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

UTP | Pereira

**<nombre estudiante>**

septiembre DE 2020

2020

# CONTENIDO

[1 CONTENIDO 1](#_Toc42134755)

[2 PRESENTACIÓN 2](#_Toc42134756)

[3 FASE 1: Dibujar y mover una bola 3](#_Toc42134757)

[4 FASE 2: Rebotando en las paredes 6](#_Toc42134758)

[5 FASE 3: Control de la pala y el teclado 9](#_Toc42134759)

[6 FASE 4: Fin del juego 18](#_Toc42134760)

[7 FASE 5: Muro de ladrillos 22](#_Toc42134761)

[8 FASE 6: Detección de colisiones 28](#_Toc42134762)

[9 FASE 7: Contar puntos y ganar 33](#_Toc42134763)

[10 FASE 8: Controlando el ratón 39](#_Toc42134764)

[11 FASE 9: Finalizando el juego 45](#_Toc42134765)

[12 CONCLUSIONES 51](#_Toc42134766)

[13 BIBLIOGRAFÍA 52](#_Toc42134767)

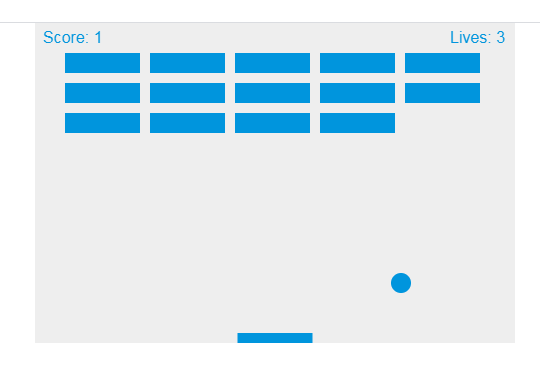
# PRESENTACIÓN

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: <NOMBRE ESTUDIANTE>**

# 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>

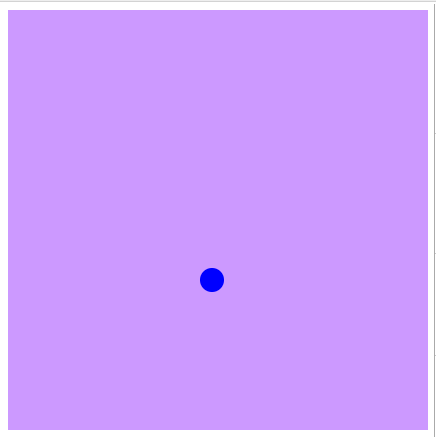
La correcta visualización del CANVAS requiere de la adición de algunas características de estilo. Una vez hecho esto, se procede a establecer la codificación pertinente del JavaScript. Debe notarse la inclusión de algunas variables que definen la funcionalidad del juego en sus aspectos básicos: las coordenadas 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>)

Al ejecutar este código se obtiene la siguiente interfaz visual:

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8" />
5. <title>Juego 2D: JavaScript - 01</title>
7. <!-- [Define](http://www.php.net/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: *#CC99Ff;*
20. display: block;
21. margin: 0 auto;
22. }
23. </style>
25. </head>
26. <body>
28. <canvas id="miCanvas" width="420" height="420"></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** x = canvas.width/5;
37. // Coloca y en la mitad de la altura del CANVAS (restando 30 a dicho valor)
38. **var** y = 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 = 30;
43. **var** dy = -30;
45. **function** dibujarBola() {
46. // Inicia el dibujo
47. ctx.beginPath();
49. /\* Define un círculo en las coordenadas (x, y) con radio 12
50. El ángulo va desde 0 hasta 2\*PI (360 grados) \*/
51. ctx.arc(x, y, 12, 0, Math.[PI](http://www.php.net/pi)\*2);
53. // Color de llenado
54. ctx.fillStyle = "#0000FF";
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, 500);
87. **</script>**
89. </body>
90. </html>



**Link juego 1**

<http://tpcg.io/JH8AuqLM>

**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.

# 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 >
3. < html >



8. < cabeza >

11. < meta charset = "utf-8" />



16. < título > Juego 2D - lección 02 </ título >



21. < estilo >

24. \* { relleno : 0 ; margen : 0 ; }

27. lienzo { fondo : *# CC99Ff; bloqueo de pantalla; margen: 0 automático; }*

30. </ estilo >


34. </ cabeza >






42. < cuerpo >



47. < canvas id = "miCanvas" width = "420" height = "420" > </ canvas >



52. < guión >



57. **var** lienzo = documento . getElementById ( "miCanvas" ) ;
59. **var** ctx = lienzo . getContext ( "2d" ) ;



64. / \* Se agrega la variable ballRadius, la cual define el tamaño de la bola \* /
65. **var** ballRadius = 11 ; // <--------------------
67. **var** x = lienzo . ancho / 2 ;
69. **var** y = lienzo . altura - 30 ;


73. **var** dx = 2 ;
75. **var** dy = - 2 ;






83. **function** dibujarBola ( ) {

86. ctx . beginPath ( ) ;


90. / \* En lugar de un número fijo, se coloca la variable ballRadius \* /
91. ctx . arco ( x , y , ballRadius , 0 , Math . [PI](http://www.php.net/pi) \* 2 ) ; // <----------------
93. ctx . fillStyle = "# 0000FF" ;
95. ctx . llenar ( ) ;


99. ctx . closePath ( ) ;



104. }





111. **function** dibujar ( ) {

114. ctx . clearRect ( 0 , 0 , lienzo . ancho , lienzo . alto ) ;

117. dibujarBola ( ) ;
119. / \* IMPORTANTE:
121. EL OPERADOR || es el operador lógico O
122. Este operador se utiliza para indicar la condición de conjunción
123. SI SE CUMPLE UNA CONDICIÓN, O SE CUMPLE OTRA CONDICIÓN, ENTONCES
124. SE CUMPLE LA CONDICIÓN
126. EL OPERADOR && es el oerador lógico AND
127. Este operador se utiliza para indicar la condición de disyunción
128. SI SE CUMPLE UNA CONDICIÓN, Y SE CUMPLE OTRA CONDICIÓN (simultánea), ENTONCES
129. SE CUMPLE LA CONDICIÓN
131. \* /
133. / \* DESPUÉS DE DIBUJAR LA BOLA, SE DEBEN CAMBIAR LAS COORDENADAS
134. EN LA lección 01 NO SE TENÍA CONTROL SOBRE LOS LÍMITES DE LA CAJA
135. -------------------------------------------------- ---------------
136. SI x + dx ES MAYOR AL ANCHO DEL CANVAS O MENOR AL TAMAÑO DEL
137. RADIO DE LA BOLA (caso en el cual se encuentra hacia la izquierda)
138. SE CAMBIA LA DIRECCIÓN DE AVANCE HORIZONTAL.
139. ESTO SE LOGRA CAMBIANDO EL SIGNO DE LA VARIABLE dx
140. ESTO HACE QUE SE CAMBIE EL SENTIDO DEL MOVIMIENTO HORIZONTAL \* /
141. if ( x + dx > lienzo . ancho - ballRadius || x + dx < ballRadius ) { // <-----------------
142. dx = dx \* ( - 1 ) ;
143. }
145. / \* SI y + dy ES MAYOR A LA ALTURA DEL CANVAS O MENOR AL TAMAÑO DEL
146. RADIO DE LA BOLA, SE CAMBIA LA DIRECCIÓN DEL AVANCE VERTICAL.
147. ESTO SE LOGRA CAMBIANDO EL SIGNO DE LA VARIABLE dy
148. ESTE CAMBIO EN dy HACE QUE SE MUEVA VERTICALMENTE EN SENTIDO
149. OPUESTO \* /
150. si ( x + dx > lona . altura - ballRadius || y + dy < ballRadius ) { // <-----------------
151. dy = dy \* ( - 1 ) ;
152. }
154. / \* AQUÍ SE CAMBIA LA POSICIÓN DE LA BOLA. SE TOMA EN CUENTA LAS
155. MODIFICACIONES A dy y dx, EN CASO DE QUE SE HUBIERAN PRODUCIDO \* /
157. x = x + dx ; // EQUIVALE A: x = x + dx; <----------------
158. y = y + dy ; // EQUIVALE A: y = y + dy; <----------------
159. }




165. setInterval ( dibujar , 5 ) ;





172. **</script>**

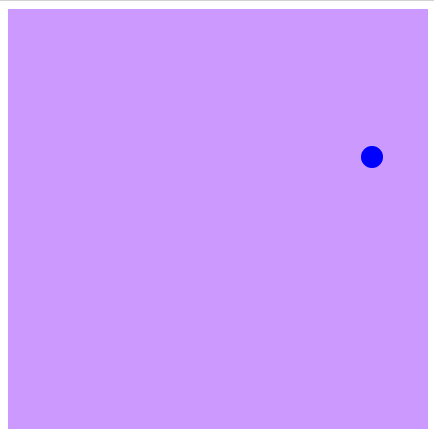


177. </ cuerpo >



182. </ html >

Al ejecutar este código se obtiene la siguiente interfaz visual:



**Link juego 2**

<http://tpcg.io/d6E87QSe>

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.

# 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

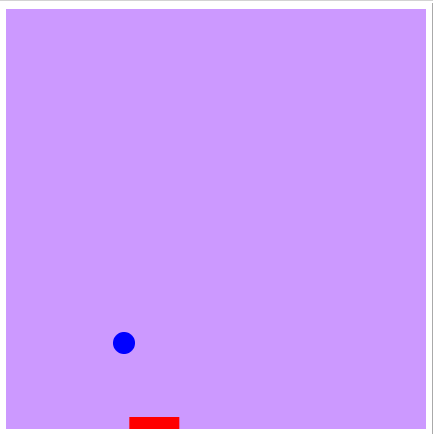
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8" />
5. <title>Juego 2D - *#03 - Paleta y Control por Teclado</title>*
6. <style>
7. \* { padding: 0; margin: 0; }
8. canvas { background: *#CC99Ff; display: block; margin: 0 auto; }*
9. </style>
10. </head>
11. <body>
13. <canvas id="miCanvas" width="420" height="420"></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 = 11; // <-----------------
47. **var** x = canvas.width/2;
48. **var** y = canvas.height-30;
49. **var** dx = 2;
50. **var** dy = -2;
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 = 12; // <---------------
91. **var** anchuraPaleta = 50; // <---------------
92. **var** paletaPosX = (canvas.width - anchuraPaleta) / 3; // <------------------
94. **var** letraAPulsada = **false**; // <-----------------
95. **var** letraBPulsada = **false**; // <-----------------

98. document.addEventListener("keydown", manejadorTeclaPresionada, **false**); // <---------
99. document.addEventListener("keyup", manejadorTeclaLiberada, **false**); // <--------
101. // Función que maneja tecla presionada
102. **function** manejadorTeclaPresionada(e) { // <----------------
103. if(e.keyCode == 68) {
104. /\* e: Es el evento que se produce, en este caso
105. tecla presionada. La propiedad: keyCode permite
106. descubrir de qué tecla se trata. Si el código es 39,
107. se ha presionado la flecha derecha. En este caso
108. se coloca la variable: flechaDerechaPulsada a true
109. \*/
110. letraAPulsada = **true**;
111. }
112. else if(e.keyCode == 65) {
113. /\* e: Es el evento que se produce, en este caso
114. tecla presionada. La propiedad: keyCode permite
115. descubrir de qué tecla se trata. Si el código es 37,
116. se ha presionado la flecha izquierda. En este caso
117. se coloca la variable: flechaIquierdaPulsada a true
118. \*/
119. letraBPulsada = **true**;
120. }
121. }
123. // Función que maneja tecla liberada
124. **function** manejadorTeclaLiberada(e) { // <--------------------
125. if(e.keyCode == 68) {
126. /\* Si la tecla liberada es la 39, se ha dejado de
127. presionar la flecha derecha. En este caso, la variable
128. se pone en: false
129. \*/
130. letraAPulsada = **false**;
131. }
132. else if(e.keyCode == 65) { // <-----------------
133. /\* Si la tecla liberada es la 37, se ha dejado de
134. presionar la flecha izquierda. En este caso, la variable
135. se pone en: false
136. \*/
137. letraBPulsada = **false**;
138. }
139. }
141. // Dibuja la bola. Código explicado en anteriores programas
142. **function** dibujarBola() {
143. ctx.beginPath();
144. ctx.arc(x, y, radioBola, 0, Math.[PI](http://www.php.net/pi)\*2);
145. ctx.fillStyle = "#0000FF";
146. ctx.fill();
147. ctx.closePath();
148. }
150. **function** dibujarPaleta() { // <--------------------
151. // Se inicia el dibujo de la paleta
152. ctx.beginPath();
154. ctx.rect(paletaPosX, canvas.height-alturaPaleta, anchuraPaleta, alturaPaleta);
155. ctx.fillStyle = "#FF0000";
156. ctx.fill();
157. // Se "cierra" la paleta, terminando su dibujo
158. ctx.closePath();
159. }
161. // Función principal. A partir de aquí se origina el proceso
162. // general del juego
163. **function** dibujar() {
164. ctx.clearRect(0, 0, canvas.width, canvas.height);
166. // En primer lugar, dibuja la bola
167. dibujarBola();
169. // Seguidamente, dibuja la paleta
170. dibujarPaleta(); // <--------------------

173. if(x + dx > canvas.width-radioBola || x + dx < radioBola) { // <--------------
174. dx = -dx;
175. }
176. if(y + dy > canvas.height-radioBola || y + dy < radioBola) { // <--------------
177. dy = -dy;
178. }
180. /\* Si se ha pulsado la flecha derecha, y la paleta aún puede
181. desplazarse hacia la derecha sin que se sobrepase el límite de la
182. pantalla, entonces se procede a cambiar su posición
183. En este caso, la función: dibujarPaleta (la cual se ejecuta de
184. manera cíclica) redibujará la paleta en la nueva posición
185. \*/
186. if(letraAPulsada && paletaPosX < canvas.width-anchuraPaleta) { // <--------
187. // Se desplaza la paleta hacia la derecha
188. // Aquí, paletaPosX += 7 equivale a: paletaPosX = paletaPosX + 7
189. paletaPosX += 7;
190. }
191. else if(letraBPulsada && paletaPosX > 0) { // <------------
192. // Se desplaza la paleta hacia la izquierda
193. // Aquí, paletaPosX -= 7 equivale a: paletaPosX = paletaPosX - 7
194. paletaPosX -= 7;
195. }
197. x += dx;
198. y += dy;
199. }


203. setInterval(dibujar, 5);
204. **</script>**
206. </body>
207. </html>

Al ejecutar este código se obtiene la siguiente interfaz visual:



**Link juego 3**

<http://tpcg.io/CVsoy0bn>

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.

# 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 codigo seria: (y + dy > canvas.height - 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 codigo: (x > paletaPosX && x < paletaPosX + 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**](http://december.com/html/4/element/html.html)>
3. <[**head**](http://december.com/html/4/element/head.html)>
4. <[**meta**](http://december.com/html/4/element/meta.html) charset="utf-8" />
5. <[**title**](http://december.com/html/4/element/title.html)>Juego 2D - #04 - Game Over</[**title**](http://december.com/html/4/element/title.html)>
6. <[**style**](http://december.com/html/4/element/style.html)>\* { padding: 0; margin: 0; } canvas { background: #eee; display: block; margin: 0 auto; }</[**style**](http://december.com/html/4/element/style.html)>
7. </[**head**](http://december.com/html/4/element/head.html)>
8. <[**body**](http://december.com/html/4/element/body.html)>
10. <canvas id="miCanvas" width="480" height="320"></canvas>
12. <[**script**](http://december.com/html/4/element/script.html)>
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. \*/
21. var canvas = document.getElementById("miCanvas");
22. var ctx = canvas.getContext("2d");
24. var radioBola = 10;
25. var x = canvas.width/2;
26. var y = canvas.height-30;
27. var dx = 2;
28. var dy = -2;
30. var alturaPaleta = 10;
31. var anchuraPaleta = 75;
32. var paletaPosX = (canvas.width-anchuraPaleta)/2;
34. var flechaDerechaPresionada = false;
35. var flechaIzquierdaPresionada = false;
37. document.addEventListener("keydown", manejadorTeclaPresionada, false);
38. document.addEventListener("keyup", manejadorTeclaLiberada, false);
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. }
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. }
72. function dibujar() {
73. ctx.clearRect(0, 0, canvas.width, canvas.height);
75. dibujarBola();
76. dibujarPaleta();
78. if(x + dx > canvas.width-radioBola || x + dx < radioBola) {
79. dx = -dx;
80. }
81. if(y + dy < radioBola) {
82. dy = -dy;
83. }
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. }
111. if(flechaDerechaPresionada && paletaPosX < canvas.width-anchuraPaleta) {
112. paletaPosX += 7;
113. }
114. else if(flechaIzquierdaPresionada && paletaPosX > 0) {
115. paletaPosX -= 7;
116. }
118. x += dx;
119. y += dy;
120. }
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**](http://december.com/html/4/element/script.html)>
129. </[**body**](http://december.com/html/4/element/body.html)>
130. </[**html**](http://december.com/html/4/element/html.html)>

Al ejecutar este código se obtiene la siguiente interfaz visual:



**Link juego 4**

<http://tpcg.io/c8Sdgng2>

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.

# FASE 5: Muro de ladrillos

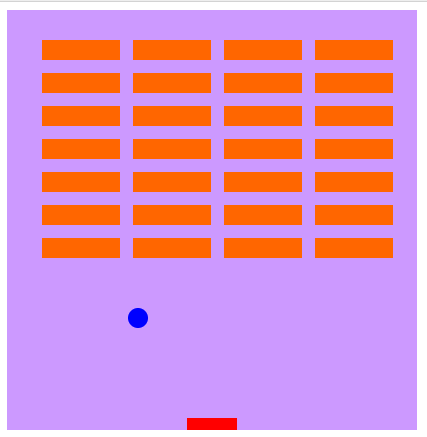
En esta parte del juego crearemos unas variables las cuales crearan un muro de ladrillos dentro del juego en los cuales rebotara la bola.

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: *#CC99Ff; display: block; margin: 0 auto; }</style>*
7. </head>
8. <body>
10. <canvas id="miCanvas" width="410" height="420"></canvas>
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 = 12;
21. **var** anchuraPaleta = 50;
22. **var** paletaPosX = (canvas.width-anchuraPaleta)/2;
23. **var** letraAPresionada = **false**;
24. **var** letraDPresionada = **false**;
26. /\* NUEVAS VARIABLES asociadas a los ladrillos
27. \*/
28. **var** nroFilasLadrillos = 4;
29. **var** nroColumnasLadrillos = 7;
30. **var** anchoLadrillo = 78;
31. **var** alturaLadrillo = 20;
32. **var** rellenoLadrillo = 13;
33. **var** vacioSuperiorLadrillo = 30;
34. **var** vacioIzquierdoLadrillo = 35;
36. // Crea el conjunto de ladrillos. Inicialmente, vacío
37. **var** ladrillos = [];
39. // Recorre cinco columnas
40. for(**var** columna=0; columna<nroColumnasLadrillos; columna++) {
41. // Define la primera columna. Es una lista vertical
42. ladrillos[columna] = [];
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

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:
56. x: Su coordenada horizontal EN LA PANTALLA
57. y: Su coordenada vertical EN LA PANTALLA
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. }
67. document.addEventListener("keydown", manejadorTeclaPresionada, **false**);
68. document.addEventListener("keyup", manejadorTeclaLiberada, **false**);
70. **function** manejadorTeclaPresionada(e) {
71. if(e.keyCode == 68) {
72. letraAPresionada = **true**;
73. }
74. else if(e.keyCode == 65) {
75. letraDPresionada = **true**;
76. }
77. }
78. **function** manejadorTeclaLiberada(e) {
79. if(e.keyCode == 68) {
80. letraAPresionada = **false**;
81. }
82. else if(e.keyCode == 65) {
83. letraBPresionada = **false**;
84. }
85. }
87. **function** dibujarBola() {
88. ctx.beginPath();
89. ctx.arc(x, y, radioBola, 0, Math.[PI](http://www.php.net/pi)\*2);
90. ctx.fillStyle = "#0000FF";
91. ctx.fill();
92. ctx.closePath();
93. }
95. **function** dibujarPaleta() {
96. ctx.beginPath();
97. ctx.rect(paletaPosX, canvas.height-alturaPaleta, anchuraPaleta, alturaPaleta);
98. ctx.fillStyle = "#FF0000";
99. ctx.fill();
100. ctx.closePath();
101. }
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** brickX = (fila\*(anchoLadrillo+rellenoLadrillo))+vacioIzquierdoLadrillo;
117. // Repite el proceso para calcular la coordenada y del ladrillo
118. **var** brickY = (columna\*(alturaLadrillo+rellenoLadrillo))+vacioSuperiorLadrillo;
120. // ASIGNA AL LADRILLO EN LA columna, fila QUE LE CORRESPONDE EN LA MATRIZ
121. // EL VALOR CALCULADO (brickX) A SU COORDENADA x
122. ladrillos[columna][fila].x = brickX;
124. // IGUAL PARA EL VALOR y EN PANTALLA
125. ladrillos[columna][fila].y = brickY;
127. // DIBUJA EL LADRILLO CON LOS VALORES ASOCIADOS:
128. // Coordenada: (brickX, brickY)
129. // Anchura: anchoLadrillo
130. // Altrua: alturaLadrillo
131. ctx.beginPath();
132. ctx.rect(brickX, brickY, anchoLadrillo, alturaLadrillo);
133. ctx.fillStyle = "#FF6600";
134. ctx.fill();
135. ctx.closePath();
137. // COMO SE RECORRE TODO EL CICLO, SE DIBUJAN TODOS LOS LADRILLOS
138. }
139. }
140. }
142. **function** dibujar() {
143. ctx.clearRect(0, 0, canvas.width, canvas.height);
145. // DIBUJA EL CONJUNTO DE LADRILLOS
146. dibujarLadrillos();
148. dibujarBola();
149. dibujarPaleta();
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("JUEGO TERMINADO :( ");
165. // RECARGA LA PÁGINA - El juego vuelve a empezar
166. document.location.reload();
167. }
168. }
170. if(letraAPresionada && paletaPosX < canvas.width-anchuraPaleta) {
171. paletaPosX += 7;
172. }
173. else if(letraDPresionada && paletaPosX > 0) {
174. paletaPosX -= 7;
175. }
177. x += dx;
178. y += dy;
179. }
181. **var** juego = setInterval(dibujar, 8);
182. **</script>**
184. </body>
185. </html>

Al ejecutar este código se obtiene la siguiente interfaz visual:



**Link juego 5**

<http://tpcg.io/YwanWPs6>

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.

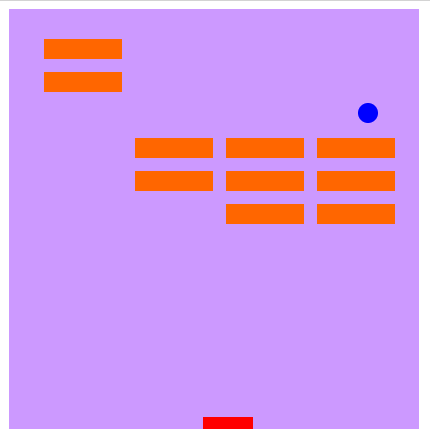
# FASE 6: Detección de colisiones

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 hara 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 asi saber si fue impactado.

1. <! DOCTYPE html >
2. < html >
3. < cabeza >
4. < meta charset = "utf-8" />
5. < title > Juego 2D - *# 05 - Detección de colisión </title>*
6. < estilo > \* { relleno : 0 ; margen : 0 ; } lienzo { fondo : *# CC99Ff; bloqueo de pantalla; margen: 0 automático; } </style>*
7. </ cabeza >
8. < cuerpo >
10. < canvas id = "miCanvas" width = "410" height = "420" > </ canvas >
12. < guión >
13. **var** lienzo = documento . getElementById ( "miCanvas" ) ;
14. **var** ctx = lienzo . getContext ( "2d" ) ;
16. **var** radioBola = 10 ;
17. **var** x = lienzo . ancho / 2 ;
18. **var** y = lienzo . altura - 30 ;
19. **var** dx = 2 ;
20. **var** dy = - 2 ;
22. **var** alturaPaleta = 12 ;
23. **var** anchuraPaleta = 50 ;
24. **var** paletaPosX = ( lienzo . ancho - anchuraPaleta ) / 2 ;
26. **var** letraAPresionada = **false**;
27. **var** letraDPresionada = **false**;
29. **var** nroFilasLadrillos = 4 ;
30. **var** nroColumnasLadrillos = 7 ;
31. **var** anchuraLadrillo = 78 ;
32. **var** alturaLadrillo = 20 ;
33. **var** rellenoLadrillo = 13 ;
34. **var** vacioSuperiorLadrillo = 30 ;
35. **var** vacioIzquierdoLadrillo = 35 ;
37. **var** ladrillos = [ ] ;
38. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
39. ladrillos [ c ] = [ ] ;
40. para ( **var** f = 0 ; f < nroFilasLadrillos ; f ++ ) {
42. / \* IMPORTANTE:
43. Como se puede observar, cada ladrillo está definido como: ==> ladrillos [c] [f]
44. Los valores cyf, se corresponda con la fila y la columna, DENTRO
45. DE LA MATRIZ DE LADRILLOS
46. -------------------------------------------------- ---------------------------
47. A cada ladrillo en la posición (c, f), se le asignan tres valores:
49. x: Su coordenada horizontal EN LA PANTALLA
50. y: Su coordenada vertical EN LA PANTALLA
51. estado: Indica si está visible o invisible. 1 = Visible, 0 = INVISIBLE
53. Inicialmente el ladrillo debe esta visible. Si la bola "toca" al ladrillo,
54. el ladrillo se debe volver INVISIBLE (status = 0)
55. -------------------------------------------------- ---------------------------
56. Los valores xyy valen originalmente cero (0)
57. Esto cambia cuando se dibujan (más adelante, en la función: dibujarLadrillos ())
58. \* /
59. ladrillos [ c ] [ f ] = { x : 0 , y : 0 , estado : 1 } ;
60. }
61. }
63. documento . addEventListener ( "keydown" , manejadorTeclaPresionada , **false**) ;
64. documento . addEventListener ( "keyup" , manejadorTeclaLiberada , **false**) ;
66. **function** manejadorTeclaPresionada ( e ) {
67. if ( e . keyCode == 68 ) {
68. letraAPresionada = **verdadero**;
69. }
70. else if ( p . ej . keyCode == 65 ) {
71. letraDPresionada = **verdadero**;
72. }
73. }
75. **función** manejadorTeclaLiberada ( e ) {
76. if ( e . keyCode == 68 ) {
77. letraAPresionada = **false**;
78. }
79. else if ( p . ej . keyCode == 65 ) {
80. letraDPresionada = **false**;
81. }
82. }
84. // EN ESTA FUNCIÓN SE DETECTA LA COLISIÓN DE LA BOLA CON EL LADRILLO
86. **function** deteccionColision ( ) {
88. // LOS DOS CICLOS SIGUIENTES RECORREN TODOS LOS LADRILLOS
89. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
90. para ( **var** f = 0 ; f < nroFilasLadrillos ; f ++ ) {
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 ] ;
96. // SI EL LADRILLO ES VISIBLE, se debe verificar si entra en contacto con la bola
97. si ( b . estado == 1 ) {
99. / \* SI LAS COORDENADAS xyy, 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. \* /
107. if ( x > b . x && x < b . x + anchuraLadrillo && y > b . y && y < b . y + alturaLadrillo ) {
108. dy = - dy ;
109. b . estado = 0 ;
110. }
111. }
112. }
113. }
114. }
116. **function** dibujarBola ( ) {
117. ctx . beginPath ( ) ;
118. ctx . arco ( x , y , radioBola , 0 , Math . [PI](http://www.php.net/pi) \* 2 ) ;
119. ctx . fillStyle = "# 0000FF" ;
120. ctx . llenar ( ) ;
121. ctx . closePath ( ) ;
122. }
124. **function** dibujarPaleta ( ) {
125. ctx . beginPath ( ) ;
126. ctx . rect ( paletaPosX , lienzo . altura - alturaPaleta , anchuraPaleta , alturaPaleta ) ;
127. ctx . fillStyle = "# FF0000" ;
128. ctx . llenar ( ) ;
129. ctx . closePath ( ) ;
130. }
132. **function** dibujarLadrillos ( ) {
133. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
134. para ( **var** f = 0 ; f < nroFilasLadrillos ; f ++ ) {
136. / \* IMPORTANTE:
138. Solamente se dibujan los ladrillos que están VISIBLES
139. Se sabe que el ladrillo es visible cuando: status == 1
140. Los ladrillos INVISIBLES NO SE DIBUJAN
142. \* /
144. if ( ladrillos [ c ] [ f ] . status == 1 ) {
146. // SE DIBUJA EL LADRILLO
147. **var** brickX = ( f \* ( anchuraLadrillo + rellenoLadrillo ) ) + vacioIzquierdoLadrillo ;
148. **var** brickY = ( c \* ( alturaLadrillo + rellenoLadrillo ) ) + vacioSuperiorLadrillo ;
149. ladrillos [ c ] [ f ] . x = ladrilloX ;
150. ladrillos [ c ] [ f ] . y = ladrilloY ;
151. ctx . beginPath ( ) ;
152. ctx . rect ( brickX , brickY , anchuraLadrillo , alturaLadrillo ) ;
153. ctx . fillStyle = "# FF6600" ;
154. ctx . llenar ( ) ;
155. ctx . closePath ( ) ;
156. }
157. }
158. }
159. }
161. **function** dibujar ( ) {
162. ctx . clearRect ( 0 , 0 , lienzo . ancho , lienzo . alto ) ;
163. dibujarLadrillos ( ) ;
164. dibujarBola ( ) ;
165. dibujarPaleta ( ) ;
166. deteccionColision ( ) ;
168. if ( x + dx > lienzo . ancho - radioBola || x + dx < radioBola ) {
169. dx = - dx ;
170. }
171. if ( y + dy < radioBola ) {
172. dy = - dy ;
173. }
174. else if ( y + dy > lienzo . altura - 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 ( "Juego terminado :(" ) ;
183. // Recarga la página, para iniciar de nuevo el juego
184. documento . ubicación . recargar ( ) ;
185. }
186. }
188. if ( letraAPresionada && paletaPosX < lienzo . ancho - anchuraPaleta ) {
189. paletaPosX + = 7 ;
190. }
191. else if ( letraDPresionada && paletaPosX > 0 ) {
192. paletaPosX - = 7 ;
193. }
195. x + = dx ;
196. y + = dy ;
197. }
199. **var** juego = setInterval ( dibujar , 8 ) ;
200. **</script>**
202. </ cuerpo >
203. </ html >

Al ejecutar este código se obtiene la siguiente interfaz visual:



**Link juego 6**

<http://tpcg.io/kFkJtN7T>

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.

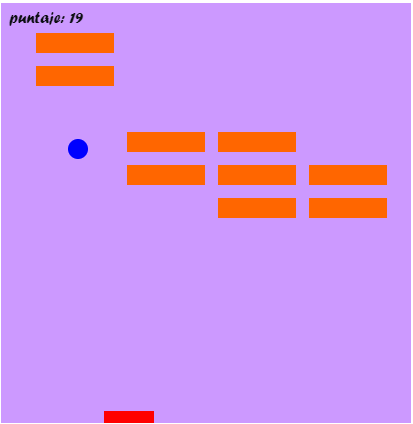
# FASE 7: Contar puntos y ganar

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. < cabeza >
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. < estilo > \* { relleno : 0 ; margen : 0 ; } lienzo { fondo : *# CC99Ff; bloqueo de pantalla; margen: 0 automático; } </style>*
10. </ cabeza >
11. < cuerpo >
13. < canvas id = "miCanvas" width = "410" height = "420" > </ canvas >
15. < guión >
16. **var** lienzo = documento . getElementById ( "miCanvas" ) ;
17. **var** ctx = lienzo . getContext ( "2d" ) ;
19. **var** radioBola = 10 ;
20. **var** x = lienzo . ancho / 2 ;
21. **var** y = lienzo . altura - 30 ;
22. **var** dx = 2 ;
23. **var** dy = - 2 ;
24. **var** alturaPaleta = 12 ;
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 = 50 ;
30. **var** paletaPosX = ( lienzo . ancho - anchuraPaleta ) / 2 ;
31. **var** letraAPresionada = **false**;
32. **var** letraDPresionada = **false**;
34. **var** nroFilasLadrillos = 4 ;
35. **var** nroColumnasLadrillos = 7 ;
36. **var** anchuraLadrillos = 78 ;
37. **var** alturaLadrillos = 20 ;
38. **var** rellenoLadrillos = 13 ;
39. **var** vacioSuperiorLadrillo = 30 ;
40. **var** vacioIzquierdoLadrillo = 35 ;
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 ;
47. **var** ladrillos = [ ] ;
48. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
49. ladrillos [ c ] = [ ] ;
50. para ( **var** f = 0 ; f < nroFilasLadrillos ; f ++ ) {
51. ladrillos [ c ] [ f ] = { x : 0 , y : 0 , estado : 1 } ;
52. }
53. }
55. documento . addEventListener ( "keydown" , manejadorTeclaPresionada , **false**) ;
56. documento . addEventListener ( "keyup" , manejadorTeclaLiberada , **false**) ;
58. **function** manejadorTeclaPresionada ( e ) {
59. if ( e . keyCode == 68 ) {
60. letraAPresionada = **verdadero**;
61. }
62. else if ( p . ej . keyCode == 65 ) {
63. letraDPresionada = **verdadero**;
64. }
65. }
66. **función** manejadorTeclaLiberada ( e ) {
67. if ( e . keyCode == 68 ) {
68. letraAPresionada = **false**;
69. }
70. else if ( p . ej . keyCode == 65 ) {
71. letraDPresionada = **false**;
72. }
73. }
74. **función** detectarColisión ( ) {
75. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
76. para ( **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 && y > b . y && y < b . y + alturaLadrillos ) {
80. dy = - dy ;
81. b . estado = 0 ;
83. // LA INSTRUCCIÓN puntaje ++ EQUIVALE A: puntaje = puntaje + 1
84. // ------------------------------------------------ ----------
85. // EN ESTE PUNTO DEL CÓDIGO LA BOLA HA IMPACTADO UN LADRILLO
86. // POR ESTE MOTIVO, SE INCREMENTA EL VALOR DE puntaje
87. // Si el puntaje es igual al número total de ladrillos (valor que
88. // se obtiene multiplicando el número de filas de ladrillos por el
89. // número de columnas de ladrillos), entonces el jugador ha ganado
90. puntaje ++;
91. if ( puntaje == nroFilasLadrillos \* nroColumnasLadrillos ) {
92. alert ( "felicidades ha ganado :)" ) ;
93. documento . ubicación . recargar ( ) ;
94. }
96. }
97. }
98. }
99. }
100. }
102. **function** dibujarBola ( ) {
103. ctx . beginPath ( ) ;
104. ctx . arco ( x , y , radioBola , 0 , Math . [PI](http://www.php.net/pi) \* 2 ) ;
105. ctx . fillStyle = "# 0000FF" ;
106. ctx . llenar ( ) ;
107. ctx . closePath ( ) ;
108. }
110. **function** dibujarPaleta ( ) {
111. ctx . beginPath ( ) ;
112. ctx . rect ( paletaPosX , lienzo . altura - alturaPaleta , anchuraPaleta , alturaPaleta ) ;
113. ctx . fillStyle = "# FF0000" ;
114. ctx . llenar ( ) ;
115. ctx . closePath ( ) ;
116. }
118. **function** dibujarLadrillos ( ) {
119. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
120. para ( **var** r = 0 ; r < nroFilasLadrillos ; r ++ ) {
121. if ( ladrillos [ c ] [ r ] . estado == 1 ) {
122. **var** posXLadrillo = ( r \* ( anchuraLadrillos + rellenoLadrillos ) ) + vacioIzquierdoLadrillo ;
123. **var** posYLadrillo = ( c \* ( alturaLadrillos + rellenoLadrillos ) ) + vacioSuperiorLadrillo ;
124. ladrillos [ c ] [ r ] . x = posXLadrillo ;
125. ladrillos [ c ] [ r ] . y = posYLadrillo ;
126. ctx . beginPath ( ) ;
127. ctx . rect ( posXLadrillo , posYLadrillo , anchuraLadrillos , alturaLadrillos ) ;
128. ctx . fillStyle = "# FF6600" ;
129. ctx . llenar ( ) ;
130. ctx . closePath ( ) ;
131. }
132. }
133. }
134. }
136. **function** dibujarPuntaje ( ) {
137. ctx . font = "16px FORTE" ;
138. ctx . fillStyle = "# 0D0E13" ;
139. ctx . fillText ( "puntaje:" + puntaje , 8 , 20 ) ;
140. }
142. **function** dibujar ( ) {
143. ctx . clearRect ( 0 , 0 , lienzo . ancho , lienzo . alto ) ;
144. dibujarLadrillos ( ) ;
145. dibujarBola ( ) ;
146. dibujarPaleta ( ) ;
147. dibujarPuntaje ( ) ;
148. detectarColision ( ) ;
150. if ( x + dx > lienzo . ancho - radioBola || x + dx < radioBola ) {
151. dx = - dx ;
152. }
153. if ( y + dy < radioBola ) {
154. dy = - dy ;
155. }
156. else if ( y + dy > lienzo . altura - radioBola ) {
157. if ( x > paletaPosX && x < paletaPosX + anchuraPaleta ) {
158. dy = - dy ;
159. }
160. else {
161. clearInterval ( juego ) ;
162. alert ( "JUEGO TERMINADO :(" ) ;
163. documento . ubicación . recargar ( ) ;
164. }
165. }
167. if ( letraAPresionada && paletaPosX < lienzo . ancho - anchuraPaleta ) {
168. paletaPosX + = 7 ;
169. }
170. else if ( letraDPresionada && paletaPosX > 0 ) {
171. paletaPosX - = 7 ;
172. }
174. x + = dx ;
175. y + = dy ;
176. }
178. **var** juego = setInterval ( dibujar , 8 ) ;
179. **</script>**
181. </ cuerpo >
182. </ html >

Al ejecutar este código se obtiene la siguiente interfaz visual:



**Link juego 7**

<http://tpcg.io/MWRsH0eb>

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.

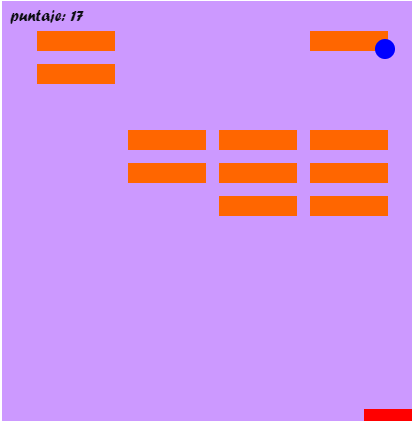
# FASE 8: Controlando el ratón

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. < cabeza >
4. < meta charset = "utf-8" />
5. < title > Juego 2D - *# 08 - Utilizando el ratón </title>*
6. < estilo > \* { relleno : 0 ; margen : 0 ; } lienzo { fondo : *# CC99Ff; bloqueo de pantalla; margen: 0 automático; } </style>*
7. </ cabeza >
8. < cuerpo >
10. < canvas id = "miCanvas" width = "410" height = "420" > </ canvas >
12. < guión >
13. **var** lienzo = documento . getElementById ( "miCanvas" ) ;
14. **var** ctx = lienzo . getContext ( "2d" ) ;
16. **var** radioBola = 10 ;
17. **var** x = lienzo . ancho / 2 ;
18. **var** y = lienzo . altura - 30 ;
19. **var** dx = 2 ;
20. **var** dy = - 2 ;
22. **var** alturaPaleta = 12 ;
23. **var** anchuraPaleta = 50 ;
24. **var** paletaPosX = ( lienzo . ancho - anchuraPaleta ) / 2 ;
26. **var** letraAPresionada = **false**;
27. **var** letraDPresionada = **false**;
29. **var** nroFilasLadrillos = 4 ;
30. **var** nroColumnasLadrillos = 7 ;
31. **var** anchuraLadrillo = 78 ;
32. **var** alturaLadrillo = 20 ;
33. **var** rellenoLadrillo = 13 ;
34. **var** vacioSuperiorLadrillo = 30 ;
35. **var** vacioIzquierdoLadrillo = 35 ;
37. **var** puntaje = 0 ;
39. **var** ladrillos = [ ] ;
40. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
41. ladrillos [ c ] = [ ] ;
42. para ( **var** f = 0 ; f < nroFilasLadrillos ; f ++ ) {
43. ladrillos [ c ] [ f ] = { x : 0 , y : 0 , estado : 1 } ;
44. }
45. }
47. documento . addEventListener ( "keydown" , manejadorTeclaPresionada , **false**) ;
48. documento . addEventListener ( "keyup" , manejadorTeclaLiberada , **false**) ;
50. // PARA DETECTAR EL MOVIMIENTO DEL RATÓN, SE COLOCA UN ESCUCHADOR (oyente)
51. // AL EVENTO "mousemove"
52. documento . addEventListener ( "mousemove" , manejadorRaton , **false**) ;
54. **function** manejadorTeclaPresionada ( e ) {
55. if ( e . keyCode == 68 ) {
56. letraAPresionada = **verdadero**;
57. }
58. else if ( p . ej . keyCode == 65 ) {
59. letraDPresionada = **verdadero**;
60. }
61. }
63. **función** manejadorTeclaLiberada ( e ) {
64. if ( e . keyCode == 68 ) {
65. letraAPresionada = **false**;
66. }
67. else if ( p . ej . keyCode == 65 ) {
68. letraDPresionada = **false**;
69. }
70. }
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 - lienzo . offsetLeft ;
88. // EL SIGUIENTE if DETERMINA SI LA POSICIÓN X DEL RATÓN ESTÁ
89. // DENTRO DEL LONA
90. if ( posXRatonDentroDeCanvas > 0 && posXRatonDentroDeCanvas < lienzo . ancho ) {
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 POSICIPÓN 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. **función** detectarColisión ( ) {
105. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
106. para ( **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. alerta ( "FELICIDADES HA GANADO :)" ) ;
115. documento . ubicación . recargar ( ) ;
116. }
117. }
118. }
119. }
120. }
121. }
123. **function** dibujarBola ( ) {
124. ctx . beginPath ( ) ;
125. ctx . arco ( x , y , radioBola , 0 , Math . [PI](http://www.php.net/pi) \* 2 ) ;
126. ctx . fillStyle = "# 0000FF" ;
127. ctx . llenar ( ) ;
128. ctx . closePath ( ) ;
129. }
130. **function** dibujarPaleta ( ) {
131. ctx . beginPath ( ) ;
132. ctx . rect ( paletaPosX , lienzo . altura - alturaPaleta , anchuraPaleta , alturaPaleta ) ;
133. ctx . fillStyle = "# FF0000" ;
134. ctx . llenar ( ) ;
135. ctx . closePath ( ) ;
136. }
137. **function** dibujarLadrillos ( ) {
138. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
139. para ( **var** r = 0 ; r < nroFilasLadrillos ; r ++ ) {
140. if ( ladrillos [ c ] [ r ] . estado == 1 ) {
141. **var** brickX = ( r \* ( anchuraLadrillo + rellenoLadrillo ) ) + vacioIzquierdoLadrillo ;
142. **var** brickY = ( c \* ( alturaLadrillo + rellenoLadrillo ) ) + vacioSuperiorLadrillo ;
143. ladrillos [ c ] [ r ] . x = ladrilloX ;
144. ladrillos [ c ] [ r ] . y = ladrilloY ;
145. ctx . beginPath ( ) ;
146. ctx . rect ( brickX , brickY , anchuraLadrillo , alturaLadrillo ) ;
147. ctx . fillStyle = "# FF6600" ;
148. ctx . llenar ( ) ;
149. ctx . closePath ( ) ;
150. }
151. }
152. }
153. }
154. **function** dibujarPuntaje ( ) {
155. ctx . font = "16px FORTE" ;
156. ctx . fillStyle = "# 0D0E13" ;
157. ctx . fillText ( "puntaje:" + puntaje , 8 , 20 ) ;
158. }
160. **function** dibujar ( ) {
161. ctx . clearRect ( 0 , 0 , lienzo . ancho , lienzo . alto ) ;
162. dibujarLadrillos ( ) ;
163. dibujarBola ( ) ;
164. dibujarPaleta ( ) ;
165. dibujarPuntaje ( ) ;
166. detectarColision ( ) ;
168. if ( x + dx > lienzo . ancho - radioBola || x + dx < radioBola ) {
169. dx = - dx ;
170. }
171. if ( y + dy < radioBola ) {
172. dy = - dy ;
173. }
174. else if ( y + dy > lienzo . altura - radioBola ) {
175. if ( x > paletaPosX && x < paletaPosX + anchuraPaleta ) {
176. dy = - dy ;
177. }
178. else {
179. clearInterval ( juego ) ;
180. alerta ( "JUEGO TERMINADO :(" ) ;
181. documento . ubicación . recargar ( ) ;
182. }
183. }
185. if ( letraAPresionada && paletaPosX < lienzo . ancho - anchuraPaleta ) {
186. paletaPosX + = 7 ;
187. }
188. else if ( letraDPresionada && paletaPosX > 0 ) {
189. paletaPosX - = 7 ;
190. }
192. x + = dx ;
193. y + = dy ;
194. }
196. **var** juego = setInterval ( dibujar , 8 ) ;
197. **</script>**
199. </ cuerpo >
200. </ html >

Al ejecutar este codigo se obtiene la siguiente interfaz visual:



**Link juego 8**

<http://tpcg.io/IRFgfhpz>

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.

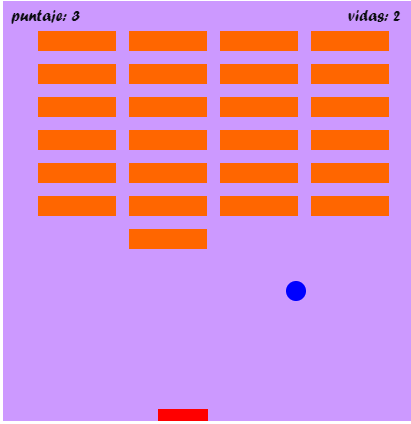
# FASE 9: Finalizando el juego

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. < cabeza >
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. < estilo > \* { relleno : 0 ; margen : 0 ; } lienzo { fondo : *# CC99Ff; bloqueo de pantalla; margen: 0 automático; \* {cursor: ninguno;}} </style>*
10. </ cabeza >
11. < cuerpo >
13. < canvas id = "miCanvas" width = "410" height = "420" > </ canvas >
15. < guión >
16. **var** lienzo = documento . getElementById ( "miCanvas" ) ;
17. **var** ctx = lienzo . getContext ( "2d" ) ;
19. **var** bolaRadio = 10 ;
20. **var** x = lienzo . ancho / 2 ;
21. **var** y = lienzo . altura - 30 ;
22. **var** dx = 2 ;
23. **var** dy = - 2 ;
25. **var** alturaPaleta = 12 ;
26. **var** anchuraPaleta = 50 ;
27. **var** paletaPosX = ( lienzo . ancho - anchuraPaleta ) / 2 ;
29. **var** letraAPresionada = **false**;
30. **var** letraDPresionada = **false**;
32. **var** nroFilasLadrillos = 4 ;
33. **var** nroColumnasLadrillos = 7 ;
34. **var** anchuraLadrillo = 78 ;
35. **var** alturaLadrillo = 20 ;
36. **var** rellenoLadrillo = 13 ;
37. **var** vacioSuperiorLadrillo = 30 ;
38. **var** vacioIzquierdoLadrillo = 35 ;
40. **var** puntaje = 0 ;
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 = 2 ;
47. // ESTA VARIABLE DEFINE UN COLOR
48. // Se pueden utilizar otros colores para los diferentes elementos del juego
49. **var** colorFigura = "# ff00" ;
50. **var** colorBola = "# 0000FF" ;
51. **var** colorPaleta = "# FF0000" ;
52. **var** colorLadrillo = "# FF6600" ;
53. **var** colorTexto = "# 0D0E13" ;
55. // ESTA INSTRUCCIÓN OCULTA EL CURSOR DEL RATON (DENTRO DEL CANVAS)
56. lienzo . estilo . cursor = 'ninguno' ;
58. **var** ladrillos = [ ] ;
59. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
60. ladrillos [ c ] = [ ] ;
61. para ( **var** f = 0 ; f < nroFilasLadrillos ; f ++ ) {
62. ladrillos [ c ] [ f ] = { x : 0 , y : 0 , estado : 1 } ;
63. }
64. }
66. documento . addEventListener ( "keydown" , manejadorTeclaPresionada , **false**) ;
67. documento . addEventListener ( "keyup" , manejadorTeclaLiberada , **false**) ;
68. documento . addEventListener ( "mousemove" , manejadorRaton , **false**) ;
70. **function** manejadorTeclaPresionada ( e ) {
71. if ( e . keyCode == 68 ) {
72. letraAPresionada = **verdadero**;
73. }
74. else if ( p . ej . keyCode == 65 ) {
75. letraDPresionada = **verdadero**;
76. }
77. }
79. **función** manejadorTeclaLiberada ( e ) {
80. if ( e . keyCode == 68 ) {
81. letraAPresionada = **false**;
82. }
83. else if ( p . ej . keyCode == 65 ) {
84. letraDPresionada = **false**;
85. }
86. }
88. **function** manejadorRaton ( e ) {
89. **var** posXRatonDentroDeCanvas = e . clientX - lienzo . offsetLeft ;
90. if ( posXRatonDentroDeCanvas > 0 && posXRatonDentroDeCanvas < lienzo . ancho ) {
91. paletaPosX = posXRatonDentroDeCanvas - anchuraPaleta / 2 ;
92. }
93. }
95. **función** detectarColisión ( ) {
96. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
97. para ( **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. alerta ( "FELICIDADES GANO :)" ) ;
106. documento . ubicación . recargar ( ) ;
107. }
108. }
109. }
110. }
111. }
112. }
114. **function** dibujarBola ( ) {
115. ctx . beginPath ( ) ;
116. ctx . arc ( x , y , bolaRadio , 0 , Math . [PI](http://www.php.net/pi) \* 2 ) ;
117. // SE UTILIZA EL COLOR PREVIAMENTE DEFINIDO
118. ctx . fillStyle = colorBola ;
119. ctx . llenar ( ) ;
120. ctx . closePath ( ) ;
121. }
122. **function** dibujarPaleta ( ) {
123. ctx . beginPath ( ) ;
124. ctx . rect ( paletaPosX , lienzo . altura - alturaPaleta , anchuraPaleta , alturaPaleta ) ;
125. ctx . fillStyle = colorPaleta ;
126. ctx . llenar ( ) ;
127. ctx . closePath ( ) ;
128. }
129. **function** dibujarLadrillos ( ) {
130. para ( **var** c = 0 ; c < nroColumnasLadrillos ; c ++ ) {
131. para ( **var** f = 0 ; f < nroFilasLadrillos ; f ++ ) {
132. if ( ladrillos [ c ] [ f ] . estado == 1 ) {
133. **var** ladrilloX = ( f \* ( anchuraLadrillo + rellenoLadrillo ) ) + vacioIzquierdoLadrillo ;
134. **var** ladrilloY = ( c \* ( alturaLadrillo + rellenoLadrillo ) ) + vacioSuperiorLadrillo ;
135. ladrillos [ c ] [ f ] . x = ladrilloX ;
136. ladrillos [ c ] [ f ] . y = ladrilloY ;
137. ctx . beginPath ( ) ;
138. ctx . rect ( ladrilloX , ladrilloY , anchuraLadrillo , alturaLadrillo ) ;
139. ctx . fillStyle = colorLadrillo ;
140. ctx . llenar ( ) ;
141. ctx . closePath ( ) ;
142. }
143. }
144. }
145. }
147. **function** dibujarPuntaje ( ) {
148. ctx . font = "16px forte" ;
149. ctx . fillStyle = colorTexto ;
150. ctx . fillText ( "puntaje:" + puntaje , 8 , 20 ) ;
151. }
153. **function** dibujarVidas ( ) {
154. ctx . font = "16px forte" ;
155. ctx . fillStyle = colorTexto ;
156. // SE MUESTRA EL NÚMERO DE VIDAS DISPONIBLES
157. ctx . fillText ( "vidas:" + vidas , lienzo . ancho - 65 , 20 ) ;
158. }
160. **function** dibujar ( ) {
161. ctx . clearRect ( 0 , 0 , lienzo . ancho , lienzo . alto ) ;
162. dibujarLadrillos ( ) ;
163. dibujarBola ( ) ;
164. dibujarPaleta ( ) ;
165. dibujarPuntaje ( ) ;
166. dibujarVidas ( ) ;
167. detectarColision ( ) ;
169. if ( x + dx > lienzo . ancho - bolaRadio || x + dx < bolaRadio ) {
170. dx = - dx ;
171. }
172. if ( y + dy < bolaRadio ) {
173. dy = - dy ;
174. }
175. else if ( y + dy > lienzo . altura - bolaRadio ) {
176. if ( x > paletaPosX && x < paletaPosX + anchuraPaleta ) {
177. dy = - dy ;
178. }
179. 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 ( "juego terminado :(" ) ;
188. documento . ubicación . recargar ( ) ;
189. }
190. else {
191. // SI vidas> 0 (diferente de CERO) EL JUEGO CONTINUA
192. x = lienzo . ancho / 2 ;
193. y = lienzo . altura - 30 ;
194. dx = 3 ;
195. dy = - 3 ;
196. paletaPosX = ( lienzo . ancho - anchuraPaleta ) / 2 ;
197. }
198. }
199. }
201. if ( letraAPresionada && paletaPosX < lienzo . ancho - anchuraPaleta ) {
202. paletaPosX + = 7 ;
203. }
204. else if ( letraDPresionada && paletaPosX > 0 ) {
205. paletaPosX - = 7 ;
206. }
208. x + = dx ;
209. y + = dy ;
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. }
219. dibujar ( ) ;
221. **</script>**
223. </ cuerpo >
224. </ html >

Al ejecutar este codigo se obtiene la siguiente interfaz visual:



**Link del juego 9**

<http://tpcg.io/9YuU3YZV>

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.

# 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 codigo html asignado a JavaSvript, 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 codigo 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 dia de hoy en el campo de la programación.

# BIBLIOGRAFÍA

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