CARS | MASTER DRIVER

UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Programacion Web

Profesor:
M.T.C.A. Erik Germán Ramos Pérez

Alumna:
Martínez Lorenzo Frida Ximena
2021020174

07 DE NOVIEMBRE DE 2023

https://github.com/FridaMartinezLorenzo/Juego_Carritos

INTRODUCCIÓN

¡Bienvenidos al detrás de cámaras de "Cars | Master Driver"! Este manual técnico es una puerta de entrada para aquellos apasionados por los secretos y el funcionamiento interno del juego.

"Master Driver" es una experiencia única que combina entretenimiento con desafíos de reflejos al volante. Todo el juego se ha construido utilizando JavaScript, HTML y CSS, y en este manual, desglosamos meticulosamente cada función y elemento de su diseño para que puedas comprenderlos fácilmente.

Sin más preámbulos, te invitamos a sumergirte en este manual para descubrir los misterios detrás de "Cars | Master Driver" y adentrarte en su emocionante universo del tech.

00 DEFINICIÓN DE NIVELES

Se almacenó en un arreglo de matrices, donde cada matriz almacena información sobre el nivel, cada elemento alberga información en forma de un json. Objeto es el número de objeto a mostrar, 1-3 es un carrito “enemigo” y el 4 es una moneda que da puntos extra y cambia el color del carrito por un determinado intervalo. Appear, refiere al número de iteración en el que van a aparecer en pantalla. X y Y indican en que posición han de aparecer. La bandera de rebaso es un indicador para cuando el carrito del jugador ha rebasado al objeto y se le debe añadir a su score, esta bandera es crucial para detectar el fin del nivel.

Se cuentan con 10 matrices, en el arreglo ocupan las posiciones del 0-9.

```
let L = [
    [   //primer nivel (0)
        [{objetos:1, appear:100,  x:canvas.width, y:y_pista+10   , bandera_rebaso:0},
         {objetos:3, appear:400,  x:canvas.width, y:y_pista+110  , bandera_rebaso:0},
         {objetos:2, appear:700,  x:canvas.width, y:y_pista+10   , bandera_rebaso:0},
         {objetos:3, appear:1000, x:canvas.width, y:y_pista+110  , bandera_rebaso:0}],
    ],
```

01 INICIO

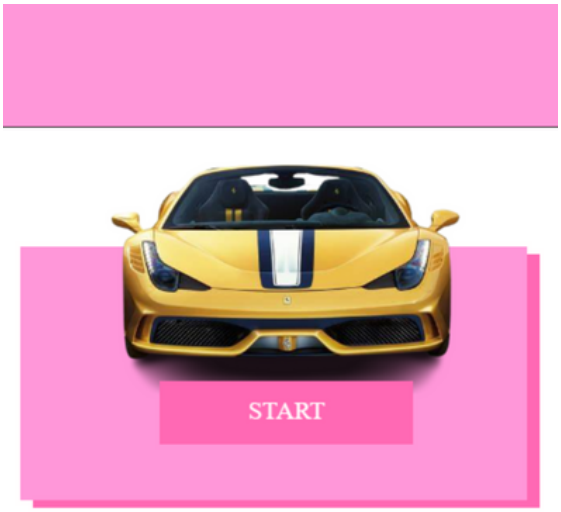
Antes que nada, se hace un pre cargado de las imágenes que se van a ocupar a lo largo del juego, a fin de que cuando hasta que se terminen de cargar no se carge la página.

```
//Precargado de imagenes
img.onload;
img_optimizado.onload;
img_respaldo.onload;
img_car_enemigo.onload;
img_car_enemigo2.onload;
img_car_enemigo3.onload;
img_meta.onload;
img_controles.onload;
img_start.onload = function(){
    cuadro_inicial();
}
```

02 CUADRO INICIAL

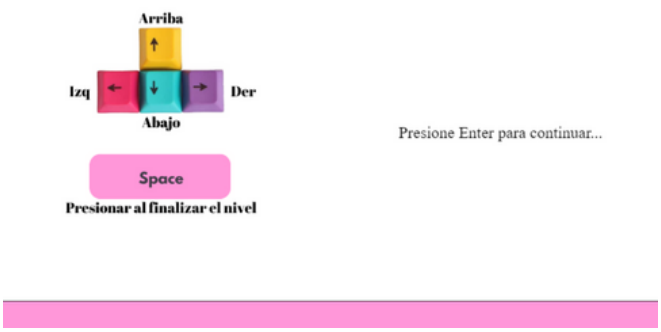
Cuadro inicial es el encargado de mostrarnos el cochecito que va a ser nuestro jugador de frente, debajo del mismo tiene dibujado un rectángulo que hará la funcionalidad del botón con ayuda de la detección de un clic en las coordenadas de ese rectángulo.

```
function cuadro_inicial(){
    ctx.fillStyle = "#FFFFFF";
    ctx.fillRect(0,0,canvas.width,canvas.height);

    ctx.fillStyle = "#FF69B4";
    ctx.fillRect(canvas.width/2-200,canvas.height/2-100,400,200);

    ctx.fillStyle = "#FF97D9";
    ctx.fillRect(canvas.width/2-210,canvas.height/2-106,400,200);

    ctx.drawImage(img_start,(canvas.width / 2)-150,10, 300, 200);

    //Dibujamos el boton
    ctx.fillStyle = "#FF69B4";
    ctx.fillRect(inicio_x_boton,inicio_y_boton,200,50);

    // Configura el estilo del texto
    ctx.font = "20px Times New Roman";
    ctx.fillStyle = "white";

    // Centra el texto en el botón
    var buttonText = "START";
    var xText = (canvas.width / 2) - 30;
    var yText = (canvas.height / 2) + 30;

    // Dibuja el texto en el botón
    ctx.fillText(buttonText, xText, yText);
}
```



CUADRO DE INSTRUCCIONES 03

Se realizó una función que dibuja la imagen en la cual se explica la forma en la que se opera el juego, en el canvas se nuestra un mensaje de presionar enter para dejar esa pantalla y empezar el juego




```
function cuadro_instrucciones(){
    ctx.fillStyle = "#FFFFFF";
    ctx.fillRect(0,0,canvas.width,canvas.height);

    ctx.drawImage(img_controles,200,20, 300,300);
}

canvas.addEventListener("click", detectarClick);
```

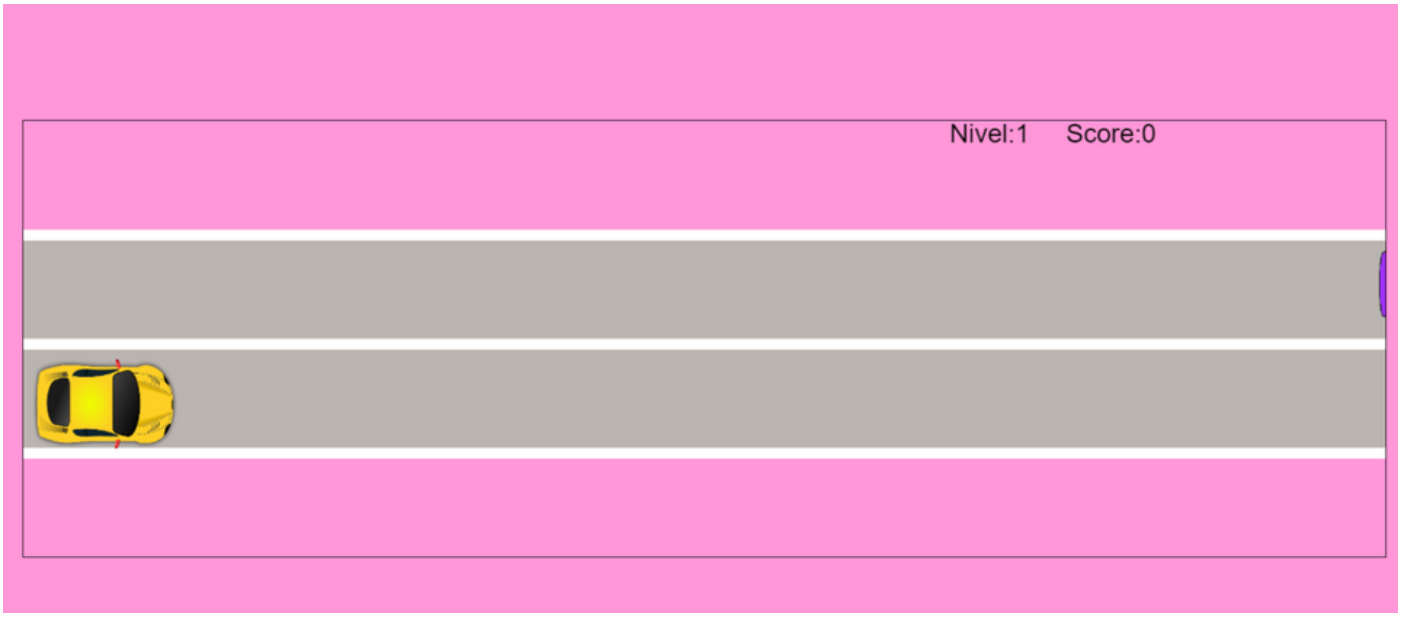
04 DETECTAR EL CLIC

El código proporcionado define una función llamada `detectarClick` que maneja los eventos de clic del ratón en un lienzo (canvas). La función calcula las coordenadas del clic en relación con el lienzo y verifica si el clic ocurrió dentro de un botón en el lienzo. Si el clic está dentro del botón y el nivel del juego es 0, muestra instrucciones y escucha eventos de teclado, en esta parte se está mandando a llamar al cuadro de instrucciones, ya que solo en el primer nivel se muestran. Si el nivel no es 0, inicia un intervalo que llama a una función llamada `draw`, presumiblemente para comenzar o continuar el juego.

El intervalo_carrito empieza en 30 y va decreciendo de 1 en 1

El contador se tiene que inicializar puesto que lleva la cuenta del número de veces que se manda a llamar a draw, con ayuda de este sabemos en qué momento deben de aparecer los carritos con los que el jugador va a competir.

```
function detectarClick(event){
    //console.log("click");
    var rect = canvas.getBoundingClientRect();
    var x_click = event.clientX - rect.left;
    var y_click = event.clientY - rect.top;

    // Verifica si el clic está dentro del botón, INICIA EL JUEGOOOOOO
    if (x_click >= inicio_x_boton &&  x_click <= inicio_x_boton+ancho_boton
        && y_click >= inicio_y_boton  && y_click <= inicio_y_boton+alto_boton ) {
            contador = 0;
            if (nivel == 0){
                cuadro_instrucciones();
                ctx.fillStyle = "black";
                ctx.fillText("Presione Enter para continuar...", 650, canvas.height/2);
            }else{
                IntervaloJuego = setInterval(main,intervalo_carrito);
            }
    }
}
```



05 COMIENZA EL JUEGO

La función `main` cumple un papel esencial para el funcionamiento del juego, ya que se manda a llamar en el setInterval y va modificando las variables, da la ilusión del movimiento en el juego al redibujar constantemente. Realiza tareas como dibujar el escenario, el carrito del jugador y objetos, detectar eventos como cruzar la línea de meta y colisiones, mostrar información de nivel y puntuación en la pantalla y controlar su rebote en los bordes de la pista. Además, registra el tiempo en niveles específicos. En conjunto, esta función es crucial para la jugabilidad y la experiencia del juego al manejar su representación gráfica y lógica.

```
function main(){
    dibuja_escenario();
    dibujar_carrito();
    detectar_rebaso();
    detectar_colision_objeto();


    ctx.font="24px Arial"
    ctx.fillStyle="black"
    ctx.fillText("Nivel:"+(nivel+1)+"     Score:"+puntaje,canvas.width-400,20);

    //ctx.fillText("Nivel:"+(nivel+1),canvas.width-300,20);

    x -= dx; //Siempre hacia atrás para que se simule el movimiento de que la pista se mueve
    dibujar_carritos_enemigos();

    if (x < 0)
        x = 0;

    if (y < y_pista || y>y_pista+ancho_pista){ //Evaluamos que no se salga de Y
        dy = -dy;
    }
    contador += 1; //Contador de apariciones (Nos ayuda a saber en que numero de repeticion se

    if ( (nivel == 8 | nivel == 9) && contador == 1){
        tiempo1 = Date.now();
    }
}
```

06 DIBUJA ESCENARIO

La función `dibuja_escenario` lo que hace es pintar un fondo de color rosa pálido (`#FF97D9`) en todo el lienzo del juego. Luego, llama a la función `dibujar_pista()` para dibujar la pista del juego en ese fondo. En resumen, esta función establece el fondo y proporciona el lienzo sobre el cual se representarán los elementos del juego.

```
function dibuja_escenario(){
    //Rosa y morado #EE97E5
    ctx.fillStyle = "#FF97D9";
    ctx.fillRect(0,0,canvas.width,canvas.height);
    dibujar_pista();
}
```

DIBUJA PISTA 07

La función `dibujar_pista` dibuja la carretera. Dependiendo del nivel de juego y el contador tiene que ser = 1 para que solo lo haga una vez y sea al principio, ya que, se ajusta diná-micamente el ancho y posición de la carretera. Luego, dibuja la carretera como un rectángulo de color gris claro en el lienzo y agrega divisiones en la carretera. 5 y 8, y en el nivel 8, también se reinicia la posición vertical de la carretera.

```
function dibujar_pista(){
    ctx.fillStyle = "#BBB5B1";

    if (nivel == 5 && contador == 1)
        ancho_pista +=100

    if (nivel == 8 && contador == 1){
        ancho_pista +=200;
        console.log(ancho_pista);
        y_pista = 0;
    }

    ctx.fillRect(0,y_pista, canvas.width, ancho_pista );
    dibujar_divisiones();
}
```

08 DIBUJA PISTA

La función `dibujar_divisiones` crea líneas blancas que separan los carriles en la carretera. Estas líneas son franjas blancas verticales de 10 píxeles de altura y se dibujan a intervalos de 100 píxeles en la carretera. Esto da la apariencia de divisiones en la carretera para indicar los carriles.

```
function dibujar_divisiones(){
    ctx.fillStyle = "#FFFFFF";

    for (i=0; i< Math.floor(ancho_pista/50); i++){
        y_divisiones = y_pista + (i*100);
        ctx.fillRect(0,y_divisiones, canvas.width, 10 );
    }
}       You, last week • Se tiene que checar la parte de de
```

09 DIBUJA CARRITO

La función `dibujar_carrito` se encarga de dibujar un carrito del jugador en el lienzo o canvas. Toma una imagen (`img`) y la coloca en la posición `(x, y)` especificada en el canvas. La imagen del carrito se dibuja con un ancho y una altura definidos por `car_ancho` y `car_altura`, respectivamente. Esto crea una representación visual del carrito en la ubicación deseada en el juego.

```
function dibujar_carrito(){
    ctx.drawImage(img, x, y, car_ancho, car_altura);
}
```

10 DETECTAR REBASO

La función `detectar_rebaso` desempeña un papel importante en el juego al rastrear el progreso del jugador y determinar cuándo se ha superado con éxito una serie de objetos en la pista. Utiliza una matriz `L` que almacena información sobre los objetos en la pista en diferentes niveles del juego. La función recorre esta matriz y evalúa cada objeto, excluyendo las monedas (con valor 4) y ciertos objetos especiales (con valor -2, son las monedas que ya tomó el usuario pero se puso un número negativo para que ya no la dibujara). Si el jugador se mueve más allá de un objeto (representado por `x + car_ancho` en la posición del objeto) y este objeto aún no ha sido rebasado (`c.bandera_rebaso` es 0), se agrega puntaje al marcador del jugador y se marca el objeto como rebasado.

La función multiplica una variable `total` por `c.bandera_rebaso` para rastrear si todos los objetos han sido rebasados. Cuando `total` es igual a 1, significa que el jugador ha superado todos los objetos en la pista. En este punto, se llama a la función `dibujar_meta` para mostrar que se ha alcanzado la meta y se está listo para avanzar al siguiente nivel. En resumen, esta función gestiona el avance del juego, el puntaje del jugador y la transición entre niveles al rastrear el progreso y verificar si se han rebasado todos los obstáculos en la pista.

```
function detectar_rebaso(){
    var total = 1;
    for (j = 0; j < L[nivel].length; j++ ){
        for (i = 0; i < L[nivel][j].length; i++)
        {
            var c = L[nivel][j][i];
            if (c.objetos != 4 && c.objetos != -2) //Permite que no se cuenten las monedas
                total *=c.bandera_rebaso;
                if (x > c.x +car_ancho && c.bandera_rebaso == 0){ //Se encuentra en una pos
                    puntaje  += c.objetos * 10;
                    c.bandera_rebaso = 1;
                }
        }
    }

    if (total == 1){  //Se hace el cambio de nivel
         dibujar_meta();
     }

}
```

11 DETECTAR COLISIÓN OBJETO

La función `detectar_colision_objeto` se encarga de manejar las colisiones entre el carrito controlado por el jugador y los objetos en la pista, en particular, las monedas y los carritos enemigos. La función recorre la matriz `L`, que almacena información sobre los objetos en la pista en diferentes niveles del juego, y evalúa las colisiones con los objetos que están actualmente en pantalla.

1. Para las monedas (objetos con valor 4), verifica si el carrito del jugador colisiona con ellas. Si hay una colisión, se triplica el puntaje, se marca la moneda como recogida (`c.objetos = -2`) y se realizan ajustes en las propiedades del carrito del jugador, como cambiar su imagen y dimensiones.

2. Para los carritos enemigos (objetos distintos de 4 y -2), verifica si hay colisiones con el carrito del jugador. Dependiendo de la dirección de la colisión (ya sea "arriba" o "abajo"), realiza los siguientes ajustes:
   - Si la colisión es "arriba", el carrito del jugador se ajusta para evitar la colisión y posiblemente rebota hacia arriba o abajo según su posición relativa al carrito enemigo.
   - Si la colisión es "abajo", se ajusta el carrito del jugador para evitar la colisión y posiblemente rebota hacia arriba o abajo. También, se reduce el puntaje en 5 puntos.

```
function detectar_colision_objeto(){
    var direccionColision = "";
    for (j = 0; j < L[nivel].length; j++ ){
        for (i = 0; i < L[nivel][j].length; i++)
        {
            var c = L[nivel][j][i];


            if (c.x + (pelotaRadio*2) > 0 && c.x < canvas.width && c.objetos == 4){ //Evaluamos que la moneda este en pantalla

                if( x < c.x+(pelotaRadio*2) && x+car_ancho > c.x && y < c.y+(pelotaRadio*2) && y+car_altura>c.y){
                    console.log("colisiono la moneda");
                    puntaje*=3; //Triplica puntaje
                    c.objetos = -2;
                    img = img_optimizado;
                    car_ancho = car_ancho_enemigo; //Se cambian las medidas
                    car_altura = car_altura_enemigo;
                    bandera_cambio_color = 1
                }
            }
```

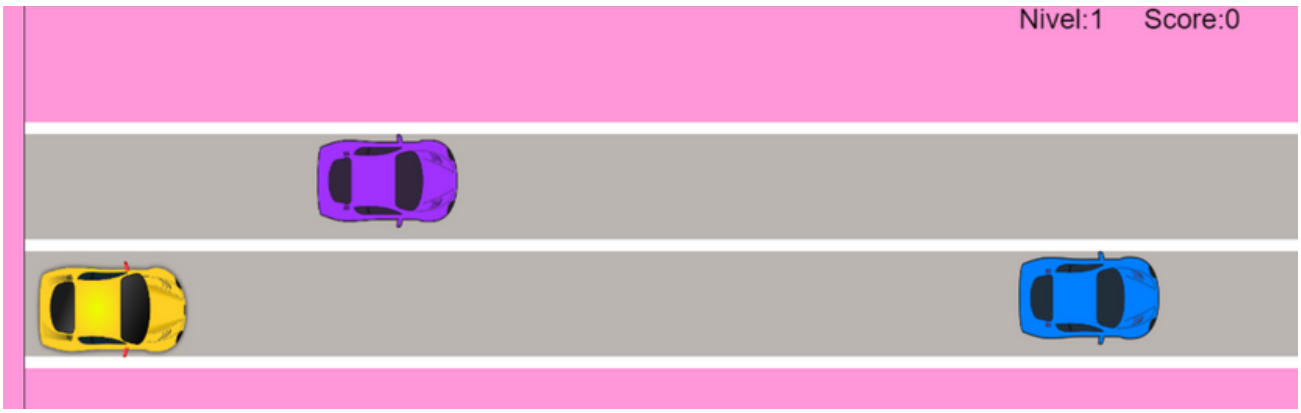
```
            if (c.x + car_ancho_enemigo > 0 && c.objetos != 4 && c.objetos != -2){ //Se encuentra en pantalla
                if(x < c.x + car_ancho_enemigo && x + car_ancho > c.x &&
                    y < c.y + car_altura_enemigo-10 && y + car_altura-10 > c.y){

                    // Determina la dirección de la colisión
                    if (y < c.y+car_altura_enemigo-10 && x < c.x+car_ancho_enemigo) {
                        direccionColision = "abajo";
                    }
                    if (y+car_altura-10 > c.y && x+car_ancho < c.x) {
                        direccionColision = "arriba";
                    }

                    // Realiza el ajuste de coordenadas para el rebote
                    if (direccionColision == "arriba") {
                        if (c.y >= y_pista + (ancho_pista/2) ) {
                            //Si se encuentra por debajo de la mitad de la pista
                            y = c.y - car_altura;
                        }
                        else {
                            y = c.y + car_altura;
                        }
                        x = c.x - car_ancho;
                    } else if (direccionColision == "abajo") {
                        if (c.y >= y_pista + (ancho_pista/2) ){
                            //Si se encuentra por debajo de la mitad de la pista
                            y = c.y - car_altura;
                        }
                        else{
                            y = c.y + car_altura;
                        }
                        x = c.x - car_ancho;
                        if(puntaje > 0){
                            puntaje -= 5;
                        }
                    }

                }
            }

        }
    }

}
```

12 DIBUJAR OBJETOS

La función `dibujar_objetos` es responsable de dibujar objetos en la pista de juego basándose en la matriz `L`, que almacena información sobre los objetos en diferentes niveles del juego. A través de bucles `for`, recorre la matriz y verifica si se debe dibujar un objeto en una posición determinada. Esto se determina comparando el contador de tiempo del juego con el valor de `c.appear` en la matriz. Si el contador es mayor o igual a `c.appear`, se llama a `dibujar_objeto` para representar el objeto en la ubicación adecuada.

En resumen, `dibujar_objetos` garantiza que los objetos aparezcan en el momento apropiado y en las posiciones correctas en la pista del juego, siguiendo la configuración del nivel actual y contribuyendo a la dinámica visual del juego.

```
function dibujar_objetos(){
    for (j = 0; j < L[nivel].length; j++ ){
        for (i = 0; i < L[nivel][j].length; i++)
        {
            c = L[nivel][j][i];
            if(c.objetos != 0){
                //Evluamos si es el momento en el que debe de aparecer el carrito
                if (contador >= c.appear){
                    dibujar_objeto(j,i);
                }
            }
        }
    }

}
```



13 DIBUJAR OBJETO

La función `dibujar_objeto` se encarga de dibujar objetos en la pista de juego, considerando la información del objeto en la matriz `L` para el nivel actual. En esta función, se realiza un seguimiento del cambio de color del carrito del jugador a través de la variable `bandera_cambio_color`. Cuando `bandera_cambio_color` es igual a 1, se incrementa `contador_carrito_optimizado` y se compara con `limite_contador_carrito_optimizado`. Si estos valores son iguales, se restaura la apariencia del carrito del jugador a su estado normal (amarillo).

Luego, la función verifica si el objeto es una moneda (`obj.objetos == 4`) o algún otro tipo de objeto. Si es una moneda, se llama a la función `dibujaMoneda` para representarla en la pista. Si no es una moneda, se utiliza `ctx.drawImage` para dibujar el objeto en la ubicación correspondiente, considerando las dimensiones del carrito enemigo. Finalmente, se actualiza la posición del objeto restando la velocidad horizontal `dx_carrito_enemigo`.

En resumen, `dibujar_objeto` administra el cambio de color del carrito del jugador, dibuja monedas y otros objetos en la pista de juego, y actualiza su posición en función de la velocidad horizontal.

```
function dibujar_objeto(j,i){
    obj = L[nivel][j][i];

    if (bandera_cambio_color == 1){//Contador de la duración el cambio de color
        contador_carrito_optimizado+=1; //  Sumamos el conta
    }

    //Cuando volver al carrito normal del juego (el amarilllo)
    if (contador_carrito_optimizado == limite_contador_carrito_optimizado){
        console.log("Detecto el limite");
        img = img_respaldo;
        car_ancho = 150;
        car_altura = 100;
        bandera_cambio_color = 0;
        contador_carrito_optimizado = 0;
    }

    if (obj.objetos == -2) //Se tomó el -2 porque el -1 nos ayuda en el proceso de verificacion de que s
        return;
    if(obj.objetos == 4) //La moneda aún no ha sido tomada por el jugador
        dibujaMoneda(obj);
    else
        ctx.drawImage(objetos[obj.objetos - 1], obj.x, obj.y, car_ancho_enemigo, car_altura_enemigo);

    obj.x -= dx_carrito_enemigo;// Actualiza la posición del objeto
}
```

14 DIBUJA MONEDA

La función `dibujaMoneda` se encarga de dibujar una moneda en el juego utilizando la imagen `img_moneda`. Esta imagen se recorta de acuerdo a los valores almacenados en el array `M` para obtener un efecto animado. La función ajusta la posición y el tamaño de la moneda en el lienzo, y el valor de `indice` se incrementa para cambiar el punto de inicio del recorte, creando así una animación visual de la moneda en movimiento.

```
function dibujaMoneda(c){
    //dibuja la estrella con efecto
    ctx.drawImage(img_moneda,M[indice][0],M[indice][1],M[indice][2],M[indice][3],c.x,c.y, M[indice][4]/2,M[indice][5]/2);
    indice = (indice+1) % 6;

}
```

15 DIBUJA META

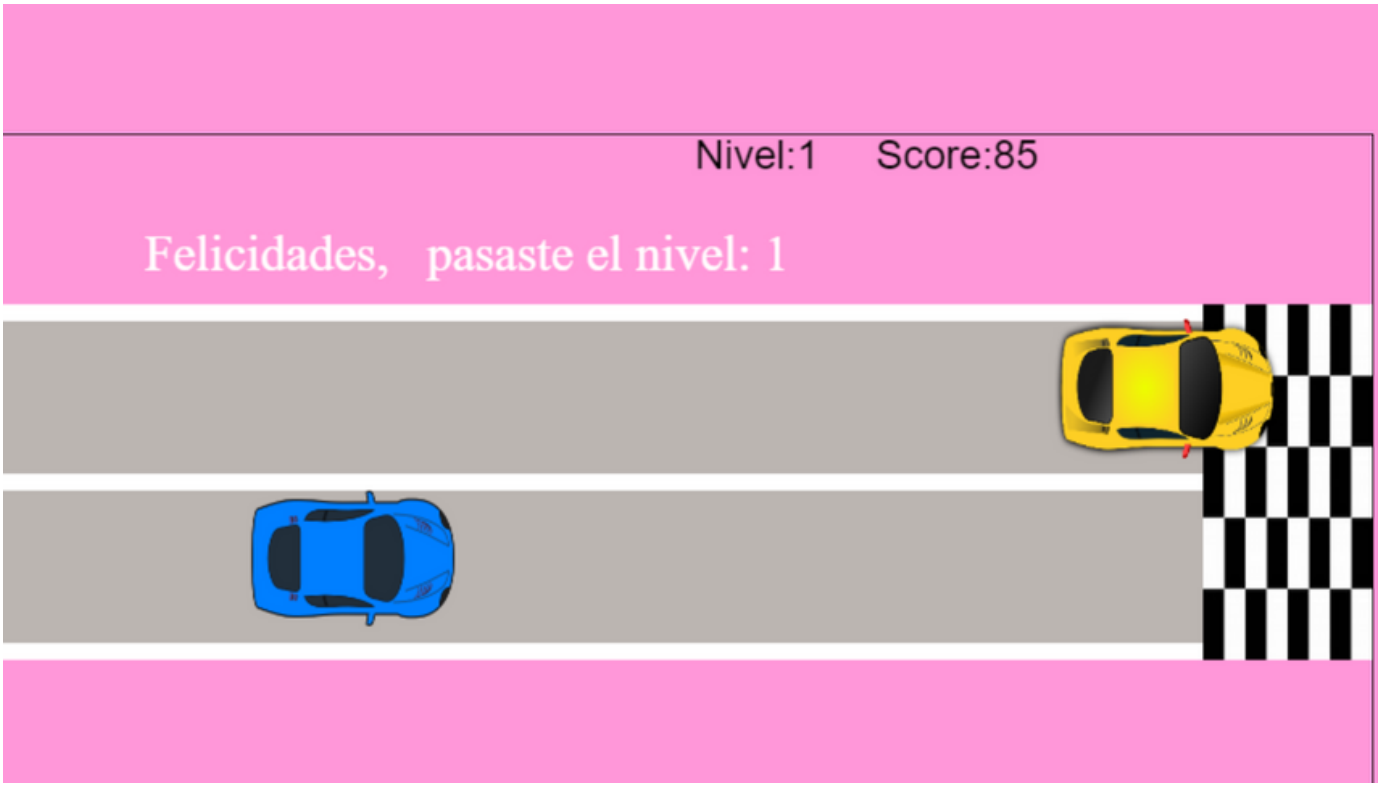
La función `dibujar_meta` tiene como propósito gestionar la aparición y el comportamiento cuando el carrito del jugador alcanza la meta al final de la pista. Dibuja una imagen (`img_meta`) que representa la meta al final de la pista. La posición de la meta es en la parte derecha de la pantalla (`canvas.width-100`) y a la altura de la pista actual (`y_pista`) con un ancho de 100 y una altura igual al ancho de la pista (`ancho_pista`).

Comprueba si la posición horizontal del carrito del jugador (`x`) supera un cierto umbral (`canvas.width-200`). Si es así, significa que el carrito ha alcanzado la meta. Dibuja el carrito del jugador en su posición actual. Esto es necesario porque el carrito debe de quedar dibujado encima de la meta, de lo contrario quedará con el cofre cubierto por la meta.

En esta función, se detiene el intervalo de juego (`IntervaloJuego`) para pausar el movimiento del carrito y evitar que siga avanzando después de llegar a la meta. Si el nivel actual es el 8 o el 9, la función mide el tiempo que ha transcurrido desde que comenzó el nivel (`tiempo1`) hasta que se alcanzó la meta (`tiempo2`). Esto calcula el tiempo en segundos y lo almacena en la variable `tiempo_resultante`. Comprueba si el tiempo resultante supera un tiempo esperado (`tiempo_esperado`) para el nivel. Si el tiempo supera el tiempo esperado, establece la bandera `bandera_fallo_nivel` en 1, lo que indica que el jugador no pasó el nivel. Se muestra un mensaje en el lienzo indicando si el jugador ha pasado o no el nivel. Si ha superado el nivel, felicita al jugador y muestra el número del nivel. Si no lo ha superado y ha excedido el tiempo esperado, muestra un mensaje de "Perdiste el nivel" junto con el número del nivel.

Agrega un escuchador de eventos para detectar la pulsación de teclas (espera se presiona la barra de espacio), lo que permitirá al jugador continuar a través de los niveles o realizar otras acciones después de llegar a la meta.

```
function dibujar_meta(){
    ctx.drawImage(img_meta,canvas.width-100, y_pista,100,ancho_pista);
    if (x > canvas.width-200){
                dibujar_carrito();

                clearInterval(IntervaloJuego);

                if (nivel == 8 | nivel == 9){
                    tiempo2 = Date.now();
                    tiempo_resultante = tiempo2 - tiempo1;
                    tiempo_resultante = Math.floor(tiempo_resultante/1000);
                    console.log(tiempo_resultante);
                }

                ctx.font="30px Times new roman"


                if ((nivel == 8 | nivel == 9) && (tiempo_resultante > tiempo_esperado)){
                    bandera_fallo_nivel = 1;
                    ctx.fillText("Perdiste el nivel "+(nivel+1),canvas.width/2-100,80);
                }else
                    ctx.fillText("Felicidades, \n pasaste el nivel: "+(nivel+1),canvas.width/2-100,80);

            }
```



16 DETECTAR TECLA

La función `detectarTecla` maneja eventos de teclado y controla el movimiento del carrito del jugador en el juego. Cuando se pulsan ciertas teclas, se ejecutan acciones específicas:

1. Si se presiona la tecla "Enter" (código 13) y el nivel actual es 0, se inicia el juego al establecer un intervalo de tiempo que llama continuamente a la función `main`, permitiendo que el carrito se mueva automáticamente.

2. Si se presiona la tecla de flecha derecha (código 39), el carrito se mueve hacia la derecha sumando una cantidad de píxeles definida en la variable `pasos`.

3. Si se presiona la tecla de flecha izquierda (código 37), el carrito se mueve hacia la izquierda restando una cantidad de píxeles igual a `pasos`.

4. Si se presiona la tecla de flecha abajo (código 40), el carrito se mueve hacia abajo sumando píxeles a la coordenada vertical `y`.

5. Si se presiona la tecla de flecha arriba (código 38), el carrito se mueve hacia arriba restando píxeles a la coordenada vertical `y`.

6. Si se presiona la tecla espaciadora (código 32), el comportamiento depende de si la bandera `bandera_fallo_nivel` es igual a 1 (indicando que el jugador perdió el nivel). En ese caso, se reduce el nivel, se ajustan algunos valores y se resta una cantidad de puntos. Si la bandera no es igual a 1, se aumenta el nivel, se ajustan valores y se procede a la transición de nivel.

Además, la función impone restricciones en el movimiento del carrito para evitar que este salga de los límites de la pantalla y se asegura de que se mantenga dentro de la pista de juego.

```
function detectarTecla(e){
    if (e.keyCode == 13 && nivel == 0)
        IntervaloJuego = setInterval(main,intervalo_carrito);

    if (e.keyCode == 39){
        //console.log("Avanzando a derecha")
        x +=  pasos;
    }

    if (e.keyCode == 37){
        //console.log("Avanzando a izquierda")
        x -= pasos;
    }

    if (e.keyCode == 40){
        //console.log("Avanzando hacia abajo")
        y += pasos;
    }

    if (e.keyCode == 38){
        //console.log("Avanzando hacia arriba")
        y -= pasos;
    }

    if (e.keyCode == 32){
       if (bandera_fallo_nivel == 1){
           nivel-=1;
           intervalo_carrito += 2;
           limite_contador_carrito_optimizado -=200;
           puntaje -= 1000; //Descontamos mil puntos
           bandera_fallo_nivel = 0;
       }
       else{
           nivel+=1;
           console.log("Nivel: "+nivel);
           if (nivel == 10) {
               fin_de_juego();
               return;
           }
           intervalo_carrito -= 1;
           limite_contador_carrito_optimizado +=200;
        }
        cuadro cambio nivel();
```

```
    }

    if (y <= y_pista)
        y = y_pista;

    if (nivel  >=6 ){
        if (y >= ancho_pista-100)
            y = ancho_pista-100;
    }else{
        if (y >= ancho_pista)
            y = ancho_pista;
    }


    if (x <= 0)
        x = 0;

    if (x >= canvas.width-margen_desaparicion_carrito)
        x = canvas.width-margen_desaparicion_carrito;
}
```

17 FIN DE JUEGO

La función `fin_de_juego` muestra una pantalla de fin de juego en el juego. Restablece las variables de nivel, puntaje e intervalo de movimiento del carrito a sus valores iniciales. Luego, muestra un mensaje de felicitaciones en la parte superior de la pantalla y un botón "JUGAR DE NUEVO" en el centro de la pantalla, que permite al jugador reiniciar el juego.

```
function fin_de_juego(){
    ctx.fillStyle = "#FFFFFF";
    ctx.fillRect(0,0,canvas.width,canvas.height);

    nivel = 0;
    puntaje = 0;
    intervalo_carrito = 25;

    ctx.font="50px Times new roman"
    ctx.fillStyle="#BF398C"
    ctx.fillText("¡¡FELICIDADES, GANASTE !!",canvas.width/2-300,150);

     //Dibujamos el boton
     ctx.fillStyle = "#FF69B4";
     ctx.fillRect(inicio_x_boton,inicio_y_boton,250,50);

     // Configura el estilo del texto
     ctx.font = "20px Times New Roman";
     ctx.fillStyle = "white";

     // Centra el texto en el botón
     var buttonText = "JUGAR DE NUEVO";
     var xText = (canvas.width / 2) - 60;
     var yText = (canvas.height / 2 + 30);

     // Dibuja el texto en el botón
     ctx.fillText(buttonText, xText, yText);

}
```

18 CUADRO CAMBIO DE NIVEL

La función `cuadro_cambio_nivel` se utiliza para dibujar un cuadro de cambio de nivel en el juego. Este cuadro consta de un fondo rectangular en tonos de rosa, el número del nivel actual, un efecto decorativo con guiones y un botón "START" centrado en la pantalla. El código configura el tamaño, posición y colores de estos elementos para crear un efecto visual atractivo y proporciona a los jugadores la opción de comenzar el nuevo nivel al presionar el botón "START".

```
function cuadro_cambio_nivel(){
    x=0;
    ctx.fillStyle = "#FFFFFF";
    ctx.fillRect(0,0,canvas.width,canvas.height);

    ctx.fillStyle = "#FF69B4";
    ctx.fillRect(canvas.width/2-200,canvas.height/2-100,400,200);

    ctx.fillStyle = "#FF97D9";
    ctx.fillRect(canvas.width/2-210,canvas.height/2-106,400,200);

    ctx.font="50px Times new roman"
    ctx.fillStyle="#BF398C"
    ctx.fillText("NIVEL: "+(nivel+1),canvas.width/2-100,150);
    ctx.fillText("_______",canvas.width/2-100,150);

    //Dibujamos el boton
    ctx.fillStyle = "#FF69B4";
    ctx.fillRect(inicio_x_boton,inicio_y_boton,200,50);
      You, 4 days ago • Se añadio el panel entre escenarios
    // Configura el estilo del texto
    ctx.font = "20px Times New Roman";
    ctx.fillStyle = "white";

    // Centra el texto en el botón
    var buttonText = "START";
    var xText = (canvas.width / 2) - 30;
    var yText = (canvas.height / 2 + 30);

    // Dibuja el texto en el botón
    ctx.fillText(buttonText, xText, yText);

}
```

EL HTML

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" href="style.css">
        <title>Cars | Master Driver </title>
    </head>
    <body>
        <div class = "zona_juego">
            <canvas id="area_juego" width="1250" height="400">
                Tu navegador no sporta canvas
            </canvas>
        </div>

        <script type="text/javascript" src="escenario.js"></script>
    </body>
</html>      You, 2 weeks ago • Carrito con animacion
```

EL CSS

```css
canvas{
    background-color: "white";
    border: 1px solid black;
    margin-top: 100px;
    margin-left: 10px;
}

body{
    background-color: #FF97D9;
}
```


GRACIAS