2020

### **GIAN FRANCO POSSO GIRALDO**

### **DANIEL ENRIQUE VILLA ARIAS**

### SEPTIEMBRE DE 2020

UTP | Pereira

# CONTENIDO

[1 CONTENIDO 1](#_TOC_250010)

[2 PRESENTACIÓN 2](#_TOC_250009)

[3 FASE 1: Dibujar y mover una bola 3](#_TOC_250008)

4 BIBLIOGRAFÍA

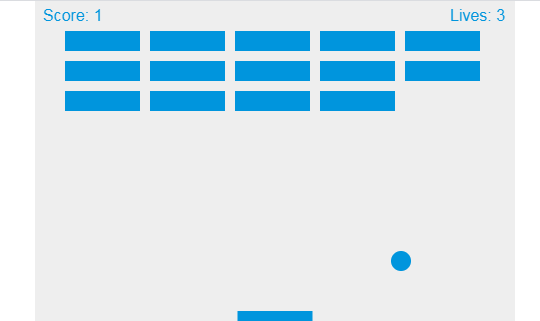
# PRESENTACIÓN

### 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\_](https://developer.mozilla.org/es/docs/Games/Workflows/Famoso_juego_2D_usando_JavaScript_puro)

#### [puro](https://developer.mozilla.org/es/docs/Games/Workflows/Famoso_juego_2D_usando_JavaScript_puro)

**AUTORES: GIAN FRANCO POSSO GIRALDO**

**DANIEL ENRIQUE VILLA ARIAS**

# 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, 18), llama a la función dibujar cada 18 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)<html](http://qbnz.com/highlighter/php_highlighter.php)%3chtml)>

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8" />
5. <title>Juego 2D: JavaScript - 01</title>
7. <!-- Define los estilos de la interfaz visual
8. padding es la distancia de un objeto en relación con el marco que lo contiene
9. margin es la distancia que separa a un objeto de otro
10. background es el color de fondo
11. display: block; Estos elementos fluyen hacia abajo
12. margin: 0 auto; Centra el canvas en la pantalla -->
13. <style>
14. \* {
15. padding: 0;
16. margin: 0;
17. }
18. canvas {
19. background: #000000;
20. display: block;
21. margin: 0 auto;
22. }
23. </style>
25. </head>
26. <body>
28. <canvas id="miCanvas" width="620" height="300"></canvas>
30. <script>
31. **var** canvas = document.getElementById("miCanvas");
32. **var** ctx = canvas.getContext("2d");
34. // Coloca x en la mitad del ancho deL CANVAS
35. **var** y = canvas.width/40;
37. // Coloca y en la mitad de la altura del CANVAS (restando 30 a dicho valor)
38. **var** x = canvas.height -30;
40. /\* DEFINE LOS INCREMENTOS EN X y en Y. El valor dy es negativo
41. para que inicialmente el movimiento de la bola sea hacia arriba \*/
42. **var** dx = -4;
43. **var** dy = 4;
45. **function** dibujarBola() {
46. // Inicia el dibujo
47. ctx.beginPath();
49. /\* Define un círculo en las coordenadas (x, y) con radio 10
50. El ángulo va desde 0 hasta 2\*PI (360 grados) \*/
51. ctx.arc(x, y, 10, 0, Math.PI\*2);
53. // Color de llenado
54. ctx.fillStyle = "#00FFFB";
56. // Se llena el círculo con el color indicado
57. ctx.fill();
59. // Finaliza el dibujo
60. ctx.closePath();
61. }
63. /\* LA FUNCIÓN dibujar REALIZA TRES TAREAS:
64. 1) Limpia el CANVAS. Inicio= (0,0) Ancho=canvas.width Altura=canvas.height
65. 2) Dibuja una bola en las coordenadas (x, y)
66. 3) Cambiar las coordenadas (x, y) agregando los valores dx, dy
67. Con este cambio cada vez que se dibuja la bola, está en una nueva posición \*/
68. **function** dibujar() {
70. // Limpia el CANVAS
71. ctx.clearRect(0, 0, canvas.width, canvas.height);
73. // Dibuja la bola
74. dibujarBola();
76. // Se incrementa x en el valor dx
77. x = x + dx;
79. // Se incrementa y en el valor dy
80. y = y + dy;
81. }
83. /\* EJECUTA LA FUNCIÓN dibujar CADA 10 MILISEGUNDOS
84. Este es el mecanismo utilizado para construir un sistema que
85. ejecuta acciones de manera permanente y periódica \*/
86. setInterval(dibujar, 18);
87. </script>
89. </body>
90. </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 del descenso en X y Y que fueron definidos.

# 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 rebotara y así mantenerse dentro del cuadro asignado.

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

### La primera condición es if(x + dx > canvas.width-ballRadius || x + dx < ballRadius)

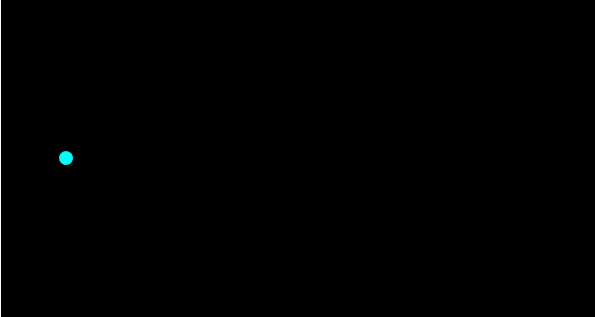
### { dx = -dx;}, esta condición crea el rango horizontal al que la pelota se puede desplazar.

### 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 (probando cosas)</title>
6. <style>
7. \* { padding: 0; margin: 0; }
8. canvas { background: #000; display: block; margin: 0  auto; }
9. </style>
10. </head>
11. <body>
13. <canvas id="miCanvas" width="600" height="320"></canvas>
15. <script>
16. **var** canvas = document.getElementById("miCanvas");
17. **var** ctx = canvas.getContext("2d");
19. /\* Se agrega la variable ballRadius, la cual define el tamaño de la bola \*/
20. **var** ballRadius = 7; // <--------------------
22. **var** y = canvas.width/50;
23. **var** x = canvas.height -30;
24. **var** dy = 3;
25. **var** dx = -3;
27. **function** dibujarBola() {
28. ctx.beginPath();
30. /\* En lugar de un número fijo, se coloca la variable ballRadius \*/
31. ctx.arc(x, y, ballRadius, 0, Math.PI\*2); // <----------------
33. ctx.fillStyle = "#00FFFB";
34. ctx.fill();
35. ctx.closePath();
36. }
38. **function** dibujar() {
39. ctx.clearRect(0, 0, canvas.width, canvas.height);
40. dibujarBola();
42. /\* IMPORTANTE:
44. EL OPERADOR || es el operador lógico OR
45. Este operador se utiliza para indicar la condición de conjunción
46. SI SE CUMPLE UNA CONDICIÓN, O SE CUMPLE OTRA CONDICIÓN, ENTONCES
47. SE CUMPLE LA CONDICIÓN
49. EL OPERADOR && es el oerador lógico AND
50. Este operador se utiliza para indicar la condicion de disyunción
51. SI SE CUMPLE UNA CONDICIÓN, Y SE CUMPLE OTRA CONDICIÓN (simultánea), ENTONCES
52. SE CUMPLE LA CONDICIÓN
54. \*/
56. /\* DESPUÉS DE DIBUJAR LA BOLA, SE DEBEN CAMBIAR LAS COORDENADAS
57. EN LA lección 01 NO SE TENÍA CONTROL SOBRE LOS LÍMITES DE LA CAJA
58. -----------------------------------------------------------------
59. SI x + dx ES MAYOR AL ANCHO DEL CANVAS O MENOR AL TAMAÑO DEL
60. RADIO DE LA BOLA (caso en el cual se encuentra hacia la izquierda)
61. SE CAMBIA LA DIRECCIÓN DE AVANCE HORIZONTAL.
62. ESTO SE LOGRA CAMBIANDO EL SIGNO DE LA VARIABLE dx
63. ESTO HACE QUE SE CAMBIE EL SENTIDO DEL MOVIMIENTO HORIZONTAL \*/
64. **if** (x + dx > canvas.width-ballRadius || x + dx < ballRadius) { // <-----------------
65. dx = dx \* (-1);
66. }
68. /\* SI y + dy ES MAYOR A LA ALTURA DEL CANVAS O MENOR AL TAMAÑO DEL
69. RADIO DE LA BOLA, SE CAMBIA LA DIRECCIÓN DEL AVANCE VERTICAL.
70. ESTO SE LOGRA CAMBIANDO EL SIGNO DE LA VARIABLE dy
71. ESTE CAMBIO EN dy HACE QUE SE MUEVA VERTICALMENTE EN SENTIDO
72. OPUESTO \*/
73. **if**(y + dy > canvas.height-ballRadius || y + dy < ballRadius) { // <-----------------
74. dy = -dy;
75. }
77. /\* AQUÍ SE CAMBIA LA POSICIÓN DE LA BOLA. SE TOMA EN CUENTA LAS
78. MODIFICACIONES A dx y dy, EN CASO DE QUE SE HUBIERAN PRODUCIDO \*/
79. x += dx; // EQUIVALE A: x = x + dx; <----------------
80. y += dy; // EQUIVALE A: y = y + dy; <----------------
81. }
83. setInterval(dibujar, 18);
84. </script>
86. </body>
87. </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.

1. **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>
7. \* { padding: 0; margin: 0; }
8. canvas { background: #000; display: block; margin: 0 auto; }
9. </style>
10. </head>
11. <body>
13. <canvas id="miCanvas" width="620" height="300"></canvas>
15. <script>
16. **var** canvas = document.getElementById("miCanvas");
17. **var** ctx = canvas.getContext("2d");
19. /\* Variables básicas:
21. radioBola: radio de la esfera
22. x: columna en la que se encuentra situada la bola
23. y: fila en la que se encuentra situada la bola
24. dx: desplazamiento horizontal de la bola
25. dy: desplazamiento vertical de la bola
27. NOTAS: originalmente, la bola está en centro del CANVAS
28. en el sentido horizontal. Y se encuentra en la
29. base inferior, pues el eje Y crece de arriba hacia
30. abajo. A este valor se le resta 30, para tomar en
31. cuenta el tamaño de la bola (que es de 20 si tomamos
32. en cuenta el diámetro)
34. NOTAS: El desplazamiento en el eje X y en el eje Y, son
35. controlados por la variable dx y la variable dy.
36. Estos valores son de 2 pixeles, y gracias a este
37. avance que se realiza en un ciclo ejecutado cada
38. 10 milisegundos, se genera el efecto de avance de
39. la bola. Dentro del ciclo se cambia la coordenada
40. (x, y) agregando los valores (dx, dy), motivo por
41. el cual la bola cambia su posición cada 10 milisegundos
42. \*/
44. /\* Se cambia el nombre de la variable \*/
45. **var** radioBola = 7; // <-----------------
47. **var** y = canvas.width/50;
48. **var** x = canvas.height-30;
49. **var** dy = 3;
50. **var** dx = -3;
52. /\* Las variables a continuación, tienen el siguiente significado:
54. Se define una paleta en la que rebotará la bola
55. La paleta está situada en la base de la pantalla de juego
56. Dicha paleta será controlada por la flecha izquierda y
57. la flecha derecha del teclado (luego será controlador por el ratón)
59. alturaPaleta: define la altura de la paleta en pixeles
60. anchuraPaleta: define la anchura de la paleta
62. NOTA: Estos dos valores determinan el tamaño de la paleta
63. La paleta se encuentra situada en la base de la pantalla
64. Para calcular la posición en X de la paleta, se debe tomar
65. el ancho del CANVAS, restarle la anchura de la paleta, y
66. el espacio que sobre debe dividirse entre dos
67. Esto garantiza que originalmente la paleta estará centrada
68. en la base de la pantalla
70. Al inicio del juego, aún no se ha presionado ninguna de las
71. flechas. Esta es la razón por la cual se definen dos variables que
72. "recuerdan" cual de las flechas se ha presionado, pero que
73. inicialmente están puestas a: false, indicando el estado inicial
74. Cuando se pulse cualquiera de las dos flechas, su valor será:
75. true (verdadero), y este valor permitira establecer en qué
76. dirección se debe mover la paleta (dentro del ciclo del juego)
77. Las variables son:
79. flechaDerechaPulsada
80. flechaIzquierdaPulsada
82. NOTA: Desde ahora debe tomarse en cuenta que cuando se pulse
83. cualquiera de las dos flechas, solamente se hará un
84. desplazamiento de la paleta a la izquierda o hacia la
85. derecha. Si se mantiene pulsada la tecla, la paleta se
86. continuará desplazando, hasta alcanzar el extremo derecho
87. o izquierdo de la pantalla del juego
88. \*/
90. **var** alturaPaleta = 10; // <---------------
91. **var** anchuraPaleta = 75; // <---------------
92. **var** paletaPosX = (canvas.width - anchuraPaleta ) / 2; // <------------------
94. **var** flechaDerechaPulsada = **false**; // <-----------------
95. **var** flechaIzquierdaPulsada = **false**; // <-----------------
97. /\* La instruccion: addEventListener, se utiliza para crear un
98. mecanismo de respuesta ante eventos que se produzcan en el juego
100. addEventListener "agregar un mecanismo que deteca y recibe eventos"
102. addEventListener recibe tres parámetros:
104. 1) El evento que se va a detectar
105. 2) El nombre que le asignamos a la función que responde ante el evento
106. 3) Valor true o false que determina la reacción ante el evento
108. Los dos primeros parámetros son fáciles de entender. Pero el tercero
109. requiere de una explicación adicional:
111. Para entender el tercer parámetro, primero hemos de saber lo que es
112. el flujo de eventos.
114. Supongamos que tenemos este tres objetos en la página:
116. <body>
117. <div>
118. <button>HAZME CLIC</button>
119. </div>
120. </body>
122. El <body> contiene un <div>, y dentro de él esta un <button>
124. Cuando hacemos clic en el botón no sólo lo estamos haciendo sobre él,
125. sino sobre los elementos que lo contienen en el árbol de la página,
126. es decir, hemos hecho clic, además, sobre el elemento <body> y sobre
127. el elemento <div>. Sí sólo hay una función asignada a una escucha
128. para el botón, no hay mayor problema, pero si además hay una
129. eschucha para el body y otra para el div,
130. ¿cuál es el orden en que se deben lanzar las tres funciones?
132. Para contestar a esa pregunta existe un modelo de comportamiento,
133. el flujo de eventos. Según éste, cuando se hace clic sobre un
134. elemento, el evento se propaga en dos fases, una que es la
135. captura —el evento comienza en el nivel superior del documento
136. y recorre los elementos de padres a hijos— y la otra la burbuja
137. —el orden inverso, ascendiendo de hijos a padres—.
139. Así, el orden por defecto de lanzamiento de las funciones
140. de escucha, sería: primero la función de escuch de body,
141. luego la función de escucha de div, y por último la función
142. de escucha de button.
144. Una vez visto esto, podemos comprender el tercer parámetro de addEventListener, que lo que hace es permitirnos escoger el orden de propagación:
146. true: El orden de propagación para el ejemplo sería, por tanto,
147. body-div-button
149. false: La propagación seguiría el modelo burbuja.
150. Así, el orden sería button-div-body.
152. NOTA: omo en nuestro ejemplo utilizamos "false", estamos
153. eaccionando primero ante el evento sobre las teclas,
154. posteriormente sobre los eventos asociados al CANVAS.
155. ste es el mecanismo más usual, pero se utilizará "true"
156. n las situaciones que lo requieran
157. \*/
158. document.addEventListener("keydown", manejadorTeclaPresionada, **false**); // <---------
159. document.addEventListener("keyup", manejadorTeclaLiberada, **false**); // <--------
161. // Función que maneja tecla presionada
162. **function** manejadorTeclaPresionada(e) { // <----------------
163. **if**(e.keyCode == 39) {
164. /\* e: Es el evento que se produce, en este caso
165. tecla presionada. La propiedad: keyCode permite
166. descubrir de qué tecla se trata. Si el código es 39,
167. se ha presionado la flecha derecha. En este caso
168. se coloca la variable: flechaDerechaPulsada a true
169. \*/
170. flechaDerechaPulsada = **true**;
171. }
172. **else** **if**(e.keyCode == 37) {
173. /\* e: Es el evento que se produce, en este caso
174. tecla presionada. La propiedad: keyCode permite
175. descubrir de qué tecla se trata. Si el código es 37,
176. se ha presionado la flecha izquierda. En este caso
177. se coloca la variable: flechaIquierdaPulsada a true
178. \*/
179. flechaIzquierdaPulsada = **true**;
180. }
181. }
183. // Función que maneja tecla liberada
184. **function** manejadorTeclaLiberada(e) { // <--------------------
185. **if**(e.keyCode == 39) {
186. /\* Si la tecla liberada es la 39, se ha dejado de
187. presionar la flecha derecha. En este caso, la variable
188. se pone en: false
189. \*/
190. flechaDerechaPulsada = **false**;
191. }
192. **else** **if**(e.keyCode == 37) { // <-----------------
193. /\* Si la tecla liberada es la 37, se ha dejado de
194. presionar la flecha izquierda. En este caso, la variable
195. se pone en: false
196. \*/
197. flechaIzquierdaPulsada = **false**;
198. }
199. }
201. // Dibuja la bola. Código explicado en anteriores programas
202. **function** dibujarBola() {
203. ctx.beginPath();
204. ctx.arc(x, y, radioBola, 0, Math.PI\*2);
205. ctx.fillStyle = "#00FFFB";
206. ctx.fill();
207. ctx.closePath();
208. }
210. **function** dibujarPaleta() { // <--------------------
211. // Se inicia el dibujo de la paleta
212. ctx.beginPath();
213. /\* Se crea un rectángulo utilizando la posición en X
214. El valor de Y está en la base de la pantalla menos la
215. altura de la paleta
216. Y a continuación se indica la anchura y la altura de la paleta
217. \*/
218. ctx.rect(paletaPosX,alturaPaleta/canvas.height, anchuraPaleta, alturaPaleta);
219. ctx.fillStyle = "#FF0000";
220. ctx.fill();
221. // Se "cierra" la paleta, terminando su dibujo
222. ctx.closePath();
223. }
225. // Función principal. A partir de aquí se origina el proceso
226. // general del juego
227. **function** dibujar() {
228. ctx.clearRect(0, 0, canvas.width, canvas.height);
230. // En primer lugar, dibuja la bola
231. dibujarBola();
233. // Seguidamente, dibuja la paleta
234. dibujarPaleta(); // <--------------------
236. /\* IMPORTANTE:
238. EL OPERADOR || es el operador lógico OR
239. Este operador se utiliza para indicar la condición de conjunción
240. SI SE CUMPLE UNA CONDICIÓN, O SE CUMPLE OTRA CONDICIÓN, ENTONCES
241. SE CUMPLE LA CONDICIÓN
243. EL OPERADOR && es el oerador lógico AND
244. Este operador se utiliza para indicar la condicion de disyunción
245. SI SE CUMPLE UNA CONDICIÓN, Y SE CUMPLE OTRA CONDICIÓN (simultánea), ENTONCES
246. SE CUMPLE LA CONDICIÓN
248. \*/
250. // Aquí se controla los límites a los que puede llegar la bola
251. // En caso de intentar sobrepasar dichos límites, se cambia
252. // el sentido del movimiento
253. // Este código se explicó en el anterior programa
254. **if**(x + dx > canvas.width-radioBola || x + dx < radioBola) { // <--------------
255. dx = -dx;
256. }
257. **if**(y + dy > canvas.height-radioBola || y + dy < radioBola) { // <--------------
258. dy = -dy;
259. }
261. /\* Si se ha pulsado la flecha derecha, y la paleta aún puede
262. desplazarse hacia la derecha sin que se sobrepase el límite de la
263. pantalla, entonces se procede a cambiar su posición
264. En este caso, la función: dibujarPaleta (la cual se ejecuta de
265. manera cíclica) redibujará la paleta en la nueva posición
266. \*/
267. **if**(flechaDerechaPulsada && paletaPosX < canvas.width-anchuraPaleta) { // <--------
268. // Se desplaza la paleta hacia la derecha
269. // Aquí, paletaPosX += 7 equivale a: paletaPosX = paletaPosX + 7
270. paletaPosX += 7;
271. }
272. **else** **if**(flechaIzquierdaPulsada && paletaPosX > 0) { // <------------
273. // Se desplaza la paleta hacia la izquierda
274. // Aquí, paletaPosX -= 7 equivale a: paletaPosX = paletaPosX - 7
275. paletaPosX -= 7;
276. }
278. x += dx;
279. y += dy;
280. }
282. /\* Con esta instrucción se crea un ciclo. Cada 10 milisegundos se
283. ejecuta la funcion: dibujar(). Esto genera el ciclo que permitirá
284. actualizar el juego, detectar eventos y cambiar el estado
285. de los objetos según las nuevas posiciones que ocupen los
286. elementos del juego
288. NOTA: La función que se ejecuta es: dibujar
289. Por tanto, dicha función es la encargada de "lanzar" el juego
290. y dentro de ella se realizarán las acciones que desencadenan
291. el juego como tal
292. \*/
293. setInterval(dibujar, 18);
294. </script>
296. </body>
297. </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.

#### [https://developer.mozilla.org/es/docs/Games/Workflows/Famoso\_juego\_2D\_usando\_JavaScript\_](https://developer.mozilla.org/es/docs/Games/Workflows/Famoso_juego_2D_usando_JavaScript_puro/Construye_grupo_bloques)

#### [puro/Construye\_grupo\_bloques](https://developer.mozilla.org/es/docs/Games/Workflows/Famoso_juego_2D_usando_JavaScript_puro/Construye_grupo_bloques)