PENSAMIENTO COMPUTACIONAL

PENSAMIENTO COMPUTACIONAL

Aplicado a dibujo 2D con p5.js

por Aarón Montoya Moraga

Santiago

CONTENIDOS RESUMIDOS

CONTENIDOS DETALLADOS

1

AUTORÍA

A pesar de que este libro está siendo escrito por una persona, nace de un trabajo colectivo que incluye a artistas, diseñadores, programadores, estudiantes, profesores, y a toda una comunidad que hace posible la reflexión y escritura detrás de este proyecto.

Aarón Montoya Moraga

Profesore asistente en la Escuela de Diseño, de la Facultad de Arquitectura, Arte y Diseño de la Universidad Diego Portales. Su formación en pregrado es en ingeniería eléctrica en la Pontificia Universidad Católica de Chile (2014). Posee un magíster del Interactive Telecommunications Program de New York University (2017), y otro magíster en ciencias y artes mediales de Massachusetts Institute of Technology (2021). Actualmente es estudiante del Doctorado de Artes y Humanidades de la Universidad de Santiago de Chile. Todos sus estudios de postgrado han sido apoyados por becas incluyendo Becas Chile, Research Assistantship MIT, Beca de arancel de doctorado USACH.

2

CURSOS

Este libro está inspirado por el nuevo curso DIS09214 Pensamiento Computacional, que creamos como curso obligatorio de tercer semestre en la Escuela de Diseño de la Universidad Diego Portales.

Este curso se dictó por primera vez en el primer semestre del año 2026, entre marzo y julio, que coincidió con la antesala del lanzamiento de la nueva versión de p5.js, la biblioteca de JavaScript que usamos para enseñar programación en este curso.

3

HERRAMIENTAS

Este libro está programado con las siguientes herramientas:

LaTeX

Este libro está escrito en LaTeX, y en particular usa la clase *nostarch*, creada por la editorial No Starch Press.

Git

Este libro está escrito con control de versiones git, que ayuda a mantener un historial de cambios de cada archivo y cada línea escrita.

GitHub

Este libro se aloja en GitHub con el sistema git, que permite trabajar de forma pública, y automatizar la compilación del código fuente PDF, usando GitHub Actions.

p5.js

p5.js es una biblioteca de JavaScript creada por Lauren Lee McCarthy, y parte de la Processing Foundation. Este libro nace en el contexto del lanzamiento de la nueva versión 2.0.0 de p5.js, que incluye nuevos paradigmas y funcionalidades.

4

AGRADECIMIENTOS

Este libro existe gracias a toda una comunidad detrás del trabajo de este proyecto.

Primero quiero agradecer a innumerables estudiantes que con su curiosidad y energía me motivan a seguir investigando, escribiendo y enseñando.

Aplaudo al equipo con el que hemos trabajado en el desarrollo de los cursos de Pensamiento Computacional en la Escuela de Diseño de la Facultad de Arquitectura, Arte y Diseño de la Universidad Diego Portales, incluyendo a profesores Matías Serrano, Sofía Suazo, Christian Oyarzún, Poli Mujica que tomaron el desafío de dictar la primera versión de este curso nuevo, y al equipo de ayudantes Morgan Aravena, Valentina Chávez, Janis Sepúlveda, Vania Paredes, Santiago Gaete, Jacinta Barrenechea, que ayudaron al desarrollo de nuestros cursos.

5

VARIABLES

Las variables son espacios de almacenamiento de información.

Definición

Sintaxis en JavaScript

Las variables en JavaScript se declaran usando la palabra reservada *let*.

```
let unaVariable = 10;
let otraVariable = "sopaipilla";
```

Hasta el año 2015 las variables se declaraban usando la palabra reservada *var*, pero esta forma de declarar variables tenía problemas, por lo que se introdujo la palabra reservada *let* en su reemplazo.

6

BUCLES

Definición

Los bucles repiten.

Los bucles también son conocidos en inglés como loops.

Sintaxis en JavaScript

Los bucles en JavaScript se declaran usando la palabra reservada *for*.

A continuación de *for*, se escriben paréntesis, dentro de los cuales se escriben tres partes separados por punto y coma.

Las llamaremos:

1. Inicialización
2. Condición
3. Actualización

Entonces en pseudocódigo resulta

```
for (inicialización; condición; actualización) {
    // código a repetir
}
```

7

CONDICIONALES

Definición

Las condicionales condicionan.

Las condicionales permiten que nuestro código se bifurque, que tome distintos caminos dependiendo de condiciones.

Las condicionales también son conocidas en inglés como conditionals, o if statements.

Sintaxis

Las condicionales en JavaScript se declaran usando la palabra reservada *if*.

Luego de la palabra reservada *if* se escribe una pregunta entre paréntesis, y luego un bloque de código entre llaves.

```
if (pregunta) {
    // código a correr si la respuesta es afirmativa
}
```

8

ARREGLOS

Definición

Los arreglos son colecciones o conjuntos de datos. Los arreglos son también conocidos en inglés como arrays.

En algunos lenguajes como Java y C, los arreglos nos permiten almacenar datos del mismo tipo, por ejemplo solamente números enteros o solamente caracteres, pero en JavaScript podemos almacenar distintos tipos de datos en el mismo arreglo.

Los arreglos en JavaScript no son primitivas, sino que son instancias de objetos del tipo *Array*. Revisaremos el significado de objetos en un capítulo posterior. Esto le permite a los arreglos tener funcionalidades internas de gran utilidad, revisaremos algunas a continuación.

Sintaxis en JavaScript

Los arreglos en JavaScript se pueden declarar con la palabra clave let, igual que las variables, y asignarle como valor inicial un arreglo vacío con corchetes. En este libro usaremos esta versión.

```
let unArreglo = [];
```

Otra versión que no usaremos es la sintaxis:

```
let otroAreglo = new Array();
```

Usos de arreglos

Hasta ahora si queremos declarar distintos nombres de colores debemos crear una variable por cada uno, por ejemplo

```
let colorPrincipal = "verde";
let colorSecundario = "rojo";
let colorFondo = "azul";
let colorEnfasis = "violeta";
```

Para usar cualquier de estos colores, debemos usar cada nombre, rápidamente empezamos a tener muchos nombres de variables, cuando quizás internamente estamos tratándoles como un conjunto de colores relacionados, por ejemplo la paleta de colores de una obra que estamos programando.

Los arreglos son una respuesta a este pensamiento, de pensar en un conjunto de datos como miembros de una misma colección, entonces en nuestro ejemplo podemos crear un arreglo llamado colores que contenga a todos los colores, y nos ahorramos la creación de una variable y un nombre de fantasía por cada dato.

```
let colores = ["verde", "rojo", "azul", "violeta"];
```

Índices en arreglos

En computación, de forma estándar contamos desde 0. Es por esto si un arreglo tiene N elementos, están indizados desde 0 hasta N-1.

Para acceder a un elemento, escribimos el nombre del arreglo, seguido de corchetes, y dentro de los corchetes ingresamos el indice al que queremos acceder.

```
    nombreDelArreglo[indice]
```

En nuestro ejemplo entonces tenemos los siguientes resultados.

```
console.log(colores[0])
// imprime verde

console.log(colores[1])
// imprime rojo

console.log(colores[2])
// imprime azul

console.log(colores[3])
// imprime violeta

console.log(colores[0])
// imprime un error
```

Largo de un arreglo

Un atributo muy importante es el largo de un arreglo, que inglés es conocido como length. El largo corresponde al número de casillas que tiene un arreglo.

La sintaxis para acceder al largo de un arreglo es el nombre del arreglo, seguido de un punto, y luego de la palabra length.

```
nombreDelArreglo.length
```

En nuestro ejemplo, si imprimimos en consola el resultado, sería el valor 4, ya que los índices son 0, 1, 2, 3.

```
console.log(colores.length)
// imprime 4
```

Iteración en arreglos

Para iterar sobre todos los elementos de un arreglo, usaremos bucles for, y el atributo length. Por ejemplo, para imprimir en consola todos los elementos de un arreglo, proponemos la siguiente sintaxis.

```
// recorrer todos los elementos del arreglo
// usando el iterador i
for(let i = 0; i < nombreDelArreglo.length; i++) {
    // imprimir valor del elemento en la posición i
    console.log(nombreDelArreglo[i]);
}
```

Referencias

1. *https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array*

9

FUNCIONES

Las funciones nos permiten escribir bloques de código reutilizables, paramétricos, y abstraer los detalles de las implementaciones de nuestras ideas. Las funciones también son conocidas en inglés como functions.

Definición

Las funciones permiten agrupar bloques de código para poder utilizarlos y volver a utilizarlos, sin necesidad de escribir el mismo código una y otra vez.

Sintaxis en JavaScript

Las funciones en JavaScript se declaran usando la palabra reservada *function*, y dándoles un nombre de fantasía.

El siguiente punto a considerar es que después del nombre de la función vienen paréntesis (), donde se definen (o no) los parámetros de la función.

Los parámetros son variables locales a la función que permiten ingresar distintos datos para ser usados, leídos e incluso modificados por la función.

Otra palabra clave importante es la palabra return, que permite que la función emita como resultado un valor al exterior, que puede ser usado por el resto del código.

Por ejemplo, aquí definiremos una función sumar() que tiene dos parámetros, a y b, los cuales son sumados internamente, asignados a una variable local resultado, cuyo valor es luego retornado.

```
// declaración de la función sumar
// con dos parámetros llamados a y b
function sumar(a, b) {
    // declaracion de variable local x
    // cuyo resultado es la suma de a y b
    let resultado = a + b;
    // la función retorna el valor almacenado en resultado
    return resultado;
}
```

Para usar una función, tiene que haber sido declarada antes. La declaración anterior nos permite luego sumar números de la siguiente forma.

```
let unaSuma = sumar(3 ,4);
```

Funciones en p5.js

El código en JavaScript que nos da el editor de p5.js cuando creamos un nuevo proyecto es el siguiente:

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```

En este código distinguimos cuatro funciones distintas, incluyendo setup(), draw(), createCanvas() y background().

Las funciones setup() y draw() están definidas, pero nunca son usadas de forma explícita. Es trabajo de la biblioteca p5.js el usarlas como las conocemos, haciendo que primero ocurra setup() una vez, y luego ocurra draw() en bucle.

Por otro lado, las funciones createCanvas() y background() están siendo usadas, pero no están declaradas acá, lo están en la biblioteca p5.js.

10

COLOR

En este capítulo veremos definiciones de color, la percepción humana de color, el modelamiento de color usando computadores, y su aplicación en p5.js

Definición

Definiremos color como la percepción de luz a través de nuestros ojos.

La velocidad de la luz es de trescientos mil kilómetros por segundo, en números $300.000\frac{km}{s}$.

Percepción humana de color

Se estima que una persona es capaz de detectar alrededor de 10 millones de colores distintos.

La percepción ocurre entre tonalidades rojas, aproximadamente con una frecuencia de $400THz$, y tonalidades violeta, con una frecuencia aproximada de $800THz$.

Las ondas de luz que tienen una frecuencia menor a rojo no son visibles y son denominadas infrarrojas, como las que se usan para la emisión de señales de controles remotos domésticos.

Las ondas de luz que tienen una frecuencia mayor a violeta no son visibles y son denominadas ultravioleta, como las que producen quemaduras en la piel.

La percepción en nuestros ojos ocurre con receptores de 3 rangos de frecuencia distintos, en torno a los colores rojo, verde y azul.

Color y computación

Para describir un color necesitamos un sistema que sea capaz de tener al menos 10 millones de combinaciones, que es la cantidad de colores que una persona puede percibir.

1 bit puede describir 2 valores posibles: 0 y 1. Si tenemos 2 bits, tenemos 4 valores posibles: 00, 01, 10 y 11. Con 3 bits, tenemos 8 valores posibles: 000, 001, 010, 011, 100, 101, 110, 111. La ecuación que gobierna cuántas combinaciones distintastenemos con N bits es:

$$combinaciones = 2^N$$

Entonces lo que estamos buscando, es cuántos bits necesitamos para tener al menos diez millones de valores posibles.

Si aumentamos N, podemos calcular que con N=23 y N=24:

$$2^{23} = 8388608$$

$$2^{24} = 16777216$$

Entonces con N = 23 no nos alcanza, recién con N = 24 superamos los 10 millones, y ese es el valor elegido por la industria computacional para representar colores.

Debido a la percepción humana de color, estos 24 bits se usan para describir cuán encendida están 3 ampolletas o canales de colores: una dedicada a rojo, una a verde, y una azul. Este sistema se denomina RGB por sus iniciales en inglés.

Cada canal de color entonces recibe 8 bits de resolución, y con eso cada canal tiene $2^8 = 256$ valores posibles. Y como en computación contamos desde 0, los valores de cada canal de color están entre el rango 0 y 255, para representar todo el espectro entre apagado y prendido.

Representación hexadecimal

Como cada canal tiene 8 bits de resolución, para describir un color necesitamos los 3 canales y con esto, el color rojo por ejemplo recibe la siguiente notación:

111111110000000000000000

Si tomamos los 8 bits de cada canal, y los dividimos entre los primeros 4 y los siguientes 4, entonces tenemos que los 24 bits son 6 grupos de 4 bits.

Los 4 bits permiten representar los siguientes números en binario:

Cuadro 10-1: Representaciones binarias, decimales y hexadecimales de valores entre 0 y 15

Binario	Decimal	Hexadecimal %
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Referencias

1. *https://en.wikipedia.org/wiki/Color*

11

LIENZO EN P5

En p5.js dibujaremos en un lienzo, una superficie rectangular en 2 dimensiones. El lienzo en p5.js recibe el nombre de canvas.

Contexto

El lienzo es una abstracción del elemento canvas introducido en HTML 5.

En Processing existe una ventana donde renderizamos las gráficas que programamos, y donde también podemos programar interactividad con interfaces humano-computador como un teclado o un ratón. En p5.js, este lugar recibe el nombre de lienzo o canvas en inglés, ya que internamente está implementado con un elemento canvas de HTML 5.

Sintaxis en p5.js

En p5.js usamos la función createCanvas() para crear un lienzo, y que tiene los parámetros ancho (en eje x / horizontal), y altura (en eje y / vertical), ambas medidas en pixeles.

```
createCanvas(ancho, altura);
```

Referencias

1. *https://en.wikipedia.org/wiki/Canvas_element*

12

SOL LEWITT

Sol Lewitt es un artista estadounidense del siglo XX, cuya obra está enmarcada en las corrientes del arte minimalista y el arte conceptual. En este capítulo nos enfocaremos en su serie de trabajos situados en el museo MASS MoCA, en Estados Unidos.

Referencias

1. *https://en.wikipedia.org/wiki/Sol_LeWitt*
2. *https://massmoca.org/sol-lewitt/*