

Juego 2D desarrollado en JavaScript puro: Introducción a la Informática

JHON ESTEBAN RESTREPO ESCOBAR
SEPTIEMBRE DE 2020



1 CONTENIDO

1	CONTENIDO.....	1
2	PRESENTACIÓN.....	2
3	FASE 1: Dibujar y mover una bola	3
4	FASE 2: Rebotando en las paredes.....	6
5	FASE 3: Control de la pala y el teclado	9
6	FASE 4: Fin del juego	18
7	FASE 5: Muro de ladrillos	22
8	FASE 6: Detección de colisiones	28
9	FASE 7: Contar puntos y ganar	33
10	FASE 8: Controlando el ratón	39
11	FASE 9: Finalizando el juego.....	45
12	CONCLUSIONES	51
13	BIBLIOGRAFÍA.....	52

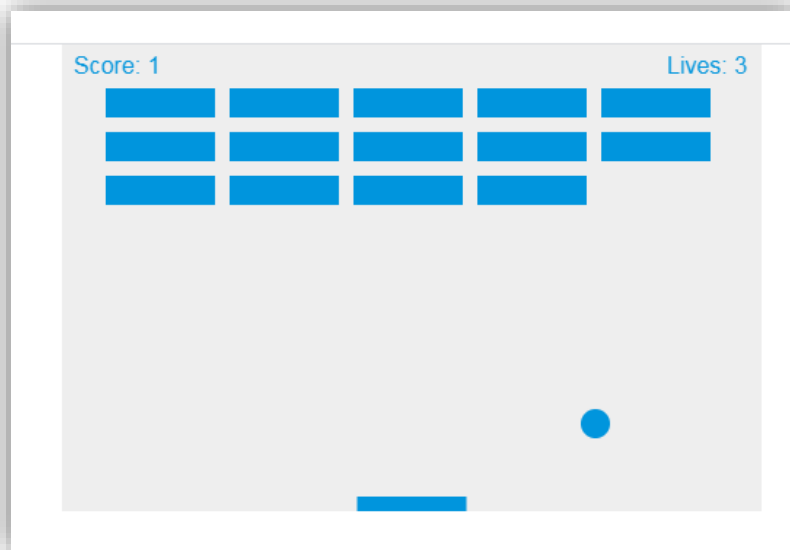
2 PRESENTACIÓN

La presente monografía describe el desarrollo metódico de un juego 2D elaborado utilizando HTML5, CSS, CANVAS y JavaScript.

El juego elaborado se crea con JavaScript puro, utilizando un enfoque metódico en el cual se avanza de versión en versión, de modo que cada nuevo programa abarca un aspecto adicional del juego.

Cada una de las fases se cubre en un apartado diferente. Se plantea el alcance de cada una de ellas, se explican las instrucciones o conceptos que son necesarios para entender el significado del trabajo realizado, se agrega el código, y finalmente se presentan fotos de la ejecución del programa

Una vez cubiertas todas las fases, se dispondrá de un clásico juego 2D que servirá como base e inspiración para desarrollar otros programas aplicados en la Web.



Gráfica 1. Juego 2D en JavaScript

El documento web que sirve como referencia para el desarrollo del juego está en el siguiente enlace:

https://developer.mozilla.org/es/docs/Games/Workflows/Famoso_juego_2D_usando_JavaScript_puro

AUTOR: Jhon Esteban Restrepo Escobar

3 FASE 1: DIBUJAR Y MOVER UNA BOLA

El primer paso consiste en elaborar una página HTML básica. Agregaremos a dicha página un elemento CANVAS, el cual nos servirá como base para el desarrollo del juego 2D.

El código JavaScript que operará sobre el CANVAS debe encerrarse entre las etiquetas `<script>...</script>`

La correcta visualización del CANVAS requiere de la adición de algunas características de estilo. Una vez hecho esto, se procede a establecer la codificación pertinente del JavaScript. Debe notarse la inclusión de algunas variables que definen la funcionalidad del juego en sus aspectos básicos: las coordenadas en las que se encuentra la bola y los valores de incremento para modificar su posición.

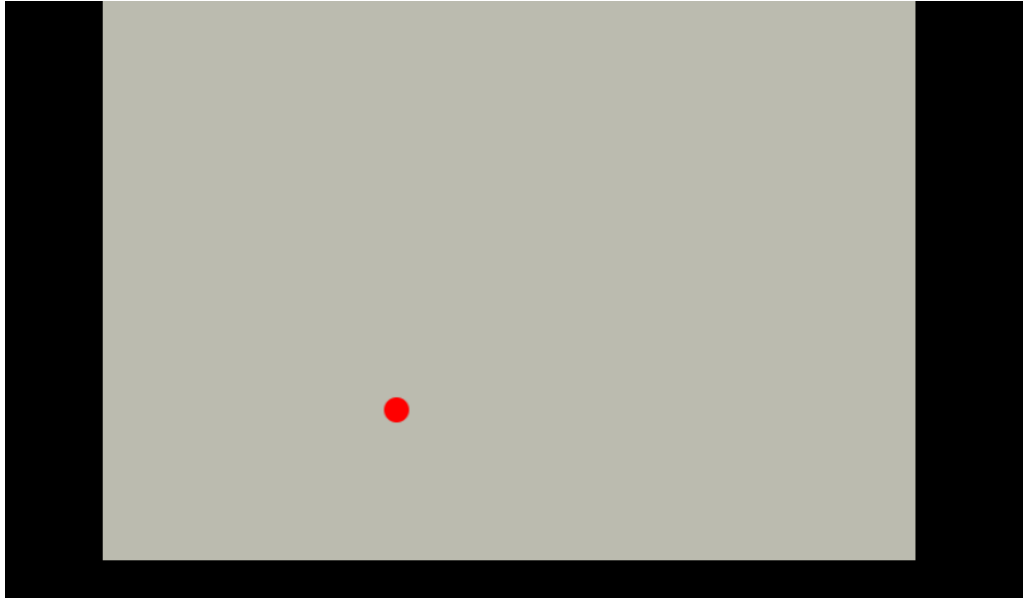
Se definen tres funciones importantes. La primera de ellas, `dibujarBola()`, se encarga de dibujar sobre la pantalla una bola con el color indicado en los estilos. La segunda función se denomina `dibujar()`, y es la encargada de limpiar el CANVAS, dibujar la bola y cambiar los valores de las coordenadas. Finalmente, la función `setInterval(dibujar, 10)`, llama a la función `dibujar` cada 10 milisegundos.

El código fuente del programa es el siguiente (para darle formato, se deben seguir las instrucciones disponibles en: <https://trabajonomada.com/insertar-codigo-word/> y seguidamente utilizar el enlace: http://qbnz.com/highlighter/php_highlighter.php)

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="utf-8" />
5.
6.     <title>Juego 2D: JS - 01</title>
7.
8.     <style>
9.         * {
10.             padding: 0;
11.             margin: 0;
12.         }
13.
14.         canvas {
15.             background: #BBBBAF;
16.             display: block;
17.             margin: 0 auto;
18.         }
19.     </style>
20.
21. </head>
22.
23. <body style="background-color: #000000;">
24.
25. <canvas id="miCanvas" width="650" height="450" ></canvas>
26.
27. <script>
28.     var canvas = document.getElementById("miCanvas");
```

```
29.     var ctx = canvas.getContext("2d");
30.     var x = canvas.width/2;
31.     var y = canvas.height-30;
32.     var dx = 30;
33.     var dy = -30;
34.
35.     function dibujarBola() {
36.
37.         ctx.beginPath();
38.
39.         ctx.arc(x, y, 10, 0, Math.PI*2);
40.
41.         ctx.fillStyle = "#FF0000";
42.
43.         ctx.fill();
44.
45.         ctx.closePath();
46.     }
47.
48.     function dibujar() {
49.
50.         ctx.clearRect(0, 0, canvas.width, canvas.height);
51.
52.         dibujarBola();
53.
54.         x = x - dx;
55.
56.         y = y + dy;
57.     }
58.
59.     setInterval(dibujar, 100);
60. </script>
61.
62. </body>
63. </html>
```

Al ejecutar este código se obtiene la siguiente interfaz visual:



Gráfica 2. La interfaz inicial del juego

En la gráfica 2 se aprecia el dibujo de la bola, y la secuencia de movimiento a partir de los incrementos en X y Y que fueron definidos.

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

4 FASE 2: REBOTANDO EN LAS PAREDES

El segundo paso consiste en elaborar los límites permitidos a los que la bola puede llegar y en los que rebotará y así mantenerse dentro del cuadro asignado.

En este paso se crean dos condiciones las cuales generan los límites permitidos a los que la bola puede llegar y va a rebotar:

La primera condición es `if(x + dx > canvas.width-ballRadius || x + dx < ballRadius) { dx = -dx;}`, esta condición crea el rango horizontal al que la pelota se puede desplazar.

La segunda condición es `if(y + dy > canvas.height-ballRadius || y + dy < ballRadius) {dy = -dy;}`, esta condición crea el rango vertical al que la pelota se puede desplazar.

```

1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="utf-8" />
5.     <title>Juego 2D - lección 02</title>
6.     <style>* { padding: 0; margin: 0; } canvas { background: #eee;
    display: block; margin: 0 auto; }</style>
7. </head>
8. <body>
9.
10.     <canvas id="miCanvas" width="480" height="320"></canvas>
11.
12.     <script>
13.         var canvas = document.getElementById("miCanvas");
14.         var ctx = canvas.getContext("2d");
15.         var ballRadius = 10;
16.         var x = canvas.width/2;
17.         var y = canvas.height-30;
18.         var dx = 2;
19.         var dy = -2;
20.
21.         function dibujarBola() {
22.             ctx.beginPath();
23.             ctx.arc(x, y, ballRadius, 0, Math.PI*2);
24.             ctx.fillStyle = "#0095DD";
25.             ctx.fill();
26.             ctx.closePath();
27.         }
28.
29.         function dibujar() {
30.             ctx.clearRect(0, 0, canvas.width, canvas.height);
31.             dibujarBola();
32.
33.             /* IMPORTANTE:
34.

```

```

35.          EL OPERADOR || es el operador lógico OR
36.          Este operador se utiliza para indicar la condición
de conjunción
37.          SI SE CUMPLE UNA CONDICIÓN, O SE CUMPLE OTRA
CONDICIÓN, ENTONCES
38.          SE CUMPLE LA CONDICIÓN
39.
40.          EL OPERADOR && es el operador lógico AND
41.          Este operador se utiliza para indicar la condición
de disyunción
42.          SI SE CUMPLE UNA CONDICIÓN, Y SE CUMPLE OTRA
CONDICIÓN (simultánea), ENTONCES
43.          SE CUMPLE LA CONDICIÓN
44.
45.          */
46.
47.          /* DESPUÉS DE DIBUJAR LA BOLA, SE DEBEN CAMBIAR LAS
COORDENADAS
48.          EN LA lección 01 NO SE TENÍA CONTROL SOBRE LOS
LÍMITES DE LA CAJA
49.          -----
50.          SI x + dx ES MAYOR AL ANCHO DEL CANVAS O MENOR AL
TAMAÑO DEL
51.          RADIO DE LA BOLA (caso en el cual se encuentra
hacia la izquierda)
52.          SE CAMBIA LA DIRECCIÓN DE AVANCE HORIZONTAL.
53.          ESTO SE LOGRA CAMBIANDO EL SIGNO DE LA VARIABLE dx
54.          ESTO HACE QUE SE CAMBIE EL SENTIDO DEL MOVIMIENTO
HORIZONTAL */
55.          if(x + dx > canvas.width-ballRadius || x + dx <
ballRadius) {
56.              dx = -dx;
57.          }
58.
59.          /* SI y + dy ES MAYOR A LA ALTURA DEL CANVAS O MENOR
AL TAMAÑO DEL
60.          RADIO DE LA BOLA, SE CAMBIA LA DIRECCIÓN DEL
AVANCE VERTICAL.
61.          ESTO SE LOGRA CAMBIANDO EL SIGNO DE LA VARIABLE dy
62.          ESTE CAMBIO EN dy HACE QUE SE MUEVA VERTICALMENTE
EN SENTIDO
63.          OPUESTO */
64.          if(y + dy > canvas.height-ballRadius || y + dy <
ballRadius) {
65.              dy = -dy;
66.          }
67.
68.          /* AQUÍ SE CAMBIA LA POSICIÓN DE LA BOLA. SE TOMA EN
CUENTA LAS
69.          MODIFICACIONES A dx y dy, EN CASO DE QUE SE
HUBIERAN PRODUCIDO */
70.          x += dx;
71.          y += dy;

```

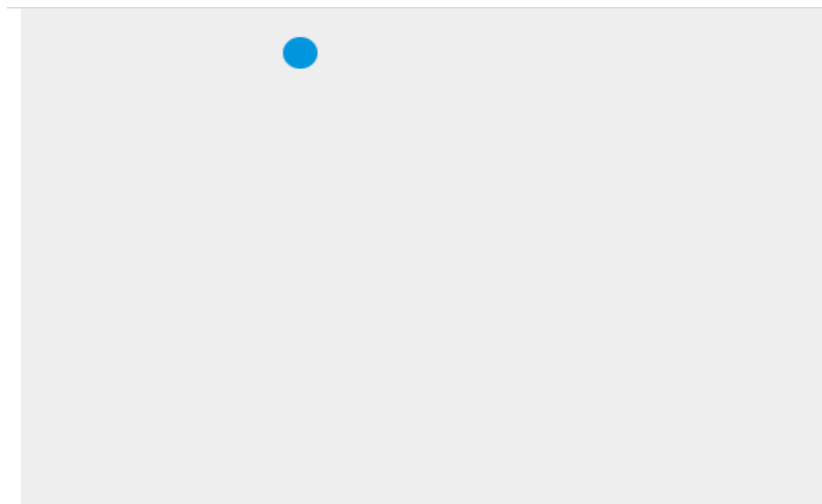


```

72.         }
73.
74.         setInterval(dibujar, 10);
75.     </script>
76.
77. </body>
78. </html>

```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la figura 3 podemos observar a la bola rebotando y cumpliendo con los límites anteriormente definidos en las condiciones.

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

5 FASE 3: CONTROL DE LA PALA Y EL TECLAD

El paso numero 3 consiste en crear la paleta en la cual la bola rebotara, la cual estará situada en la parte inferior del juego y sera controlada por ambas flechas del teclado.

Para empezar con el tercer paso primero se deben crear dos variables a las cuales se les asignara el movimiento de la paleta con las flechas del teclado. Estas variables llevaran el siguiente nombre flechaDerechaPulsada y flechaIzquierdaPulsada. Luego de esto se crea una función la cual maneja el movimiento de la tecla presionada y otro de la tecla liberada

```

1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="utf-8" />
5.     <title>Juego 2D - #03 - Paleta y Control por Teclado</title>
6.     <style>* { padding: 0; margin: 0; } canvas { background: #eee;
    display: block; margin: 0 auto; }</style>
7. </head>
8. <body>
9.
10.     <canvas id="miCanvas" width="480" height="320"></canvas>
11.
12.     <script>
13.         var canvas = document.getElementById("miCanvas");
14.         var ctx = canvas.getContext("2d");
15.
16.         /* Variables básicas:
17.
18.         radioBola: radio de la esfera
19.         x: columna en la que se encuentra situada la bola
20.         y: fila en la que se encuentra situada la bola
21.         dx: desplazamiento horizontal de la bola
22.         dy: desplazamiento vertical de la bola
23.
24.         NOTAS: originalmente, la bola está en centro del
    CANVAS
25.               en el sentido horizontal. Y se encuentra en la
26.               base inferior, pues el eje Y crece de arriba
    hacia
27.               abajo. A este valor se le resta 30, para tomar
    en
28.               cuenta el tamaño de la bola (que es de 20 si
    tomamos
29.               en cuenta el diámetro)
30.

```

```

31.      NOTAS: El desplazamiento en el eje X y en el eje Y,
    son
32.      controlados por la variable dx y la variable
    dy.
33.      Estos valores son de 2 pixeles, y gracias a
    este
34.      avance que se realiza en un ciclo ejecutado
    cada
35.      10 milisegundos, se genera el efecto de avance
    de
36.      la bola. Dentro del ciclo se cambia la
    coordenada
37.      (x, y) agregando los valores (dx, dy), motivo
    por
38.      el cual la bola cambia su posición cada 10
    milisegundos
39.      */
40.      var radioBola = 10;
41.      var x = canvas.width/2;
42.      var y = canvas.height-30;
43.      var dx = 2;
44.      var dy = -2;
45.
46.      /* Las variables a continuación, tienen el siguiente
    significado:
47.
48.      Se define una paleta en la que rebotará la bola
49.      La paleta está situada en la base de la pantalla
    de juego
50.      Dicha paleta será controlada por la flecha
    izquierda y
51.      la flecha derecha del teclado (luego será
    controlador por el ratón)
52.
53.      alturaPaleta: define la altura de la paleta en
    pixeles
54.      anchuraPaleta: define la anchura de la paleta
55.
56.      NOTA: Estos dos valores determinan el tamaño de la
    paleta
57.      La paleta se encuentra situada en la base de
    la pantalla
58.      Para calcular la posición en X de la paleta,
    se debe tomar
59.      el ancho del CANVAS, restarle la anchura de
    la paleta, y
60.      el espacio que sobre debe dividirse entre
    dos
61.      Esto garantiza que originalmente la paleta
    estará centrada
62.      en la base de la pantalla
63.
64.      Al inicio del juego, aún no se ha presionado
    ninguna de las

```

```

65.         flechas. Esta es la razón por la cual se definen
        dos variables que
66.         "recuerdan" cual de las flechas se ha presionado,
        pero que
67.         inicialmente están puestas a: false, indicando el
        estado inicial
68.         Cuando se pulse cualquiera de las dos flechas, su
        valor será:
69.         true (verdadero), y este valor permitira
        establecer en qué
70.         dirección se debe mover la paleta (dentro del
        ciclo del juego)
71.         Las variables son:
72.
73.         flechaDerechaPulsada
74.         flechaIzquierdaPulsada
75.
76.         NOTA: Desde ahora debe tomarse en cuenta que
        cuando se pulse
77.         cualquiera de las dos flechas, solamente se
        hará un
78.         desplazamiento de la paleta a la izquierda o
        hacia la
79.         derecha. Si se mantiene pulsada la tecla, la
        paleta se
80.         continuará desplazando, hasta alcanzar el
        extremo derecho
81.         o izquierdo de la pantalla del juego
82.         */
83.         var alturaPaleta = 10;
84.         var anchuraPaleta = 75;
85.         var paletaPosX = (canvas.width-anchuraPaleta)/2;
86.         var flechaDerechaPulsada = false;
87.         var flechaIzquierdaPulsada = false;
88.
89.         /* La instruccion: addEventListener, se utiliza para
        crear un
90.         mecanismo de respuesta ante eventos que se produzcan
        en el juego
91.
92.         addEventListener "agregar un mecanismo que detecta y
        recibe eventos"
93.
94.         addEventListener recibe tres parámetros:
95.
96.         1) El evento que se va a detectar
97.         2) El nombre que le asignamos a la función que
        responde ante el evento
98.         3) Valor true o false que determina la reacción ante
        el evento
99.
100.        Los dos primeros parámetros son fáciles de entender.
        Pero el tercero
101.        requiere de una explicación adicional:

```

```

102.
103.      Para entender el tercer parámetro, primero hemos de
104.      saber lo que es
105.      el flujo de eventos.
106.
107.      Supongamos que tenemos este tres objetos en la
108.      página:
109.
110.      <body>
111.      <div>
112.      <button>HAZME CLIC</button>
113.      </div>
114.      </body>
115.
116.      El <body> contiene un <div>, y dentro de él esta
117.      un <button>
118.
119.      Cuando hacemos clic en el botón no sólo lo estamos
120.      haciendo sobre él,
121.      sino sobre los elementos que lo contienen en el árbol
122.      de la página,
123.      es decir, hemos hecho clic, además, sobre el
124.      elemento <body> y sobre
125.      el elemento <div>. Si sólo hay una función asignada a
126.      una escucha
127.      para el botón, no hay mayor problema, pero si además
128.      hay una
129.      escucha para el body y otra para el div,
130.      ¿cuál es el orden en que se deben lanzar las tres
131.      funciones?
132.
133.      Para contestar a esa pregunta existe un modelo de
134.      comportamiento,
135.      el flujo de eventos. Según éste, cuando se hace
136.      clic sobre un
137.      elemento, el evento se propaga en dos fases, una
138.      que es la
139.      captura –el evento comienza en el nivel superior
140.      del documento
141.      y recorre los elementos de padres a hijos– y la
142.      otra la burbuja
143.      –el orden inverso, ascendiendo de hijos a padres–.
144.
145.      Así, el orden por defecto de lanzamiento de las
146.      funciones
147.      de escucha, sería: primero la función de escuch de
148.      body,
149.      luego la función de escucha de div, y por último
150.      la función
151.      de escucha de button.
152.
153.      Una vez visto esto, podemos comprender el tercer
154.      parámetro de addEventListener, que lo que hace es permitirnos
155.      escoger el orden de propagación:

```

```

137.
138.      true: El orden de propagación para el ejemplo sería,
    por tanto,
139.      body-div-button
140.
141.      false: La propagación seguiría el modelo burbuja.
142.      Así, el orden sería button-div-body.
143.
144.      NOTA: omo en nuestro ejemplo utilizamos "false",
    estamos
145.      eaccionando primero ante el evento sobre las
    teclas,
146.      posteriormente sobre los eventos asociados al
    CANVAS.
147.      ste es el mecanismo más usual, pero se utilizará
    "true"
148.      n las situaciones que lo requieran
149.      */
150.      document.addEventListener("keydown",
    manejadorTeclaPresionada, false);
151.      document.addEventListener("keyup",
    manejadorTeclaLiberada, false);
152.
153.      // Función que maneja tecla presionada
154.      function manejadorTeclaPresionada(e) {
155.          if(e.keyCode == 39) {
156.              /* e: Es el evento que se produce, en este
    caso
157.              tecla presionada. La
    propiedad: keyCode permite
158.              descubrir de qué tecla se
    trata. Si el código es 39,
159.              se ha presionado la flecha
    derecha. En este caso
160.              se coloca la variable:
    flechaDerechaPulsada a true
161.              */
162.              flechaDerechaPulsada = true;
163.          }
164.          else if(e.keyCode == 37) {
165.              /* e: Es el evento que se produce, en este
    caso
166.              tecla presionada. La
    propiedad: keyCode permite
167.              descubrir de qué tecla se
    trata. Si el código es 37,
168.              se ha presionado la flecha
    izquierda. En este caso
169.              se coloca la variable:
    flechaIzquierdaPulsada a true
170.              */
171.              flechaIzquierdaPulsada = true;
172.          }
173.      }

```

```

174.
175.      // Función que maneja tecla liberada
176.      function manejadorTeclaLiberada(e) {
177.          if(e.keyCode == 39) {
178.              /* Si la tecla liberada es la 39, se ha
dejado de
179.                  presionar la flecha derecha. En este caso,
la variable
180.                      se pone en: false
181.              */
182.              flechaDerechaPulsada = false;
183.          }
184.          else if(e.keyCode == 37) {
185.              /* Si la tecla liberada es la 37, se ha
dejado de
186.                  presionar la flecha izquierda. En este
caso, la variable
187.                      se pone en: false
188.              */
189.              flechaIzquierdaPulsada = false;
190.          }
191.      }
192.
193.      // Dibuja la bola. Código explicado en anteriores
programas
194.      function dibujarBola() {
195.          ctx.beginPath();
196.          ctx.arc(x, y, radioBola, 0, Math.PI*2);
197.          ctx.fillStyle = "#0095DD";
198.          ctx.fill();
199.          ctx.closePath();
200.      }
201.
202.      function dibujarPaleta() {
203.          // Se inicia el dibujo de la paleta
204.          ctx.beginPath();
205.          /* Se crea un rectángulo utilizando la posición en X
206.             El valor de Y está en la base de la pantalla menos
la
207.             altura de la paleta
208.             Y a continuación se indica la anchura y la altura
de la paleta
209.             */
210.          ctx.rect(paletaPosX, canvas.height-alturaPaleta,
anchuraPaleta, alturaPaleta);
211.          ctx.fillStyle = "#0095DD";
212.          ctx.fill();
213.          // Se "cierra" la paleta, terminando su dibujo
214.          ctx.closePath();
215.      }
216.
217.      // Función principal. A partir de aquí se origina el
proceso
218.      // general del juego

```

```

219.         function dibujar() {
220.             ctx.clearRect(0, 0, canvas.width, canvas.height);
221.
222.             // En primer lugar, dibuja la bola
223.             dibujarBola();
224.
225.             // Seguidamente, dibuja la paleta
226.             dibujarPaleta();
227.
228.             /* IMPORTANTE:
229.
230.                 EL OPERADOR || es el operador lógico OR
231.                 Este operador se utiliza para indicar la condición
de conjunción
232.                 SI SE CUMPLE UNA CONDICIÓN, O SE CUMPLE OTRA
CONDICIÓN, ENTONCES
233.                 SE CUMPLE LA CONDICIÓN
234.
235.                 EL OPERADOR && es el operador lógico AND
236.                 Este operador se utiliza para indicar la condición
de disyunción
237.                 SI SE CUMPLE UNA CONDICIÓN, Y SE CUMPLE OTRA
CONDICIÓN (simultánea), ENTONCES
238.                 SE CUMPLE LA CONDICIÓN
239.
240.             */
241.
242.             // Aquí se controla los límites a los que puede llegar
la bola
243.             // En caso de intentar sobrepasar dichos límites, se
cambia
244.             // el sentido del movimiento
245.             // Este código se explicó en el anterior programa
246.             if(x + dx > canvas.width-radioBola || x + dx <
radioBola) {
247.                 dx = -dx;
248.             }
249.             if(y + dy > canvas.height-radioBola || y + dy <
radioBola) {
250.                 dy = -dy;
251.             }
252.
253.             /* Si se ha pulsado la flecha derecha, y la paleta aún
puede
254.             desplazarse hacia la derecha sin que se sobrepase
el límite de la
255.             pantalla, entonces se procede a cambiar su posición
256.             En este caso, la función: dibujarPaleta (la cual se
ejecuta de
257.             manera cíclica) redibujará la paleta en la nueva
posición
258.             */
259.             if(flechaDerechaPulsada && paletaPosX < canvas.width-
anchuraPaleta) {

```

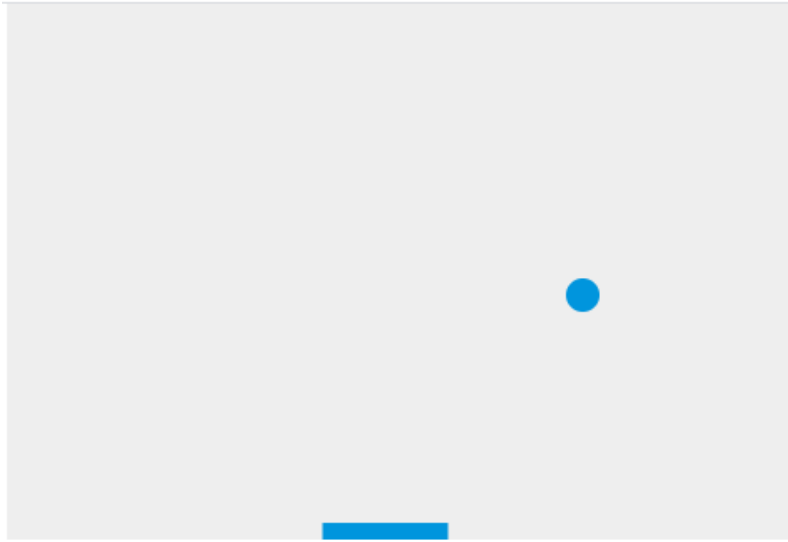


```

260.                // Se desplaza la paleta hacia la derecha
261.                // Aquí, paletaPosX += 7 equivale a:
    paletaPosX = paletaPosX + 7
262.                paletaPosX += 7;
263.            }
264.            else if(flechaIzquierdaPulsada && paletaPosX > 0) {
265.                // Se desplaza la paleta hacia la izquierda
266.                // Aquí, paletaPosX -= 7 equivale a:
    paletaPosX = paletaPosX - 7
267.                paletaPosX -= 7;
268.            }
269.
270.            x += dx;
271.            y += dy;
272.        }
273.
274.        /* Con esta instrucción se crea un ciclo. Cada 10
    milisegundos se
275.        ejecuta la funcion: dibujar(). Esto genera el ciclo
    que permitirá
276.        actualizar el juego, detectar eventos y cambiar el
    estado
277.        de los objetos según las nuevas posiciones que ocupen
    los
278.        elementos del juego
279.
280.        NOTA: La función que se ejecuta es: dibujar
281.        Por tanto, dicha función es la encargada de
    "lanzar" el juego
282.        y dentro de ella se realizarán las acciones que
    desencadenan
283.        el juego como tal
284.        */
285.        setInterval(dibujar, 10);
286.    </script>
287.
288.    </body>
289.    </html>

```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la figura 4 podemos observar la bola y la paleta en la parte inferior del juego creadas anteriormente en la parte numero 3 del código.

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

6 FASE 4: FIN DEL JUEGO

En esta parte del programa programaremos que se pueda detectar cuando la bola toca la base de la pantalla, en una coordenada diferente a la de donde se encuentra la paleta, lo que hará que el juego se pierda.

Para este caso analizaremos un código en la función dibujar, el código sería: $(y + dy > \text{canvas.height} - \text{radioBola})$ el cual se utilizaría para cuando la bola toque la parte inferior del juego lo cual haría que el juego se pierda. Pero para estar seguros de que el juego se ha perdido analizaremos el siguiente código: $(x > \text{paletaPosX} \ \&\& \ x < \text{paletaPosX} + \text{anchuraPaleta})$ el cual hace que se analice la posición de la bola y en caso de que lo bola toque la parte inferior hace que se detenga el ciclo de animación del juego y se pierda.

```

1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="utf-8" />
5.     <title>Juego 2D - #04 - Game Over</title>
6.     <style>* { padding: 0; margin: 0; } canvas { background: #eee;
    display: block; margin: 0 auto; }</style>
7. </head>
8. <body>
9.
10.     <canvas id="miCanvas" width="480" height="320"></canvas>
11.
12.     <script>
13.         /* Este programa detecta cuando la bola toca la base
    de la pantalla
14.             Lo anterior significa que la paleta está en otra
    posición distinta
15.             al punto de toque de la bola con la base de la
    pantalla
16.             En este caso, se considera que el jugador ha
    perdido una vida
17.             El sistema lo informa generando una alerta
18.             El código se encuentra dentro de la función
    dibujar
19.             */
20.
21.         var canvas = document.getElementById("miCanvas");
22.         var ctx = canvas.getContext("2d");
23.
24.         var radioBola = 10;
25.         var x = canvas.width/2;
26.         var y = canvas.height-30;
27.         var dx = 2;
28.         var dy = -2;
29.

```

```

30.         var alturaPaleta = 10;
31.         var anchuraPaleta = 75;
32.         var paletaPosX = (canvas.width-anchuraPaleta)/2;
33.
34.         var flechaDerechaPresionada = false;
35.         var flechaIzquierdaPresionada = false;
36.
37.         document.addEventListener("keydown",
manejadorTeclaPresionada, false);
38.         document.addEventListener("keyup",
manejadorTeclaLiberada, false);
39.
40.         function manejadorTeclaPresionada(e) {
41.             if(e.keyCode == 39) {
42.                 flechaDerechaPresionada = true;
43.             }
44.             else if(e.keyCode == 37) {
45.                 flechaIzquierdaPresionada = true;
46.             }
47.         }
48.         function manejadorTeclaLiberada(e) {
49.             if(e.keyCode == 39) {
50.                 flechaDerechaPresionada = false;
51.             }
52.             else if(e.keyCode == 37) {
53.                 flechaIzquierdaPresionada = false;
54.             }
55.         }
56.
57.         function dibujarBola() {
58.             ctx.beginPath();
59.             ctx.arc(x, y, radioBola, 0, Math.PI*2);
60.             ctx.fillStyle = "#0095DD";
61.             ctx.fill();
62.             ctx.closePath();
63.         }
64.         function dibujarPaleta() {
65.             ctx.beginPath();
66.             ctx.rect(paletaPosX, canvas.height-alturaPaleta,
anchuraPaleta, alturaPaleta);
67.             ctx.fillStyle = "#0095DD";
68.             ctx.fill();
69.             ctx.closePath();
70.         }
71.
72.         function dibujar() {
73.             ctx.clearRect(0, 0, canvas.width, canvas.height);
74.
75.             dibujarBola();
76.             dibujarPaleta();
77.
78.             if(x + dx > canvas.width-radioBola || x + dx <
radioBola) {
79.                 dx = -dx;

```

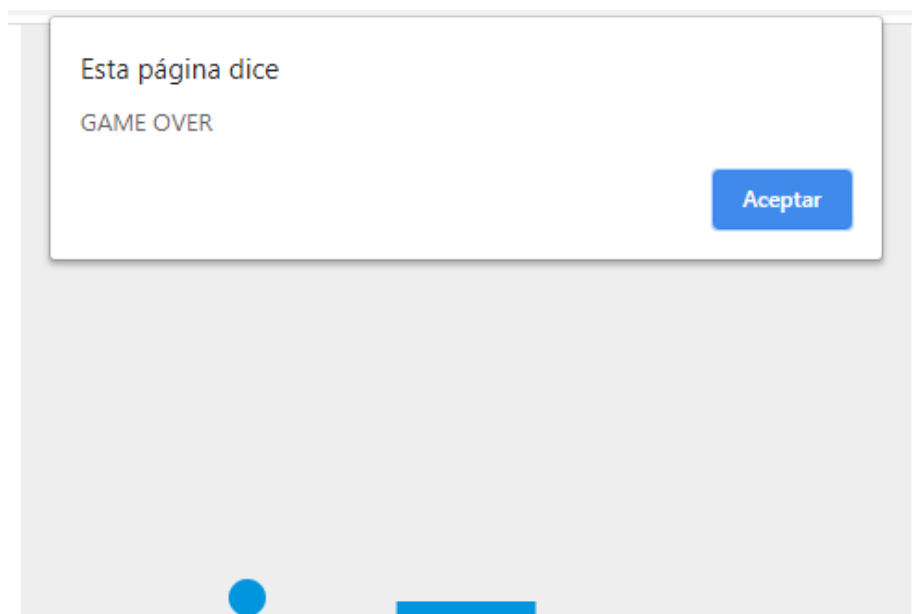
```

80.         }
81.         if(y + dy < radioBola) {
82.             dy = -dy;
83.         }
84.
85.         /* Si y + dy alcanza la frontera inferior de la
pantalla
86.             (y + dy > canvas.height - radioBola)
87.             existe la posibilidad de que el jugador pierda el
juego
88.             Para ello debe evaluarse una segunda opción:
89.             La variable x determina la posición de la bola
90.             Lo que debe hacerse es mirar si x está DENTRO de
la palata:
91.             (x > paletaPosX && x < paletaPosX + anchuraPaleta)
92.             -----
93.             Si x está dentro de la paleta, todo va
bien y se incrementa y
94.             -----
95.             Si x NO ESTÁ dentro de la paleta (else),
la bola ha llegado
96.             a la frontera inferior, y no encuentra la
paleta en su camino
97.             En este caso, SE DETIENE EL CICLO DE
ANIMACIÓN, y se genera
98.             un ALERT indicando que el jugador ha
perdido (GAME OVER)
99.             */
100.            else if(y + dy > canvas.height-radioBola) {
101.                if(x > paletaPosX && x < paletaPosX +
anchuraPaleta) {
102.                    dy = -dy;
103.                }
104.                else {
105.                    clearInterval(juego);
106.                    alert("GAME OVER");
107.                    document.location.reload();
108.                }
109.            }
110.
111.            if(flechaDerechaPresionada && paletaPosX <
canvas.width-anchuraPaleta) {
112.                paletaPosX += 7;
113.            }
114.            else if(flechaIzquierdaPresionada && paletaPosX > 0)
{
115.                paletaPosX -= 7;
116.            }
117.
118.            x += dx;
119.            y += dy;
120.        }

```

```
121.  
122.      /* En este programa se asigna a una variable el proceso  
    cíclico  
123.      Esto tiene mucha importancia, porque si en algún  
    momento se requiere  
124.      eliminar el ciclo, se utilizará la variable asignada  
125.      */  
126.      var juego = setInterval(dibujar, 10);  
127.      </script>  
128.  
129.      </body>  
130.      </html>
```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la figura 5 podemos observar como la bola al tocar la parte inferior del juego y al estar en una coordenada diferente a la paleta aparece un game over que significa que el juego se ha perdido y se ha acabado

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

7 FASE 5: MURO DE LADRILLOS

En esta parte del juego crearemos unas variables las cuales crearan un muro de ladrillos dentro del juego en los cuales rebotara la bola.

Analizaremos la sgte funcion: `function dibujarLadrillos()` , esta función se apoya de varias variables para la creación del muro de los ladrillos la cual la hace analizando la columna y la fila en la que quedara asignado cada ladrillo.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.   <meta charset="utf-8" />
5.   <title>Juego 2D: #05 - Construcción de los ladrillos</title>
6.   <style>* { padding: 0; margin: 0; } canvas { background: #eee;
   display: block; margin: 0 auto; }</style>
7. </head>
8. <body>
9.
10.   <canvas id="miCanvas" width="480" height="320"></canvas>
11.
12.   <script>
13.     var canvas = document.getElementById("miCanvas");
14.     var ctx = canvas.getContext("2d");
15.     var radioBola = 10;
16.     var x = canvas.width/2;
17.     var y = canvas.height-30;
18.     var dx = 2;
19.     var dy = -2;
20.     var alturaPaleta = 10;
21.     var anchuraPaleta = 75;
22.     var paletaPosX = (canvas.width-anchuraPaleta)/2;
23.     var flechaDerechaPresionada = false;
```

```

24.         var flechaIzquierdaPresionada = false;
25.
26.         /* NUEVAS VARIABLES asociadas a los ladrillos
27.         */
28.         var nroFilasLadrillos = 5;
29.         var nroColumnasLadrillos = 3;
30.         var anchoLadrillo = 75;
31.         var alturaLadrillo = 20;
32.         var rellenoLadrillo = 10;
33.         var vacioSuperiorLadrillo = 30;
34.         var vacioIzquierdoLadrillo = 30;
35.
36.         // Crea el conjunto de ladrillos. Inicialmente, vacío
37.         var ladrillos = [];
38.
39.         // Recorre cinco columnas
40.         for(var columna=0; columna<nroColumnasLadrillos;
columna++) {
41.             // Define la primera columna. Es una lista vertical
42.             ladrillos[columna] = [];
43.
44.             // Para la columna, recorre las tres filas, una
después de otra
45.             for(var fila=0; fila<nroFilasLadrillos; fila++) {
46.                 // Para cada (columna, fila) se define un ladrillo
47.
48.
49.                 /* IMPORTANTE:
50.                 Como se puede observar, cada ladrillo está
definido como: ==> ladrillos[c][f]
51.                 Los valores c y f, se corresponden con la fila
y la columna, DENTRO
52.                 DE LA MATRIZ DE LADRILLOS
53.                 -----
54.                 A cada ladrillo en la posicion (c, f), se le
asignan tres valores:
55.
56.                 x: Su coordenada horizontal EN LA PANTALLA
57.                 y: Su coordenada vertical EN LA PANTALLA
58.
59.                 -----
60.                 Los valores x y y valen originalmente cero (0)
61.                 Esto cambia cuando se dibujan (más adelante,
en la función: dibujarLadrillos())
62.                 */
63.                 ladrillos[columna][fila] = { x: 0, y: 0 };
64.             }
65.         }
66.
67.         document.addEventListener("keydown",
manejadorTeclaPresionada, false);

```



```

68.         document.addEventListener("keyup",
manejadorTeclaLiberada, false);
69.
70.         function manejadorTeclaPresionada(e) {
71.             if(e.keyCode == 39) {
72.                 flechaDerechaPresionada = true;
73.             }
74.             else if(e.keyCode == 37) {
75.                 flechaIzquierdaPresionada = true;
76.             }
77.         }
78.         function manejadorTeclaLiberada(e) {
79.             if(e.keyCode == 39) {
80.                 flechaDerechaPresionada = false;
81.             }
82.             else if(e.keyCode == 37) {
83.                 flechaIzquierdaPresionada = false;
84.             }
85.         }
86.
87.         function dibujarBola() {
88.             ctx.beginPath();
89.             ctx.arc(x, y, radioBola, 0, Math.PI*2);
90.             ctx.fillStyle = "#0095DD";
91.             ctx.fill();
92.             ctx.closePath();
93.         }
94.
95.         function dibujarPaleta() {
96.             ctx.beginPath();
97.             ctx.rect(paletaPosX, canvas.height-alturaPaleta,
anchuraPaleta, alturaPaleta);
98.             ctx.fillStyle = "#0095DD";
99.             ctx.fill();
100.            ctx.closePath();
101.        }
102.
103.        /* FUNCIÓN QUE DIBUJA LOS LADRILLOS
104.        -----
105.        */
106.        function dibujarLadrillos() {
107.            // Recorre todas las columnas
108.            for(var columna=0; columna<nroColumnasLadrillos;
columna++) {
109.                // Para cada columna, recorre sus filas
110.                for(var fila=0; fila<nroFilasLadrillos; fila++) {
111.                    // Calcula la coordenada x del ladrillo, según
en que fila se encuentre
112.                    // según el ancho del ladrillo, el valor de
relleno interno
113.                    // y el espacio que debe dejar a la izquierda
114.                    // NOTA: Se sugiere asignar valores y dibujar
el esquema a mano

```

```

115.             var
116.             brickX = (fila*(anchoLadrillo+rellenoLadrillo))+vacioIzquierdoLadri
117.             llo;
118.             // Repite el proceso para calcular la
119.             coordenada y del ladrillo
120.             var
121.             brickY = (columna*(alturaLadrillo+rellenoLadrillo))+vacioSuperiorLa
122.             drillo;
123.             // ASIGNA AL LADRILLO EN LA columna, fila QUE
124.             LE CORRESPONDE EN LA MATRIZ
125.             // EL VALOR CALCULADO (brickX) A SU COORDENADA
126.             x
127.             ladrillos[columna][fila].x = brickX;
128.             // IGUAL PARA EL VALOR y EN PANTALLA
129.             ladrillos[columna][fila].y = brickY;
130.             // DIBUJA EL LADRILLO CON LOS VALORES
131.             ASOCIADOS:
132.             // Coordenada: (brickX, brickY)
133.             // Anchura: anchoLadrillo
134.             // Altrua: alturaLadrillo
135.             ctx.beginPath();
136.             ctx.rect(brickX, brickY, anchoLadrillo,
137.             alturaLadrillo);
138.             ctx.fillStyle = "#0095DD";
139.             ctx.fill();
140.             ctx.closePath();
141.             // COMO SE RECORRE TODO EL CICLO, SE DIBUJAN
142.             TODOS LOS LADRILLOS
143.             }
144.             }
145.             }
146.             function dibujar() {
147.             ctx.clearRect(0, 0, canvas.width, canvas.height);
148.             // DIBUJA EL CONJUNTO DE LADRILLOS
149.             dibujarLadrillos();
150.             dibujarBola();
151.             dibujarPaleta();
152.             if(x + dx > canvas.width-radioBola || x + dx <
153.             radioBola) {
154.             dx = -dx;
155.             }
156.             if(y + dy < radioBola) {
157.             dy = -dy;
158.             }
159.             else if(y + dy > canvas.height-radioBola) {

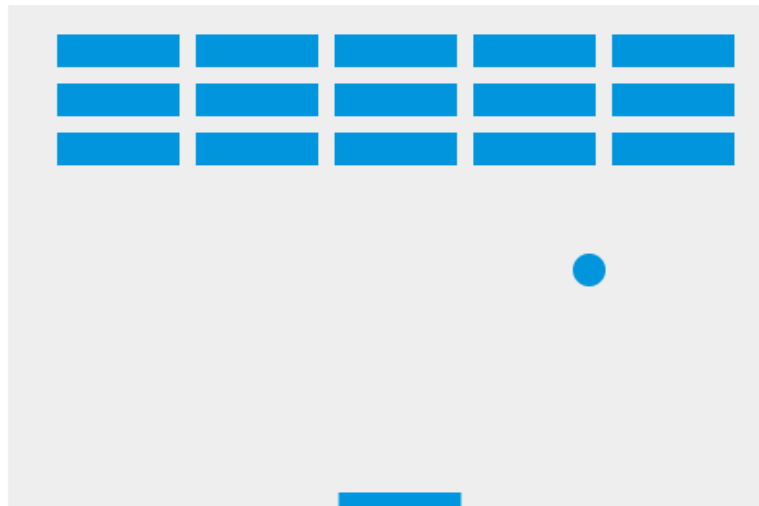
```

```

158.         if(x > paletaPosX && x < paletaPosX +
    anchuraPaleta) {
159.             dy = -dy;
160.         }
161.         else {
162.             clearInterval(juego);
163.             alert("GAME OVER");
164.
165.             // RECARGA LA PÁGINA - El juego vuelve a
    empezar
166.             document.location.reload();
167.         }
168.     }
169.
170.     if(flechaDerechaPresionada && paletaPosX <
    canvas.width-anchuraPaleta) {
171.         paletaPosX += 7;
172.     }
173.     else if(flechaIzquierdaPresionada && paletaPosX > 0)
    {
174.         paletaPosX -= 7;
175.     }
176.
177.     x += dx;
178.     y += dy;
179. }
180.
181.     var juego = setInterval(dibujar, 10);
182. </script>
183.
184. </body>
185. </html>

```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la figura 6 podemos observar la creación de la pared de ladrillos dentro del campo del juego.

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

8 FASE 6: DETECCIÓN DE COLISIONES

En esta parte del programa realizaremos la función que hará que se detecte la colisión de la bola con alguno de los ladrillos y al ocurrir esto hará que el ladrillo con el que colisiono desaparezca.

Procederemos a analizar la función que hace esto posible, la función será la siguiente: function deteccionColision(), esta es la función que permite que cuando la bola colisione con alguno de los ladrillos desaparezca, esto se realiza creando una variable temporal en la cual se asigna el ladrillo y analizando su columna y su fila y así saber si fue impactado.

```

1. Tambien se crean las siguientes variables: la primera es clearInterval(juego); la cual hace
   que se detenga el ciclo del juego, otra es alert("GAME OVER"); la cual hace que al perder
   el juego salga un letrero con la palabra GAME OVER que significa que se ha acabado el
   juego y por ultimo la siguiente variable document.location.reload(); que hace que el juego
   se recargue nuevamente y se pueda volver a empezar <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="utf-8" />
5.     <title>Juego 2D - #06 - Detección de colisión</title>
6.     <style>* { padding: 0; margin: 0; } canvas { background: #eee;
   display: block; margin: 0 auto; }</style>
7. </head>
8. <body>
9.
10.     <canvas id="miCanvas" width="480" height="320"></canvas>
11.
12.     <script>
13.         var canvas = document.getElementById("miCanvas");
14.         var ctx = canvas.getContext("2d");
15.
16.         var radioBola = 10;
17.         var x = canvas.width/2;
18.         var y = canvas.height-30;
19.         var dx = 2;
20.         var dy = -2;
21.
22.         var alturaPaleta = 10;
23.         var anchuraPaleta = 75;
24.         var paletaPosX = (canvas.width-anchuraPaleta)/2;
25.
26.         var flechaDerechaPresionada = false;
27.         var flechaIzquierdaPresionada = false;
28.
29.         var nroFilasLadrillos = 5;
30.         var nroColumnasLadrillos = 3;
31.         var anchuraLadrillo = 75;

```

```

32.         var alturaLadrillo = 20;
33.         var rellenoLadrillo = 10;
34.         var vacioSuperiorLadrillo = 30;
35.         var vacioIzquierdoLadrillo = 30;
36.
37.         var ladrillos = [];
38.         for(var c=0; c<nroColumnasLadrillos; c++) {
39.             ladrillos[c] = [];
40.             for(var f=0; f<nroFilasLadrillos; f++) {
41.
42.                 /* IMPORTANTE:
43.                     Como se puede observar, cada
44.     ladrillo está definido como: ==> ladrillos[c][f]
45.                     Los valores c y f, se corresponden
46.     con la fila y la columna, DENTRO
47.                     DE LA MATRIZ DE LADRILLOS
48.     -----
49.
50.                     A cada ladrillo en la posicion (c,
51.     f), se le asignan tres valores:
52.
53.                     x: Su coordenada horizontal EN
54.     LA PANTALLA
55.                     y: Su coordenada vertical EN LA
56.     PANTALLA
57.                     status: Indica si está visible
58.     o invisible. 1 = Visible, 0 = INVISIBLE
59.
60.                     Inicialmente el ladrillo debe esta
61.     visible. Si la bola "toca" al ladrillo,
62.                     el ladrillo se debe volver
63.     INVISIBLE (status = 0)
64.     -----
65.
66.                     Los valores x y y valen
67.     originalmente cero (0)
68.                     Esto cambia cuando se dibujan (más
69.     adelante, en la función: dibujarLadrillos())
70.
71.                     */
72.                     ladrillos[c][f] = { x: 0, y: 0, status: 1 };
73.                 }
74.             }
75.
76.             document.addEventListener("keydown",
77.     manejadorTeclaPresionada, false);
78.             document.addEventListener("keyup",
79.     manejadorTeclaLiberada, false);
80.
81.             function manejadorTeclaPresionada(e) {
82.                 if(e.keyCode == 39) {
83.                     flechaDerechaPresionada = true;
84.                 }
85.                 else if(e.keyCode == 37) {
86.                     flechaIzquierdaPresionada = true;

```

```

72.         }
73.     }
74.
75.     function manejadorTeclaLiberada(e) {
76.         if(e.keyCode == 39) {
77.             flechaDerechaPresionada = false;
78.         }
79.         else if(e.keyCode == 37) {
80.             flechaIzquierdaPresionada = false;
81.         }
82.     }
83.
84.     // EN ESTA FUNCIÓN SE DETECTA LA COLISIÓN DE LA BOLA CON
      EL LADRILLO
85.
86.     function deteccionColision() {
87.
88.         // LOS DOS CICLOS SIGUIENTES RECORREN TODOS LOS
      LADRILLOS
89.         for(var c=0; c<nroColumnasLadrillos; c++) {
90.             for(var f=0; f<nroFilasLadrillos; f++) {
91.
92.                 // EN ESTE PUNTO SE TIENE EL LADRILLO SITUADO
      EN: (c, f)
93.                 // SE CREA UNA VARIABLE TEMPORAL PARA EL
      LADRILLO
94.                 var b = ladrillos[c][f];
95.
96.                 // SI EL LADRILLO ES VISIBLE, se debe
      verificar si entra en contacto con la bola
97.                 if(b.status == 1) {
98.
99.                     /* SI LAS COORDENADAS x y y, SE
      ENCUENTRAN DENTRO DE LAS COORDENADAS
100.                     DEL LADRILLO (aspecto que se
      verifica con las condiciones mostradas)
101.                     LA BOLA HA IMPACTADO CONTRA EL
      LADRILLO
102.                     En este caso, se modifica la
      coordenada y, PERO LÓ MÁS IMPORTANTE
103.                     ES QUE SE COLOCA EL VALOR DE
      status A CERO, HACIENDO QUE EL LADRILLO
104.                     SE VUELVA INVISIBLE
105.                     */
106.
107.                     if(x > b.x && x < b.x+anchuraLadrillo && y
      > b.y && y < b.y+alturaLadrillo) {
108.                         dy = -dy;
109.                         b.status = 0;
110.                     }
111.                 }
112.             }
113.         }
114.     }

```

```

115.
116.     function dibujarBola() {
117.         ctx.beginPath();
118.         ctx.arc(x, y, radioBola, 0, Math.PI*2);
119.         ctx.fillStyle = "#0095DD";
120.         ctx.fill();
121.         ctx.closePath();
122.     }
123.
124.     function dibujarPaleta() {
125.         ctx.beginPath();
126.         ctx.rect(paletaPosX, canvas.height-alturaPaleta,
anchuraPaleta, alturaPaleta);
127.         ctx.fillStyle = "#0095DD";
128.         ctx.fill();
129.         ctx.closePath();
130.     }
131.
132.     function dibujarLadrillos() {
133.         for(var c=0; c<nroColumnasLadrillos; c++) {
134.             for(var f=0; f<nroFilasLadrillos; f++) {
135.
136.                 /* IMPORTANTE:
137.
138.                 Solamente se dibujan los ladrillos que
139.                 están VISIBLES
140.                 Se sabe que el ladrillo es visible cuando:
141.                 status == 1
142.                 Los ladrillos INVISIBLES NO SE DIBUJAN
143.
144.                 */
145.                 if(ladrillos[c][f].status == 1) {
146.
147.                     // SE DIBUJA EL LADRILLO
148.                     var
brickX = (f*(anchuraLadrillo+rellenoLadrillo))+vacioIzquierdoLadrillo;
149.                     var
brickY = (c*(alturaLadrillo+rellenoLadrillo))+vacioSuperiorLadrillo;
150.                     ladrillos[c][f].x = brickX;
151.                     ladrillos[c][f].y = brickY;
152.                     ctx.beginPath();
153.                     ctx.rect(brickX, brickY, anchuraLadrillo,
alturaLadrillo);
154.                     ctx.fillStyle = "#0095DD";
155.                     ctx.fill();
156.                     ctx.closePath();
157.                 }
158.             }
159.         }
160.

```

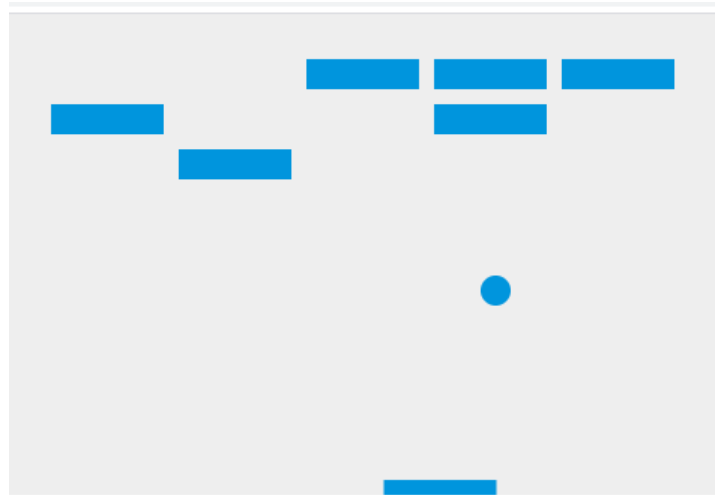


```

161.         function dibujar() {
162.             ctx.clearRect(0, 0, canvas.width, canvas.height);
163.             dibujarLadrillos();
164.             dibujarBola();
165.             dibujarPaleta();
166.             deteccionColision();
167.
168.             if(x + dx > canvas.width-radioBola || x + dx <
radioBola) {
169.                 dx = -dx;
170.             }
171.             if(y + dy < radioBola) {
172.                 dy = -dy;
173.             }
174.             else if(y + dy > canvas.height-radioBola) {
175.                 if(x > paletaPosX && x < paletaPosX +
anchuraPaleta) {
176.                     dy = -dy;
177.                 }
178.                 else {
179.                     // Detiene el ciclo del juego
180.                     clearInterval(juego);
181.                     // Genera mensaje, pues el jugador ha perdido
182.                     alert("GAME OVER");
183.                     // Recarga la página, para iniciar de nuevo
el juego
184.                     document.location.reload();
185.                 }
186.             }
187.
188.             if(flechaDerechaPresionada && paletaPosX <
canvas.width-anchuraPaleta) {
189.                 paletaPosX += 7;
190.             }
191.             else if(flechaIzquierdaPresionada && paletaPosX > 0)
{
192.                 paletaPosX -= 7;
193.             }
194.
195.             x += dx;
196.             y += dy;
197.         }
198.
199.         var juego = setInterval(dibujar, 10);
200.     </script>
201.
202. </body>
203. </html>

```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la figura 7 podemos observar como algunos ladrillos se desaparecieron luego de ser golpeados por la bola.

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

9 FASE 7: CONTAR PUNTOS Y GANAR

En esta parte del programa se realiza la variable para darle algún valor cuando la bola golpee algún ladrillo y se convierta en un punto y se sumen estos puntos hasta ganar el juego con el máximo de puntos que se puedan obtener

Se crea una variable llamada puntaje la cual controla la cantidad de ladrillos que han sido golpeados por la bola, cada que la bola impacta un ladrillo se le agrega un valor a esta variable hasta que el puntaje es igual al numero de ladrillos haciendo que el juego se gane.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.   <meta charset="utf-8" />
5.   <title>Juego 2D - #07 - Control de juego ganado</title>
```

```

6.      <!-- EN ESTE EJEMPLO SE CAMBIA LA ANCHURA DE LA PALETA
7.          ESTO ES CLAVE PARA PERMITIR QUE EL JUEGO SEA AUTOMÁTICO
8.          Y SE PUEDA VERIFICAR EL OBJETIVO DEL JUEGO Y EL JUGADOR
      GANE -->
9.      <style>* { padding: 0; margin: 0; } canvas { background: #eee;
      display: block; margin: 0 auto; }</style>
10.     </head>
11.     <body>
12.
13.     <canvas id="miCanvas" width="480" height="320"></canvas>
14.
15.     <script>
16.         var canvas = document.getElementById("miCanvas");
17.         var ctx = canvas.getContext("2d");
18.
19.         var radioBola = 10;
20.         var x = canvas.width/2;
21.         var y = canvas.height-30;
22.         var dx = 2;
23.         var dy = -2;
24.         var alturaPaleta = 10;
25.
26.         // EL ANCHO DE LA PALETA ES 480. ESTE ES EL MISMO ANCHO
      DEL CANVAS
27.         // Con esto se garantiza que el juego termine
28.         var anchuraPaleta = 480;
29.
30.         var paletaPosX = (canvas.width-anchuraPaleta)/2;
31.         var flechaDerechaPresionada = false;
32.         var flechaIzquierdaPresionada = false;
33.
34.         var nroFilasLadrillos = 5;
35.         var nroColumnasLadrillos = 3;
36.         var anchuraLadrillos = 75;
37.         var alturaLadrillos = 20;
38.         var rellenoLadrillos = 10;
39.         var vacioSuperiorLadrillo = 30;
40.         var vacioIzquierdoLadrillo = 30;
41.
42.         // LA VARIABLE puntaje CONTROLA EL NÚMERO DE LADRILLOS
      QUE HAN SIDO
43.         // IMPACTADOS POR LA BOLA. Cada vez que la bola golpee un
      ladrillo,
44.         // la variable "puntaje" se incrementa en uno
45.         var puntaje = 0;
46.
47.         var ladrillos = [];
48.         for(var c=0; c<nroColumnasLadrillos; c++) {
49.             ladrillos[c] = [];
50.             for(var f=0; f<nroFilasLadrillos; f++) {
51.                 ladrillos[c][f] = { x: 0, y: 0, estado: 1 };
52.             }
53.         }
54.

```

```

55.     document.addEventListener("keydown",
manejadorTeclaPresionada, false);
56.     document.addEventListener("keyup", manejadorTeclaLiberada,
false);
57.
58.     function manejadorTeclaPresionada(e) {
59.         if(e.keyCode == 39) {
60.             flechaDerechaPresionada = true;
61.         }
62.         else if(e.keyCode == 37) {
63.             flechaIzquierdaPresionada = true;
64.         }
65.     }
66.     function manejadorTeclaLiberada(e) {
67.         if(e.keyCode == 39) {
68.             flechaDerechaPresionada = false;
69.         }
70.         else if(e.keyCode == 37) {
71.             flechaIzquierdaPresionada = false;
72.         }
73.     }
74.     function detectarColision() {
75.         for(var c=0; c<nroColumnasLadrillos; c++) {
76.             for(var f=0; f<nroFilasLadrillos; f++) {
77.                 var b = ladrillos[c][f];
78.                 if(b.estado == 1) {
79.                     if(x > b.x && x < b.x+anchuraLadrillos
80. && y > b.y && y < b.y+alturaLadrillos) {
81.                         dy = -dy;
82.                         b.estado = 0;
83.
84.                                     // LA INSTRUCCIÓN puntaje++ EQUIVALE
85.                                     // -----
86.                                     // EN ESTE PUNTO DEL CÓDIGO LA BOLA
HA IMPACTADO UN LADRILLO
87.                                     // POR ESTE MOTIVO, SE INCREMENTA EL
VALOR DE puntaje
88.                                     // Si el puntaje es igual al número
total de ladrillos (valor que
89.                                     // se obtiene multiplicando el número
de filas de ladrillos por el
90.                                     // número de columnas de ladrillos),
entonces el jugador ha ganado
91.                                     puntaje++;
92.                                     if(puntaje ==
nroFilasLadrillos*nroColumnasLadrillos) {
93.                                         alert("USTED GANA!
FELICITACIONES!!!");
94.                                         document.location.reload();
95.                                     }
96.
97.     }

```

```

98.         }
99.     }
100. }
101. }
102.
103.     function dibujarBola() {
104.         ctx.beginPath();
105.         ctx.arc(x, y, radioBola, 0, Math.PI*2);
106.         ctx.fillStyle = "#0095DD";
107.         ctx.fill();
108.         ctx.closePath();
109.     }
110.
111.     function dibujarPaleta() {
112.         ctx.beginPath();
113.         ctx.rect(paletaPosX, canvas.height-alturaPaleta,
anchuraPaleta, alturaPaleta);
114.         ctx.fillStyle = "#0095DD";
115.         ctx.fill();
116.         ctx.closePath();
117.     }
118.
119.     function dibujarLadrillos() {
120.         for(var c=0; c<nroColumnasLadrillos; c++) {
121.             for(var r=0; r<nroFilasLadrillos; r++) {
122.                 if(ladrillos[c][r].estado == 1) {
123.                     var
posXLadrillo = (r*(anchuraLadrillos+rellenoLadrillos))+vacioIzquier
doLadrillo;
124.                     var
posYLadrillo = (c*(alturaLadrillos+rellenoLadrillos))+vacioSuperior
Ladrillo;
125.                     ladrillos[c][r].x = posXLadrillo;
126.                     ladrillos[c][r].y = posYLadrillo;
127.                     ctx.beginPath();
128.                     ctx.rect(posXLadrillo, posYLadrillo,
anchuraLadrillos, alturaLadrillos);
129.                     ctx.fillStyle = "#0095DD";
130.                     ctx.fill();
131.                     ctx.closePath();
132.                 }
133.             }
134.         }
135.     }
136.
137.     function dibujarPuntaje() {
138.         ctx.font = "16px Arial";
139.         ctx.fillStyle = "#0095DD";
140.         ctx.fillText("puntaje: "+puntaje, 8, 20);
141.     }
142.
143.     function dibujar() {
144.         ctx.clearRect(0, 0, canvas.width, canvas.height);
145.         dibujarLadrillos();

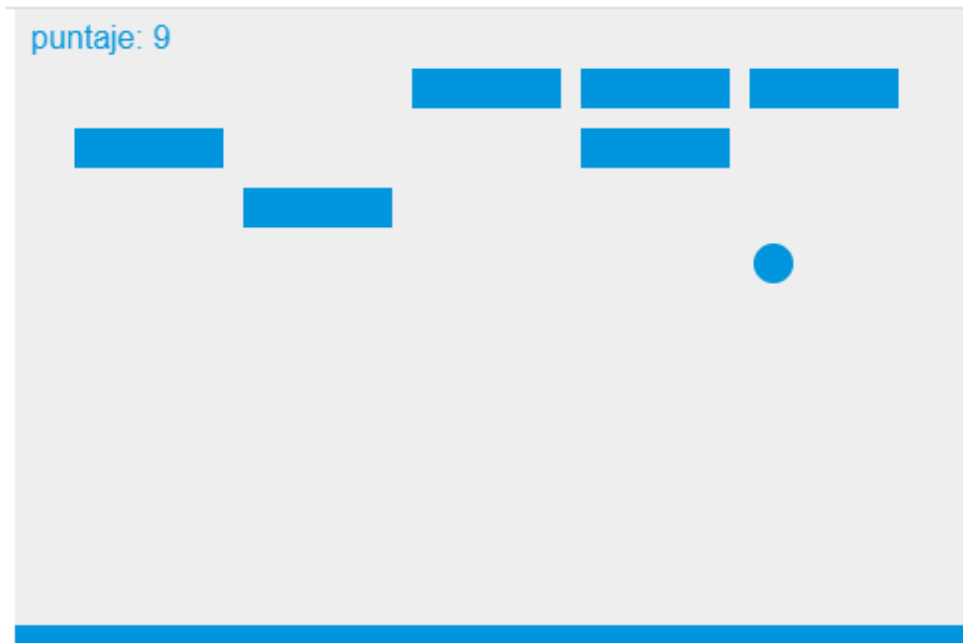
```

```

146.         dibujarBola();
147.         dibujarPaleta();
148.         dibujarPuntaje();
149.         detectarColision();
150.
151.         if(x + dx > canvas.width-radioBola || x + dx <
radioBola) {
152.             dx = -dx;
153.         }
154.         if(y + dy < radioBola) {
155.             dy = -dy;
156.         }
157.         else if(y + dy > canvas.height-radioBola) {
158.             if(x > paletaPosX && x < paletaPosX +
anchuraPaleta) {
159.                 dy = -dy;
160.             }
161.             else {
162.                 clearInterval(juego);
163.                 alert("GAME OVER");
164.                 document.location.reload();
165.             }
166.         }
167.
168.         if(flechaDerechaPresionada && paletaPosX <
canvas.width-anchuraPaleta) {
169.             paletaPosX += 7;
170.         }
171.         else if(flechaIzquierdaPresionada && paletaPosX > 0)
{
172.             paletaPosX -= 7;
173.         }
174.
175.         x += dx;
176.         y += dy;
177.     }
178.
179.     var juego = setInterval(dibujar, 10);
180. </script>
181.
182. </body>
183. </html>

```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la figura 8 se puede observar como la bola al impactar en los ladrillos estos desaparecen y el puntaje incrementa hasta desaparecer todos los ladrillos y ganar el juego.

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

10 FASE 8: CONTROLANDO EL RATÓN

En esta parte del programa haremos que la paleta en lugar de ser movida por las flechas sea movida por el mouse.

Esto se obtiene creando una función llamada `function manejadorRaton(e)` a la cual se le da una variable y una condición que al cumplirla hace que la paleta pueda ser desplazada mediante el mouse.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="utf-8" />
5.     <title>Juego 2D - #08 - Utilizando el ratón</title>
6.     <style>* { padding: 0; margin: 0; } canvas { background: #eee;
7.     display: block; margin: 0 auto; }</style>
8. </head>
9. <body>
10.     <canvas id="miCanvas" width="480" height="320"></canvas>
11.
12.     <script>
13.         var canvas = document.getElementById("miCanvas");
14.         var ctx = canvas.getContext("2d");
15.
16.         var radioBola = 10;
17.         var x = canvas.width/2;
18.         var y = canvas.height-30;
19.         var dx = 2;
20.         var dy = -2;
21.
22.         var alturaPaleta = 10;
23.         var anchuraPaleta = 75;
24.         var paletaPosX = (canvas.width-anchuraPaleta)/2;
25.
26.         var flechaDerechaPresionada = false;
27.         var flechaIzquierdaPresionada = false;
28.
29.         var nroFilasLadrillos = 5;
30.         var nroColumnasLadrillos = 3;
31.         var anchuraLadrillo = 75;
32.         var alturaLadrillo = 20;
33.         var rellenoLadrillo = 10;
34.         var vacioSuperiorLadrillo = 30;
35.         var vacioIzquierdoLadrillo = 30;
36.
37.         var puntaje = 0;
38.
39.         var ladrillos = [];
40.         for(var c=0; c<nroColumnasLadrillos; c++) {
```



```

41.         ladrillos[c] = [];
42.         for(var f=0; f<nroFilasLadrillos; f++) {
43.             ladrillos[c][f] = { x: 0, y: 0, estado: 1 };
44.         }
45.     }
46.
47.     document.addEventListener("keydown",
manejadorTeclaPresionada, false);
48.     document.addEventListener("keyup", manejadorTeclaLiberada,
false);
49.
50.     // PARA DETECTAR EL MOVIMIENTO DEL RATÓN, SE COLOCA UN
ESCUCHADOR (listener)
51.     // AL EVENTO "mousemove"
52.     document.addEventListener("mousemove", manejadorRaton,
false);
53.
54.     function manejadorTeclaPresionada(e) {
55.         if(e.keyCode == 39) {
56.             flechaDerechaPresionada = true;
57.         }
58.         else if(e.keyCode == 37) {
59.             flechaIzquierdaPresionada = true;
60.         }
61.     }
62.
63.     function manejadorTeclaLiberada(e) {
64.         if(e.keyCode == 39) {
65.             flechaDerechaPresionada = false;
66.         }
67.         else if(e.keyCode == 37) {
68.             flechaIzquierdaPresionada = false;
69.         }
70.     }
71.
72.     // ESTE ES EL MANEJADOR DEL RATÓN
73.     // -----
74.     // La instrucción: "offsetLeft" calcula la distancia desde
el borde izquierdo
75.     // de la pantalla hasta un componente html
76.     // -----
77.     // Por tanto, la instrucción: "canvas.offsetLeft" calcula
el espacio a la izquierda
78.     // del objeto CANVAS
79.     // -----
80.     // Dentro del manejador del ratón, la
instrucción: "e.clientX" calcula la posición
81.     // del ratón en la pantalla. Para calcular la posición del
ratón DENTRO del CANVAS
82.     // debemos RESTAR a la posición X del ratón, el valor
izquierdo del CANVAS
83.     // -----
84.     // Es decir: "e.clientX - canvas.offsetLeft"
85.     // -----

```

```

86.         function manejadorRaton(e) {
87.             var posXRatonDentroDeCanvas = e.clientX -
               canvas.offsetLeft;
88.             // EL SIGUIENTE if DETERMINA SI LA POSICIÓN X DEL
               RATÓN ESTÁ
89.             // DENTRO DEL CANVAS
90.             if(posXRatonDentroDeCanvas > 0 &&
               posXRatonDentroDeCanvas < canvas.width) {
91.                 // SI LA RESPUESTA ES POSITIVA, EL RATÓN ESTÁ
               DENTRO DEL CANVAS
92.                 // EN ESTE CASO, SE RECALCULA LA POSICIÓN DE LA
               PALETA
93.                 // SU VALOR X ES AHORA LA POSICIÓN X DEL RATÓN
94.                 // -----
95.                 // PERO DEBE RECORDARSE QUE LA PALETA TIENE UN
               ANCHO. ESTA ES LA RAZÓN
96.                 // POR LA CUAL SE DEBE RESTAR A LA POSICIÓN X DE
               LA PALETA LA MITAD DEL
97.                 // ANCHO DE LA PALETA
98.                 // -----
99.                 // AL HACER ESTO, LA PALETA MODIFICA SU POSICIÓN
               CON BASE EN EL
100.                // MOVIMIENTO DEL RATÓN
101.                paletaPosX = posXRatonDentroDeCanvas -
                  anchuraPaleta/2;
102.            }
103.        }
104.        function detectarColision() {
105.            for(var c=0; c<nroColumnasLadrillos; c++) {
106.                for(var r=0; r<nroFilasLadrillos; r++) {
107.                    var b = ladrillos[c][r];
108.                    if(b.estado == 1) {
109.                        if(x > b.x && x < b.x+anchuraLadrillo && y
               > b.y && y < b.y+alturaLadrillo) {
110.                            dy = -dy;
111.                            b.estado = 0;
112.                            puntaje++;
113.                            if(puntaje ==
               nroFilasLadrillos*nroColumnasLadrillos) {
114.                                alert("USTED GANA,
               FELICITACIONES!!!");
115.                                document.location.reload();
116.                            }
117.                        }
118.                    }
119.                }
120.            }
121.        }
122.
123.        function dibujarBola() {
124.            ctx.beginPath();
125.            ctx.arc(x, y, radioBola, 0, Math.PI*2);

```

```

126.         ctx.fillStyle = "#0095DD";
127.         ctx.fill();
128.         ctx.closePath();
129.     }
130.     function dibujarPaleta() {
131.         ctx.beginPath();
132.         ctx.rect(paletaPosX, canvas.height-alturaPaleta,
133.             anchuraPaleta, alturaPaleta);
134.         ctx.fillStyle = "#0095DD";
135.         ctx.fill();
136.         ctx.closePath();
137.     }
138.     function dibujarLadrillos() {
139.         for(var c=0; c<nroColumnasLadrillos; c++) {
140.             for(var r=0; r<nroFilasLadrillos; r++) {
141.                 if(ladrillos[c][r].estado == 1) {
142.                     var
143.                     brickX = (r*(anchuraLadrillo+rellenoLadrillo))+vacioIzquierdoLadrillo;
144.                     var
145.                     brickY = (c*(alturaLadrillo+rellenoLadrillo))+vacioSuperiorLadrillo;
146.                     ladrillos[c][r].x = brickX;
147.                     ladrillos[c][r].y = brickY;
148.                     ctx.beginPath();
149.                     ctx.rect(brickX, brickY, anchuraLadrillo,
150.                         alturaLadrillo);
151.                     ctx.fillStyle = "#0095DD";
152.                     ctx.fill();
153.                     ctx.closePath();
154.                 }
155.             }
156.         }
157.     }
158.     function dibujarPuntaje() {
159.         ctx.font = "16px Arial";
160.         ctx.fillStyle = "#0095DD";
161.         ctx.fillText("puntaje: "+puntaje, 8, 20);
162.     }
163.     function dibujar() {
164.         ctx.clearRect(0, 0, canvas.width, canvas.height);
165.         dibujarLadrillos();
166.         dibujarBola();
167.         dibujarPaleta();
168.         dibujarPuntaje();
169.         detectarColision();
170.         if(x + dx > canvas.width-radioBola || x + dx <
171.             radioBola) {
172.             dx = -dx;
173.         }
174.         if(y + dy < radioBola) {
175.             dy = -dy;

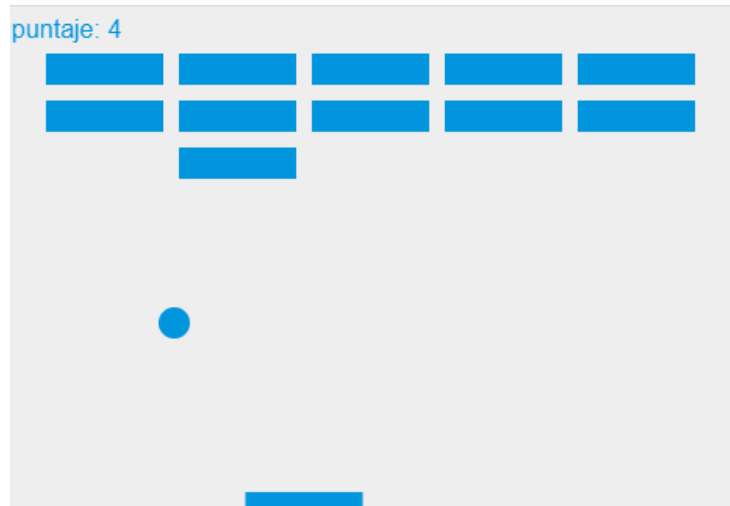
```

```

173.         }
174.         else if (y + dy > canvas.height-radioBola) {
175.             if (x > paletaPosX && x < paletaPosX +
anchuraPaleta) {
176.                 dy = -dy;
177.             }
178.             else {
179.                 clearInterval(juego);
180.                 alert("GAME OVER");
181.                 document.location.reload();
182.             }
183.         }
184.
185.         if (flechaDerechaPresionada && paletaPosX <
canvas.width-anchuraPaleta) {
186.             paletaPosX += 7;
187.         }
188.         else if (flechaIzquierdaPresionada && paletaPosX > 0)
{
189.             paletaPosX -= 7;
190.         }
191.
192.         x += dx;
193.         y += dy;
194.     }
195.
196.     var juego = setInterval(dibujar, 10);
197. </script>
198.
199. </body>
200. </html>

```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la figura 9 se puede observar como la paleta es controlada de derecha a izquierda mediante el uso del mouse

En el siguiente apartado se explicará la siguiente fase del juego. En caso de ser necesario, se agregarán todas las explicaciones que sean necesarias para que el juego quede debidamente explicado.

11 FASE 9: FINALIZANDO EL JUEGO

En esta parte del programa ya se agregan los toques finales tales como vidas para el jugador, ocultar el mouse entre otras.

Se crea una variable `var vidas = 3` con la instrucción de controlar las vidas que tiene dentro del juego cada participante y se crea otra variable `canvas.style.cursor = 'none'` para ocultar el mouse dentro del campo del juego, también se crea la instrucción `vidas--`; la cual lleva la cuenta de las vidas que tiene y que ha perdido.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="utf-8" />
5.     <title>Juego 2D - #09 - Juego completo</title>
6.     <!-- 1. Se oculta el ratón
7.           2. Se agregan vidas al jugador
8.           3. Ya no se utiliza "setInterval" -->
9.     <style>* { padding: 0; margin: 0; } canvas { background: #eee;
display: block; margin: 0 auto; * {cursor: none;} } </style>
10. </head>
11. <body>
12.
13.     <canvas id="miCanvas" width="480" height="320"></canvas>
14.
15.     <script>
16.         var canvas = document.getElementById("miCanvas");
17.         var ctx = canvas.getContext("2d");
18.
19.         var bolaRadio = 10;
20.         var x = canvas.width/2;
21.         var y = canvas.height-30;
22.         var dx = 2;
23.         var dy = -2;
24.
25.         var alturaPaleta = 10;
26.         var anchuraPaleta = 75;
27.         var paletaPosX = (canvas.width-anchuraPaleta)/2;
28.
29.         var flechaDerechaPresionada = false;
30.         var flechaIzquierdaPresionada = false;
31.
32.         var nroFilasLadrillos = 5;
33.         var nroColumnasLadrillos = 3;
34.         var anchuraLadrillo = 75;
35.         var alturaLadrillo = 20;
36.         var rellenoLadrillo = 10;
37.         var vacioSuperiorLadrillo = 30;
38.         var vacioIzquierdoLadrillo = 30;
```

```

39.
40.         var puntaje = 0;
41.
42.         // ESTA INSTRUCCIÓN CONTROLA EL NÚMERO DE VIDAS DEL
    JUGADOR
43.         // CUANDO LA INSTRUCCIÓN vidas DISMINUYE A CERO, EL
    JUGADOR PIERDE,
44.         // PUESTO QUE HA PERDIDO TRES VECES
45.         var vidas = 3;
46.
47.         // ESTA VARIABLE DEFINE UN COLOR
48.         // Se pueden utilizar otros colores para los diferentes
    elementos del juego
49.         var colorFigura = "#ff0000";
50.         var colorBola = "#137B13";
51.         var colorPaleta = "#0000ff";
52.         var colorLadrillo = "#dd2244";
53.         var colorTexto = "#000000";
54.
55.         // ESTA INSTRUCCIÓN OCULTA EL CURSOR DEL RATON (DENTRO
    DEL CANVAS)
56.         canvas.style.cursor = 'none';
57.
58.         var ladrillos = [];
59.         for(var c=0; c<nroColumnasLadrillos; c++) {
60.             ladrillos[c] = [];
61.             for(var f=0; f<nroFilasLadrillos; f++) {
62.                 ladrillos[c][f] = { x: 0, y: 0, estado: 1 };
63.             }
64.         }
65.
66.         document.addEventListener("keydown",
    manejadorTeclaPresionada, false);
67.         document.addEventListener("keyup", manejadorTeclaLiberada,
    false);
68.         document.addEventListener("mousemove", manejadorRaton,
    false);
69.
70.         function manejadorTeclaPresionada(e) {
71.             if(e.keyCode == 39) {
72.                 flechaDerechaPresionada = true;
73.             }
74.             else if(e.keyCode == 37) {
75.                 flechaIzquierdaPresionada = true;
76.             }
77.         }
78.
79.         function manejadorTeclaLiberada(e) {
80.             if(e.keyCode == 39) {
81.                 flechaDerechaPresionada = false;
82.             }
83.             else if(e.keyCode == 37) {
84.                 flechaIzquierdaPresionada = false;
85.             }

```

```

86.     }
87.
88.     function manejadorRaton(e) {
89.         var posXRatonDentroDeCanvas = e.clientX -
canvas.offsetLeft;
90.         if(posXRatonDentroDeCanvas > 0 &&
posXRatonDentroDeCanvas < canvas.width) {
91.             paletaPosX = posXRatonDentroDeCanvas -
anchuraPaleta/2;
92.         }
93.     }
94.
95.     function detectarColision() {
96.         for(var c=0; c<nroColumnasLadrillos; c++) {
97.             for(var f=0; f<nroFilasLadrillos; f++) {
98.                 var b = ladrillos[c][f];
99.                 if(b.estado == 1) {
100.                    if(x > b.x && x < b.x+anchuraLadrillo && y
> b.y && y < b.y+alturaLadrillo) {
101.                        dy = -dy;
102.                        b.estado = 0;
103.                        puntaje++;
104.                        if(puntaje ==
nroFilasLadrillos*nroColumnasLadrillos) {
105.                            alert("USTED GANA,
FELICITACIONES!");
106.                            document.location.reload();
107.                        }
108.                    }
109.                }
110.            }
111.        }
112.    }
113.
114.    function dibujarBola() {
115.        ctx.beginPath();
116.        ctx.arc(x, y, bolaRadio, 0, Math.PI*2);
117.        // SE UTILIZA EL COLOR PREVIAMENTE DEFINIDO
118.        ctx.fillStyle = colorBola;
119.        ctx.fill();
120.        ctx.closePath();
121.    }
122.    function dibujarPaleta() {
123.        ctx.beginPath();
124.        ctx.rect(paletaPosX, canvas.height-alturaPaleta,
anchuraPaleta, alturaPaleta);
125.        ctx.fillStyle = colorPaleta;
126.        ctx.fill();
127.        ctx.closePath();
128.    }
129.    function dibujarLadrillos() {
130.        for(var c=0; c<nroColumnasLadrillos; c++) {
131.            for(var f=0; f<nroFilasLadrillos; f++) {
132.                if(ladrillos[c][f].estado == 1) {

```



```

133.             var
134. ladrilloX = (f*(anchuraLadrillo+rellenoLadrillo))+vacioIzquierdoLad
135. rillo;
136.             var
137. ladrilloY = (c*(alturaLadrillo+rellenoLadrillo))+vacioSuperiorLadri
138. llo;
139.             ladrillos[c][f].x = ladrilloX;
140.             ladrillos[c][f].y = ladrilloY;
141.             ctx.beginPath();
142.             ctx.rect(ladrilloX, ladrilloY,
143. anchuraLadrillo, alturaLadrillo);
144.             ctx.fillStyle = colorLadrillo;
145.             ctx.fill();
146.             ctx.closePath();
147.         }
148.     }
149. }
150.
151. function dibujarPuntaje() {
152.     ctx.font = "16px Arial";
153.     ctx.fillStyle = colorTexto;
154.     ctx.fillText("puntaje: "+puntaje, 8, 20);
155. }
156.
157. function dibujarVidas() {
158.     ctx.font = "16px Arial";
159.     ctx.fillStyle = colorTexto;
160.     // SE MUESTRA EL NÚMERO DE VIDAS DISPONIBLES
161.     ctx.fillText("vidas: "+vidas, canvas.width-65, 20);
162. }
163.
164. function dibujar() {
165.     ctx.clearRect(0, 0, canvas.width, canvas.height);
166.     dibujarLadrillos();
167.     dibujarBola();
168.     dibujarPaleta();
169.     dibujarPuntaje();
170.     dibujarVidas();
171.     detectarColision();
172.
173.     if(x + dx > canvas.width-bolaRadio || x + dx <
174. bolaRadio) {
175.         dx = -dx;
176.     }
177.     if(y + dy < bolaRadio) {
178.         dy = -dy;
179.     }
180.     else if(y + dy > canvas.height-bolaRadio) {
181.         if(x > paletaPosX && x < paletaPosX +
182. anchuraPaleta) {
183.             dy = -dy;
184.         }
185.     }
186.     else {

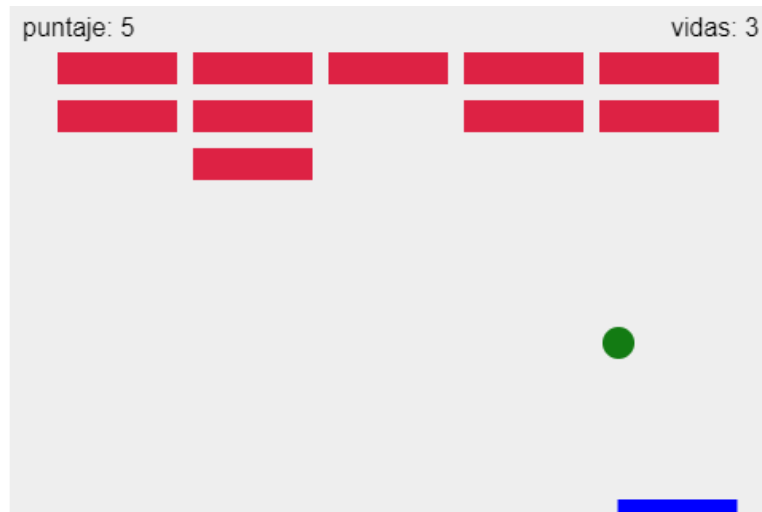
```

```

180.          // SI SE PRODUCE UN CONTACTO DE LA BOLA CON
    LA BASE DEL CANVAS
181.          // SE PIERDE UNA VIDA. PARA ELLO, LA
    INSTRUCCIÓN vidas--;
182.          // LO CUAL EQUIVALE A: vidas = vidas - 1
183.          vidas--;
184.          if(!vidas) {
185.              // SI vidas == 0 (lo cual también
    puede escribir: !vidas)
186.              // EL JUGADOR HA PERDIDO
187.              alert("GAME OVER");
188.              document.location.reload();
189.          }
190.          else {
191.              // SI vidas > 0 (diferente de CERO)
    EL JUEGO CONTINUA
192.              x = canvas.width/2;
193.              y = canvas.height-30;
194.              dx = 3;
195.              dy = -3;
196.              paletaPosX = (canvas.width-
    anchuraPaleta)/2;
197.          }
198.      }
199.  }
200.
201.      if(flechaDerechaPresionada && paletaPosX <
    canvas.width-anchuraPaleta) {
202.          paletaPosX += 7;
203.      }
204.      else if(flechaIzquierdaPresionada && paletaPosX > 0)
    {
205.          paletaPosX -= 7;
206.      }
207.
208.      x += dx;
209.      y += dy;
210.
211.      // ESTE ES UN SEGUNDO MÉTODO PARA REALIZAR LA
    ANIMACIÓN DEL JUEGO
212.      // LA INSTRUCCIÓN: requestAnimationFrame SE EJECUTA
    60 VECES POR SEGUNDO
213.      // Y AL EJECUTARSE LLAMA A LA FUNCIÓN ENTRE
    PARÉNTESIS
214.      // POR TANTO, dibujar SE EJECUTA 60 VECES POR SEGUNDO
215.      // GENERANDO EL CICLO DEL JUEGO
216.      requestAnimationFrame(dibujar);
217.  }
218.
219.      dibujar();
220.  </script>
221.
222. </body>
223. </html>

```

Al ejecutar este código se obtiene la siguiente interfaz visual:



En la imagen 10 podemos observar el juego ya completado totalmente, y en el podemos observar las vidas y el puntaje que lleva el jugador durante el juego y la desaparición del mando dentro del canvas.



12 CONCLUSIONES

En conclusión podemos observar como después de seguir una cierta cantidad de pasos pudimos llegar a nuestro objetivo el cual era construir un juego en 2D.

Este juego realizado a través de un código html asignado a JavaScript, en el cual usando las herramientas prestadas por html y creando y probando las funciones correctas con sus variables y problemas que surgen dentro de este código podemos llegar a tener un juego en la red virtual.

Este es un juego que nos ayuda para el aprendizaje dentro del campo de la programación tanto con el lenguaje html como con tantos lenguajes que existen el día de hoy en el campo de la programación.



13 BIBLIOGRAFÍA

https://developer.mozilla.org/es/docs/Games/Workflows/Famoso_juego_2D_usando_JavaScript_puro/Construye_grupo_bloques