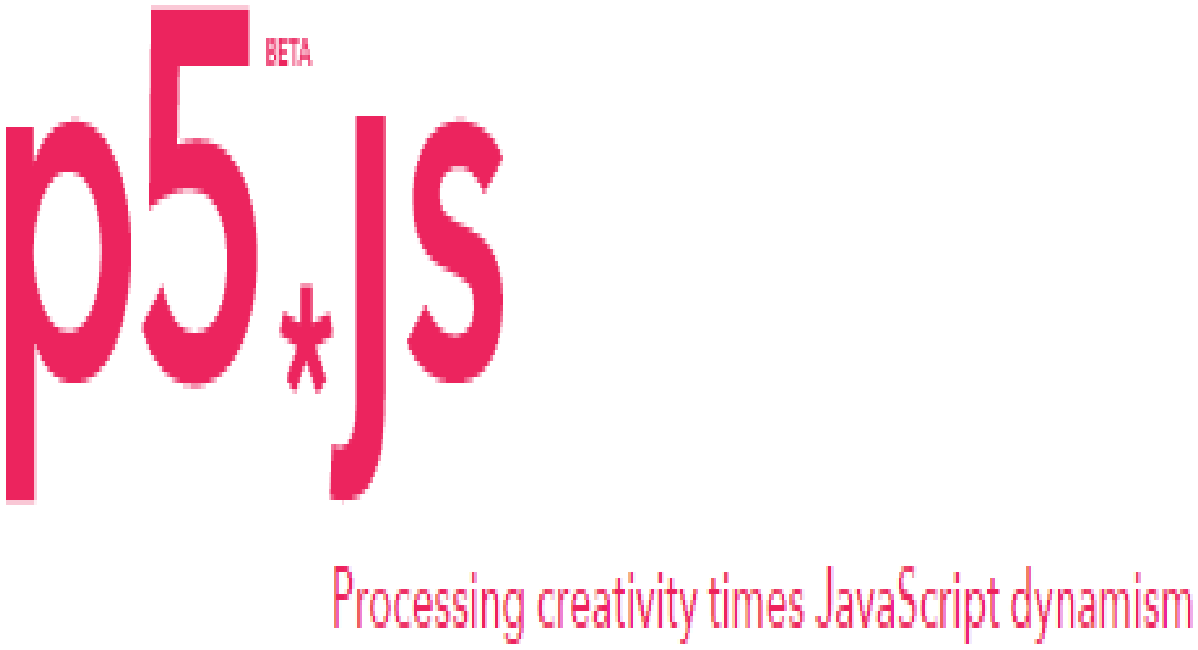

Introducción

La idea sera conocer la librería play.js con ejemplos sencillos y muy funcionales. Al final se compartirá un juego escrito usando la biblioteca y se explicara a modo raso el código y como se crearon las animaciones usando flash para generar los spritesheets. Si desea conocer el juego directamente, haga click aqui

Esta guia se hace en referencia a este gran tutorial de Allison Parrish:Link

Programación de juegos con play.js

En este tutorial, vamos a ver como funciona una biblioteca de la librería p5 llamada p5.play.js. Escrita por **Paolo Pedercini**p5-play

Esta va a ser una guía muy simple, con algunos ejemplos para entender como funciona a modo raza la librería. Por favor siempre este consultando la referencia para conocer mas a fondo la librería.

Play.p5.js es una librería que ofrece una serie de objetos y funciones útiles para escribir videojuegos y otras aplicaciones interactivas. Los objetos y funciones que introduce se incorporan en p5.js, es como si fuera una biblioteca del framework.

Instalación

En el sitio oficial esta un guía de referencia para instalar la biblioteca. Puede descargar la biblioteca haciendo click

aqui

; después de descomprimir el archivo, vaya al directorio libraries y copie el archivo *p5.play.js* y péguelo en la biblioteca libraries de su directorio. Hay que recalcar que en su propio vocero index.html, no olvide usar la etiqueta script para importar la biblioteca a su vocero. Si desea mas información al respecto, Aquí hay un gran tutorial de como importar esta librería a su proyecto.

Sprite

Para crear un elemento en el mundo de *p5.play* logramos esto usando la función **createSprite()**. Esta función devuelve un objeto Sprite, que a su vez posee una serie de atributos y métodos que nos permite consultar y modificar las propiedades del sprite.

Es muy importante que si desea usar los ejemplos aquí descritos, use un servidor local para correr los ejemplos.

Un ejemplo muy sencillo aquí para la creación de un sprite:

```
var forma;
function setup() {
  createCanvas(400, 400);
  forma = createSprite(
    width/2, height/2, 40, 40);
  forma.shapeColor = color(255);
  forma.velocity.y = 0.5;
}
function draw() {
  background(50);
  drawSprites();
}
function mousePressed() {
  forma.position.x = mouseX;
  forma.position.y = mouseY;
}
```

La función **createSprite()** para poderse crear toma cuatro parámetros. La posición en **X** y en **Y** ademas del **ancho** y el **alto**. El atributo **.shapeColor** es una función que recibe como parámetro un color, que en este caso establecerá el color de nuestro sprite, que por defecto siempre sera un cuadrado. Para que la librería p5.play pueda mostrar el sprite, tenemos que añadir la función drawSprites() antes del final de la función draw().

Cada objeto creado con **createSprite()**, posee ademas atributos como la **posición** y la **velocidad**. Ambos se pueden configurar para controlar o establecer la posición y la velocidad del sprite. La biblioteca *p5.play* se encarga de actualizar la velocidad o de preguntar la posición actual para nosotros. Y por ello no tendremos que preocuparnos de las matemáticas complejas que hay detrás de ese proceso.

En el ejemplo anterior el sprite se mueve constantemente hacia abajo. Su posición inicial siempre sera fijada a la posición en X y en Y del mouse cuando se hace click sobre el lienzo.

Moviendo los sprites

Como mencionamos anteriormente, podemos establecer una velocidad inicial o una posición inicial para nuestro sprite. Para establecer la velocidad directamente, referenciamos el atributo **.velocity.x** para la coordenada en **x** y **.velocity.y** para la coordenada en **y**. Si lo que queremos es indicar velocidades fijas a una dirección determinada, usamos la función **setSpeed()**. En el ejemplo que vamos a ver a continuación, desplazamos el sprite usando las teclas de nuestro teclado:

```
var forma;
function setup() {
  createCanvas(400, 400);
  forma = createSprite(
    width/2, height/3, 40, 40);
  forma.shapeColor = color(255);
}
function draw() {
  background(50);
  fill(255);
  noStroke();
  textAlign(CENTER, CENTER);
  text("use arrow keys, or SPACE to stop",
    width/2, height*0.67);
  drawSprites();
}
function keyPressed() {
  if (keyCode == RIGHT_ARROW) {
    forma.setSpeed(1.5, 0);
  }
  else if (keyCode == DOWN_ARROW) {
    forma.setSpeed(1.5, 90);
  }
  else if (keyCode == LEFT_ARROW) {
    forma.setSpeed(1.5, 180);
  }
  else if (keyCode == UP_ARROW) {
    forma.setSpeed(1.5, 270);
  }
  else if (key == ' ') {
    forma.setSpeed(0, 0);
  }
  return false;
}
```

La variable key en *p5.js* solo funciona para los caracteres alfanuméricos. Con el fin de detectar las teclas de las flechas, usamos la variable **KeyCode.** No olvide escribirla con K mayúscula al iniciar.

Para añadir gravedad al dibujo, vasta con usar la función **.setSpeed()** para añadir una fuerza

constante.

En las siguientes lineas de código hay un ejemplo que hace que un sprite se dibuje en la pantalla, para que se mueva hacia abajo en cada fotograba y luego rebota cuando llega a la parte inferior.

```
var forma;
function setup() {
  createCanvas(400, 400);
  forma = createSprite(width/2, height/2,
    40, 40);
  forma.shapeColor = color(255);
  forma.velocity.y = 0;
}
function draw() {
  background(50);
  if (forma.position.y >= height) {
    forma.velocity.y *= -1;
    // set to height to prevent "tunneling"
    forma.position.y = height;
  }
  // constant downward speed
  // (i.e., gravity)
  forma.addSpeed(0.25, 90);
  drawSprites();
}
function mousePressed() {
  forma.position.y = mouseY;
}
```

Siguiendo el mouse

Hay muchas formas de hacer que un sprite siga la posición del mouse. Inicialmente estableceremos la posición directamente.

```
var forma;
function setup() {
  createCanvas(400, 400);
  forma = createSprite(
    width/2, height/2, 40, 40);
  forma.shapeColor = color(255);
}
function draw() {
  background(50);
  forma.position.x = mouseX;
  forma.position.y = mouseY;
  drawSprites();
}
```

Podemos añadir un tipo de **easing** (retrazo) a la forma que esta siguiendo el mouse. Este retrazo se establece en los ejes X y Y, restando la posición del sprite y la posición del mouse.

```
var forma;
function setup() {
  createCanvas(400, 400);
  forma = createSprite(
    width/2, height/2, 40, 40);
  forma.shapeColor = color(255);
}
function draw() {
  background(50);
  forma.velocity.x = (mouseX - spr.position.x) * 0.2;
  forma.velocity.y = (mouseY - spr.position.y) * 0.2;
  drawSprites();
}
```

Por último, podemos usar el metodo **.attractionPoint()** para establecer una fuerza que empuja la forma a la dirección y la posición del ratón:

```
var forma;
function setup() {
  createCanvas(400, 400);
  forma = createSprite(
    width/2, height/2, 40, 40);
  forma.shapeColor = color(255);
  forma.rotateToDirection = true;
  forma.maxSpeed = 2;
  forma.friction = 0.99;
}
function draw() {
  background(50);
  if (mouseIsPressed) {
    forma.attractionPoint(0.5, mouseX, mouseY);
  }
  drawSprites();
}
```

En el ejemplo anterior, establecimos también algunos atributos como **.maxSpeed** del objeto. (Este controla la velocidad con la cual un sprite se mueve, independientemente de las fuerzas que operan en el). También el atributo **.friction** (El cual es un multiplicador que reduce lentamente la velocidad del objeto en cada frame) y finalmente el atributo .rotateToDirection (Si se declara en true inicialmente, hace que el objeto gire a la dirección en la cual se esta moviendo).

Eventos con el mouse

Los sprites dentro de p5.play vienen con un mecanismo incorporado para detectar si el usuario esta interactuando con el sprite usando el ratón. Por ello, hay dos formas de comprobar si esta interactuando con el ratón: callbacks o atributos booleanos.

Hay cuatro atributos que tiene un objeto sprite, que se asignan como funciones para definir el comportamiento del sprite en relación a si el usuario esta moviendo el mouse. En el siguiente ejemplo

se ilustran las 4.

```
var forma1;
var forma2;
function setup() {
  createCanvas(400, 400);

  forma1 = createSprite(width/2, height/3,
    100, 100);
  forma1.shapeColor = color(255);
  forma1.onMouseOver = function() {
    this.scale = 2;
  }
  forma1.onMouseOut = function() {
    this.scale = 1;
  }

  forma2 = createSprite(width/2, height*0.67,
    100, 100);
  forma2.shapeColor = color(0);
  forma2.onMousePressed = function() {
    this.shapeColor = color(128);
  }
  forma2.onMouseReleased = function() {
    this.shapeColor = color(0);
  }
}
function draw() {
  background(50);
  drawSprites();
}
```

Estos cuatro atributos son:

onMouseOver: Cuando el cursor se mueve sobre el objjeto.

onMouseOut: Cuando el mouse sale del objeto.

onMousePressed: Cuando el usuario presiona el botón el mouse, y el cursor del mouse esta encima del objeto.

onMouseReleased: Cuando el usuario suelta el boton del mouse, despues de un evento onMousePressed.

La función que se asigna a estos atributos se ejecuta cada vez que se produzca el evento especificado. Dentro de la función, la expresión se refiere *al objeto el cual llama la interacción*. Esto es útil para escribir controladores de eventos que se pueden aplicar a más de un objeto:

Cada **objeto Sprite** también tiene un atributo *MouseIsOver*, que tiene un valor booleano: **true** si el ratón está actualmente por encima del objeto, y **falso** en caso contrario. En el siguiente ejemplo, los dos sprites responden cuando el ratón está sobre ellos, pero solo el segundo sprite, reacciona si el mouse esta encima y se hace click sobre el también.

```
var forma1;
var forma2;
function setup() {
  createCanvas(400, 400);
  forma1 = createSprite(width/2, height/3,
    100, 100);
  forma1.shapeColor = color(255);
  forma1.mouseActive = true;
  forma2 = createSprite(width/2, height*0.67,
    100, 100);
  forma2.shapeColor = color(0);
  forma2.mouseActive = true;
}
function draw() {
  background(50);
  if (forma1.mouseIsOver) {
    background(100);
  }
  if (forma2.mouseIsOver && mouseIsPressed) {
    forma2.rotation += 4;
  }
  drawSprites();
}
```

Note que también en este ejemplo, usamos el atributo *.rotation*, que establece la rotación actual del sprite en grados.

Multiples sprites

Podemos llamar a la función *createSprite()* tantas veces como se desee!. El framerowk **play.p5** realiza un seguimiento de todos los sprites que ha añadido detrás de las escenas (por lo que no es necesario crear su propia estructura de datos para almacenarlos). En el siguiente ejemplo, se ha escrito algo de código con *mousePressed()*, el cual crea un nuevo sprite cada vez que el usuario hace clic con el ratón:

```
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(50);
  drawSprites();
}
function mousePressed() {
  var forma = createSprite(width/2, height/2,
    random(10, 50), random(10, 50));
  forma.shapeColor = color(255);
  forma.velocity.y = random(3);
  forma.velocity.x = random(-3, 3);
  forma.position.x = mouseX;
```

```
  forma.position.y = mouseY;
  forma.friction = 0.995;
  forma.life = 120;
}
```

Deténgase un momento a analizar el código y encontrara el uso del atributo *.life*, que es el **número máximo de fotogramas** que el sprite puede mostrarse en pantalla, antes de que sea eliminado automáticamente por la librería *p5.play*

Para **modificar los sprites** después de que estos son creados, por excepción de los cambios que realiza la librería **p5.play** por sí sola, usted necesita iterar sobre cada elemento sprite en el método *draw()*. El marco proporciona una matriz llamada **allSprites** incorporada que contiene todos los sprites activos en el boceto. En el ejemplo siguiente, se utiliza la variable *allSprites* para aplicar "gravedad". (ósea una fuerza descendente constante) para cada sprite añadido a la escena en mousepressed(). Otra sentencia if comprueba para ver si el sprite se ha ido mas haya de la altura del lienzo y hace que rebote. Otra sentencia if elimina cualquier *sprites* que se han excedido del limite del boceto en el eje X.

```
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(50);
  for (var i = 0; i < allSprites.length; i++) {
    // gravity
    allSprites[i].addSpeed(0.1, 90);
    if (allSprites[i].position.y > height) {
      allSprites[i].velocity.y *= -1;
    }
    // any code that removes sprites should be
    // the *last* thing in the loop!
    if (allSprites[i].position.x > width ||
        allSprites[i].position.x < 0) {
      allSprites[i].remove();
    }
  }
  textAlign(RIGHT, TOP);
  text("sprite count: " + allSprites.length,
    width-10, 10);
  drawSprites();
}
function mousePressed() {
  var forma = createSprite(width/2, height/2,
    random(10, 50), random(10, 50));
  forma.shapeColor = color(255);
  forma.velocity.y = random(3);
  forma.velocity.x = random(-3, 3);
  forma.position.x = mouseX;
  forma.position.y = mouseY;
}
```

De ahora en adelante solo mostraremos ejemplos

Eventos con multiples sprites

```
var score = 0;
function setup() {
  createCanvas(400, 400);
  for (var i = 0; i < 10; i++) {
    var forma = createSprite(
      random(width), random(height),
      random(10, 50), random(10, 50));
    forma.shapeColor = random(255);
    forma.onMouseOver = removeAndScore;
  }
}
function draw() {
  background(50);
  drawSprites();
  fill(255);
  noStroke();
  textSize(72);
  textAlign(CENTER, CENTER);
  if (score < 10) {
    text(score, width/2, height/2);
  }
  else {
    text("you win!", width/2, height/2);
  }
}
function removeAndScore() {
  score += 1;
  this.remove();
}
```

grupos de sprites - sprite group

```
var clouds;
var birds;
function setup() {
  createCanvas(400, 400);
  clouds = new Group();
  birds = new Group();

  for (var i = 0; i < 10; i++) {
    var c = createSprite(
      random(width), random(height),
      random(25, 100), random(25, 100));
    c.shapeColor = color(random(200, 255));
```

```
    clouds.add(c);
  }
  for (var i = 0; i < 5; i++) {
    var b = createSprite(
      random(width), random(height),
      random(10, 50), random(5, 25));
    b.shapeColor = color(255, 0, random(255));
    b.friction = random(0.97, 0.99);
    b.maxSpeed = random(1, 4);
    b.rotateToDirection = true;
    birds.add(b);
  }
}
function draw() {
  background(0, 150, 240);
  for (var i = 0; i < clouds.length; i++) {
    clouds[i].position.x += clouds[i].width * 0.01;
    if (clouds[i].position.x > width) {
      clouds[i].position.x = 0;
    }
  }
  for (var i = 0; i < birds.length; i++) {
    birds[i].attractionPoint(0.2, mouseX, mouseY);
  }
  drawSprites();
}
```

colisiones

Para mas información ir a la referencia de play.p5, en este apartado hay un ejemplo con colisiones.

```
var spr1;
var spr2;
function setup() {
  createCanvas(400, 400);
  spr1 = createSprite(
    width/2, height/2, 150, 150);
  spr1.shapeColor = color(0);
  spr2 = createSprite(0, 0, 50, 50);
  spr2.shapeColor = color(128);
}
function draw() {
  background(50);
  spr2.velocity.x = (mouseX-spr2.position.x)*0.2;
  spr2.velocity.y = (mouseY-spr2.position.y)*0.2;
  if (spr2.overlap(spr1)) {
    spr1.shapeColor = color(255);
  }
  else {
    spr1.shapeColor = color(0);
```

```
  }
  drawSprites();
}
```

Bloques

.collide

```
var spr1;
var spr2;
function setup() {
  createCanvas(400, 400);
  spr1 = createSprite(
    width/2, height/2, 100, 100);
  spr1.shapeColor = color(0);
  spr2 = createSprite(0, 0, 50, 50);
  spr2.shapeColor = color(128);
}
function draw() {
  background(50);
  spr2.velocity.x = (mouseX-spr2.position.x)*0.2;
  spr2.velocity.y = (mouseY-spr2.position.y)*0.2;
  spr2.collide(spr1);
  drawSprites();
}

<code>

.displace()

<code javascript>
var spr1;
var spr2;
function setup() {
  createCanvas(400, 400);
  spr1 = createSprite(
    width/2, height/2, 100, 100);
  spr1.shapeColor = color(0);
  spr2 = createSprite(0, 0, 50, 50);
  spr2.shapeColor = color(128);
}
function draw() {
  background(50);
  spr2.velocity.x = (mouseX-spr2.position.x)*0.2;
  spr2.velocity.y = (mouseY-spr2.position.y)*0.2;
  spr2.displace(spr1);
  drawSprites();
}
```

Grupo de colisiones

```
var walls;
var boxes;
var player;
function setup() {
  createCanvas(400, 400);
  walls = new Group();
  boxes = new Group();
  player = createSprite(100, 100, 40, 40);
  player.shapeColor = color(255);
  for (var i = 0; i < 5; i++) {
    var w = createSprite(
      random(125, width-125), (height/5)*i,
      random(10, 100), random(10, 100));
    w.shapeColor = color(0);
    walls.add(w);
  }
  for (var i = 0; i < 4; i++) {
    var b = createSprite(
      random(50, 100), random(100, height-100),
      25, 25);
    b.shapeColor = color(255, 0, 0);
    boxes.add(b);
  }
}
function draw() {
  background(50);
  player.velocity.x =
    (mouseX-player.position.x)*0.1;
  player.velocity.y =
    (mouseY-player.position.y)*0.1;
  player.collide(walls);
  player.displace(boxes);
  boxes.collide(walls);
  boxes.displace(boxes);
  drawSprites();
}
```

Grupo de callbacks

```
var coins;
var player;
var score = 0;
function setup() {
  createCanvas(400, 400);
  coins = new Group();
  for (var i = 0; i < 10; i++) {
    var c = createSprite(
```

```
      random(100, width-100),
      random(100, height-100),
      10, 10);
    c.shapeColor = color(255, 255, 0);
    coins.add(c);
  }
  player = createSprite(50, 50, 40, 40);
  player.shapeColor = color(255);
}
function draw() {
  background(50);
  player.velocity.x =
    (mouseX-player.position.x)*0.1;
  player.velocity.y =
    (mouseY-player.position.y)*0.1;
  player.overlap(coins, getCoin);
  drawSprites();
  fill(255);
  noStroke();
  textSize(72);
  textAlign(CENTER, CENTER);
  if (coins.length > 0) {
    text(score, width/2, height/2);
  }
  else {
    text("you win!", width/2, height/2);
  }
}
function getCoin(player, coin) {
  coin.remove();
  score += 1;
}
```