Tabla de contenido

[Guía básica de creación de videojuegos. 2](#_Toc32428180)

[Porqué GameMaker Studio 2. 2](#_Toc32428181)

[Conociendo GameMaker Studio 2. 2](#_Toc32428182)

[Interfaz de GameMaker Studio 2 3](#_Toc32428183)

[Creando nuestro primer Sprite 5](#_Toc32428184)

[Creando nuestro primer Objeto 7](#_Toc32428185)

[Creando nuestra primer room 8](#_Toc32428186)

[Guía rápida de GML 9](#_Toc32428187)

[Variables. 9](#_Toc32428188)

[Cadena de texto. 9](#_Toc32428189)

[Números reales 10](#_Toc32428190)

[Booleanos. 10](#_Toc32428191)

[Expresiones. 11](#_Toc32428192)

[Expresiones aritméticas 11](#_Toc32428193)

[Expresiones Comparativas 11](#_Toc32428194)

[Expresión Combinativas 12](#_Toc32428195)

[Estructura condicional. 13](#_Toc32428196)

[If 13](#_Toc32428197)

[While 15](#_Toc32428198)

# Guía básica de creación de videojuegos.

El propósito de este curso es crear videojuegos de una manera fácil y rápida, con la posibilidad de expandir tu proyecto. Adquirirás los conocimientos básicos y esenciales que podrás utilizar para crear tus propios videojuegos.

# Porqué GameMaker Studio 2.

GameMaker Studio 2, es un programa creado por “YoYo Games” para desarrollar tus propios videojuegos. GameMaker Studio 2(GMS) pone a nuestra disposición todas las herramientas requeridas para desarrollar videojuegos de manera profesional, de una forma que sea cómodo para cualquier persona desde un desarrollador avanzado, hasta un estudiante de 10 años sin experiencia desarrollando videojuegos.

En definitiva, GameMaker Studio 2 es una excelente herramienta para iniciar con el desarrollo de videojuegos, esa es la razón por la cual fue seleccionada para este curso.

## Conociendo GameMaker Studio 2.

Tú puedes crear juegos con GMS sin saber escribir código, esto es porque GMS tiene la opción de usar drag-and-drop que proporciona una interfaz para decirle a GMS que, como, cuando y donde hacer algo en nuestro videojuego.

Las drag-and-drop proporciona una forma rápida de hacer el código necesario para nuestro videojuego. Pero para desarrollar un proyecto serio, o incluso un proyecto profesional debemos tener la mente abierta para aprender como programar, esto beneficiará nuestros juegos, y nos dará una idea más profunda de cómo funciona nuestro juego.

## Interfaz de GameMaker Studio 2

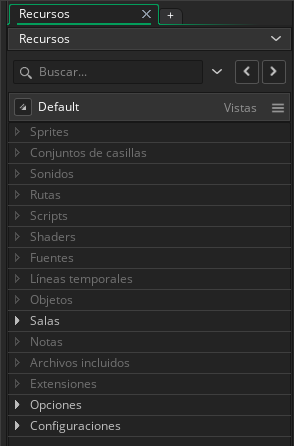
Aquí se muestra como es la interfaz inicial de GMS 2, donde podemos principalmente, crear un nuevo proyecto o cargar un nuevo proyecto.

 Al crear un nuevo proyecto, nos dará dos opciones: Arrastrar y soltar o Idioma GameMaker

Como mencionábamos en la introducción a GameMaker, existen dos opciones para el desarrollo de nuestro videojuego, Drag-and-drop (arrastrar y soltar) que nos permite mediante una interfaz de asistencia programar nuestro videojuego y el idioma GameMaker (GML) que nos permite desarrollar nuestro juego de una manera más profesional y entendible con un lenguaje de programación propio de GameMaker. Es conveniente empezar con GML ya que esto nos dará pie para movernos a otros programas de desarrollo más complejos, que de otra forma si usamos Drag-and-drop la transición nos será sumamente difícil.

Una vez seleccionada la opción, y creado el proyecto, estaremos en la interfaz de desarrollo, en la parte derecha nos mostrara un menú de recursos, donde estarán las principales herramientas de desarrollo de nuestro videojuego.

En el listado de recursos nos encontramos muchos apartados, pero en los que nos enfocaremos serán:

Sprites: Son Las imágenes necesarias para nuestro videojuego.

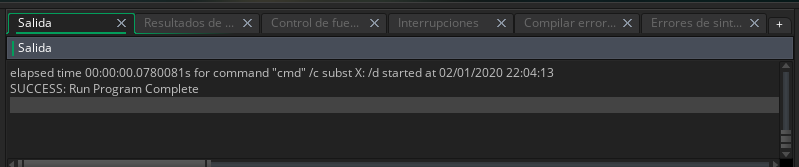
Sonidos: Son los efectos de sonido necesarios para nuestro videojuego.

Objetos: Son todos los elementos de nuestro videojuego como plataformas, enemigos, el personaje del jugador etc.

Salas: Es el lugar donde almacenaremos nuestros elementos del juego, cada sala es un nivel del juego.

Al hacer clic derecho en el recurso podemos crear cada uno de ellos.

El panel inferior nos servirá para detectar errores o depurar nuestro juego.

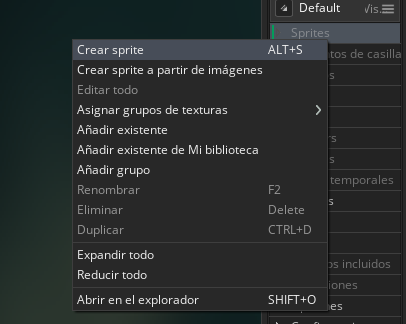


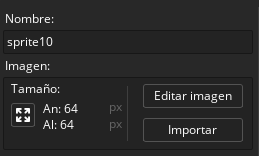
En el panel superior nos servirá para navegar en el espacio de trabajo, guardar, o ejecutar nuestro juego



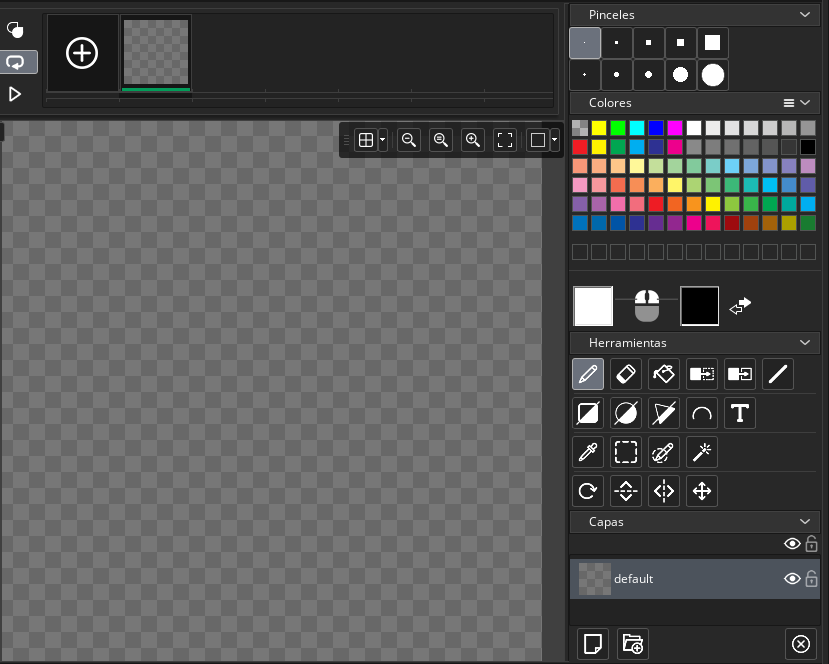
# Creando nuestro primer Sprite

Los sprites generalmente son representaciones visuales de los objetos dentro de los juegos que creamos. Un Sprite puede estar hecha con el editor integrado de GMS o pueden importarse imágenes de un editor externo.

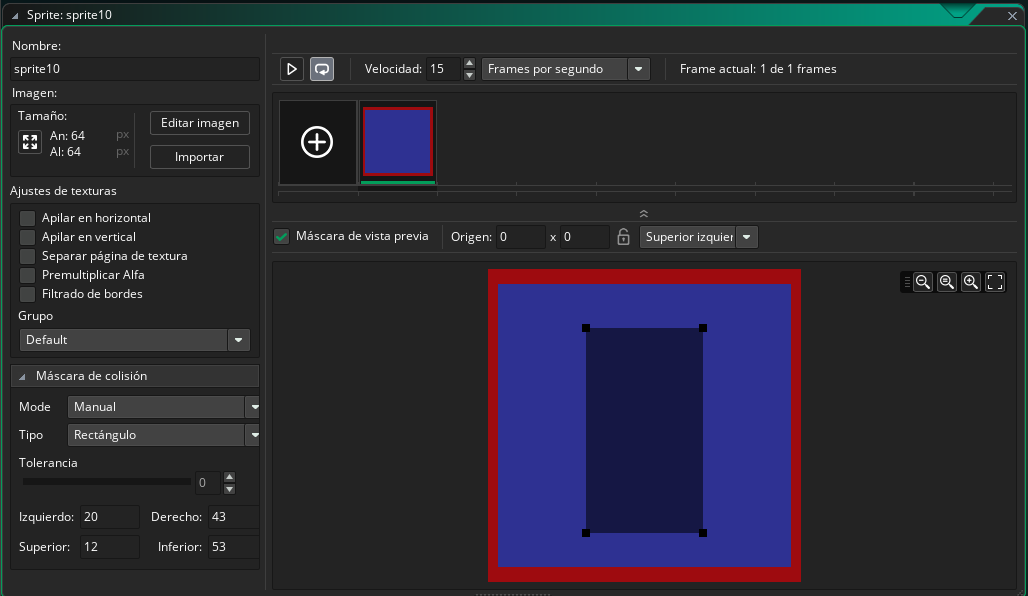
Al hacer clic derecho en el recurso de Sprites se despliega un menú que nos permitirá crear un nuevo Sprite, en la pestaña de espacio de trabajo se abrirá un menú en el que nos permite configurar nuestro Sprite, los sprites pueden estar conformados en base a varias imágenes para formar una animación, o pueden están formado de una sola imagen, en el menú podremos abrir una imagen que ya tengamos echa o crear nuestra propia imagen.



Si ingresamos en la opción editar imagen tendremos a nuestra disposición herramientas para crear una imagen en 2D como lápiz, borrador, opciones de figuras, selección etc.

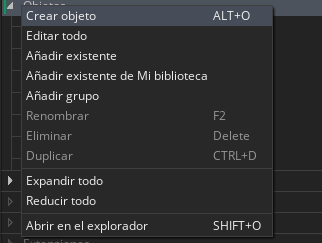


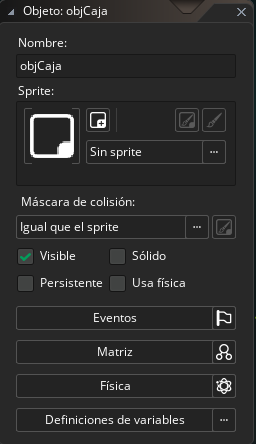
Una vez creada la imagen tenemos que saber que nuestros sprites necesitan una máscara de colisión, Una máscara de colisión es lo que representa el punto de contacto entre un objeto y otro, esto nos servirá para saber si nuestra imagen que creamos tiene contacto con otros elementos, volviendo al espacio de trabajo en el submenú de Mascara de colisión del Sprite en el tipo de menú por defecto lo encontraremos en automático, si queremos modificar la forma de esta mascara de colisión.



# Creando nuestro primer Objeto

Los objetos son un recurso especial que utilizamos para controlar aspectos de un juego, es decir son todos los elementos para interactuar en el juego. La mayoría de las veces tienen un Sprite asociado con ellos para que los veas en la sala de juegos.

Al hacer clic derecho en el recurso objeto, podremos crear objetos, al crear un objeto tendremos que poner un nombre al objeto, es una buena practica utilizar el prefijo “obj” en el nombre de nuestro objeto, para cuando tengamos interacciones con este lo podamos identificar fácilmente, en este caso creamos el objCaja.

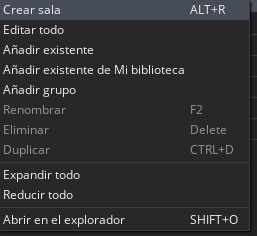


Al crearlo, podemos establecer algunos atributos, como por ejemplo podemos asignar los sprites que creamos con anterioridad a nuestro objeto, esto haciendo clic en el recuadro con el texto “Sin Sprite”.

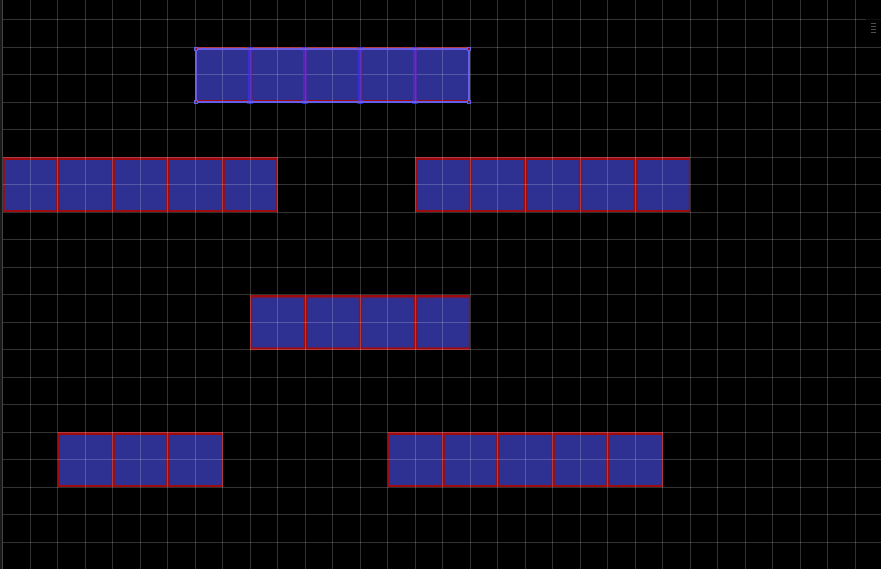
Nos mostrara una lista de los sprites disponibles, al hacer clic en alguno de ellos, lo asignaremos en nuestro objeto

Creando nuestra primer room**.**

Todos los juegos que se crean necesitan al menos una room para funcionar (pero pueden tener muchas más rooms), una room es básicamente el espacio donde colocas los objetos que conforman tu juego.



Haciendo clic derecho en el recurso room, podremos crear salas, en la sala que creamos podremos arrastrar los objetos que creamos, objetos a los que posteriormente podremos asignar tareas. Al crear una room tendremos que asignarle un nombre.

En la Room creada, en el menú de recursos haremos clic sobre el objeto que queramos colocar, una vez echo eso presionando alt+clic podremos “pintar” o colocar libremente con el objeto en nuestra sala

# Guía rápida de GML

Lenguaje GameMaker (GML) debe tener una estructura básica para funcionar, a continuación, veremos un ejemplo general de esta estructura.

Un código de GML consta de un conjunto de instrucciones, que GameMaker Studio interpreta y utiliza para que suceda algo dentro del juego. Ese “algo puede ser tan simple como sumar 2 más 2 para obtener 4, o tan complejo como hacer que un enemigo huya cuando su salud se pone por debajo de cierto valor.

Las instrucciones deben separarse con un símbolo “;” para evitar errores y mantener el código limpio y ordenado.

También hay instrucciones que se “agrupan utilizando los símbolos “{“ y “}” que veremos más adelante.

# Variables.

En GML se usan variables para guardar datos, con datos nos referimos a que se puede guardar información, a la que podemos acceder cuando queramos.

Las variables pueden tener cualquier nombre mientras no tengan espacios y que no los hayas utilizado anteriormente, y utilizaremos el operador “=” para asignarle información. Sin embargo, esta información puede venir en diversos “sabores”, como, por ejemplo, puede ser un número, o un texto, a esto se le denomina tipos de datos y puede ser cualquiera de los siguientes:

## Cadena de texto.

Es simplemente cualquier texto que se haya colocado entre comillas

Por ejemplo:

nombre = “Javier”;

como podemos observar en el ejemplo, nuestra variable seria “nombre”, para asignarle un valor utilizaremos el “=”, y el valor que asignaremos como será una cadena de texto tendría que estar encerrada entre comillas “Javier” y el punto y coma para indicar el fin de la instrucción.

Y así dentro de nuestra variable nombre almacenaremos el nombre de “Javier”.

## Números reales

Un Numero no es una cadena de texto, por lo que para almacenar este número no usaremos las comillas, sino que escribiremos solamente el número, pero también podremos guardar decimales utilizando el punto para indicar las decimales, o para guardar un numero negativo utilizaremos el símbolo menos para indicar el numero negativo.

Por ejemplo

Numero1 = 10;

Aquí a la variable Numero1 le asignaremos el valor 10 y “;” para indicar el fin de la instrucción.

Numero2 = 20.20;

Aquí a la variable Numero2 le asignaremos el valor decimal 20.20.

Numero3 = -30;

Aquí a la variable Numero3 le asignaremos el valor negativo -30.

## Booleanos.

Una variable booleana básicamente puede almacenar dos valores, false(falso) o true(verdadero), estas variables son muy útiles al momento de controlar lo que sucede en nuestro videojuego, el valor (false o true) no se pone entre comillas “” ¡Sino se convertiría en una cadena de texto!, aquí un pequeño ejemplo de su uso.

estaVivo = true;

Aquí a la variable “estaVivo” le estamos asignando el valor true(verdadero), entonces podemos usar esta variable como un indicador de que nuestro jugador está vivo.

En caso contrario;

estaVivoEnemigo = false;

aquí la variable “estaVivoEnemigo” le estamos asignando el valor false(falso) entonces podemos usar esta variable como un indicador de que el enemigo está muerto.

# Expresiones.

## Expresiones aritméticas

Como en las funciones matemáticas, para realizar una operación utilizaremos los siguientes símbolos:

|  |  |  |
| --- | --- | --- |
| Expresión |  | Ejemplo |
| + | Para sumar | numeroFinal = numero1 + numero2; |
| - | Para restar | resultadoResta = numero1 - numero2; |
| \* | Para multiplicar | resultadoMultiplicacion = numero1 \* numero2; |
| / | Para dividir | numeroDivision = numero1 / numero2; |

## Expresiones Comparativas

Existen otro tipo de expresiones, estas se usan de forma comparativa, como por ejemplo para saber si un numero es mayor a otro, menor o igual. Estas expresiones generalmente dan como resultado “Verdadero” si se cumple o “Falso” si no se cumple. Y se usan en bloques de código especiales que veremos mas adelante.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expresión** | **Significado** | **Ejemplo** | |
| < | Menor | 1 < 2 | 5 < 6 |
| Esto es **Verdadero**. | Esto es **Falso**. |
| <= | Menor o igual | 5 <= 5 | 4 <= 5 |
| Esto es **Verdadero**. | Esto es **Falso**. |
| == | Igual | 7 == 7 | 7 == 6 |
| Esto es **Verdadero**. | Esto es **Falso**. |
| != | Diferente que | 5 != 6 | 5 != 5 |
| Esto es **Verdadero**. | Esto es **Falso**. |
| > | Mayor | 5 > 3 | 7 > 3 |
| Esto es **Verdadero**. | Esto es **Falso**. |
| >= | Mayor que | 5 >= 5 | 5 >= 7 |
| Esto es **Verdadero**. | Esto es **Falso**. |

## Expresión combinativas

Se usan entre las expresiones comparativas para agrupar dos o más comparaciones

|  |  |  |
| --- | --- | --- |
| Expresión | Significado | Ejemplo |
| && | Se usa “&&” cuando quieres que se cumplan 2 o más expresiones comparativas, si no se cumple alguna de las 2 expresiones da como resultado “falso”, si se cumplen las 2 expresiones da como resultado “verdadero”. | 10 == 10 && 20 >10 |
| Esto es **Verdadero**.  Porque se cumplen las 2 expresiones |
| 10 == 10 && 3 < 1 |
| Esto es **Falso.**  Porque no se cumple la segunda expresión |
| || | Se usa “||” Cuando quieres que se cumpla alguna de las 2 o más expresiones comparativas, si no se cumple NINGUNA de las expresiones da como resultado “falso”, si se cumple alguna de las dos expresiones da como resultado “verdadero”. Si se cumplen las 2 también da “verdadero” | 10 == 10 || 20 >10 |
| Esto es **Verdadero**.  Porque se cumple la primera expresión, aunque la segunda no. |
| 10 == 5 || 3 < 1 |
| Esto es **Falso.**  Porque no se cumple ninguna de las expresiones. |

# Estructura condicional.

En los lenguajes de programación en general hay palabras que están reservadas porque tienen un funcionamiento especial dentro del lenguaje, estas palabras en este caso son “if” y “while” que nos permitirán hacer muchísimas cosas dentro de nuestro videojuego, estas palabras deben tener una estructura en especial para funcionar.

## If

If en español se traduce como si…, es como decir “Si esto es correcto, entonces has esto”

Por lo que en esta estructura tendremos que utilizar las expresiones comparativas vistas anteriormente, también si recordamos al inicio explicábamos que hay instrucciones de código que se agrupan mediante “{“ y “}” en este caso el if (y el while también) es una de ellas.

Aquí un ejemplo de la estructura del if:

|  |  |
| --- | --- |
| If vida == 0 | básicamente esto significa  Si(if) la variable vida tiene un valor de 0 entonces has lo que hay dentro de los siguientes “{ }” |
| { | Si se cumple la expresión comparativa entonces se ejecutarán las instrucciones dentro del { |
| nombre = “Daniel” | Como la vida es igual a 0, entonces la variable nombre tomará el valor de “Daniel” |
| } | Cuando se acaben estas instrucciones se indicara que terminamos el if utilizando el } de cierre.  También si la expresión no se cumple inmediatamente pasara a esta parte y el código de en medio de { código } no se ejecutará. |

Entonces la estructura seria la siguiente:

if *expresión comparativa*

{

*Si se cumple la expresión.*

*Código que queramos ejecutar…*

}

*Si no se cumple la expresión.*

Y Si siempre inmediatamente después de colocar la expresión comparativa del if se debe seguir con los { }.

## While

while en español significa “Mientras” es decir, “Mientras esto sea correcto, entonces has esto”, el while tiene la misma estructura que la del if, solamente que en este caso el while se repetirá por siempre hasta que la expresión comparativa se deje de cumplir.

Aquí un ejemplo de la estructura del while:

|  |  |
| --- | --- |
| While vida > 0 | básicamente esto significa.  Mientras(while) la variable vida tenga un mayor a 0 entonces has lo que hay dentro de los siguientes “{ }” |
| { | Si se cumple la expresión comparativa entonces se ejecutarán las instrucciones dentro del { |
| Estado=”vivo”; | Como la vida es mayor a 0, entonces la variable estado tomará el valor de “vivo” |
| } | Cuando se deje de cumplir la expresión comparativa se dejara de ejecutar el código y pasara a la } de cierre.  También si la expresión no se cumple inmediatamente pasara a esta parte y el código de en medio de { código } no se ejecutará. |

En general Esta es una vista rápida y completa de los aspectos básicos de GameMaker Studio 2, aun así, hay muchísimas más opciones para crear nuestro juego prácticamente como deseemos, también tocamos aspectos importantísimos que son las bases de la programación de una forma interactiva y aplicada, si nos familiarizamos rápidamente con estos términos, no tendremos problemas al momento de desarrollar cualquier aplicación.

Una vez entendidos estos términos podremos empezar a crear nuestro propio videojuego ¡en GameMaker Studio 2 !.