ACTIVIDAD UT4

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tetris Personalizado</title>
  <style>
    canvas {
      display: block;
      margin: 0 auto;
      background: #000;
    }
  </style>
</head>
<body>
  <canvas id="tetris" width="300" height="600"></canvas>
  <script src="tetris.js"></script>
</body>
</html>
```

Para su realización usaremos la API para creación de gráficos canva

Algunas variables que utilizaremos serán las siguientes

```javascript
const canvas = document.getElementById("tetris");
const lienzo = canvas.getContext("2d");

const filas = 20;
const columnas = 10;
const tamanoCelda = 30; // Tamaño de las celdas en píxeles
```

Además, tendremos una variable que representará todas las posiciones posibles en nuestro tablero. Donde inicialmente todas ellas serán 0, para ello utilizaremos un array bidimensional denominado **tablero[filas][columnas]**

Para definir las piezas que se usarán en nuestro tablero, utilizaremos un array que almacenará cada una de nuestras piezas en un objeto. Con la siguiente estructura:

const piezas = [pieza1, pieza2, …]

Cada objeto pieza contendrá las siguientes propiedades:

- nombre: String con el nombre de la pieza
- forma: array bidimensionales de 0 o 1 que indica las posiciones que se deben pintar
- probabilidad: entre 0 y 1 de que la pieza aparezca (siendo la suma de todas las probabilidades igual a 1)
- color: String con el nombre del color CSS que deseamos pintar nuestra pieza

El juego deberá contener un array con al menos 5 piezas entre ellas deberá estar esta.

```
{ nombre: "C", forma: [[1, 1, 1], [1, 0, 1]], probabilidad: 0.2, color: "red" },
```

Lista de funcionalidades mínimas a implementar:

- **dibujarTablero()** : encargada de pintar cada una de las celdas del tablero, si el valor de la celda es 1 pintará de gris sino será negro.

- **dibujarPieza(pieza, x, y)**: le pasamos el objeto pieza y la posición donde tiene que dibujarla en el eje x y en el eje y de nuestro lienzo.

- **generarPieza()** : retorna la pieza que se muestra en base a la probabilidad que tiene cada una.

- **chequearColisiones(pieza, x, y)**: detecta si se produce una colisión es decir salirse de nuestro tablero o hay un 1 en el tablero por lo que ese hueco está ocupado. Retorna verdadero o falso.

- **posicionaPieza(pieza, x, y)**: coloca la pieza en el tablero

- **eliminarLinea():** elimina de nuestro tablero una fila si está es todo 1 y añade una nueva fila de ceros

- **actualizar():** Es la función que controla la lógica del juego, mediante la utilización de las funciones chequearcolisiones, posicionaPieza, eliminanarLinea y generarPieza nueva, por último comprueba si la nueva pieza generada tiene colisiones, si es así el juego debe finalizar mostrando una alerta por pantalla que dice “FIN DE LA PARTIDA, este juego ha sido desarrollado por [Tu nombre y apellidos]”

- **jugar()** es una función que primero actualiza el estado del juego, y después borra el tablero para volver a dibujarlo todo (tablero y pieza). Este proceso debe repetirse cada medio segundo.

Para el control del juego se usarán las letras

A - moverse a la izquierda
D - moverse a la derecha
S - bajar la ficha más rápidamente

Implementaciones extra que se pueden realizar.

- Rotar la pieza con la tecla W
- Añadir sistema de puntuaciones
- Mostrar la próxima ficha que va a salir por pantalla.
- Incrementar el nivel de dificultad del juego aumentando progresivamente la velocidad.