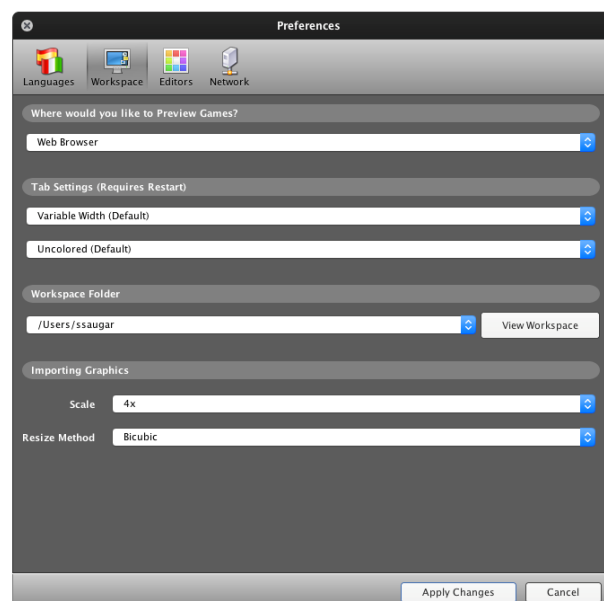


### 3.1 Configuración de un espacio de trabajo

Lo primero que debemos hacer cuando arranquemos la aplicación, es configurar la carpeta que conformará nuestro espacio de trabajo. En esta carpeta, *Stencyl* guardará todos los juegos y elementos que generemos, permitiéndonos hacer copias de seguridad, pasarle los juegos a nuestro@s amig@s, etcétera.

La configuración del programa está incluida en el menú de "Preferencias". De todas las preferencias (idioma, configuración de la red, ...) la más importante es la que define la carpeta del espacio de trabajo (*workspace folder*). Antes de realizar ninguna operación con la aplicación, crea una carpeta dentro de *Documentos* y selecciónala como espacio de trabajo (llámala, por ejemplo, *juegos*).

A partir de ese momento, en esa carpeta se encontrarán todos los juegos que vayas creando (por ejemplo, al terminar el taller, ¡cópiatela en un USB para llevarte el trabajo a casa!).



## 4 Comenzando a crear un juego

Vamos a crear un juego basado en una plantilla ofrecida por *Stencyl*. Para ello, en primer lugar, tendremos que cargar la plantilla.

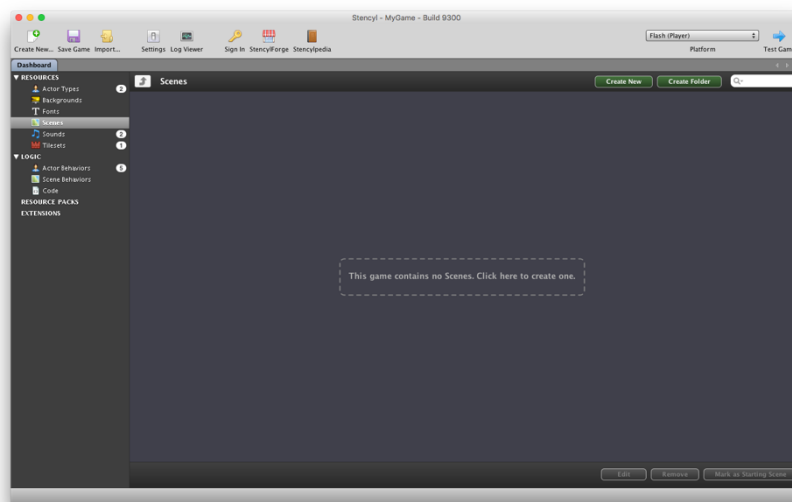
- Pulsa en **File** -> **Import Game** y carga el fichero que se llamado: *Crash Course Kit.stencyl*. Si no encuentras el fichero en el escritorio del ordenador, puedes descargarlo directamente desde la página del tutorial de *Stencyl* [4] .

Una vez tengamos la plantilla creada, crearemos un nuevo juego basado en ella:

1. Para ello, pulsa en **Create** -> **New Game**, selecciona un juego de tipo *Crash Course kit* y después pulse sobre **Next**.
2. Pon nombre a tu juego y haz click en **Create**.

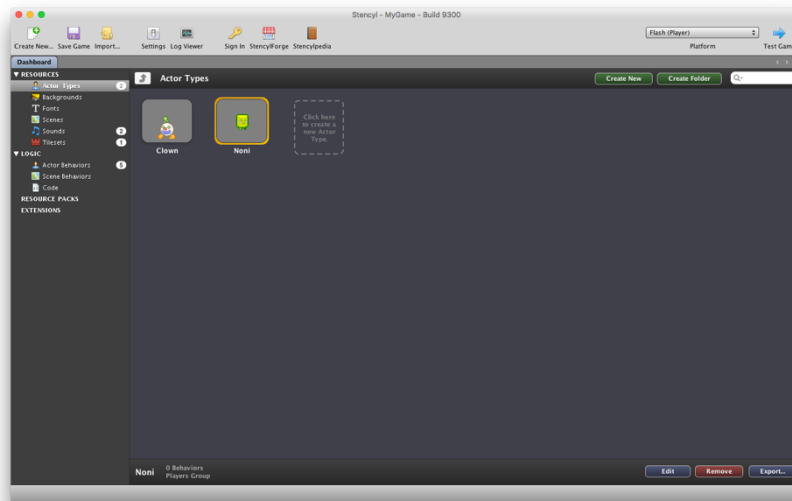


A partir de aquí, podremos comenzar a definir todos los aspectos necesarios para jugar.



## 4.1 Los actores de mi juego

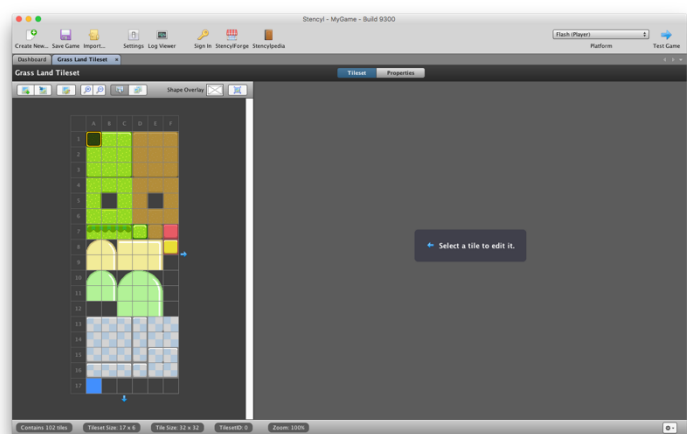
Si pulsas sobre **Actor Types**, verás los diferentes actores que van a formar parte de tu juego. En *Stencyl*, todos los elementos del juego que pueden interactuar de alguna manera, se denominan actores. En nuestro juego tenemos dos actores: **Noni** y **Clown**. Si haces doble click sobre **Noni** podrás ver la pantalla de edición del personaje.



Recuerda que siempre puedes volver a la pantalla principal del juego pulsando sobre **Dashboard**.

## 4.2 Otros elementos para construir los escenarios

Por otra parte, si desde **Dashboard** pulsas sobre **Tilesets**, podrás ver todos los elementos que tenemos para construir la parte estática de los escenarios. En este caso, la plantilla nos ofrece gran cantidad de recursos que se denominan **Grass Land Tileset**, que contienen piezas de césped, montañas, plataformas... Con todas esas piezas podrás crear el paisaje de los diferentes escenarios del juego.



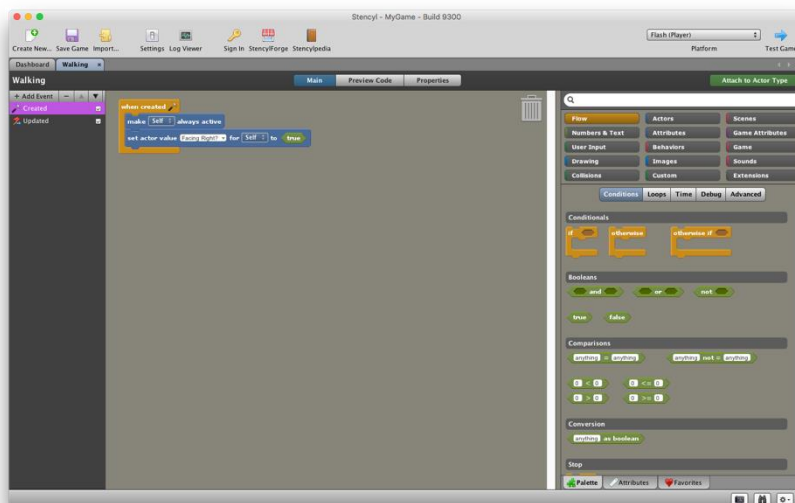
También puedes ver que la plantilla dispone de dos sonidos, que pueden utilizarse en distintos puntos del juego.

## 4.3 El comportamiento de los actores

De vuelta a la pantalla [Dashboard](#), pulsa sobre [Actor Behaviors](#). Esta acción nos mostrará los diferentes comportamientos que se pueden asociar a cada uno de nuestros personajes. Los comportamientos sirven para describir cómo van a reaccionar los actores cada vez que suceda alguna cosa dentro del juego. A las cosas que suceden dentro del juego las denominaremos "eventos". Por ejemplo: que aparezca un nuevo personaje, que se choquen dos tipos de personajes, que pase el tiempo, que el jugador haga click con el ratón, etcétera.

Para que te vayas familiarizando con la manera de programar que tiene *Stencyl*, vamos a ver alguno de estos comportamientos por dentro, así comprobarás cómo están definidos:

1. Pulsa, por ejemplo, sobre el comportamiento [Walking](#) para ver la pantalla de desarrollo:
  - a. En esta pantalla, a la izquierda tienes un menú con el que podremos añadir cómo reaccionará el actor a los diferentes eventos que, como programador del juego, consideres adecuados.
  - b. En el centro de la pantalla se muestra el diagrama de bloques, que contiene el "código fuente" (el programa) que especifica lo que hará el actor que tenga este comportamiento asociado cada vez que suceda el evento indicado.
  - c. Por último, a la derecha puedes ver todos los bloques que Stencyl te ofrece para programar estos comportamientos (y el resto de la aplicación). Si te das cuenta, en este caso, la programación es muy fácil puesto que se realiza moviendo bloques y agrupándolos (siempre que sean compatibles mediante su forma). De esta manera, ¡programar es tan fácil como resolver un puzzle!



Recuerda que es muy importante que, mientras que estés desarrollando el juego, presiones el botón [Save Game](#) para que no pierdas los cambios que hagas.

---

## 4.4 Agrupando los actores

A continuación, vamos a clasificar a los actores de nuestro juego. Clasificar los actores en grupos nos permite especificar las interacciones y/o colisiones que va a haber entre ellos de una manera más sencilla.

En nuestro caso, **Noni** será nuestro jugador (y por eso pertenecerá al grupo **Players**) y el payaso **Clown**, será nuestro enemigo, y por ello, pertenecerá a un grupo denominado **Enemies**). Para crear los grupos y asignar a los actores, haremos lo siguiente:

- Desde la pantalla principal, pulsa sobre **Settings** -> **Groups**. En la derecha de la pantalla podrás observar los grupos que están predefinidos para la aplicación. Ya está creado el grupo **Players** (que, como dijimos, será el grupo de **Noni**). Sin embargo, ¡falta uno que agrupe a sus enemigos!
- Por lo tanto, vamos a crear un nuevo grupo de actores. En primer lugar, pulsa el botón **Create New**. En segundo lugar, configura el grupo: le llamaremos **Enemies** (y ahí meteremos a los enemigos de **Noni**). Intenta averiguar cómo configurarías el grupo de tal manera que interactúe (o colisione) con **Players** y con **Tiles**.

Perfecto, una vez creados los grupos, asociaremos a nuestros actores dentro de ellos.

- Empecemos por **Noni**. Para agregarle a un grupo, necesitaremos abrir su ventana de edición (¿recuerdas cómo se hacía?).
  - Pulsa sobre **Properties** para comprobar el grupo al que pertenece **Noni**. Debería ser **Players**. Ten en cuenta que **Noni** es un actor y también pertenece al grupo **Actors** por defecto, sin embargo, de manera específica, **Noni** es un "jugador").
- Sigamos con **Clown**. Sigue los mismos pasos, pero asigna a **Clown** al grupo de enemigos que hemos creado.

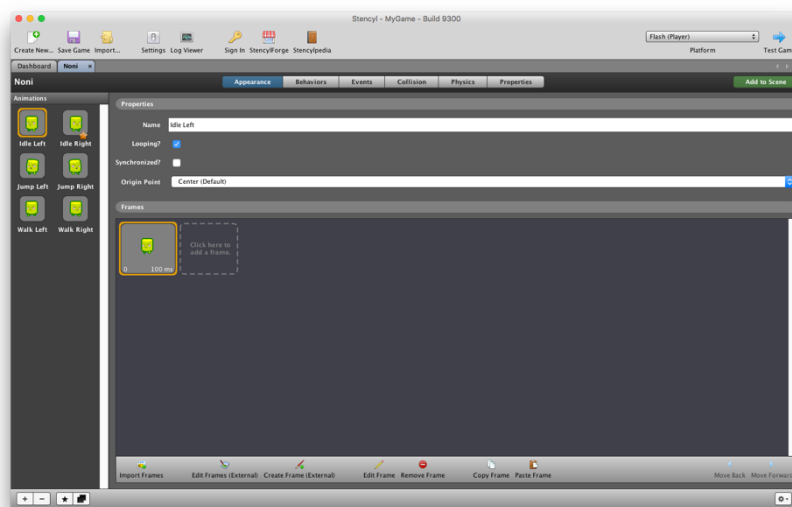
## 4.5 Asignando un comportamiento a cada actor

Precisamente, una de las partes más importantes de un juego es definir las cosas que puede (o no puede) hacer cada uno de los personajes. También hay que establecer cómo los distintos eventos que suceden a lo largo del juego (por ejemplo: un personaje recibe el impacto de un disparo, un personaje es pisado por otro, etcétera).

En *Stencyl*, la pantalla de edición de cada uno de los actores sirve para definir precisamente esos aspectos:

- **Appearance**: En este menú podemos definir las diferentes animaciones que el actor tendrá en cada caso que creamos conveniente (qué animación tiene al andar a la izquierda, al estar parado mirando a la derecha, cuando salta...).
- **Behaviors**: Comportamientos que tiene asociado el actor.

- **Events**: Cómo reacciona el actor a cada uno de los eventos que van sucediendo a lo largo del juego.
- **Collision**: Define el área de colisión del actor para cada una de las posibles animaciones que hemos establecido en la apariencia.
- **Physics**: En este apartado, podrás decidir cómo se comporta tu actor ante las diferentes leyes de la física. Cuestiones cómo: ¿le afecta la fuerza de la gravedad a este personaje?
- **Properties**: Tal y como has visto en el apartado anterior, en este menú se establecen aspectos como el nombre y el grupo al que pertenece el personaje que estás editando.



Perfecto, pues ahora que nos hemos familiarizado con la ventana de edición de los personajes, vamos a configurar a nuestros actores.

#### 4.5.1 Configuración de *Noni*

Abre la pantalla de edición de *Noni*, tal y como hemos visto anteriormente, e intenta configurar los siguientes aspectos de este actor:

1. **Physics**: en este caso, vamos a establecer que *Noni* NO pueda rotar libremente por la pantalla. Por el contrario, a *Noni* Sí que le afectará la gravedad.
2. **Behaviors**: pasemos a asociarle a *Noni* los siguientes comportamientos, asegurándonos de su correcto funcionamiento. Ten en cuenta que, para añadir el primer comportamiento, tendrás que pulsar sobre [Click here to choose a Behavior to attach to this Actor Type](#). Después, aparecerá, en la parte baja de la pantalla, dos botones (uno que permitirá añadir un comportamiento adicional y otro que permitirá borrarlos).
  - **Walking**: añadamos un comportamiento de este tipo a *Noni*. Una vez añadido, tendremos que configurarlo correctamente. Por ejemplo, habrá que



seleccionar qué controles se quieren utilizar para mover al personaje. También tendremos que establecer las animaciones que queremos que tenga el personaje para cada tipo de posición o movimiento.

- **Jumping**: como ya hemos añadido un comportamiento, utilizaremos la opción de **Add Behavior**, para configurar las opciones de salto de **Noni**. Una vez añadidas el comportamiento, ajústalo de la siguiente manera:



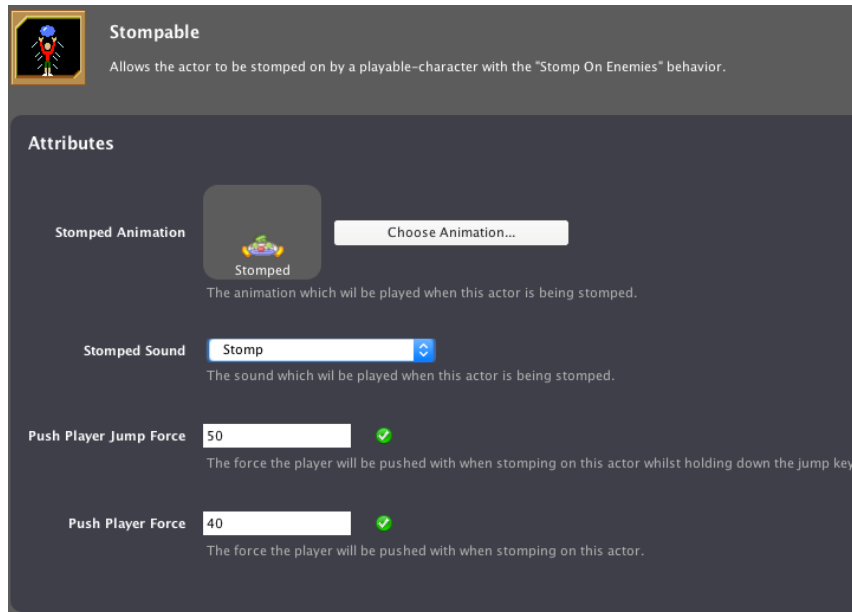
- **Stomp on Enemies**: haz que Noni pueda aplastar a sus enemigos. Para ello, añádele este comportamiento y configúralo de tal manera que caiga sobre el grupo de actores denominado **Enemies**.
- **Die in Pit and Reload**: por ultimo, si Noni cae fuera de los límites del escenario, debería morir y reiniciarse el juego. Este comportamiento, precisamente, nos asegura ese hecho. Añádeselo para que no haya problemas.

Genial, una vez estemos en este punto, habremos definido todas las cosas que **Noni** puede hacer.

#### 4.5.2 Configuración de **Clown**

Sólo nos queda configurar a **Clown**. En nuestro caso, los enemigos estarán a lo largo de los escenarios, esperando a que **Noni** los aplaste. Por lo tanto, tendremos que asociar el siguiente comportamiento:

- **Stompable**: este comportamiento definirá qué sucede cada vez que **Clown** es aplastado por **Noni**. En este caso, deberás configurar los parámetros para que coincidan con lo siguiente:



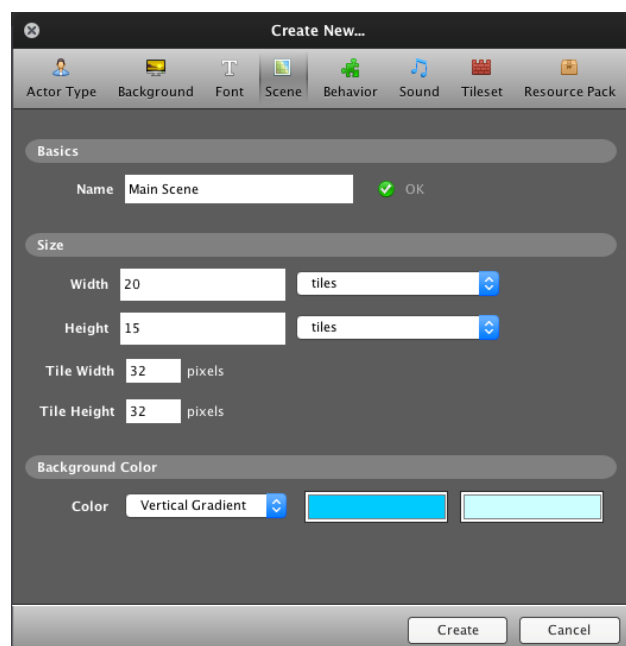
Perfecto. En este punto ya tendrás una configuración inicial de los personajes del juego. Luego, te animaremos a ir modificando los distintos parámetros de los comportamientos hasta que consigas el efecto adecuado a tu gusto.

## 4.6 Creación de escenas

Una vez que hemos configurado los personajes y establecido cómo se comportarán a lo largo del juego, es necesario crear las distintas escenas en las que se batirán **Noni** y **Clown**.

Para crear las escenas, vuelve a la pantalla **Dashboard**, y selecciona **Scenes**. Una vez ahí, pulsa en el centro y crea una nueva escena que se ajuste a la configuración que tienes en esta imagen de la derecha.

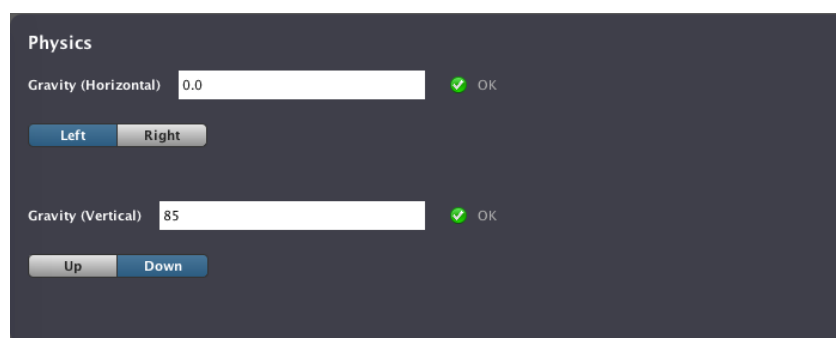
Una vez tengas la escena, que representa un día soleado (en el que se puede ver el cielo completamente azul),



selecciona las piezas (tiles) que están definidas para crear un escenario completo. También puedes colocar la posición de los diferentes actores dentro de la escena. Por ejemplo, la primera escena podría tener este aspecto:



Finalmente, para que nuestros actores puedan caer al suelo por la fuerza de la gravedad, deberemos definir (para esta escena), dentro del apartado **Physics**, las siguientes modificaciones:



## 4.7 Ejecución del juego

Llegados a este punto, lo único que nos queda es comprobar que nuestro juego funciona correctamente. Para ello, desde la pantalla **Dashboard** pulsa el botón **Test Game** y comprueba que todo funciona tal y como esperabas.

## 5 ¡Atrévete a investigar!

Muy bien, acabas de crear la base de tu juego, sin embargo, todavía queda tiempo de taller, ¿verdad? ¿estás motivado? ¿quieres que **Noni** sea el protagonista del juego más famoso de todos los tiempos? Pues atrévete con las diferentes actividades que te proponemos a continuación.

## 5.1 Intenta insertar nuevos elementos en las escenas

¿Qué sucedería si pones plataformas elevadas a las que **Noni** pueda acceder saltando? ¿Qué pasaría si, con los **Tiles** adecuados, formas montañas que **Noni** deba evitar? Intenta complicar la escena. Juega también con la gravedad que afecta a los personajes.

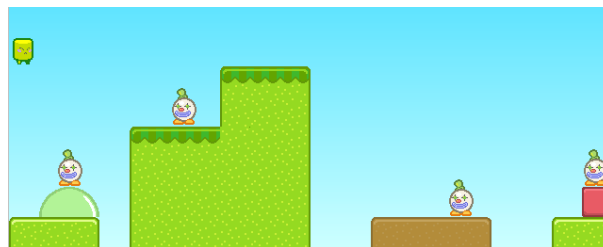
## 5.2 Juega con los distintos parámetros de los comportamientos

En primer lugar, estaría genial familiarizarse con todos los parámetros que se pueden configurar en los comportamientos de los actores y la escena. ¿Qué sucedería si cambiases la gravedad de la escena? ¿te has preguntado para qué sirven los parámetros **Push Player Force** y **Push Player Jump Force** del comportamiento de **Clown**? Quizá puedan servirte para que Noni acceda a diferentes alturas para seguir aplastando enemigos...

## 5.3 Crea diferentes escenas

Con lo que has aprendido en los dos apartados anteriores, intenta crear una nueva escena para que, el jugador habilidoso pase de pantalla.

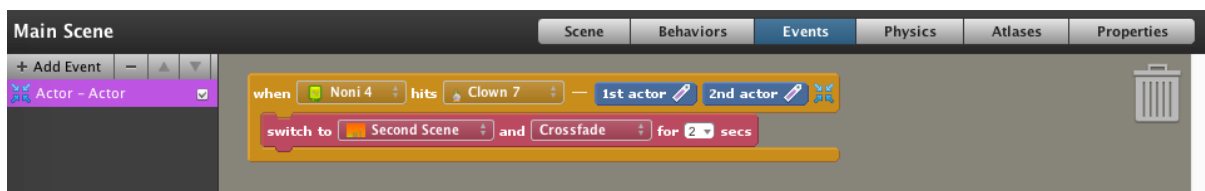
1. Crea una escena principal que sea compleja. Por ejemplo, que **Noni** tenga que superar distintos obstáculos (a diferentes alturas) y que para ello tenga que pisar a **Clown** (mientras pulsa la tecla de salto, como has aprendido en 5.2). Podría ser algo similar a:



2. Diseña otra pantalla. En este caso, por ejemplo, podría tratarse del atardecer, ¡ya que el día va avanzando!



3. Ahora haz que, cuando el jugador que maneja a **Noni** aplaste el **Clown** que se encuentra encima del bloque rojo de la primera escena, consiga pasar de pantalla.
  - a. Para ello, acuérdate de lo que hablamos de los eventos (definen cosas que suceden en el juego).
  - b. Abre escena principal y pulsa el menú de **Events**.
  - c. Pulsa **Add Event** y añade un evento de tipo **Collisions** que especifique **Any actor collides with Specific actor**. Configúralo de tal modo que quede de la siguiente manera:



Para ello, busca las acciones que puedes realizar en los bloques que tienes a la derecha de la pantalla. Están organizados dependiendo de los distintos aspectos que tratan. En este caso, como lo que queremos es controlar las escenas del juego, no estaría mal que mirases la sección de **Scenes**. Juega con las distintas transiciones que hay entre escenas (*Crossfade, Fade Out, ...*).

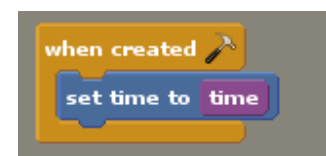
## 5.4 Añade un temporizador

No hay mejor manera de picar a un jugador que poniéndole un tiempo muy justo para que pueda pasarse la pantalla. Si quieres que tu juego sea muy adictivo, ¡mezcla lo visto en apartados anteriores con un tiempo muy ajustado para pasarse la pantalla!

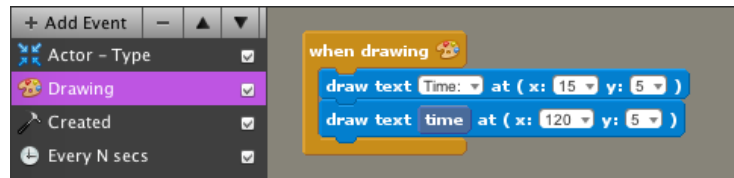
1. Para poner un temporizador, en primer lugar, debemos de crear un atributo de juego que lo almacene. En primer lugar, en la pantalla de eventos de la escena, busca en los bloques de la derecha **Game Attributes**. Una vez ahí, selecciona **Create New Game Attribute** para crear un atributo. Denomínalo **time**, que sea de tipo **Number** y que el valor inicial sea **30**.
2. Después, selecciona el bloque **Attributes** y crea un atributo que también se denomine **time** y que sea de tipo **Number**.

En este momento, ya habrás creado un atributo de juego denominado **time** que será de 30 segundos y un atributo **time** local que utilizaremos para cada escena.

Lo siguiente que debemos hacer es inicializar el atributo local cada vez que comience la escena, para ello, añadimos un evento de tipo **When Creating** con el código de la derecha.



Como queremos que los usuarios puedan ver el tiempo de la pantalla, tendremos que añadir un evento **When Drawing** (que ocurre cada vez que se actualiza alguna parte del juego) para que pinte el texto "Time:" y el valor del tiempo en cada momento (utilizando el atributo **time**).



Por último, añadiremos un evento cuya funcionalidad sea, cada segundo, restar un segundo del tiempo restante en esta pantalla, cambiar el color de **Noni** cuando queden 10 segundos del límite y, finalmente, volver a cargar la pantalla si el usuario no ha conseguido pasársela antes del límite.

1. Para ello, agregamos un evento de tipo **Time** -> **Every N Seconds**. Y lo configuramos de la siguiente manera (intenta no sólo copiar las instrucciones, sino intentar comprender lo que se hace 😊):



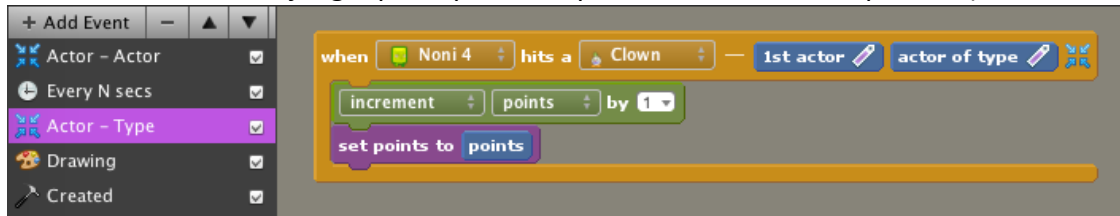
## 5.5 Añade un marcador

Está claro que a todo el mundo le gusta sacar la mejor puntuación posible en los juegos. ¿Porqué no le añades un marcador que lleve la cuenta de cuantos **Clown** ha aplastado **Noni**?

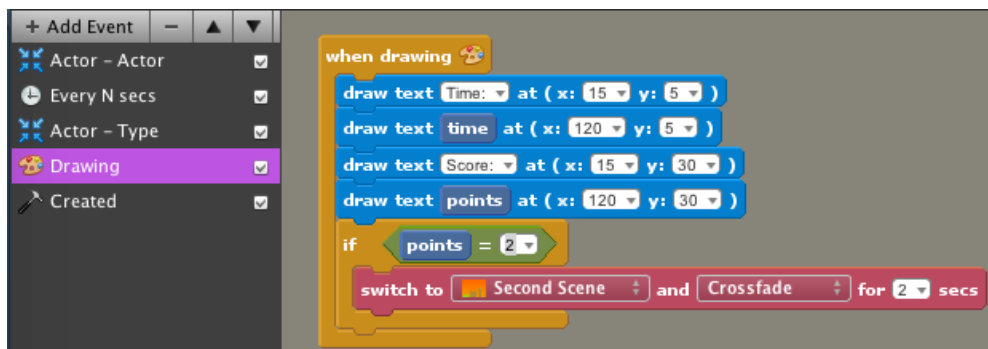
Para ello, sigue los primeros pasos de la sección 5.4, creando un atributo de juego (**Game Attribute**) denominado **points** y otro atributo también llamado **points** (ambos de tipo **Number**). Añade también (al mismo evento **Drawing** que tenías) el código adecuado para que también se pinten en pantalla los puntos (poniendo, por ejemplo, "Score:" con el valor del atributo **points**).

Una vez hayas hecho eso, tendrás que indicar que, cada vez que exista una colisión entre **Noni** y uno de los actores **Clown**, se suma un punto al marcador.

1. En la escena agrega un nuevo evento de tipo **Collisions** -> **Any Actor collides with Actor of Type**. Definiendo que el agente que colisiona es **Noni**, con cualquier actor de tipo **Clown** (si hubiésemos tenido más enemigos, podríamos haber seleccionado el evento de colisión de un actor con un grupo). Lo único que queda por hacer es incrementar los puntos (y después almacenarlos en el atributo de juego, para que no se pierdan al terminar la pantalla).



Ahora que sabes incrementar el marcador, y sabes cómo se define cuándo **Noni** choca con un **Clown** concreto (sección 5.3), podrías intentar asignar una puntuación extra a algunos **Clown** de difícil acceso. De esa manera tu juego podría pasar de pantalla de dos maneras: si se alcanza el **Clown** que se encuentra sobre el bloque rojo o cuando **Noni** haya aplastado algunos de los **Clown** más difíciles. Para ello, podríamos modificar el evento de **When Drawing**, añadiendo el código que comprueba los puntos en cada momento y cambia de pantalla cuando llegue la puntuación que queremos.



## 5.6 Haz que tus enemigos se muevan

Tenemos un juego muy adictivo, pero podemos complicarlo todavía más, ¡seguro que lo declaran juego del año!

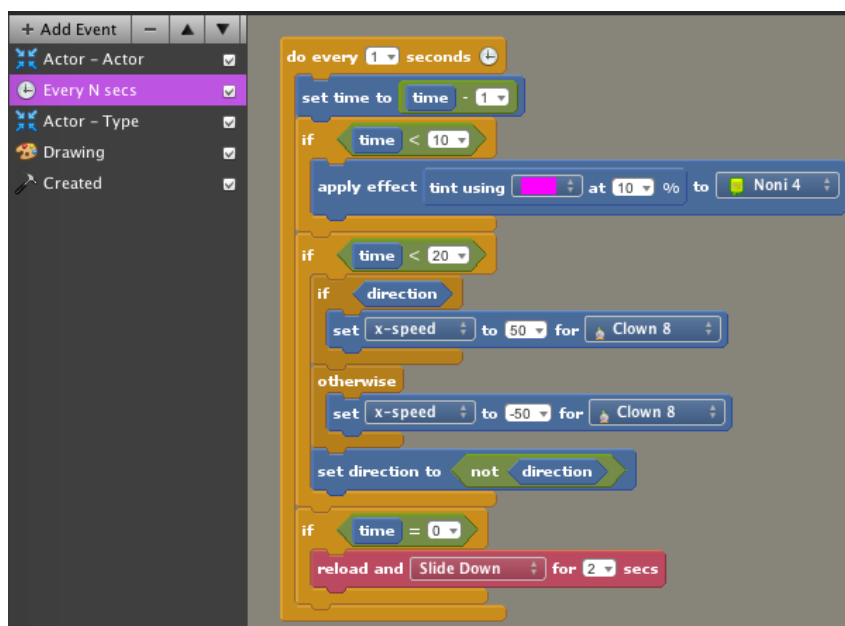
Como **Noni** utiliza a los **Clown** para poder saltar más alto y llegar a ciertas áreas que le permiten progresar por la pantalla, estaría genial que los **Clown** se moviesen, ¡así se lo pondrían más difícil a nuestro protagonista!

Existen muchas maneras de establecer este movimiento, pero una manera sencilla es hacer que determinados **Clown** cambien de posición cuando ha pasado cierto tiempo. De esta manera conseguiremos que Noni se ponga más nervioso y no pueda cumplir con su cometido.

1. Haz click en el evento **Every N secs** que definimos anteriormente para disminuir el tiempo y cambiar el color de **Noni**.

2. En ese evento estableceremos una condición que, cuando llegue determinado tiempo de juego, haga que se mueva uno de nuestros **Clown**.
  - a. Vamos a cambiar la velocidad en el *eje x* de uno de los actores.
  - b. Cada segundo cambiará el sentido de la velocidad, haciendo que se mueva de derecha a izquierda.
    - i. Para saber si la última vez que nos movimos lo hicimos a la derecha o a la izquierda, necesitaremos añadir un atributo (**Attribute**) de tipo **Boolean** que denominaremos **direction** y que inicializaremos a **false**.
  - c. De esta manera, cada vez que estemos por debajo del tiempo que hace que se mueva el **Clown**, miraremos la dirección anterior, nos moveremos en el sentido que toque y actualizaremos la dirección para dirigirnos al lado contrario la próxima vez.
  - d. ¡Ojo, no te olvides de poner unos **tiles** extra en tu escenario! En caso contrario, es posible que el **Clown** se caiga al vacío cuando se mueva.

Para conseguir este efecto, modifica el evento **Every N secs** de la siguiente manera:



## 5.7 ¡Hasta el infinito y más allá!

Ya lo decía *Buzz Lightyear* que era un juguete y sabe mucho de juegos: "el límite lo pone tu imaginación". Hasta este punto te hemos contado cómo puedes ir modificando el juego inicial para conseguir nuevas metas. Ahora te proponemos que investigues, que juegues, que pienses cómo te gustaría mejorar tu juego, que navegues entre los distintos bloques que *Stencyl* te ofrece e intentes ver qué otras cosas podrías añadir a tu aplicación.



## 6 Referencias

- [1] Stencyl. Página principal de la aplicación.  
<http://www.stencyl.com>
- [2] JAVA® JRE - versión 1.8. Página de descarga.  
<http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>
- [3] Stencyl. Página con tutoriales.  
<http://www.stencyl.com/help/start/>
- [4] Stencyl. Descarga de Crash Course.  
<http://www.stencyl.com/help/viewArticle/143>