TIC-80

TIC-80 es un entorno de desarrollo de videojuegos que simula un ordenador de los años 80. Dichas limitaciones permiten escribir un juego en poco tiempo comparado con el tiempo de desarrollo de un juego actual.
Los juegos creados en TIC-80 pueden exportarse como “cartuchos” o exportalos a distintas plataformas, como Android, Linux, MacOS, Windows, Raspberry Pi, Nintendo 3DS, RetroArch, y HTML5.
Se pueden emplear varios lenguajes de programación como Lua, Javacript, MoonScript, Ruby, Wren, Fennel, Squirrel, Python y D.

Para usarlo sólo tenemos que descargarlo desde su página web y ejecutarlo en nuestro equipo tanto en Linux, Windows, Android, HTML,… o instalarlo por el gestor de paquetes de nuestra distribución de Linux correspondiente.

El entorno cuenta con una consola, un editor de texto, un editor de sprites, editor de mapas, editor de música y editor de sonidos. A estas herramientas se accede usando las teclas F1, F2, F3,…

El manual de uso de TIC-80 se encuentra en el siguiente enlace:

https://tic80.com/learn

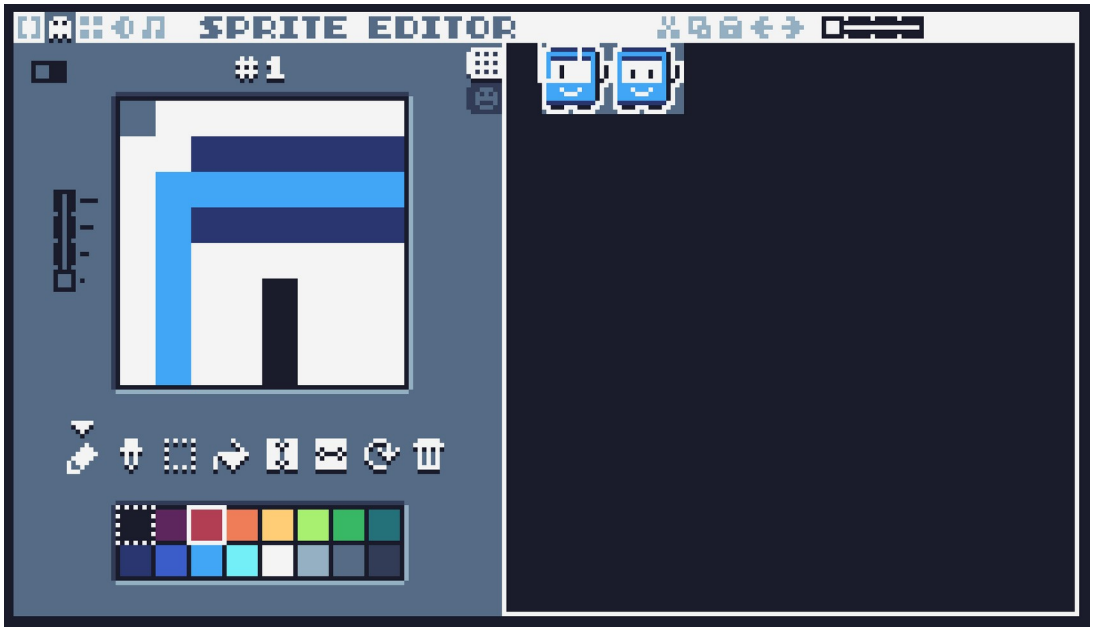
La pantalla se va a dividir en 240x136 píxeles y 16 colores. Los sprites tienen por defecto 8x8 píxeles.

Uso básico de la consola

Para crear un nuevo proyecto en Javascript, se debe ejecutar el comando:

```
new js
```

Se creará un proyecto con un ejemplo, mínimo que permite mover un sprite por la pantalla. Pulsando F1 se puede consultar el código. Pulsando F2 se puede ver el editor de sprites. Se puede volver a la consola usando la tecla ESC.



En el caso de se quiera guardar el proyecto, se pulsará ESC y en la consola ejecutará:
```
save nombre
```

Cuando se está trabajando, se puede escribir:
```
save
```
sin poner el nombre para ir guardando el proyecto actual.

Para volver a cargar el proyecto se ejecutará:
```
load nombre.tic
```

Para ejecutar el proyecto se escribirá:
```
run
```

Para salir del juego se pulsará ESC.

Para poder añadir o sacar los archivos ".tic" (cartuchos en la notación de TIC-80) que son donde se almacenan los proyectos, se puede ejecutar:
```
folder
```
que abre un navegador de ficheros con el que se tienen acceso a los contenidos de TIC-80.

Con el comando help se tiene acceso a la ayuda.

Limitaciones de Javascript

La versión de Javascript incluida en TIC-80 tiene algunas limitaciones.

- No se puede usar let. Se usará var en su lugar.
- En la función print() no se puede concatenar un string y un número.

El primer ejemplo

Si se estudia el código de este primer ejemplo:

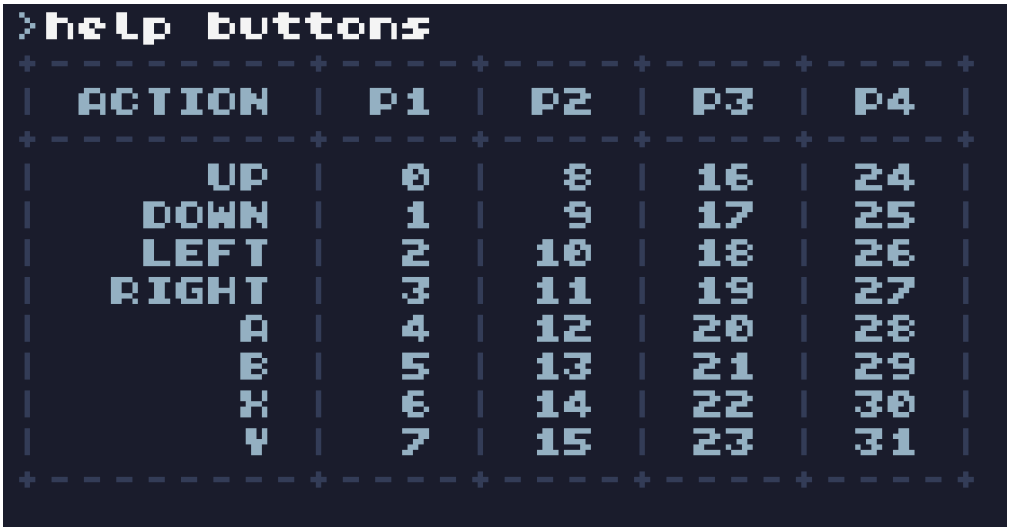
```
var t=0
var x=96
var y=24

function TIC()
{
      if(btn(0))y--
      if(btn(1))y++
      if(btn(2))x--
      if(btn(3))x++

      cls(13)
      spr(1+((t%60)/30|0)*2,x,y,14,3,0,0,2,2)
      print("HELLO WORLD!",84,84)
      t++
}
```

La función **TIC()** se ejecuta 60 veces por segundo y es la encargada de manejar la lógica del juego.

La función **btn(número)** devuelve true o false en función de si la tecla ha sido pulsada o no. Cada número corresponde a un botón del mando virtual que simula. En el caso de usar un ordenador cada valor corresponde a las teclas del cursor. En el caso de usar Android se mostrará un mando virtual en la pantalla con las flechas y los botones de disparo.

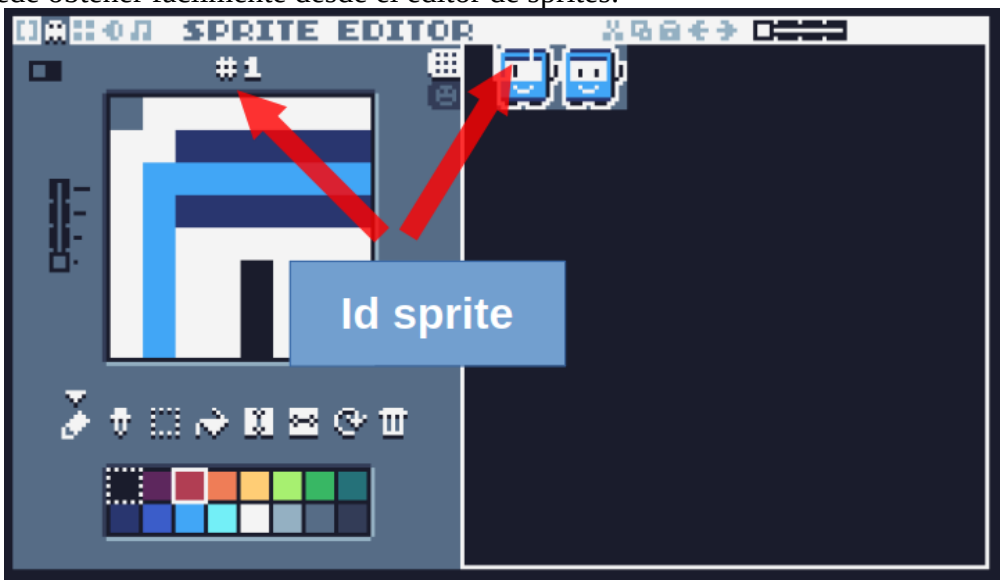


La función **cls(número)** limpia la pantalla y pone el color que se indique en el número. Cada color tiene asociado un número.

La función **spr(id, x, y, colorkey=-1, scale=1, flip=0, rotate=0, width=1, height=1)** dibuja un sprite en pantalla en la posición que se le indique. Según la documentación:
Dibuja el sprite número índice id en las coordenadas x e y.
Puede especificar una clave de color, colorkey, en la paleta que se utilizará como color transparente o utilizar un valor de -1 para un sprite opaco.
El sprite puede ser escalado por un factor deseado. Por ejemplo, un factor de escala, scale, de 2 significa que un sprite de 8x8 píxeles se dibuja en un área de 16x16 de la pantalla.
Puede reflejar , flip, el sprite donde:
- 0 = No voltear
- 1 = Reflejar horizontalmente
- 2 = Reflejar verticalmente
- 3 = Reflejar vertical y horizontalmente

Cuando giras el sprite, rotate, se gira en el sentido de las agujas del reloj en 90 grados:
- 0 = Sin rotación
- 1 = 90 de rotación
- 2 = 180 rotación
- 3 = 270 rotación
Puedes dibujar un sprite compuesto (consistente en una región rectangular de sprites de la hoja de sprites) especificando los parámetros `width` y `height` (que por defecto son 1).

El id se puede obtener fácilmente desde el editor de sprites:



En el ejemplo, aparece el código:

```
spr(id=1+((t%60)/30|0)*2,x,y,colorkey=14,scale=3,flip=0,rotate=0,width=2,height=2)
```

que significa:
En este caso el id es 1+((t%60)/30|0)*2, lo que está haciendo es una función matemática que depende del tiempo y cambia el resultado del id desde la posición 1 a la 3. Por ello se verá que el personaje mueve los ojos.

La función **print(texto,x,y)** muestra un texto en la posición indicada.

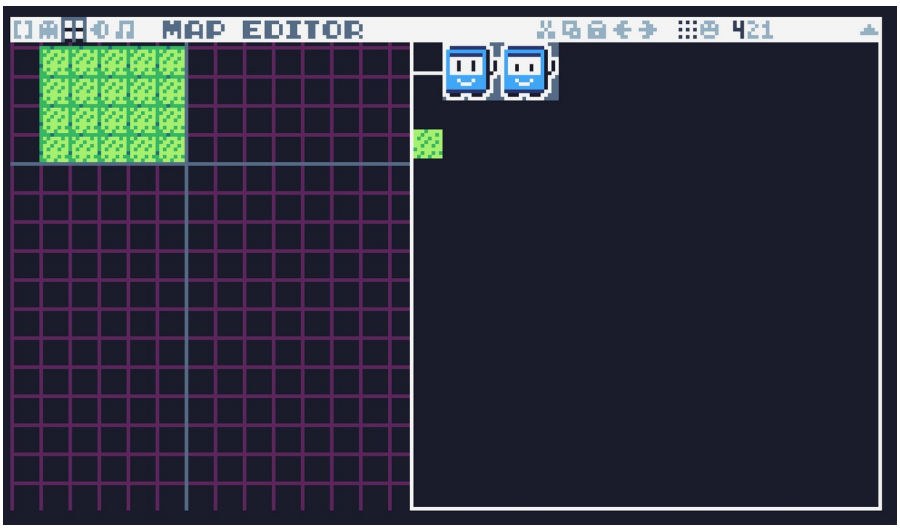
En el código también se puede ver que se definen tres variables globales, aunque se pueden definir las que se necesiten:

```
var t=0     // Representa el tiempo
var x=96    // Representa la posición x del héroe
var y=24    // Representa la posición y del héroe
```

Con estos conocimientos tan básicos se puede crear un juego del tipo el dinosaurio de Google Chrome en poco tiempo.

El mapa

Con la tecla F3 se puede acceder al editor de mapas:



Pulsando la flecha que hay en la parte superior derecha, se puede acceder a los sprites y seleccionar el que se quiere dibujar.

Es importante distinguir la posición en la que estamos dibujando. Por ello cuando pongamos el cursor sobre el mapa **aparecerán las coordenadas de la casilla en la que estamos**. Estas coordenadas serán después importantes.

Con los iconos de la parte superior:

se puede seleccionar la parte del mapa que se desea editar o quitar o poner la rejilla.

También hay un icono de una mano en la parte superior derecha que permite coger y desplazar el mapa de un lado a otro.

El manual de TIC-80 describe la función map que sirve para dibujar una porción del mapa en la posición que se le indique de la pantalla. Por ejemplo:

```
map(5,6,30,17,0,0)
```

Dibujará una porción del mapa de un tamaño de 30x17 desde las coordenadas del mapa (5,6) a las coordenadas de pantalla (0,0). 30x17 es justamente el tamaño de un pantalla de TIC-80.

La descripción completa de map según el manual:

```
map(x=0, y=0, w=30, h=17, sx=0, sy=0, colorkey=-1, scale=1, remap=nil)
The map consists of cells of 8x8 pixels, each of which can be filled with a
sprite using the map editor.
The map can be up to 240 cells wide by 136 deep.
This function will draw the desired area of the map to a specified screen
position.
For example, map(5,5,12,10,0,0) will draw a 12x10 section of the map, starting
from map coordinates (5,5) to screen position (0,0).
The map function's last parameter is a powerful callback function for changing
how map cells (sprites) are drawn when map is called.
It can be used to rotate, flip and replace sprites while the game is running.
Unlike mset, which saves changes to the map, this special function can be used
to create animated tiles or replace them completely.
```

```
Some examples include changing sprites to open doorways, hiding sprites used to
spawn objects in your game and even to emit the objects themselves.
The tilemap is laid out sequentially in RAM - writing 1 to 0x08000 will cause
tile(sprite) #1 to appear at top left when map() is called.
To set the tile immediately below this we need to write to 0x08000 + 240, ie
0x080F0.
```

También son interesantes las funciones:

```
mget(x, y) -> tile_id
Gets the sprite id at the given x and y map coordinate.
```

Y:

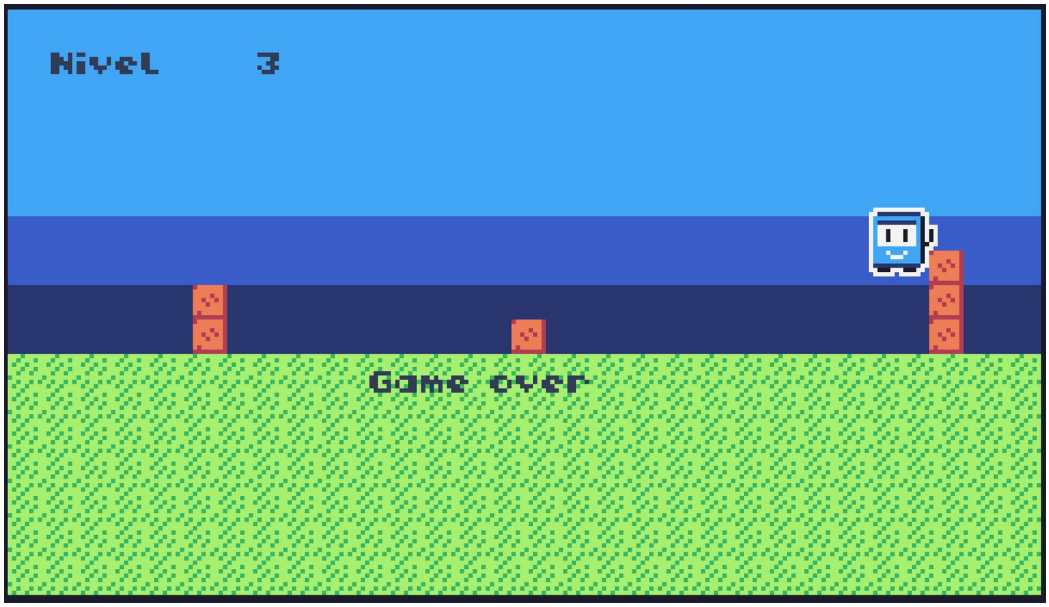
```
mset(x, y, tile_id)
This function will change the tile at the specified map coordinates.
By default, changes made are only kept while the current game is running.
```

Se pueden usar para consultar o cambiar los sprites que hay en una posición del mapa. Se podría poner un sprite de un tesoro en una posición, consultar si la posición en la que está el héroe está el sprite correspondiente al tesoro y cambiar el sprite del tesoro por un sprite de suelo cuando se haya recogido el tesoro.

Nota: En este manual se ha limitado al uso del teclado o mando, pero también se puede usar el ratón. Se recomienda consultar la función mouse.

Un ejemplo de juego

Este ejemplo es un simple juego que consiste en saltar obstáculos:



Nota: Para saltar se usarán los 4 botones de disparo o las teclas “z” o “x” en un teclado.

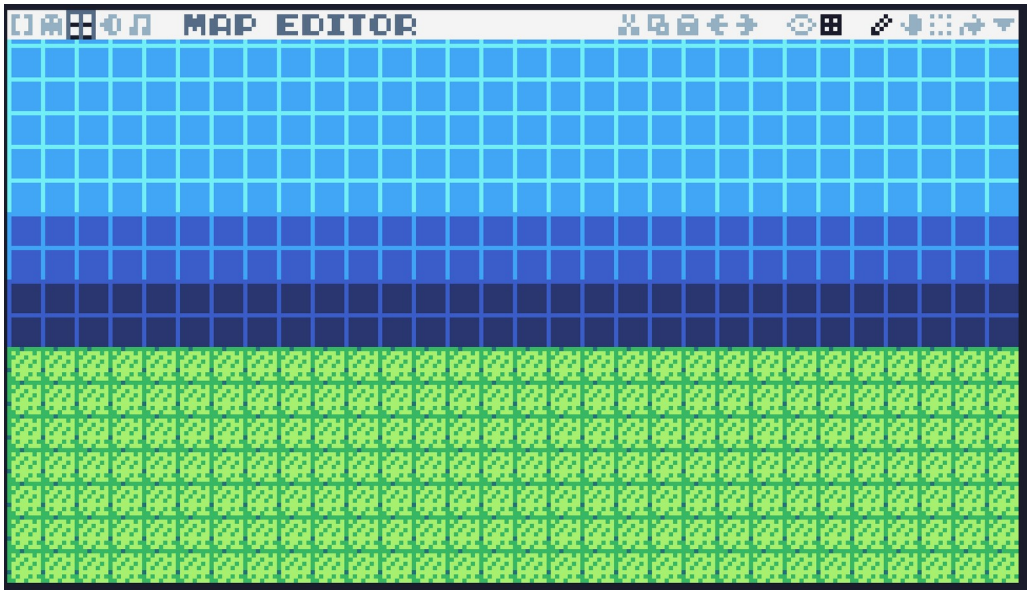
Se ha creado el proyecto usando el comando:

```
new js
```

Los sprites que se han generado son los siguientes:

Como se puede ver se han reusado los que vienen por defecto cuando se crea un nuevo proyecto y se han añadido 5 sprites más. Es importante mantener las posiciones de los sprites para que el código funcione sin problemas.

Como mapa, en la posición (0,0) del mapa se ha puesto (importante: mover el mapa a la posición 0,0 antes de ponerse a dibujar):



El javascript correspondiente es el siguiente:

```
// title:   game title
// author:  game developer, email, etc.
// desc:    short description
// site:    website link
// license: MIT License (change this to your license of choice)
// version: 0.1
// script:  js

var t=0
var fin=false
const GRAVEDAD = 0.6
```

```
const SUELO = 64
const heroe = {x:200, y:SUELO, v:10}
function Objeto(altura) {
      this.altura = altura
      this.x = -8 - Math.random()*128
      this.y = SUELO +16 - altura*8
      this.dibujar = function(){
       var i;
            for(i=0;i<altura;i++)
             spr(6,this.x,this.y+i*8)
      }
      this.avanzar = function(){
            this.x += 2
            if(this.x>240){
                  this.x = -8 - Math.random()*64
            }
      }
      this.colision = function(){
            if(
             heroe.x<this.x&&this.x<(heroe.x+16)
             &&
                  (heroe.y+16)>this.y
            ){
                  return true
            }
            return false
      }
}
var obs = []
var nivel = 0

function TIC()
{
 if(!fin){
  // Se aumenta el nivel
  if((t%600)==0){
      obs.push(new Objeto((nivel%4)+1))
      nivel++
  }
       if(btn(4)||btn(5)||btn(6)||btn(7)){
            if(heroe.y==SUELO){
            heroe.v = -9
        }
       }
            if(heroe.y!=SUELO || heroe.v!=0){
            heroe.y += heroe.v
            heroe.v += GRAVEDAD
       }
            if(heroe.y>SUELO){
                  heroe.y=SUELO
                  heroe.v=0
            }
            map(0,0,30,17,0,0)
            print("Nivel",10,10)
            print(nivel,58,10)
            for(var i=0;i<obs.length;i++){
                  obs[i].avanzar()
                  obs[i].dibujar()
            }
            spr(1+((t%60)/30|0)*2,heroe.x,heroe.y,14,1,0,0,2,2)
            for(i=0;i<obs.length;i++){
                  if(obs[i].colision()){
                        fin=true
                        print("Game over",84,84)
```

```
                    }
                }
                t++
        }
}
```

Usando el proyecto en un navegador

Sólo hay que ir a la página de Github de TIC-80 y descargar la versión para HTML:

https://github.com/nesbox/TIC-80/releases

Una vez descargada, se descomprime y se guarda en la misma carpeta que el archivo “index.html” nuestro proyecto con el nombre “cart.tic”.

Para que funcione correctamente se debe usar un servidor web. Desde Linux sólo hay que ejecutar el comando:

```
python3 -m http.server
```

y conectarse a la URL:

http://localhost:8000