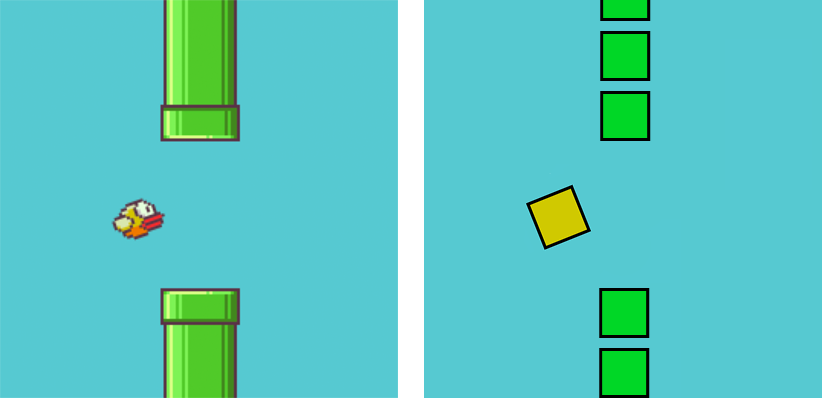
# **Flappy Bird en HTML5 con Phaser**



Vamos a hacer una versión simplificada de Flappy Bird en solo 65 líneas de Javascript con el marco Phaser.

Phaser es un marco gratuito, de código abierto y sorprendente para que los juegos se puedan jugar en cualquier navegador web.

Si quieres jugar el juego que vamos a construir, haz [clic aquí](http://www.lessmilk.com/game/flappy-bird/) . Debes presionar la barra espaciadora o tocar el juego para saltar.

## **Prepararación**

Para comenzar este tutorial, debe descargar [esta plantilla vacía](http://www.lessmilk.com/content/flappy-bird/empty.zip) que hice. En ella encontrarás:

* **phaser.min.js**, el marco Phaser v2.4.3.
* **index.html**, donde se mostrará el juego.
* **main.js**, un archivo donde escribiremos todo nuestro código.
* assets /, un directorio con 2 imágenes y un efecto de sonido.

## **Proyecto vacío**

Lo primero que vamos a hacer es construir un proyecto vacío.

Abra el archivo **index.html** y agregue este código.

|  |
| --- |
| <!DOCTYPE html>  <**html**>  <**head**>  <**meta** charset="utf-8" />  <**title**> Flappy Bird Clone </**title**>  <**script** type="text/javascript" src="phaser.min.js"></**script**>  <**script** type="text/javascript" src="main.js"></**script**>  </**head**>  <**body**>  </**body**>  </**html**> |

Simplemente cargará nuestros 2 archivos Javascript.

En el archivo **main.js** agregamos esto para crear un juego Phaser vacío.

|  |
| --- |
| // Esta es la base de todo juego que hagammos con phaser  var mainState = {  preload: function() {  // Esta función será ejecutada al inicio  // Esta es la función que carga las imagenes y sonidos  },  create: function() {  // TESta función es llamada despupes de la punción preload  // Esta función iniccializa las escenas, sprites y posiciones  },  update: function() {  // Esta función se llama 60 veces por segundio  // Contiene la lógica del juego  },  };  // Inicializamos Phaser, y creamos un juego de 400px by 490px  var game = new Phaser.Game(400, 490);  // Agregamos el 'mainState'y lo llamamos 'main'  game.state.add('main', mainState);  // Inicializamos el juego  game.state.start('main'); |

Todo lo que tenemos que hacer para hacer un juego con Phaser es llenar las funciones preload() , create() y update() .

## **El pájaro**

Primero centrémonos en agregar un pájaro al juego que saltará cuando presionemos la tecla de la barra espaciadora.

Todo es bastante simple y está bien comentado, por lo que debería poder entender el código a continuación. Para una mejor legibilidad, eliminé la inicialización de Phaser y el código de administración de estados que se puede ver arriba.

Primero actualizamos las funciones preload() , create() y update() .

|  |
| --- |
| preload: function() {  // carga la imagen del pajaro  game.load.image('bird', 'assets/bird.png');  },  create: function() {  // Cambia el color de fondo a azul  game.stage.backgroundColor = '#71c5cf';  // Seteo de las fisicas del juego  game.physics.startSystem(Phaser.Physics.ARCADE);  // Muestra al pajaro en la posición x=100 y y=245  this.bird = game.add.sprite(100, 245, 'bird');  // Agregamos la fisica al pajaro  // Se necesita la fisica para los movimientos, saltos,colisiones.  game.physics.arcade.enable(this.bird);  // Le agregamos gravedad al pajarro para hacerlo caer  this.bird.body.gravity.y = 1000;  // Llamamos a la funcion ´jump’ cuando se presiona la tecla SPPACEBAR  var spaceKey = game.input.keyboard.addKey(  Phaser.Keyboard.SPACEBAR);  spaceKey.onDown.add(this.jump, this);  },  update: function() {  // Si el pajaro estpa fuera de la pantalla (demmasiado bajo o alto)  // Llamamos a la función 'restartGame'}  if (this.bird.y < 0 || this.bird.y > 490)  this.restartGame();  },  // Función saltar  jump: function() {  // Agregado de velocidad vertical al pajaro  this.bird.body.velocity.y = -350;  },  // Restart del jeugo  restartGame: function() {  // Inicia el estado 'main', el cual resetea le juego  game.state.start('main');  }, |

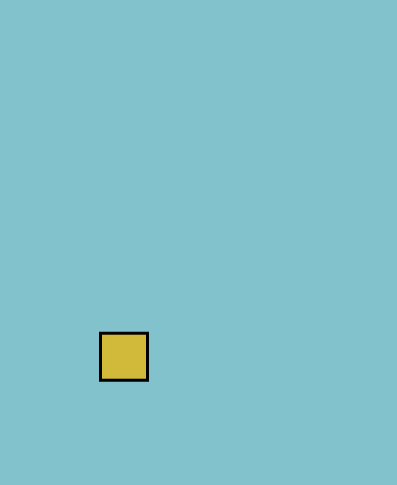
## **Testing**

Ejecutar un juego Phaser directamente en un navegador no funciona, eso se debe a que Javascript no puede cargar archivos desde su sistema de archivos local. Para resolver eso, tendremos que usar un servidor web para jugar y probar nuestro juego.

Hay muchas maneras de configurar un servidor web local en una computadora y vamos a cubrir rápidamente 3 a continuación.

* Use Brackets. Abra el directorio que contiene el juego en el editor Brackets y haga clic en el pequeño icono de perno que se encuentra en la esquina superior derecha de la ventana. Esto abrirá directamente su navegador con una vista previa en vivo de un servidor web. Esa es probablemente la solución más fácil.
* Usando aplicaciones. Puede descargar [WAMP](https://translate.googleusercontent.com/translate_c?depth=1&rurl=translate.google.com&sl=auto&sp=nmt4&tl=es&u=http://www.wampserver.com/en/&xid=17259,15700023,15700186,15700191,15700256,15700259,15700262,15700265,15700271&usg=ALkJrhjsNe0oLxW_AGSdM2_hI6LiJWQ6QQ) (Windows) o [MAMP](https://translate.googleusercontent.com/translate_c?depth=1&rurl=translate.google.com&sl=auto&sp=nmt4&tl=es&u=https://www.mamp.info/en/&xid=17259,15700023,15700186,15700191,15700256,15700259,15700262,15700265,15700271&usg=ALkJrhiDBY_c-2ehCZ8fsBUL8z0tf7cmAg) (Mac). Ambos tienen una interfaz de usuario limpia con guías de configuración simples disponibles.
* Usa la línea de comando. Si tiene instalado Python y está familiarizado con la línea de comandos, escriba python -m SimpleHTTPServer para tener un servidor web ejecutándose en el directorio actual. Luego usa la url 127.0.0.1:8000 para jugar.

Una vez hecho esto, deberías ver esto en tu pantalla.



## **Las pipas**

Un juego de Flappy Bird sin obstáculos (los tubos verdes) no es realmente interesante, así que cambiemos eso.

Primero, cargamos el sprite de la tubería en la función **preload()** .

|  |
| --- |
| game.load.image('pipe', 'assets/pipe.png'); |

Dado que vamos a manejar muchas tuberías en el juego, es más fácil usar una función de Phaser llamada "grupo". El grupo simplemente contendrá todas nuestras tuberías. Para crear el grupo, agregamos esto en la función **create()** .

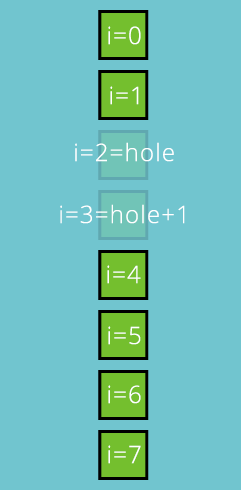
|  |
| --- |
| // Crreamos un grupo vacio  this.pipes = game.add.group(); |

Ahora necesitamos una nueva función para agregar una tubería en el juego. Podemos hacer eso con una nueva función.

|  |
| --- |
| addOnePipe: function(x, y) {  // Crea un caño en la posicion x e y  var pipe = game.add.sprite(x, y, 'pipe');  // Agregga el caño al grupo creado anteriormente  this.pipes.add(pipe);  // Habilita fisiccas a los caños  game.physics.arcade.enable(pipe);  // Agregamos velocidad a los caños para que se muuevan a la izquierda  pipe.body.velocity.x = -200;  // Automaticamente “matammos” al caño cuando ya no es visible  pipe.checkWorldBounds = true;  pipe.outOfBoundsKill = true;  }, |

La función anterior crea una tubería, pero necesitamos mostrar 6 tuberías en una fila con un agujero en algún lugar en el medio. Así que creemos una nueva función que haga exactamente eso.

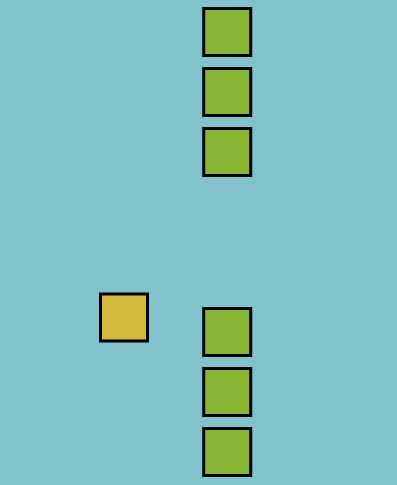
|  |
| --- |
| addRowOfPipes: function() {  // Elige in nummero random entre el 1 y el 5  // Este será la posición del agujero  var hole = Math.floor(Math.random() \* 5) + 1;  // Agregamos 6 caños  // Con un agujero gande en la posición 'hole' y 'hole + 1'  for (var i = 0; i < 8; i++)  if (i != hole && i != hole + 1)  this.addOnePipe(400, i \* 60 + 10);  }, |

Aquí hay una imagen para aclarar las cosas:  
  


Para agregar tuberías en nuestro juego, debemos llamar a la función addRowOfPipes() cada 1.5 segundos. Podemos hacer esto agregando un temporizador en la función create() .

|  |
| --- |
| this.timer = game.time.events.loop(1500, this.addRowOfPipes, this); |

Ahora puede guardar su archivo y probar el código. Poco a poco, esto comienza a parecer un juego real.



## **Puntuación y colisiones**

Necesitamos agregar un puntaje y manejar las colisiones.

Eesto es bastante fácil de hacer.

Agregamos esto en la función **create()** para mostrar el puntaje en la esquina superior izquierda.

|  |
| --- |
| this.score = 0;  this.labelScore = game.add.text(20, 20, "0",  { font: "30px Arial", fill: "#ffffff" }); |

Y en **addRowOfPipes()** ponemmos, para aumentar la puntuación en 1 cada vez que se crean nuevas canalizaciones.

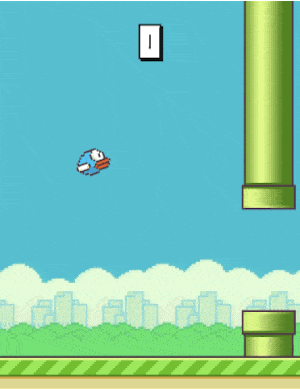
|  |
| --- |
| this.score += 1;  this.labelScore.text = this.score; |

A continuación, agregamos esta línea en la función **update()** para llamar a restartGame() cada vez que el pájaro colisiona con una tubería del grupo de pipes .

|  |
| --- |
| game.physics.arcade.overlap(  this.bird, this.pipes, this.restartGame, null, this); |

## **Agregar animación**

## El pájaro se mueve hacia arriba y hacia abajo de una manera bastante aburrida. Mejoremos eso agregando algunas animaciones como en el juego original.



Podes ver esto:

* El pájaro gira lentamente hacia abajo, hasta cierto punto.
* Y cuando el pájaro salta, gira hacia arriba.

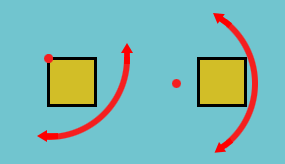
El primero giro es facil. Solo necesitamos agregar esto en la función **update()** .

|  |
| --- |
| if (this.bird.angle < 20)  this.bird.angle += 1; |

Para el salto, simplemente podríamos agregar this.bird.angle = -20; en la función jump() . Sin embargo, cambiar instantáneamente el ángulo se verá raro. En cambio, haremos que el pájaro cambie su ángulo en un corto período de tiempo. Podemos hacerlo creando una animación en la función **jump()**

|  |
| --- |
| // Creamos la animacion en el pajaro  var animation = game.add.tween(this.bird);  // Cambiamos el angulo del pajaro a -20° en 100 millisegundos  animation.to({angle: -20}, 100);  // Iniciamos la animación  animation.start(); |

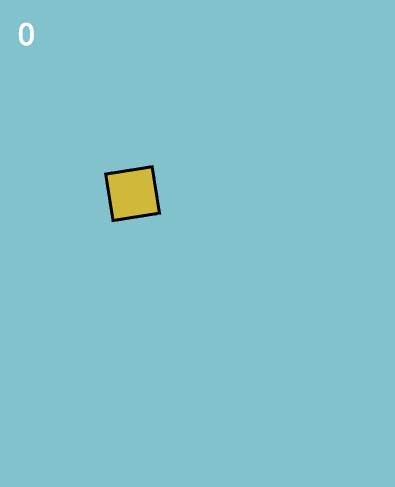
Si probas el juego en este momento, notarás que el pájaro no gira como el Flappy Bird original. Está girando como el dibujo de la izquierda, y queremos que se vea como el de la derecha.

****

Lo que debemos hacer es cambiar el centro de rotación del ave (el punto rojo arriba) llamado "ancla". Entonces agregamos esta línea de código en la función **create()** .

|  |
| --- |
| **this.bird.anchor.setTo(-0.2, 0.5);** |

Si probas el juego ahora, la animación debería verse mucho mejor.



## **Añadir animación de muerte**

Cuando el pájaro muere, reiniciamos el juego al instante.

Para darle más dramatismo , vamos a hacer que el pájaro se caiga de la pantalla.

Primero, actualizamos esta línea de código en la función **update()** para llamar a hitPipe() lugar de restartGame() cuando el pájaro golpea una tubería.

|  |
| --- |
| game.physics.arcade.overlap(  this.bird, this.pipes, this.hitPipe, null, this); |

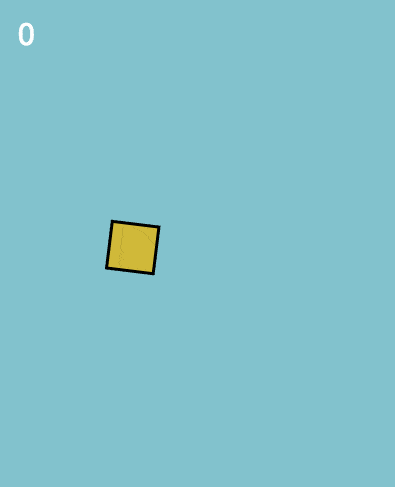
Ahora creamos la nueva función hitPipe() .

|  |
| --- |
| hitPipe: function() {  // Si el pajaro ya se golpeo, no puede haccer nada más  // Esto significa que el pajaro se tiene que caer de la pantalla  if (this.bird.alive == false)  return;  // SSeteamos el estado de vida de pajarro a falso  this.bird.alive = false;  // Hacemos que no aparezccan mpas ccaños  game.time.events.remove(this.timer);  // Hacemos que los caños dejen de moverse  this.pipes.forEach(function(p){  p.body.velocity.x = 0;  }, this);  }, |

Lo último, no queremos poder hacer que el pájaro salte cuando está muerto. Así que editamos la función **jump()** agregando estas 2 líneas al comienzo de la función.

|  |
| --- |
| if (this.bird.alive == false)  return; |

Y hemos terminado de agregar animaciones.



## **Añadir sonido**

Agregar sonidos es muy fácil con Phaser.

Comenzamos cargando el sonido de salto en la función **preload()** .

|  |
| --- |
| game.load.audio('jump', 'assets/jump.wav'); |

Ahora agregamos el sonido en el juego al poner esto en la función create() .

|  |
| --- |
| this.jumpSound = game.add.audio('jump'); |

Finalmente, agregamos esta línea en la función jump() para reproducir realmente el efecto de sonido.

|  |
| --- |
| this.jumpSound.play(); |

Y eso es todo, ¡ahora tenemos animaciones y sonidos en el juego!