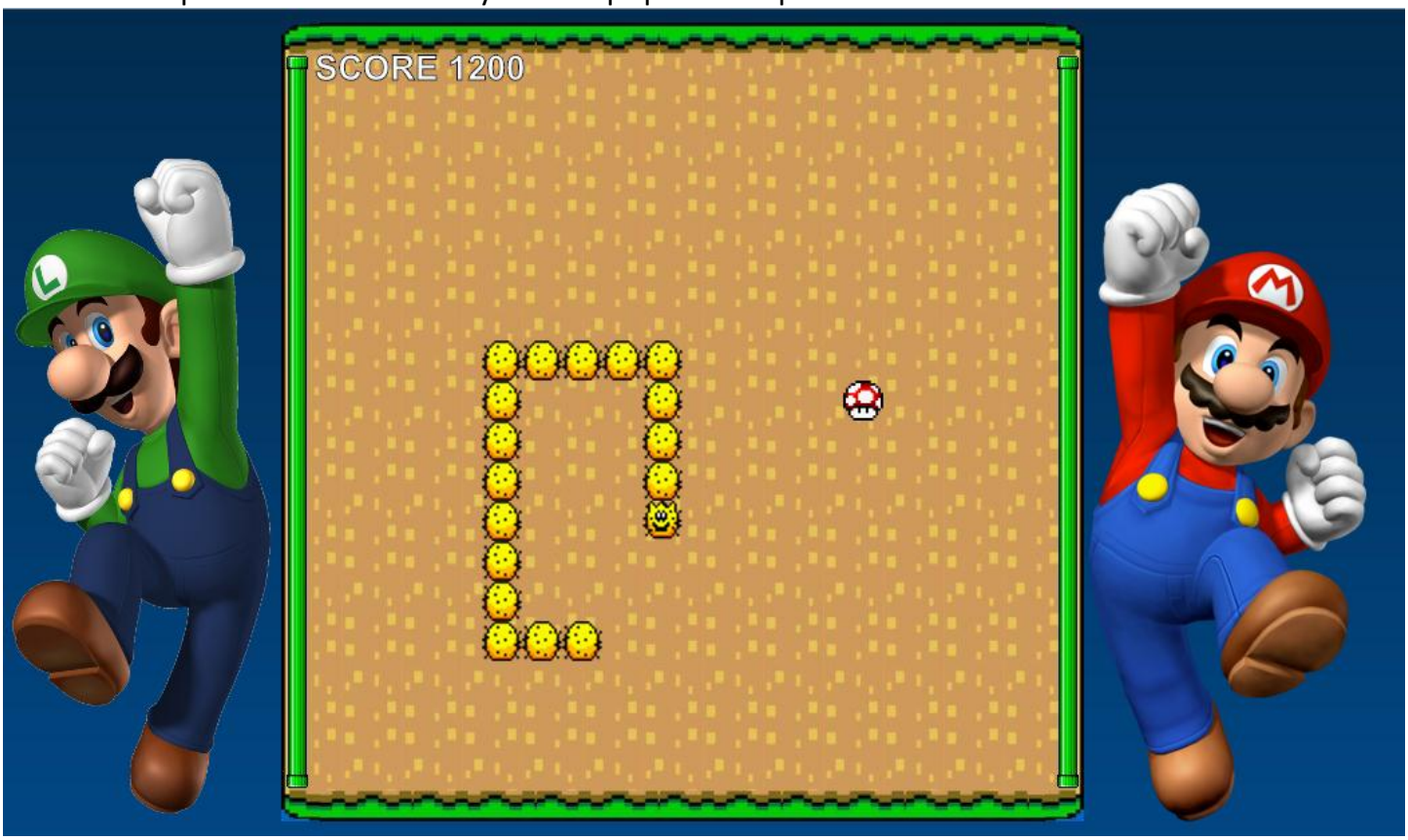
# Práctica Canvas – Juego Snake

**Objetivo**
Aplicar los conocimientos adquiridos sobre el componente Canvas de HTML para crear un juego de serpiente.

**Instrucciones**
Lee toda la práctica antes de empezar.
En esta práctica crearás un juego de serpiente usando el componente Canvas de HTML.
La serpiente se desplazará por el tablero evitando chocar con las paredes y consigo misma. El objetivo es juntar la mayor puntuación (score) comiendo hongos. Para controlar a la serpiente, usarás las teclas de dirección del teclado.

Utilizarás lo que sabes de CANVAS y Javascript para esta práctica.




## *Parte 1 – Preparar el ambiente*

1. **Archivo HTML**
    a. Crea un nuevo archivo HTML llamado serpiente.html que contenga jQuery.
2. **Directorios**
    a. Crea un directorio js donde coloques jQuery.
    b. Crea un directorio images donde colocarás las imágenes incluidas con la práctica.
        i. Nota que hay dos subdirectorios: sprites y ui. Conserva esa estructura.

## *Parte 2 – Componentes del HTML*

3. **Imágenes precargadas**
   a. Dentro del cuerpo de la página crea un nuevo DIV y define el siguiente estilo para él.

```
<div style="display: none;">
```

   La propiedad *display: none* provocará que cualquier objeto que se encuentre dentro del div se cargue en memoria pero no sea visible.
   b. Hay 5 imágenes en el directorio de sprites. Estas imágenes será las que se usen para mostrar en el juego. Por cada una de ellas, crea un elemento *img* como el siguiente y asegúrate de indicarle un Id distinto a cada uno y referente a su nombre y crearlas dentro del DIV que no es visible, para que estén ocultas.

```
<img id="bg" src="images/sprites/bg.jpg">
<img id="cabeza" src="images/sprites/cabeza.png">
<img id="cuerpo" src="images/sprites/cuerpo.png">
<img id="comida" src="images/sprites/comida.png">
<img id="perdiste" src="images/sprites/perdiste.png">
```

4. **Nuestro lienzo**
   a. Crea un elemento CANVAS y asígnale un Id con el que puedas recuperarlo en *JavaScript*. El tamaño del *canvas* debe ser de 600x600.

```
<canvas id="miCanvas" width="600" height="600" >
```

En este punto, tu página ya contiene los elementos multimedia necesarios para ser dibujados.

## *Parte 3 – Parte de JavaScript - Constantes y variables globales.*

Asegúrate de que tu página tenga incluido *jQuery*. Las siguientes variables y constantes mantienen el estado del juego. Créalas dentro del bloque <*script*> globalmente (fuera de cualquier función).

| | |
|---|---|
| El tablero está dividido en bloques por donde se mueve la serpiente. Crea las siguientes constantes para definir la cantidad de filas y columnas que tiene nuestro tablero, así como el tamaño del bloque. | `var FILAS = 20;`<br>`var COLUMNAS = 20;`<br><br>`var TAM_CUADRO = 30;` |
| Crea las siguientes constantes para definir los códigos del teclado que corresponden con la dirección a la que se debe mover la serpiente. | `var ARRIBA = 38;`<br>`var ABAJO = 40;`<br>`var IZQUIERDA = 37;`<br>`var DERECHA = 39;` |
| Variables para guardar las imágenes de los *sprites* a usar en el juego. | `var bg;`<br>`var imagenCabeza;`<br>`var imagenCuerpo;`<br>`var imagenComida;`<br>`var imagenPerdiste;` |
| Variable para indicar la dirección a la que se está moviendo la serpiente en un momento dado. Inicia moviéndose hacia arriba. | `var direccion = ARRIBA;` |
| Variable para indicar que has perdido el juego. | `var perdiste = false;` |
| Variable para almacenar el puntaje. | `var score = 0;` |
| Variable para las posiciones de los nodos de la serpiente y la posición de la | `var posicionComida = null;`<br>`var serpiente = null;` |

| | |
|---|---|
| comida. | |
| Variables para almacenar el *canvas* y su contexto. | `var canvas;`<br>`var c;` |

## *Parte 4 – Funciones utilitarias*

Crea las siguientes funciones para usarlas durante el proceso del juego.

| | |
|---|---|
| Se encarga de crear el objeto inicial de la serpiente. Es un arreglo de estructuras que posee las variables *x* y *y*.<br>Primero crea la cabeza en la posición 10, 10 y luego agrega 5 nodos a la serpiente usando la función *alargaCola*. | `function inicializaSerpiente(){`<br>`  serpiente = [{x: 10, y: 10}];`<br>`  for(var i = 0; i < 5; i++){`<br>`  alargaCola();`<br>`  }`<br>`}` |
| Esta función agrega nuevos nodos al final de la serpiente usando el método *push*() para arreglos. La posición en la que la crea el nuevo nodo, es en donde se encuentra la cola actual (último nodo). | `function alargaCola(){`<br>`  var ultimoNodo = serpiente[serpiente.length - 1];`<br>`  serpiente.push({x: ultimoNodo.x, y: ultimoNodo.y});`<br>`}` |
| Esta función es la asignada a la ventana para que se ejecute cada vez que se presiona una tecla.<br>En la estructura evento viene una variable con el código de la tecla presionada. El código se guarda en la variable global *dirección* para ser usada más adelante. | `function teclaPresionada(evento){`<br>`  direccion = evento.keyCode;`<br>`  }` |
| Esta función dibuja una imagen en el *canvas* y traduce la posición x, y del tablero a pantalla, multiplicando por el tamaño de un cuadro. Así, si la imagen se encuentra en la posición de tablero 10, 5, esta se dibujará en la posición real 300, 150, al ser multiplicada por 30. | `function dibujaImagen(imagen, x, y){`<br>`  c.drawImage(imagen, x * TAM_CUADRO, y * TAM_CUADRO);`<br>`}` |
| Esta función crea una posición aleatoria para colocar comida en el tablero. Usa la biblioteca *Math* de *JavaScript* para calcular un número aleatorio entre 0 y la cantidad máxima de filas y columnas.<br>La posición calculada es almacenada en la variable *posicionComida*. | `function crearComida(){`<br>`  var nuevaX = Math.round(Math.random() * (COLUMNAS - 1));`<br>`  var nuevaY = Math.round(Math.random() * (FILAS - 1));`<br>`  console.log("Nueva comida en " + nuevaX + ", " + nuevaY);`<br><br>`  posicionComida = {x: nuevaX, y: nuevaY};`<br>`}` |

## *Parte 5 – Función del motor de acciones del juego IA.*

La siguiente función tiene la tarea de realizar el procesamiento lógico del juego, aplica reglas y cambia el estado de los objetos de acuerdo a lo que el usuario introduzca. Crea una función *updateIA()* que no reciba parámetros y coloca los siguientes bloques de código dentro de ella.

| | |
|---|---|
| Obtén el primer nodo de la serpiente (la cabeza) en una variable para referenciarlo fácilmente más adelante. | `var cabeza = serpiente[0];` |

| | |
|---|---|
| Recorre el arreglo de la serpiente de atrás para adelante (excluyendo la cabeza) y "hereda" la posición de un nodo al siguiente. Esto hace que cada nodo de la serpiente se mueva a la posición que tiene el nodo que le antecede. | `for(var i = serpiente.length - 1; i > 0; i--){`<br>`  serpiente[i].x = serpiente[i - 1].x;`<br>`  serpiente[i].y = serpiente[i - 1].y;`<br>`}` |
| Cambia la dirección a la que se mueve la serpiente usando la variable dirección, que fue actualizada al presionar el teclado.<br>Las teclas de flecha tienen códigos fijos que definiste en constantes previamente. | `switch (direccion){`<br>`  case DERECHA: cabeza.x++;`<br>`  break;`<br>`  case IZQUIERDA: cabeza.x--;`<br>`  break;`<br>`  case ARRIBA: cabeza.y--;`<br>`  break;`<br>`  case ABAJO: cabeza.y++;`<br>`  break;`<br>`}` |
| Compara la posición de la cabeza con la de la comida. Si son iguales en sus componentes x, y, significa que la serpiente ha comido. Se incrementa el score, se crea una nueva posición para la comida y se alarga la serpiente. | `if (cabeza.x == posicionComida.x && cabeza.y == posicionComida.y){`<br>`  score += 100;`<br>`  crearComida();`<br>`  alargaCola();`<br>`}` |
| Recorre el arreglo de la serpiente comparando la posición de la cabeza con sus nodos. Si alguna es igual, entonces la cabeza chocó con el cuerpo. En caso de que así sea, el juego termina indicándolo en la variable *perdiste*. | `for(var i = 1; i < serpiente.length; i++){`<br>`  var nodo = serpiente[i];`<br>`  if (cabeza.x == nodo.x && cabeza.y == nodo.y){`<br>`  perdiste = true;`<br>`  }`<br>`}` |
| Compara la posición de la cabeza con las orillas del tablero. Si está fuera de los límites, el juego termina indicándolo en la variable *perdiste*. | `if (serpiente[0].x < 0 || serpiente[0].x >= COLUMNAS ||`<br>`  serpiente[0].y < 0 || serpiente[0].y >= FILAS){`<br>`  perdiste = true;`<br>`}` |

## *Parte 6 – Función del Loop del juego*

Crea una función *update*() que no reciba ningún parámetro. Esta es la función principal del *loop* del juego, que es repetida en intervalos regulares. Coloca los siguientes bloques de código dentro de la función *update()*.

| | |
|---|---|
| Dibuja la imagen de fondo. | `c.drawImage(bg, 0, 0);` |
| Llama a la función *updateIA*() para actualizar la lógica del juego antes de pintar los objetos. | `updateIA();` |
| El primer nodo de la serpiente (el cero) es la cabeza. Dibuja la imagen de la cabeza en dicha posición x, y usando la función que traduce la posición de tablero a posición real (*dibujaImagen*). | `dibujaImagen(imagenCabeza, serpiente[0].x, serpiente[0].y);` |
| Recorre el arreglo de la serpiente (excluyendo la cabeza) y dibuja la imagen de los nodos en las respectivas posiciones. | `for(var i = 1; i < serpiente.length; i++){`<br>`  dibujaImagen(cuerpo, serpiente[i].x, serpiente[i].y);`<br>`}` |
| Dibuja la comida en la posición indicada por la variable *posicionComida*. Valida que siempre exista. | `if (posicionComida != null){`<br>`  dibujaImagen(comida, posicionComida.x, posicionComida.y);`<br>`}` |

| | |
|---|---|
| Dibuja el texto del puntaje en posición 25, 45. Primero lo rellena de color blanco y luego dibuja un borde negro usando la función *strokeText*. | `c.fillText("SCORE " + score, 25, 45);`<br>`c.strokeText("SCORE " + score, 25, 45);` |
| Verifica si la variable de perdiste está activada. En caso de que así sea, se dibuja la imagen de *Game Over* y se corta la ejecución del *loop*. | `if(perdiste){`<br>`  dibujaImagen(imagenPerdiste, 3, 7);`<br>`  return;`<br>`}` |
| Se programa la siguiente repetición del *loop* del juego 100 milisegundos después. | `setTimeout(update, 100);` |

## *Parte 7 – Arranque del juego*

Crea una función *inicializacion*() que no reciba parámetros y define que sea ejecutada cuando se carga al página Web usando la siguiente instrucción dentro del bloque <*script*>.

```
$(document).ready(incializacion);
```

Coloca los siguientes bloques de código dentro de la función *inicializacion*().

| | |
|---|---|
| Indica la acción que se ejecuta al presionar algo en el teclado asignando un *event listener* a la ventana. | `window.addEventListener("keydown", teclaPresionada, true);` |
| Obtén los elementos del *canvas HTML* y su contexto. | `canvas = document.getElementById("miCanvas");`<br>`c = canvas.getContext("2d");` |
| Obtén los elementos *IMG* de las imágenes de los sprites precargadas. | `bg = document.getElementById("bg");`<br>`imagenCabeza = document.getElementById("cabeza");`<br>`imagenCuerpo = document.getElementById("cuerpo");`<br>`imagenComida = document.getElementById("comida");`<br>`imagenPerdiste = document.getElementById("perdiste");` |
| Define el estilo de fuente a usar para dibujar el score. | `c.fillStyle = "#FFFFFF";`<br>`c.strokeStyle = "#000000";`<br>`c.font = "bold 26px 'Arial'";` |
| Crea el arreglo de la serpiente. | `inicializaSerpiente();` |
| Crea la primera instancia de la comida. | `crearComida();` |
| Haz el primer llamado al *loop* del juego. | `update();` |

Prueba tu juego en el navegador. Cambia la dirección de la serpiente usando las flechas del teclado y junta muchos puntos. Recuerda mantener la consola del navegador abierta para detectar cualquier error.

## *Parte 7 – Formato bonito*

Dentro de las imágenes incluidas en la práctica puedes encontrar un fondo para la página. Aplícalo usando CSS.
También encontrarás imágenes de Luigi y Mario. Colócalas como *IMGs* a los lados del canvas para terminar de dar una presentación agradable a tu juego.

Felicidades, haz terminado un videojuego usando tus súper-poderes de programador.