

Tutoriales
Diseño en
Medios
Interactivos

04

Gestos e interacción

- a.*** Arrastrar
- b.*** Velocidad del ratón
- c.*** Lanzar
- d.*** Clics por minuto
- e.*** Tamaño de la ventana
- f.*** Dentro del canvas

Un **gesto** es una forma de comunicación no verbal capaz de transmitir sentimientos, intenciones y pensamientos. ¿Cómo podemos inferir gestos a partir de acciones que realiza una persona con un medio digital?

En este tutorial se explican 6 ejemplos sencillos para leer diferentes **tipos de interacción** en un programa de p5.js. Estos ejemplos buscan inspirar y funcionar como **punto de partida** para exploraciones más complejas e interesantes.

El código de todos los ejemplos se puede ver y descargar en:
github.com/laurajunco/medios/tree/gh-pages/04_gestos

¿Cómo saber si una persona quiere arrastrar algún
componente o cómo permitir que estos puedan ser
arrastrados?

Se puede ver el código de este ejemplo en:

alpha.editor.p5js.org/laurajunco/sketches/SkozlcA0Z

a.

Arrastrar

1. Saber si el ratón se encuentra sobre el objeto

```
function mouseDragged() {  
    //recorre la lista de bolas  
    for (var i = 0; i < numBolas; i++) {  
        //revisa si el mouse esta encima de alguna bola  
        if (dist(mouseX, mouseY, bolas[i].x, bolas[i].y) < bolas[i].tam + 10) {  
            //llama a la funcion arrastrar  
            bolas[i].arrastrar();  
        }  
    }  
}
```

Si se tiene una lista de objetos se debe recorrer uno por uno con un **for()**

Utilizando el comando **dist()** dentro de una **sentencia condicional** se puede saber si el ratón se encuentra cercano al objeto que se va a arrastrar.

2. Saber si el ratón esta siendo arrastrado

```
function mouseDragged() {  
    //recorre la lista de bolas  
    for (var i = 0; i < numBolas; i++) {  
        //revisa si el mouse esta encima de alguna bola  
        if (dist(mouseX, mouseY, bolas[i].x, bolas[i].y) < bolas[i].tam + 10) {  
            //llama a la funcion arrastrar  
            bolas[i].arrastrar();  
        }  
    }  
}
```

La función **mouseDragged()** se activa cada vez que el ratón es arrastrado dentro de la pantalla.

Así, lo que se desee que ocurra cuando el ratón se arrastra debe ir dentro de esta función.

mouseDragged() solo se activa cuando el mouse se arrastra, una vez se deje de arrastrar el mouse todos los comportamientos dentro de la función **dejarán de suceder**.

3. Mover los objetos una vez se cumplan ambas condiciones

```
function mouseDragged() {  
  
    //recorre la lista de bolas  
    for (var i = 0; i < numBolas; i++) {  
  
        //revisa si el mouse esta encima de alguna bola  
        if (dist(mouseX, mouseY, bolas[i].x, bolas[i].y) < bolas[i].tam + 10) {  
  
            //llama a la funcion arrastrar  
            bolas[i].arrastrar();  
        }  
    }  
}
```

En este caso cuando el ratón esta sobre uno de los objetos y es arrastrado se llama a la función **arrastrar()** de la bola.

```
//cuando se activa esta funcion se cambia la posicion de la bola  
por la del mouse  
this.arrastrar = function() {  
    this.x = mouseX;  
    this.y = mouseY;  
}
```

Cuando la función `arrastrar()` es llamada el objeto **toma la posición del ratón.**

¿Cómo conocer la velocidad con la que la persona está
moviendo el ratón en la pantalla?

Se puede ver el código de este ejemplo en:

alpha.editor.p5js.org/laurajunco/sketches/H11vmq0Ab

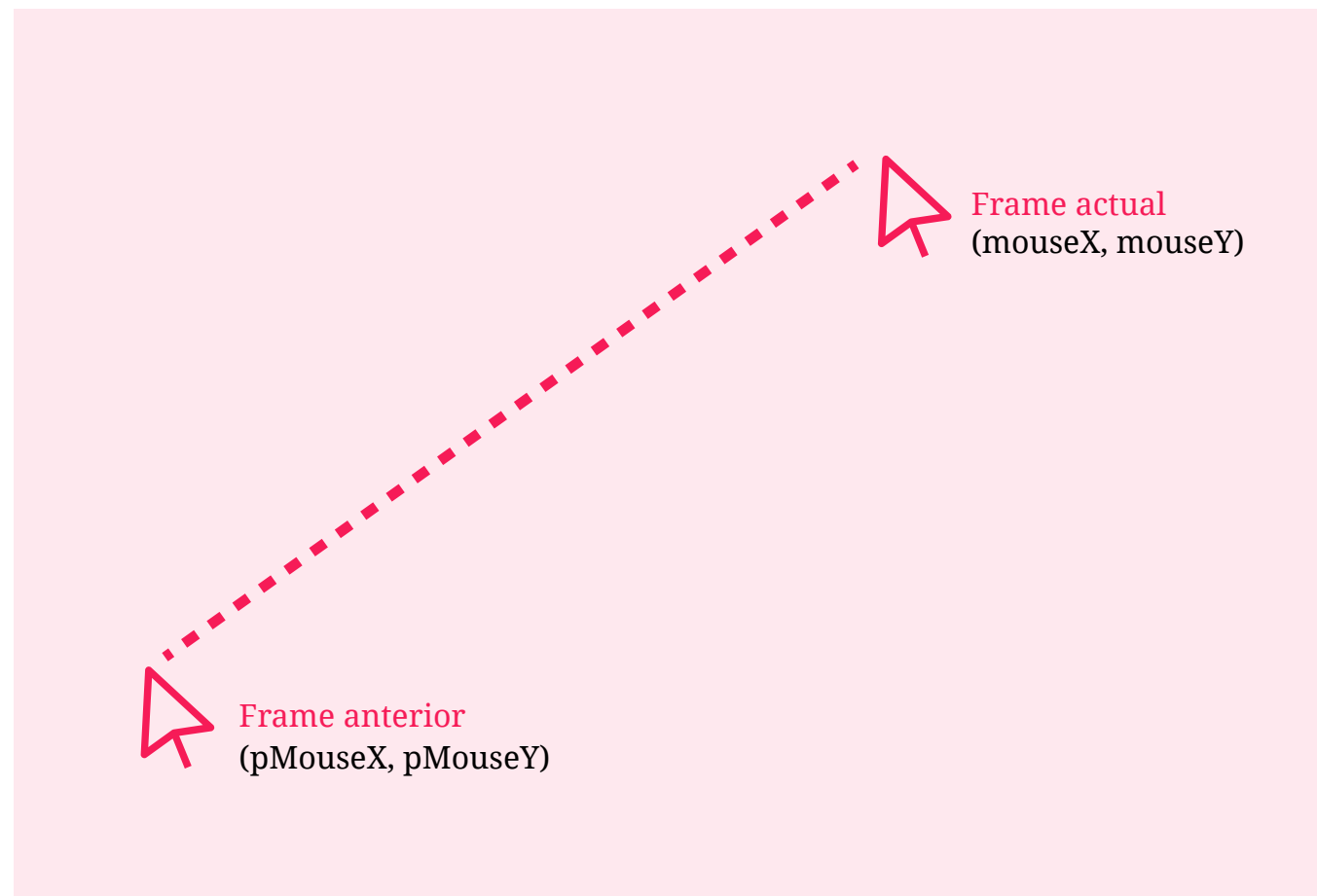
b.

Velocidad del ratón

1. Conocer la posición actual y la posición anterior del ratón

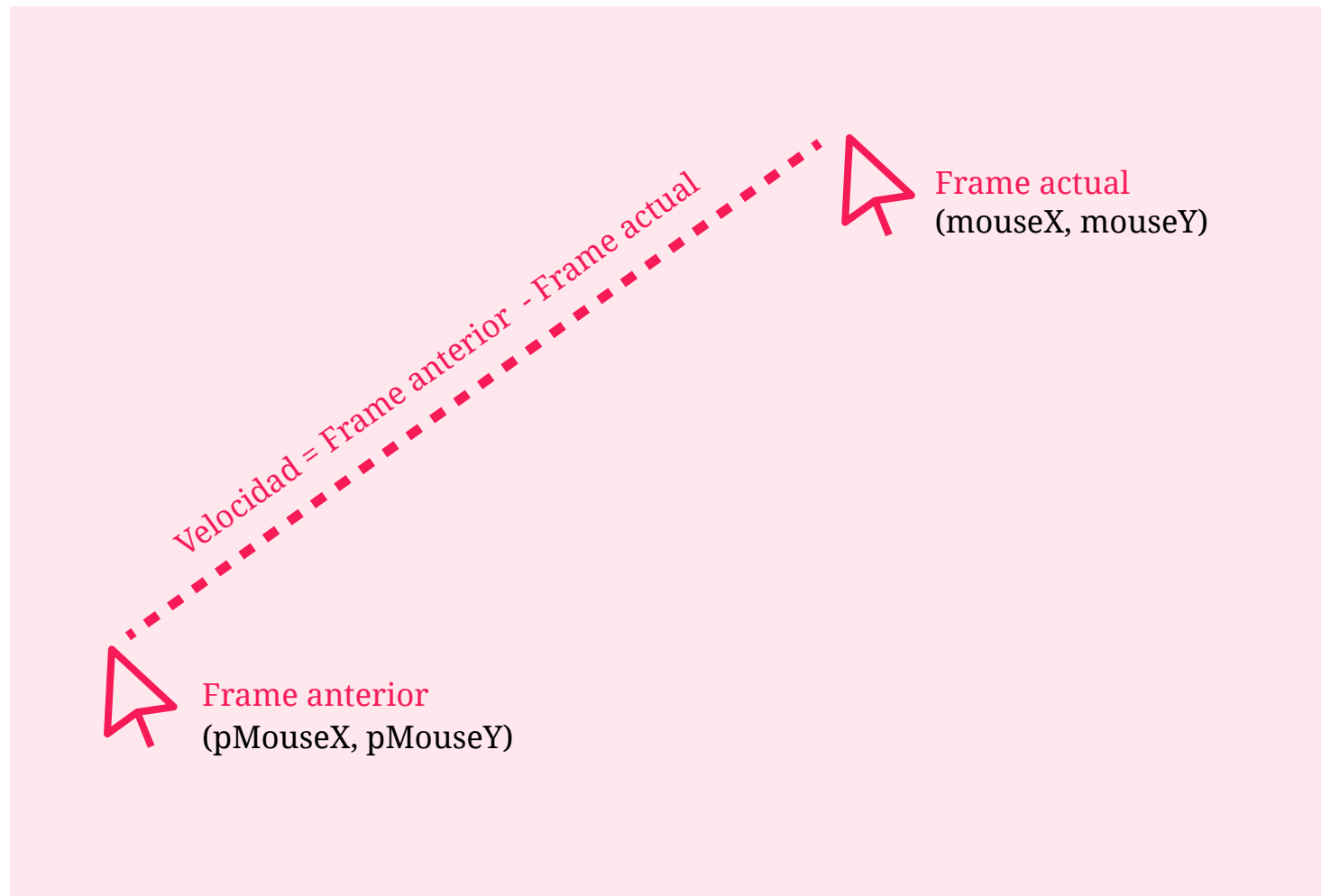
Para conocer la rapidez con la que se esta moviendo algún objeto, es necesario conocer por lo menos **dos posiciones y compararlas en el tiempo**.

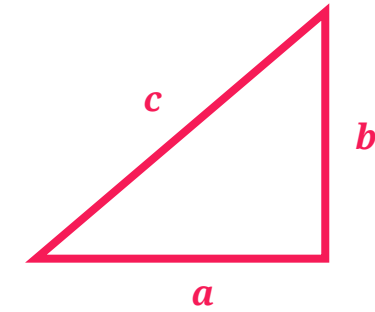
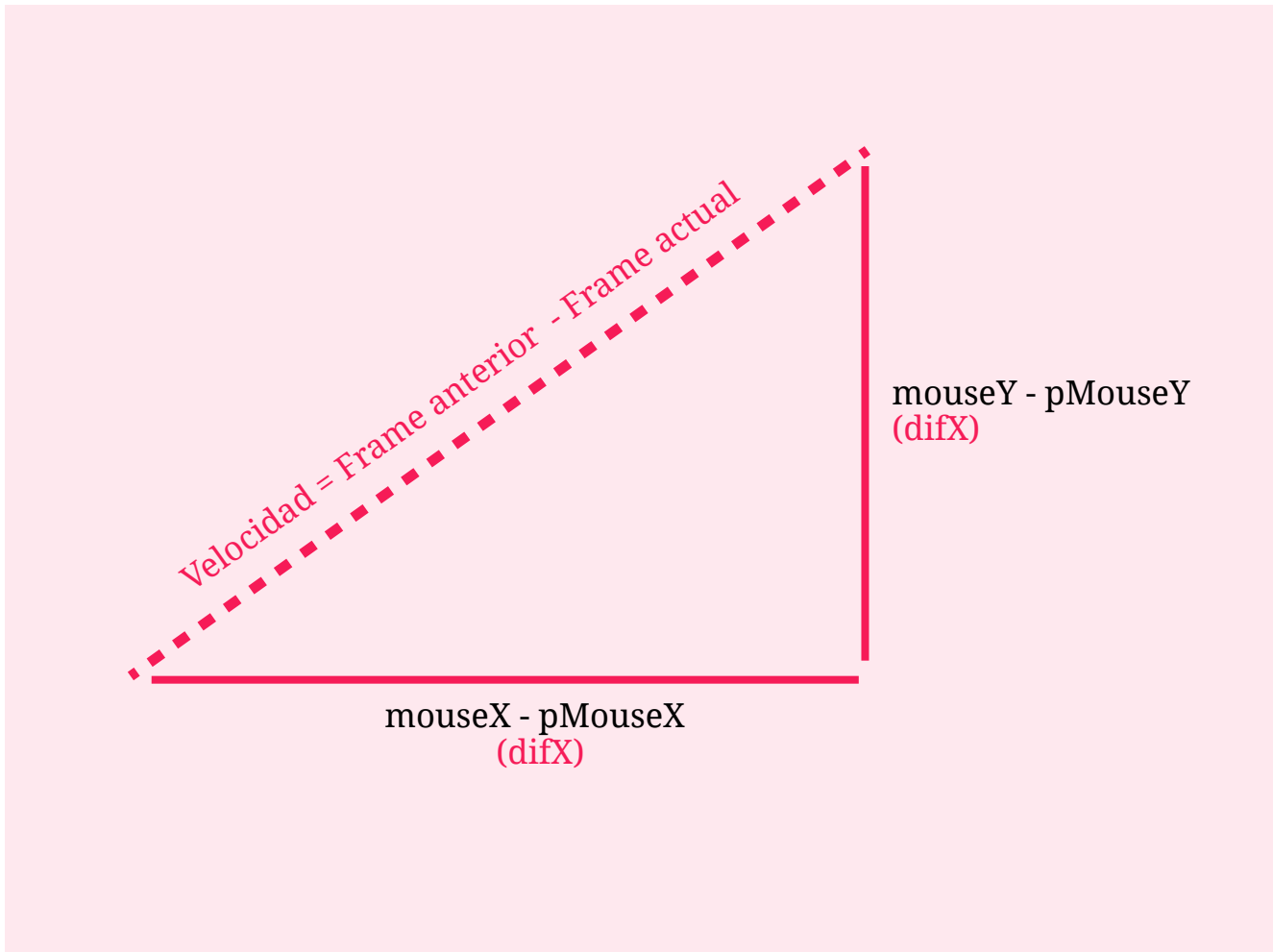
pMouseX y **pMouseY** son dos variables de p5.js que siempre guardan la posición en X y Y del ratón del frame anterior al actual.



2. Calcular la diferencia entre ambas posiciones

La velocidad con la que se esta moviendo el ratón en cada frame se puede calcular al hacer la resta de la posición actual del mouse con la posición que tuvo en el frame inmediatamente anterior.





$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$

$$\text{vel}^2 = \text{difX}^2 + \text{difY}^2$$

$$\text{vel} = \sqrt{\text{difX}^2 + \text{difY}^2}$$

Utilizando el **teorema de Pitágoras** se puede obtener la velocidad del ratón

3. Realizar el cálculo en p5.js

```
//difX es la la resta de la posicion x del mouse actual con la posicion del mouse anterior  
var difX = abs(mouseX - pmouseX);  
  
//difY es la la resta de la posicion y del mouse actual con la posicion del mouse anterior  
var difY = abs(mouseY - pmouseY);  
  
//se elevan las restas al poder de 2  
difX = pow(difX, 2);  
difY = pow(difY, 2);  
  
//la velocidad es dada por el teorema de pitagoras c = sqrt(a^2 + b^2)  
//con floor() se redondea la velocidad a un numero entero  
var vel = floor(sqrt(difX + difY));
```

Se utiliza **abs(valor)** - valor absoluto - para que el resultado de la resta sea siempre positivo.

La función **pow(valor, potencia)** eleva un número a la potencia elegida.

sqrt(valor) devuelve la raíz cuadrada de un número.

floor(valor) redondea un número decimal a un número entero.

4. Mostrar la velocidad del ratón en la pantalla

```
//texto que muestra la velocidad en la pantalla  
fill(0);  
text(vel, 30, 30);
```

La función **text()** recibe tres argumentos: el texto a mostrar, su posición en X y su posición en Y

5. Utilizar la velocidad del ratón para generar algún comportamiento en el programa

En este caso cada vez que el mouse este oprimido y tenga una velocidad mayor a 80 pixeles por frame se creará un nuevo objeto.

```
//revisa que la velocidad del mouse es mayor a 80 y el mouse este presionado  
if (vel > 80 && mouseIsPressed) {  
    //crea una nueva bola en la posicion mouseX, mouseY y envía la velocidad como parametro  
    bolas[numBolas] = new bola(mouseX, mouseY, vel);  
  
    //aumenta el numero de bolas de la lista  
    numBolas++;  
}
```

```
//función bola recibe (posicion del mouse en X, posicion del
mouse en Y, v la velocidad del ratón)
function bola(px, py, v) {

    //variables de posicion, la bola se crea en la posicion X y Y
del mouse que recibe por parametro
    this.x = px;
    this.y = py;

    //variables de velocidad
    //la velocidad de cada bola se determina por la velocidad del
mouse
    this.velx = v;
    this.vely = v;
```

Al definir la función, también se definen los **datos o variables** que necesita para funcionar. En este caso: posición en X y Y del ratón y la velocidad del mismo.

¿Cómo puede el comportamiento de un objeto ser guiado y determinado por una acción?

Se puede ver el código de este ejemplo en:

alpha.editor.p5js.org/laurajunco/sketches/Skjpy50AW

C.

Lanzar

En este ejemplo se creará la posibilidad de lanzar objetos a partir de la posición y velocidad con la que se mueva el ratón

1. Leer la velocidad con la que se está moviendo el ratón en los ejes X y Y

```
//posicion del mouse Actual
var x1 = mouseX;
var y1 = mouseY;

//posicion anterior del mouse
var x2 = pmouseX;
var y2 = pmouseY;

//la velocidad se mide restando la posicion actual del
mouse con la posición del frame anterior
var velx = x1 - x2;
var vely = y1 - y2;
```

Esta es otra manera de medir la velocidad del ratón de forma más sencilla.

2. Utilizar la información del ratón para crear un nuevo objeto

```
// se crea una nueva bola que recibe por parametro las
velocidades en x y y del mouse
bolas[numBolas] = new bola(velx, vely);

//aumenta el numero de bolas de la lista
numBolas++;
```

Envía la velocidad en X y Y del ratón a la función bola.

3. Utilizar la información del ratón para crear un objeto nuevo

```
//función bola
function bola(px, py) {
    //variables de posicion, la bola se crea en la
    posición del mouse
    this.x = mouseX;
    this.y = mouseY;

    //variables de velocidad que recibe la función
    this.velx = px;
    this.vely = py;
}
```

Recibe velocidad en X y
velocidad en Y

Asigna la posición inicial del
objeto en la posición del mouse

La velocidad del objeto es la
velocidad del mouse

4. Controlar la frecuencia con la que se crean los objetos

```
//solo se pueden crear bolas en frames multiples de 10
if (frameCount % 10 != 0) {
    return;
}
```

El operador % (módulo) devuelve el
resto de la división entre dos
numeros.

Ej. (20 % 7 = 6)

Porque $20 / 7 = 14$ y sobran 6

Este código revisa si el número de frame actual es **múltiplo de 10**, y sólo permite seguir avanzando en el programa hasta que esto pase. De esta manera, las bolas se pueden crear con 10 frames de diferencia.

5. Por último, revisar que todo suceda mientras el mouse está siendo arrastrado en la pantalla

```
//funcion que se activa si el mouse es arrastrado
function mouseDragged() {

    //solo se pueden crear bolas en frames multiplos de 10, esto hace que no se puedan crear
    bolas tan seguido
    if (frameCount % 10 != 0) {
        return;
    }

    //posicion del mouse Actual
    var x1 = mouseX;
    var y1 = mouseY;

    //posicion anterior del mouse
    var x2 = pmouseX;
    var y2 = pmouseY;

    //la velocidad se mide restando la posicion guardada del mouse con la posición que se habia
    guardado de este
    var velx = x1 - x2;
    var vely = y1 - y2;

    // se crea una nueva bola que recibe por parametro las velocidades en x y y del mouse
    bolas[numBolas] = new bola(velx, vely);

    //aumenta el numero de bolas de la lista
    numBolas++;
}
```

¿Cómo conocer la frecuencia con la que se realiza una acción?

Se puede ver el código de este ejemplo en:

alpha.editor.p5js.org/laurajunco/sketches/S1dBY5RRW

d.

Clics por minuto

1. Conocer el tiempo entre un clic y otro.

Clic 1



Millis() = 4000

Clic 2



Millis() = 7000

Tiempo entre clics = Tiempo clic 2 - Tiempo clic 1
= 7000 ms - 4000ms
= 3000 ms

El primer paso para conocer la frecuencia con la que se realiza algo es medir la **diferencia de tiempo** entre una acción y la siguiente

2. Generalizar ese intervalo para una cantidad de tiempo mayor

¿Si entre un clic y otro hubo 3000 milisegundos de diferencia, cuantos clics se podrían hacer en 1 minuto?

Clics por minuto = 60000 ms / 3000 ms
= 200 clics por segundo

Después de conocer el tiempo entre dos acciones, es posible saber cuantas acciones se pueden hacer en un intervalo de tiempo determinado.

- En un minuto hay 60000 milisegundos

3. Crear variables para guardar la información de los clics

```
//crea una lista para guardar el tiempo entre dos clics
var tiempo = [0, 0];

//variable para guardar la velocidad de los clics
var vel = 0;
```

La variable tiempo es una **lista** de 2 posiciones en la que se guardarán los tiempos en los que se haga clic.

4. Guardar el tiempo cuando se haga clic

```
//funcion que se llama cuando se hace clic
function mouseClicked() {

    //el tiempo del segundo clic se corre a la segunda posición
    tiempo[0] = tiempo[1];

    // se guarda el tiempo del primer clic en el arreglo
    tiempo[1] = millis();
}
```

Cuando se hace clic por segunda vez se **corren** los valores de la lista.

Se guarda el tiempo del clic en la **segunda posición** de la lista que se creó.

5. Obtener la cantidad de clics por minuto

```
//la velocidad de la bola se obtiene de la division de  
60 segundos entre el intervalo de dos clics  
  
//tiempo[1] = tiempo en el que se hizo el primer clic  
//tiempo[0] = tiempo en el que se hizo el segundo clic  
  
var vel = 60000 / (tiempo[1] - tiempo[0]);
```

Para hacer más sencilla la medición, es posible definir el numero de clics por minuto como 0 si el intervalo entre un clic y otro mayor a un valor determinado.

```
//si han pasado mas de 200 milisegundos entre un clic y  
otro se asigna un valor de 0 a la velocidad  
  
if (millis() - tiempo[1] > 200) {  
    vel = 0;  
}
```

6. Imprimir el numero de clics por minuto en la pantalla

```
//texto de la pantalla  
//floor() redondea el numero de vel a un valor entero  
text(floor(vel) + " clics por minuto", 30, 30);
```

¿Puede la ventana de navegación ser un elemento de interacción en si?

Se puede ver el código de este ejemplo en:

alpha.editor.p5js.org/laurajunco/sketches/BkEKWjCCW

e.

Tamaño de la ventana

En este ejemplo se utilizará el tamaño de la ventana de navegación como un elemento que influye en el comportamiento de los objetos de un programa

1. Guardar el tamaño inicial de la ventana

```
//variables para guardar el ancho y alto de la pantalla  
var w = 0;  
var h = 0;
```

```
//crea un canvas de pantalla completa  
createCanvas(windowWidth, windowHeight);  
  
//guarda el ancho de la pantalla en la variable w  
w = windowWidth;  
  
//guarda el alto de la pantalla en la variable h  
h = windowHeight;
```

windowWidth y **windowHeight** son variables de p5.js que guardan automáticamente el **tamaño** de la ventana del navegador

2. Utilizando la función **windowResized**, conocer si el tamaño de la ventana cambio.

```
//funcion que se activa cuando cambia el tamaño de la
ventana del navegador
function windowResized() {

    //se actualiza el nuevo tamaño de la pantalla en las
    variables w y h
    w = windowWidth;
    h = windowHeight;

    //se cambia el tamaño del canvas para que sea del
    nuevo tamaño de la pantalla
    resizeCanvas(w, h);
}
```

Cada vez que la ventana del navegador cambia de tamaño, se guardan las nuevas dimensiones en las variables w y h.

Con la función **resizeCanvas()** se asigna el **nuevo tamaño** al canvas.

3. Generar un comportamiento a partir del evento

```
//si el nuevo tamaño de la ventana es mayor al tamaño guardado en w y h
if (windowWidth > w || windowHeight > h) {
    //se pintan las bolas de verde
    fill(0, 255, 0);

    //se llama a la función crecer para cada una de las bolas
    for (var i = 0; i < numBolas; i++) {
        bolas[i].crecer();
    }
}
```

Revisa si la nueva posición de la pantalla es **mayor** a la que se tiene guardada.

¿Cómo jugar con las formas esperadas de interacción
en un programa?

Se puede ver el código de este ejemplo en:

alpha.editor.p5js.org/laurajunco/sketches/SyKFXj00Z



Dentro del canvas

En este ejemplo se detectará cuando el mouse de la persona esté dentro del canvas en un programa de p5.js

1. Crear una variable para el canvas

```
//crea una variable para guardar el canvas  
var canvas;
```

Para este ejercicio, se utilizará el canvas como **elemento html** de la página. Por esta razón se declara como una variable. Más sobre esto en: p5js.org/reference/#/p5.Element

2. Crear el canvas en el setup

```
//a la variable canvas se le asigna crear un canvas  
de pantalla completa  
  
canvas = createCanvas(windowWidth, windowHeight);
```

3. Crear funciones para que respondan a eventos de interacción con el canvas

```
//cuando el mouse este sobre el canvas se llama a la
funcion detener
canvas.mouseOver(detener);

//cuando el mouse este sobre el canvas se llama a
la funcion mover
canvas.mouseOut(mover);
```

Las funciones **mouseOver()** y **mouseOut()** viene incluidas en p5.js y se activan cuando el ratón entra y sale de algún elemento en una página web.

El tipo de funciones que se utiliza acá es llamado **Callback functions**. Son funciones que **llaman a otra función**. En este caso mouseOver y mouseOut llaman a las funciones mover y detener de los objetos.

4. Crear nuevas funciones para responder a mouseOver y mouseOut

```
function mover() {  
  
    //recorre la lista de bolas  
    for (var i = 0; i < numBolas; i++) {  
  
        //moverse le asigna true a la variable  
        bolas[i].moverse = true;  
    }  
}  
  
//funcion detener: se activa cuando el mouse entra al  
canvas  
function detener() {  
  
    //recorre la lista de bolas  
    for (var i = 0; i < numBolas; i++) {  
  
        //le asigna false a la variable moverse  
        bolas[i].moverse = false;  
    }  
}
```