ISSN 2448-8054
Núm. 62, marzo 2022

Cuadernos del centro
de investigación
en **economía creativa (CIEC)**

Manual
STEAM Lab
Código Creativo

centro.
Centro de Investigación
en Economía Creativa

Díaz, B. y Álvarez, E. (2022). Manual – Código Creativo. STEAM Lab. Cuadernos del Centro de Investigación en Economía Creativa (CIEC), (62), marzo, México: Centro de Diseño, Cine y Televisión.

centro.

Dirección general
Kerstin Scheuch

Dirección de Gestión y Desarrollo Académico
Gabriela Traverso

Coordinación CIEC
Graciela Kasep

Coordinación STEAM Licenciaturas
Sandra Barrón

Director de Licenciatura en Medios Digitales y Tecnología
Roberto Cabezas

Edición
Eduardo Álvarez

Diseño editorial
Daniel Berkstein

ISSN 2448-8054

CUADERNOS DEL CENTRO DE INVESTIGACIÓN EN ECONOMÍA CREATIVA (CIEC), Año 9. núm. 62, marzo 2022, editado por Centro de Diseño y Comunicación S.C., con domicilio en Av. Constituyentes 455, Colonia América, Alcaldía Miguel Hidalgo, C.P. 11820, Ciudad de México, T. (55) 2789 9000, centro.edu.mx, https://www3.centro.edu.mx/cuadernos-de-investigacion/ Editor Responsable: Graciela Kasep, Centro de Diseño y Comunicación S.C., Reserva de Derechos al Uso Exclusivo No. 04-2016-052014385000-203, ISSN: 2448-8054, ambos otorgados por el Instituto Nacional del Derecho de Autor; persona responsable de la última actualización de esta publicación: Graciela Kasep a través del Centro de Investigación en Economía Creativa de Centro de Diseño y Comunicación, S.C. con domicilio en el antes indicado, fecha de última modificación 11 de marzo de 2022.

centro.edu.mx/ciec

INTRODUCCIÓN GENERAL AL PROGRAMA STEAM

El trabajo en CENTRO para profesionalizar la creatividad ha transformado esquemas preestablecidos e integrado nuevas visiones, así como prácticas de innovación educativa para formar profesionales acordes a su tiempo.

En 2014, el Consejo Académico y un grupo de profesores identificó la necesidad de integrar un acercamiento más profundo a la tecnología y el desarrollo de habilidades indispensables en el mundo laboral como la alfabetización digital. Después de un proceso de investigación para elaborar el diseño curricular adecuado a todos los programas de Licenciatura y el establecimiento de un área especializada dentro del Campus para estos propósitos (Piso 7), nació STEAMLab. La evolución de este programa sea ha transformado en una base pedagógica para todos los estudiantes que integran un espacio de investigación, de desarrollo de ideas y la construcción de puentes entre el pensamiento y la implementación.

STEAM es un acrónimo cuyas letras se refieren a cinco áreas de conocimiento — ***S****cience,* ***T****echnology,* ***E****ntrepreneurship,* ***A****rts & Creativity y* ***M****eaning* —; que se traducen en líneas de formación en CENTRO.

Los objetivos del programa STEAM Lab son:

- Potenciar la creatividad a través de conocimientos, metodologías y herramientas complementarias a cada programa (*Science, Technology, Entrepreneurship, Arts-Creativity & Innovation and Meaning)*.
- Fomentar la enseñanza interdisciplinaria para escuchar diferentes perspectivas, experiencias, puntos de vista y metodologías de otras disciplinas creativas de CENTRO.
- Construir vínculos y colaboraciones profesionales a través de disciplinas creativas, reflejando así dinámicas del mundo profesional.
- Reflexionar sobre las fuerzas y circunstancias que determinan el desarrollo de cada disciplina y profesión para explorar sus posibilidades.

La implementación de este programa busca que los estudiantes tengan los pilares esenciales para desarrollar un aprendizaje basado en el entendimiento de un contexto integrando la programación, el pensamiento lógico matemático, las técnicas creativas, el uso de los fundamentos de la visualidad y el autoconocimiento. STEAM tiene una asignatura por área de conocimiento que se cursa de segundo a sexto semestre y concluye con un proyecto integral de emprendimiento, donde se pone en práctica las habilidades antes mencionadas.

OBJETIVOS MODULARES

STEAM LAB

Science	Technology		Arts - Creativity & Innovation	Meaning
Pensamiento lógico matemático	**Tecnología creativa**	**Código creativo**	**Introducción al análisis de problemas complejos**	**Autoconocimiento**
Aplicar el pensamiento numérico en diferentes niveles de complejidad para abordar retos creativos.	Identificar, analizar, prospectar y evaluar las implicaciones de la interacción entre los seres humanos y la tecnología, con base en la exploración de casos de alto impacto social y cultural.	Comprender los lenguajes de programación computacional como una herramienta para potenciar la creatividad.	Identificar, conocer, comprender y analizar problemas de distinta escala y grado de complejidad explorando su contexto y alcance, e identificando los métodos y las herramientas que, en cada caso, resultan más pertinentes para descifrarlos.	Comprender recursos científicos, filosóficos y humanos que a nivel personal u organizacional, inciden en la manera de relacionarnos con nosotros mismos, los otros y los retos que enfrentamos.
Entrepreneurship				
Emprendimiento Aplicar los conocimientos, los métodos y las herramientas adquiridos en STEAM Lab a un proyecto de emprendimiento				

Como parte del trabajo editorial producido desde el Centro de Investigación de Economía Creativa se inicia una línea editorial de los cuadernos del CIEC que se enfocara en contenido relativo al programa STEAM.

INTRODUCCIÓN AL MANUAL DE CÓDIGO CREATIVO

Para el caso de código creativo, materia principal de este manual y asignatura perteneciente al área de Tecnología (***T****echnology*), los objetivos por semestre son:

Objetivo 2º Semestre

Comprender los lenguajes de programación computacional como una práctica creativa para diseño y arte en el entorno de desarrollo *Processing*.

Objetivo 3º Semestre

Aplicar procesos algorítmicos para la conceptualización y realización de productos digitales de diseño y arte mediante programación orientada a objetos.

Objetivo 4º Semestre

Aplicar procesos algorítmicos avanzados para la conceptualización y realización de productos digitales de diseño y arte mediante programación orientada a objetos.

Objetivo 5º Semestre

Analizar y desarrollar estrategias personalizadas de creatividad computacional basadas en lenguajes de programación contemporáneos.

Manual STEAM Lab | Código Creativo

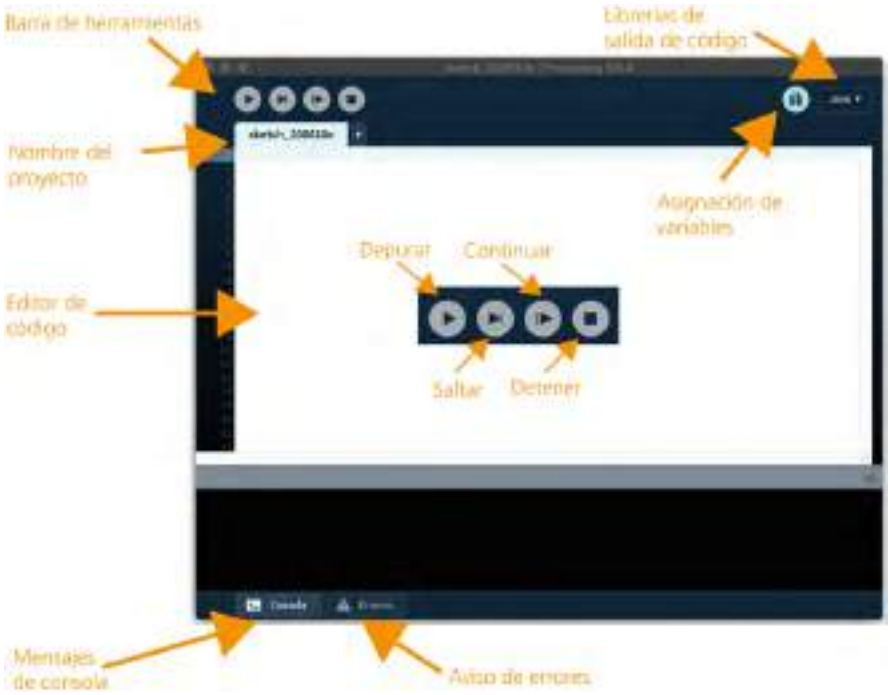
El siguiente manual contiene una breve introducción al *software Processing* y a los conceptos necesarios para entender el proceso de programación. Además de un temario con sesiones en Youtube realizadas por Bruno Díaz, profesor de CENTRO.

Interfaz

Processing es un lenguaje de programación libre, gratuito (*open source*) y en un entorno de trabajo para programar imágenes, animaciones y sonido. Es utilizado por estudiantes, artistas, diseñadores, arquitectos, investigadores, etc. para aprender, prototipar, proyectar y producir contenidos creativos digitales.

- Libre y gratuito
- Escrito sobre Java
- Permite 3 formas de programar: básica, procedural/estructurada y orientada a objetos
- Orientado a *media y screen-based arts*

Processing cuenta con un entorno de desarrollo (o interfaz) simple que permite concentrarse exclusivamente en el conjunto de instrucciones y en la observación de resultados visuales de manera rápida (ventana de dibujo, también llamado lienzo).

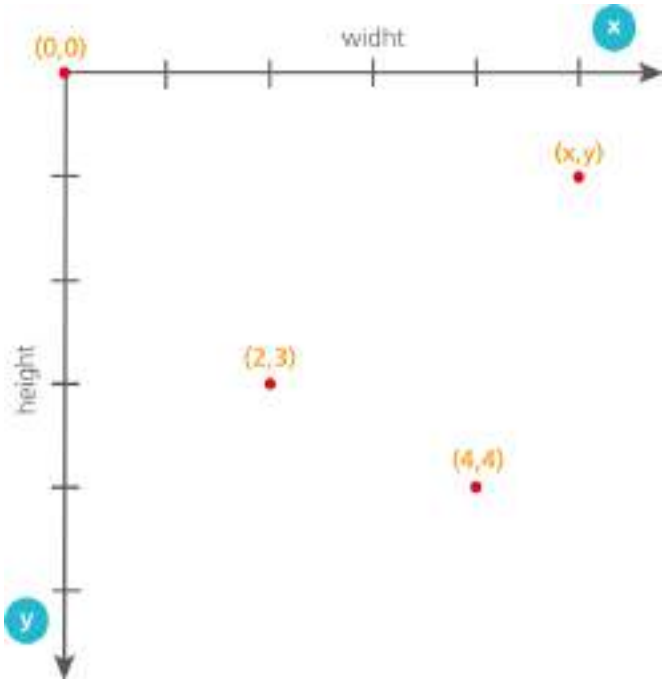


Al ejecutar el código, *Processing* abre una ventana nueva (ventana de dibujo) en donde muestra el resultado del código.

Ventana de dibujo de *Processing* (versión 3.5.4 para Mac). La ventana de dibujo tiene un tamaño de 100x100px., por defecto y el color predeterminado es gris claro.

Pixel

Se refiere a la unidad menor de los que componen una imagen digital. El comando para representar esta unidad es px y debe asignarse en las coordenadas de la ventana de dibujo.



Representación del píxel (px) sobre las coordenadas de la ventana de dibujo.

¿Qué es programar?
Programar es hacer o escribir, algoritmos para manipular datos. Un algoritmo es una lista secuencial de instrucciones finitas que resuelve un problema particular. Está fundado en la premisa del "desarrollo incremental" (dividir en problemas cada vez más pequeños).

Para escribir un algoritmo en *processing* se debe describir detalladamente una tarea específica para que la computadora pueda replicar la acción.

Operación Básica

Variables

Como en todos los lenguajes de programación es imprescindible el uso de datos, así como contar con la posibilidad de almacenarlos y manipularlos. Las variables son espacios reservados que nos permiten guardar en la memoria un valor determinado.
Para ello debemos conocer los tipos de datos que nos ofrece *Processing* según sea su naturaleza:

- Enteros (int): permiten representar valores numéricos de tipo entero.
- Real (float): permiten representar valores numéricos de tipo real que pueden tener decimales.
- Caracteres (char): permite representar caracteres.
- Cadena de caracteres (string): permite representar cadenas de caracteres como palabras o frases.
- Valores lógicos (boolean): permite representar valores de tipo lógico que solo tienen dos posibilidades de verdadero o falso.

¿Qué es una condicional?
Las **condicionales** funcionan para tomar decisiones sobre qué líneas de código deben ejecutarse y cuáles no. La estructura necesaria para tomar estas decisiones es:

```
if (condición) {
acción
}
```

La **condición** debe ser una expresión que se resuelve con verdadero o falso. Si se resuelve como verdadero se ejecuta el código dentro de las llaves. Si la expresión es falsa se ignora este código.

if — el código se ejecuta cuando la condición es verdad.
else — se usa cuando la condición es falsa para que se ejecute una segunda acción.

```
if (condición) {
acción si la condición es verdadera
}else{
acción si la condición es falsa
}
```

Para la función *background* tenemos los siguientes parámetros:

background(rgb)
background(rgb, alpha)
background(gray)
background(gray, alpha)
background(v1, v2, v3)
background(v1, v2, v3, alpha)
background(image)

Parámetros

rgb	int: *any value of the color* / cualquier valor del tipo de datos de color
v1	*float*: *red or hue value (depending on the current color mode)* / valor de rojo o matiz (dependiendo del modo de color actual)
v2	*float*: *green or saturation value (depending on the current color mode)* / valor de verde o saturación
v3	*float*: *blue or brightness value (depending on the current color mode)* / azul o valor de brillo (dependiendo del modo de color actual)
image	*PImage*: PImage *to set as background (must be same size as the sketch window)* / PImage para establecer una imágen como fondo (debe ser del mismo tamaño que la ventana de boceto).

Funciones

Processing se compone de dos funciones o bloques, los cuales son:

- void **setup**()
{
Las instrucciones de esta función se interpretan una sola vez cuando se ejecutan el programa, antes del primer frame. Estas instrucciones se componen de comandos y objetos.
}
- void **draw**()
{
Las instrucciones de esta función se ejecutan una vez por frame(*loop*) hasta que se detenga el programa.
}

Ejemplo

Si deseamos cambiar el tamaño y el color de nuestra ventana de dibujo o lienzo se deberá programar el cambio, es decir, dar la instrucción precisa para que la computadora realice el cambio en el lienzo:

- void **setup**()
{
size(800,600); // Con la función *size* determinamos el tamaño de la ventana.
background(138, 43, 226); // Con la función *background* cambiamos el color de fondo. }
- void **draw**()
{
// En este ejemplo no se utiliza esta función.
}

Código de personalización de lienzo

Lienzo con valores predeterminados

Lienzo personalizado

Representación de la personalización de lienzo

Temario en Youtube

En el siguiente temario se encuentran videotutoriales realizados por Bruno Díaz (***Brunirows Code***) sobre introducción a la programación con *Processing*, donde muestra teoría de programación en diversos ejemplos:

Processing | https://youtu.be/vy3HN3uuhco
Sistema de coordenadas | https://youtu.be/_rj0bHJzStI
Referencias | https://youtu.be/cXF6C7U9sJ0
Comentarios y orden de ejecución | https://youtu.be/Ec_O5GXPT30
Color (Parte 1) | https://youtu.be/WfOFNvHLhsU
Color (Parte 2) | https://youtu.be/Mq8fJVQrEgY
Variables | https://youtu.be/UKLPHV9gkUQ
Variables propias | https://youtu.be/kiwx0610eLs
Porcentaje y Map | https://youtu.be/Al47tPDDQks
Valor retorno Map | https://youtu.be/FHcxbYhSny8
Random | https://youtu.be/SkkRF4NIFno
If | https://youtu.be/kUj_yL5ag-M
Ejercicio pelota rebotando | https://youtu.be/4Cr44nCJWEU
And Or | https://youtu.be/D-OhrrlKIW4
Else | https://youtu.be/lPngvIouFho
Variable booleana | https://youtu.be/PkJL1wj1G2c
While | https://youtu.be/lupMpu_1kHw
For | https://youtu.be/YcLkNSKDJuA
Loops Anidados | https://youtu.be/JkPpneqZ7R0
Funciones Modularidad | https://youtu.be/MYxl5lmjcMs
Funciones Reusabilidad | https://youtu.be/Es8H4i6fXPk
Funciones Valor de Retorno | https://youtu.be/dv3s7xXJfmI
Arrays | https://youtu.be/bIRTX1z8Sz4
Arrays Limites | https://youtu.be/j4LRVMj7rv8
Arrays y Ciclos For | https://youtu.be/tNTwSTBCi3E
Arrays Acceso individual | https://youtu.be/zx2Rnoj-9UI
POO Atributos | https://youtu.be/W5JvI3SbSIw
POO Constructor | https://youtu.be/UCjyyldneE4
POO Constructor con argumentos | https://youtu.be/-Zdci6Xt_qs
POO Métodos | https://youtu.be/TD-kr5rfvJQ
POO Arreglos y Objetos | https://youtu.be/F1x-T066LP4
Transformaciones Translate y Scale | https://youtu.be/bMBzwnwD5cc
Transformaciones Rotate | https://youtu.be/KgmNlw9JxVA
Transformaciones Push y Pop | https://youtu.be/rzaNk7_RISs
Transformaciones Jerarquia | https://youtu.be/jrXMAZTNr10

Glosario

Abstracción (*abstraction*). Se refiere a ocultar detalles de un proceso para centrarse en el resultado. Por ejemplo, en la línea line(x1, y1, x2, y2), la función abstrae las muchas líneas de código necesarias para dibujar una línea en la pantalla. Esto para que el programador pueda centrarse en la posición de la línea.

Algoritmo (*algorithm*). Conjunto de pasos ordenados para lograr un objetivo.

Argumentos. Variable que puede ser recibida por una función o método.

Array. Una colección de elementos de datos referenciados con un nombre. Se accede a cada elemento de acuerdo con el orden dentro de la lista.

Bloque (*block*). Un grupo de código definido mediante llaves coincidentes, los caracteres {y}. Los bloques se usan para agrupar código en clases, funciones a diferencia de *if* y *for que son para* estructuras.

***Bug*.** Un error dentro de un programa que impide ejecutarse, o bien, que el programa se comporta de manera diferente.

Byte. Grupo de 8 bits.

Clase (*class*). Son los bloques de construcción de programación orientada a objetos; un objeto es una instancia de una clase.

Compilar (*compile*). Cuando se ejecuta un programa de procesamiento que es traducido de notación de código a una notación que puede ser ejecutada por una computadora.

Conjunción. Operador que devuelve el valor de verdadero cuando ambas proposiciones son verdaderas.

Condición. Sentencia que se puede evaluar como verdadera o falsa.

DFD. Diagrama de Flujo de Datos.

Disyunción. Operador que devuelve el valor verdadero cuando alguna proposición es verdadera.

Evento (*event*). Una acción como presionar una tecla, el movimiento del *mouse*, o una nueva pieza de datos convirtiéndose disponible para ser leído. Un evento interrumpe el flujo normal de un programa para ejecutar el código dentro de un bloque de eventos.

Expresión (*expression*). Una combinación de operadores, variables y literales. Una expresión siempre tiene un valor que es determinado por sus elementos.

HSB (*Hue, Saturation, Brightness*). Modelo de color (tono, saturación y brillo).

Instrucción. Conjunto de datos insertados en una secuencia estructurada o específica que el procesador interpreta y ejecuta.

Interfaz. Conexión física y funcional entre dos sistemas o dispositivos de cualquier tipo.

Iteración. Acto de repetir un proceso con el objetivo de alcanzar una meta deseada.

Lenguajes de programación. Es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por las computadoras. Ejemplos: Java, Pascal, C, C + +, Lisp, Fortran, Python, MatLab, etc.

Operadores incremento y decremento. Símbolos que sirven para aumentar o disminuir el valor de una variable de manera automática (++, --).

Operadores lógicos. Símbolos que se utilizan para conectar dos o más expresiones y determinar un valor compuesto.

Operadores relacionales. Símbolos que se usan para comparar dos valores (<,>,==,!=).

Pixel. *Picture element.* Es la menor unidad homogénea en color que forma parte de una imagen digital.

Programa. Es una secuencia de instrucciones escritas para realizar una tarea específica con una computadora.

Random. Proceso cuyo resultado no es previsible más que en razón de la intervención del azar.
RGB. Modelo de color rojo, verde, azul (*Red, Green, Blue*).

Sintaxis. Forma visible de un lenguaje de programación que describe las combinaciones posibles de los símbolos que forman un programa sintácticamente correcto.

Variables. Espacio en el sistema de almacenaje (memoria principal de una computadora) y un nombre simbólico que está asociado a dicho espacio.

Bibliografía

Glasnner, A. (2018). *PROCESSING FOR VISUAL ARTISTS : how to create expressive images and interactive art*. CRC Press.

Greenberg, I. (2007). *Processing: creative coding and computational art*. Friends of Ed, an Apress Co.

Greenberg, I. (2009). *The essential guide to Processing for Flash developers*. Springer-Verlag.

Nyhoff, J. & Nyhoff, L. (2017). *Processing : an introduction to programming*. CRC Press.

Reas, C. & Fry, B. (2015). *Getting started with Processing*. Maker Media.

Reas, C. & Fry, B. (2007). *Processing : a programming handbook for visual designers and artists*. MIT Press.

Agradecimiento especial programa STEAM

Iván Abreu

Francisco Aviléz

Sandra Barrón

Roberto Cabezas

Gabriel Charles

Karla Paniagua

Kerstin Scheuch

Gabriela Traverso

CENTRO
DE INVESTIGACIÓN
EN **ECONOMÍA**
CREATIVA

Centro de diseño, cine y televisión
Constituyentes 455, Col. América, Ciudad de México, 11820
T. 2789 9000 | centro.edu.mx