



Processing creativity times JavaScript dynamism

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](#)

aquí

; 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** además 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 además 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, basta con usar la función **.setSpeed()** para añadir una fuerza constante.

En las siguientes líneas de código hay un ejemplo que hace que un sprite se dibuje en la pantalla, para que se mueva hacia abajo en cada fotograma 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)

From:
<https://wiki.unloquer.org/> -

Permanent link:
https://wiki.unloquer.org/proyectos/nombre_proyecto_2?rev=1473868984

Last update: **2016/09/14 16:03**



