

Damian Barragán Romero
María Paula Barón Aristizábal
Jaime Patarroyo Godoy

A Designer's Approach
to Teaching Programming

Una visión desde el diseño para enseñar programación

Damian Barragán Romero. Dean of the Faculty of Arts and Humanities and Full Professor in the Faculty of Architecture and Design at Universidad de los Andes. An artist and designer, he holds a Master of Arts in Interaction Design from the Interaction Design Institute Ivrea (Italy). Decano de la Facultad de Artes y Humanidades y profesor titular de la Facultad de Arquitectura y Diseño de la Universidad de los Andes, artista y diseñador, egresado de la Maestría en Artes en Diseño de Interacción del Interaction Design Institute Ivrea (Italia).

María Paula Barón Aristizábal. Associate Professor in the Faculty of Architecture and Design. A multidisciplinary designer and creator, she studied design at Universidad de los Andes and earned a master's degree from the Glasgow School of Art in Scotland. Profesora asociada de la Facultad de Arquitectura y Diseño. Diseñadora multidisciplinar, creadora, estudió diseño en la Universidad de los Andes y obtuvo una maestría en la Glasgow School of Art en Escocia.

Jaime Patarroyo Godoy. Associate Professor in the Faculty of Architecture and Design. A Colombian artist, designer, and educator, he is interested in researching and developing new ways of relating to—and through—digital technologies, exploring both their practical and experimental applications. Profesor asociado de la Facultad de Arquitectura y Diseño. Artista, diseñador y educador colombiano, interesado en investigar y crear nuevas formas de relacionarnos con —y a través de— las tecnologías digitales desde sus aplicaciones prácticas y experimentales.

[illegible]

A Designer's Approach
to Teaching Programming

Una visión desde el diseño para enseñar programación

A Designer's Approach
to Teaching Programming

Una visión desde el diseño para enseñar programación

Damian Barragán Romero
María Paula Barón Aristizábal
Jaime Patarroyo Godoy

Universidad de los Andes RAQUEL BERNAL – Rectora
Facultad de Arquitectura y Diseño CLAUDIA MEJÍA – Decana
Departamento de Diseño SANTIAGO DE FRANCISCO – Director

FACULTAD DE ARQUITECTURA Y DISEÑO – Calle 19A n.º 1-37 este, bloque C
Teléfonos: 601 332 4512 - 339 4949, ext. 2480 – Bogotá, D. C. (Colombia)
<http://arqdis.uniandes.edu.co>

ARQDIS®
FACULTAD DE
ARQUITECTURA Y DISEÑO

Nombre: Barragán Romero, Damian, autor. | Barón Aristizábal, María Paula, autor. | Patarroyo Godoy, Jaime, autor.
Título: Una visión desde el diseño para enseñar programación = A Designer's Approach to Teaching Programming / Damian Barragán Romero, María Paula Barón Aristizábal, Jaime Patarroyo Godoy.
Descripción: Bogotá : Universidad de los Andes, Facultad de Arquitectura y Diseño, Departamento de Diseño, Ediciones Uniandes, 2026. | 232 páginas: ilustraciones ; 17,5 x 24,5 cm.
Identificadores: ISBN 978-958-798-979-3 (rústica) | 978-958-798-980-9 (e-book) | 978-958-798-981-6 (e-pub)
Materias: Programación (Computadores electrónicos) | Processing (Lenguaje de programación de computadores)

Clasificación: CDD 006.76–dc23

SBUA

Primera edición: abril del 2026

© Damian Barragán Romero, María Paula Barón Aristizábal y
Jaime Patarroyo Godoy

© Universidad de los Andes,
Facultad de Arquitectura y Diseño,
Departamento de Diseño

Ediciones Uniandes
Carrera 1.ª n.º 18A-12, bloque Tm
Bogotá, D. C., Colombia
Teléfono: 601 339 4949, ext. 2133
<http://ediciones.uniandes.edu.co>
ediciones@uniandes.edu.co

ISBN: 978-958-798-979-3
ISBN e-book: 978-958-798-980-9
ISBN e-pub: 978-958-798-981-6
DOI: <http://doi.org/10.51573/Andes.9789587989793.9789587989809.9789587989816>

Corrección de estilo en inglés y español: Martha Méndez
Diseño y diagramación: Juan David Rendón y Adriana Páramo

Impresión:
Torreblanca Agencia Gráfica S. A. S.
Bogotá, D. C., Colombia

Impreso en Colombia – *Printed in Colombia*

Universidad de los Andes | Vigilada Mineducación. Reconocimiento como universidad: Decreto 1297 del 30 de mayo de 1964. Reconocimiento de personería jurídica: Resolución 28 del 23 de febrero de 1949, Minjusticia. Acreditación institucional de alta calidad, 10 años: Resolución 000194 del 16 de enero del 2025, Mineducación.

Todos los derechos reservados. Esta publicación no puede ser reproducida ni en su todo ni en sus partes, ni registrada en o transmitida por un sistema de recuperación de información, en ninguna forma ni por ningún medio, sea mecánico, fotoquímico, electrónico, magnético, electro-óptico, por fotocopia o cualquier otro, sin el permiso previo por escrito de la editorial.

p. 11	The Processing Working Environment Entorno de trabajo de Processing	1
p. 17	Form and Color Forma y color	2
p. 31	Variables and Repetition Variables y repetición	3
p. 53	Movement Movimiento	4
p. 69	Interacting with the Mouse Interacción con el mouse	5
p. 87	Interacting with the Keyboard Interacción con el teclado	6
p. 95	Data Structures: Arrays Estructuras de datos: arreglos	7
p. 113	Functions Funciones	8
p. 125	Classes and Objects Clases y objetos	9
p. 145	States Estados	10
p. 155	Basic 3D Drawing Dibujo 3D básico	11
p. 169	Images Imágenes	12
p. 185	Time Tiempo	13
p. 193	Vector Files (PDF export) Archivos vectoriales (exportar archivos PDF)	14
p. 203	Vector Files (Loading) Archivos vectoriales (cargar)	15
p. 211	Data (Generating, Loading, Visualizing) Datos: generar, cargar y visualizar	16
p. 223	Conclusions Conclusiones	C



[Repositorio-ArqDis /designprogramming](#)

This book is designed for educators in STEM areas, with previous knowledge in programming languages, such as Java, or experience using Processing, who are interested in a curriculum to teach computer programming, for beginners of all ages, and people who might not necessarily have a technological or engineering background. The book proposes a base, lesson-by-lesson script, which is also modular, and extensible over time. These features allow educators to adjust their exercises and curricula to the needs, frequency, demands, and particularities of different learning environments—from a short ice breaker programming workshop, to a regular class, distributed across a couple of months.

This text is the outcome of hundreds of workshops and class sessions that helped us identify sensitive exploration vehicles that contribute to motivation, fun, and resilience in these audiences' learning process. They help relieve the frustration that arises when learning how to program, allowing the visual and interactive aspects to emerge from the beginning. At the end of this trip, the students are expected to develop programming skills at a threshold level that enables them to go beyond the topics dealt with, and delve into more advanced subjects confidently.

The Processing material is available online at processing.org. From there you can download the software for the different platforms, and access the complete language reference, along with examples and tutorials.

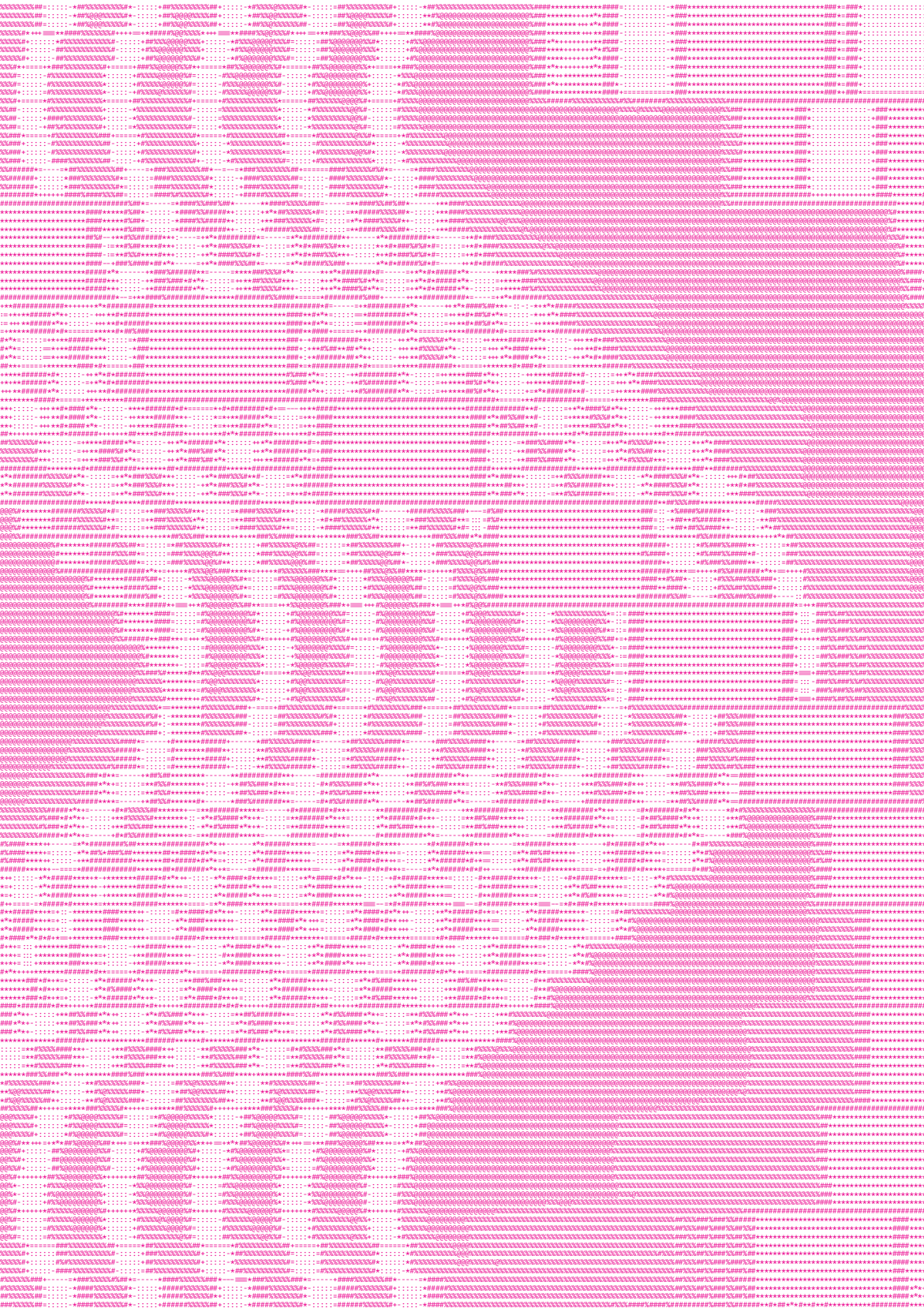
Este libro está diseñado pensando en educadores de áreas STEM, con conocimientos previos de programación en lenguajes como Java o con experiencia usando [Processing](#), que están interesados en un currículo para enseñar programación orientado a principiantes de todas las edades y, en general, a personas que no necesariamente vienen de disciplinas tecnológicas o de la ingeniería. El libro plantea un guion base para el docente por lecciones, modular, extensible en el tiempo, que le permite diseñar ejercicios y currículos adaptados a las necesidades, frecuencia, exigencia y particularidades de diferentes entornos de aprendizaje, desde un taller de unas cuantas horas para romper el hielo con la programación, hasta una clase distribuida a lo largo de un periodo de meses.

Este material es el resultado de la realización de cientos de talleres y sesiones de clase que nos ayudaron a identificar vehículos sensibles de exploración que contribuyen a la motivación, la diversión y la resiliencia en el proceso de aprendizaje de estas audiencias. Gracias a ellos, es posible aliviar la frustración que surge cuando se aprende a programar, y los aspectos visual e interactivo emergen desde el principio. Se espera que al final de este viaje los estudiantes desarrollen las competencias en programación en un nivel de umbral que les permita explorar más allá de los temas planteados y profundizar con mayor seguridad en temas más avanzados.

Todo el material de Processing se encuentra disponible en línea en processing.org. Desde allí es posible descargar el *software* para las diferentes plataformas y tener acceso a la referencia completa del lenguaje, así como a ejemplos y tutoriales.

The Processing Working Environment

Entorno de trabajo de Processing



The best way to start is by giving students a general context of the tool and the environment in which they are going to work, so that they can understand its importance. This is a suitable time to talk about John Maeda's Design By Numbers, which precedes Processing, and about other Processing Foundation projects, such as p5.js or Processing.py. The students may have different approaches to the environment they are about to work with, and we must be clear about the reasons why we prefer one software or programming language over another.

Why Processing? For us, as authors, Processing is still the best tool for beginners of all ages for four main reasons:

1. It is a free, open-source tool.
2. It gives us a guarantee of stability. For someone who is just starting to program it is very difficult to determine whether an error is his or her fault, or it is due to the malfunctioning of the tool they are using. With twenty years of experience, Processing offers us a stability that is difficult to match by younger developments which may have other advantages, such as portability or web compatibility.
3. Its documentation is very robust, as it has been documented freely and voluntarily by the community for many years, which has allowed a lot of information to become available.
4. Its visual approach helps those coming from creative disciplines understand some concepts more quickly.



Download and install Processing

Introduce the environment and the application interface

Introduce the 2D coordinate system

La mejor forma de comenzar es dando a los estudiantes un contexto general de la herramienta y el ambiente en el que vamos a trabajar, lo cual permite entender su importancia. Este es un buen momento para hablar de Design by Numbers, de John Maeda, que precede a Processing, y de otros proyectos de Processing Foundation, como p5.js o Processing.py. Los estudiantes pueden tener diferentes acercamientos al medio con el que nos disponemos a trabajar, y debemos ser claros sobre por qué preferimos un *software* o un lenguaje de programación y no otro.

¿Por qué Processing? Los autores consideramos que Processing sigue siendo la mejor herramienta para principiantes de todas las edades por cuatro razones principales:

1. Es una herramienta gratuita y de código abierto.
2. Brinda una garantía de estabilidad. Para quien está empezando a programar es muy difícil distinguir cuándo un error es culpa suya o del programa con el que está trabajando. Los veinte años de Processing nos ofrecen una estabilidad que es difícil de igualar por desarrollos más jóvenes que pueden tener otras ventajas, como portabilidad o compatibilidad web.
3. Su documentación es muy robusta. Esto obedece a que el programa lleva muchos años siendo documentado de forma gratuita y voluntaria por la comunidad, lo que ha permitido que exista mucha información disponible.
4. Su enfoque visual les permite a quienes vienen de disciplinas creativas entender algunos conceptos más rápidamente.



[Descargar e instalar Processing](#)

[Presentar el ambiente y la interfaz de la aplicación](#)

[Presentar el sistema de coordenadas 2D](#)

How to know if your students are ready for the next lesson?

Technical component

- ☐ Do the students understand the screen coordinate system and make decisions based on it?

Competencies

- ☐ Do they understand the set of processes and tools within a programming environment?
- ☐ Toolbox: keywords, definitions, QR, links or resources that can help you deepen a theme or overcome obstacles.

[illegible]

¿Cómo saber si los estudiantes están listos para la siguiente lección?

Aspecto técnico

- ☐ ¿Comprenden los estudiantes el sistema de coordenadas de la pantalla y pueden tomar decisiones basados en él?

Competencias

- ☐ ¿Comprenden el conjunto de procesos y herramientas del entorno de programación?
- ☐ Caja de herramientas: palabras clave, definiciones, QR, hipervínculos o recursos que pueden ayudar a profundizar en el tema o superar obstáculos.



Video lesson contents
**Contenidos de
la lección en video**

[Environment and Interface](#) [Entorno e interfaz](#)
[Coordinate System](#) [Sistema de coordenadas](#)

Additional Resources
Recursos adicionales

[Welcome to Processing](#) [Bienvenidos a Processing](#)
[Hello Processing](#) [Hola Processing](#)

Toolbox
Caja de herramientas

keywords, definitions, QR, links or resources
that can help you deepen a theme or
overcome obstacles.

palabras clave, definiciones, QR,
hipervínculos o recursos que pueden ayudar a
profundizar en el tema o superar obstáculos.



Downloading and Installing Processing
Descargar e instalar Processing



Processing Environment and Interface
Entorno e interfaz de Processing

LESSON / LECCIÓN

Form and color

Forma y color

2

Introduce comments in the code, in one line `//` and in multiple lines `/* */`.

Introduce syntax, and how syntax highlighting and error checking are done by Processing.

Introduce the `size()` command.

Introduce the `point(x, y)` command.

Student Activities

1. Draw 5 points on the screen.

Introduce the `line(x1, y1, x2, y2)` command.

2. Draw 5 lines, one of them horizontal, and another, vertical.

Introduce the `rect(x, y, w, h)` command.

3. Draw 5 rectangles.

Introduce the RGB color scheme and the color commands

`fill()`, `stroke()`, `noFill()`, `noStroke()`.

4. Change the color of the above drawings.

Introduce the `ellipse(x, y, w, h)` command.

5. Draw 5 ellipses.

Introduce transparency in color.

6. Change the color transparency of the above drawings.

Introduce the `background()` command.

Introduce the `triangle(x1, y1, x2, y2, x3, y3)` command.

7. Draw 5 triangles, each one of a different color.

Introduce the `quad(x1, y1, x2, y2, x3, y3, x4, y4)` command.

8. Draw 5 quads, each one of a different transparency.

Introduce polygons, the `beginShape()`, `vertex()`, and `endShape()` commands.

9. Draw an eight-sided polygon.

Introduce los comentarios dentro del código, en una línea `//` y en múltiples líneas `/* */`.

Presenta la sintaxis, y cómo Processing resalta la sintaxis y verifica si hay errores.

Presenta el comando `size()`.

Presenta el comando `point(x, y)`.

Actividades para el estudiante

1. Haz 5 puntos en la pantalla.
Presenta el comando `line(x1, y1, x2, y2)`.
2. Dibuja 5 líneas; una de ellas debe ser horizontal y otra, vertical.
Presenta el comando `rect(x, y, w, h)`.
3. Dibuja 5 rectángulos.
Presenta el sistema de color RGB y los comandos de color `fill()`, `stroke()`, `noFill()`, `noStroke()`.
4. Cambia el color de los dibujos anteriores.
Presenta el comando `ellipse(x, y, w, h)`.
5. Dibuja 5 elipses.
Presenta la transparencia en el color.
6. Cambia la transparencia del color de los dibujos anteriores.
Presenta el comando `background()`.
Presenta el comando `triangle(x1, y1, x2, y2, x3, y3)`.
7. Dibuja 5 triángulos, cada uno de diferente color.
Presenta el comando `quad(x1, y1, x2, y2, x3, y3, x4, y4)`.
8. Dibuja 5 cuadriláteros, cada uno con diferente transparencia.
Presenta los polígonos, y los comandos `beginShape()`, `vertex()` y `endShape()`.
9. Dibuja un polígono de 8 lados.

Explain how instructions are executed sequentially by the computer. The order in the sequence alters the outcome, even if sometimes it is not self-evident. Good examples include drawing overlapping shapes, so that the latest one covers the previous ones visually; also, choosing a color before, and then drawing a shape.

This is a suitable time to introduce the Processing reference and Processing book as supplementary tools of the educational process. During the class sessions, we will hardly cover all the commands and possibilities of Processing. Therefore, it is very important for students to develop skills that allow them to look for their own tools and find out how to use them.

Give students time to search, test, and use the following commands in the reference or the book:

The curve() command and draw a curve.

The bezier() command and draw a Bezier curve.

The arc() command and draw an arc.

Usually, curves are not easy to understand, so it is worth it reviewing them as a group. If available, applications such as Adobe Illustrator can be used, as an example of how anchoring and control points work.



This is a suitable time to approach and learn about the work of surreal artists. For the next exercise, we invite you to review the work of artists such as Paul Klee or Max Ernst.

Explica cómo la computadora ejecuta secuencialmente las instrucciones; el orden en la secuencia altera el resultado, aunque a veces no sea evidente. Algunos buenos ejemplos son dibujar formas superpuestas, de manera que la última que se dibuje cubra visualmente las anteriores, o seleccionar el color antes y después de dibujar una forma.

Este es un buen momento para presentar la referencia y el libro de Processing como herramientas complementarias del proceso educativo. Las clases son un espacio en el que difícilmente se alcanzan a cubrir todos los comandos y posibilidades del *software*, por esto es muy importante que los estudiantes desarrollen habilidades que les permitan buscar sus propias herramientas y descubrir cómo utilizarlas.

Deles tiempo para que busquen, prueben y usen los siguientes comandos en la referencia o el libro:

Comando `curve()` y dibujar una curva.

Comando `bezier()` y dibujar una curva Bezier.

Comando `arc()` y dibujar un arco.

Las curvas por lo general no son fáciles de entender, y vale la pena repasarlas de forma colectiva. Dado el caso, se pueden usar aplicaciones como Adobe Illustrator para explicar cómo funcionan los puntos de anclaje y de control.



Este es un buen momento para acercarse y conocer la obra de artistas surrealistas. Los invitamos a revisar el trabajo de artistas como Paul Klee o Max Ernst para el siguiente ejercicio.





```
// Fragment of Paul Klee's Witch scene
// https://www.wikiart.org/en/paul-klee/witch-scene-1921
```

```
size(500, 500);
fill(28, 24, 13);
rect(0, 0, 500, 500);

noStroke();

// Light cones
// Left
fill(69, 57, 41);
triangle(0, 500, 423, 92, 302, 500);
// Right
fill(58, 49, 34);
quad(426, 92, 500, 146, 500, 500, 457, 500);

// Wolf body
beginShape();
fill(157, 106, 51);
vertex(127, 500);
vertex(193, 113);
vertex(218, 113);
vertex(247, 309);
bezierVertex(243, 330, 288, 340, 265, 336);
vertex(394, 350);
bezierVertex(407, 352, 403, 367, 404, 358);
bezierVertex(399, 368, 378, 375, 392, 370);
vertex(361, 371);
bezierVertex(349, 373, 349, 397, 349, 385);
vertex(362, 402);
vertex(211, 369);
vertex(211, 407);
vertex(249, 412);
vertex(312, 413);
vertex(322, 420);
vertex(320, 432);
vertex(275, 431);
vertex(278, 500);
vertex(127, 500);
endShape();
```

```

// Wolf dress
beginShape();
fill(211, 152, 93);
vertex(80, 500);
vertex(115, 390);
bezierVertex(114, 413, 141, 438, 131, 426);
bezierVertex(154, 433, 190, 422, 172, 427);
bezierVertex(198, 421, 220, 443, 209, 432);
bezierVertex(227, 455, 236, 485, 230, 471);
bezierVertex(238, 490, 283, 490, 259, 490);
bezierVertex(299, 490, 313, 490, 305, 490);
vertex(304, 500);
vertex(80, 500);
endShape();

// Wolf eyes
stroke(91, 70, 54);
strokeWeight(2);
fill(243, 225, 210);
// Left eye
ellipse(230, 316, 30, 30);
line(215, 312, 241, 319);
line(218, 305, 239, 328);
line(216, 324, 236, 308);
line(226, 330, 228, 303);
// Right eye
ellipse(259, 319, 24, 24);
line(248, 325, 263, 310);
line(255, 331, 255, 308);
line(266, 331, 247, 312);
line(246, 319, 271, 322);

// Wolf teeth
// Upper teeth
triangle(225, 373, 220, 387, 236, 376);
triangle(236, 376, 229, 392, 251, 379);
triangle(306, 391, 294, 400, 322, 395);
triangle(322, 394, 314, 405, 336, 398);
triangle(342, 399, 337, 409, 352, 401);
// lower teeth
triangle(255, 409, 244, 388, 270, 409);
triangle(269, 409, 253, 386, 281, 409);
triangle(281, 409, 271, 388, 291, 409);
triangle(291, 409, 284, 387, 301, 409);
// Wolf nose
fill(196, 143, 85);
strokeWeight(1);
quad(328, 355, 365, 357, 365, 362, 330, 360);
quad(393, 350, 387, 359, 398, 367, 405, 356);

```

```

// Wolf hair
beginShape();
strokeWeight(2);
stroke(41, 28, 15);
fill(87, 68, 36);
vertex(185, 156);
vertex(193, 113);
vertex(218, 113);
vertex(224, 156);
bezierVertex(220, 168, 200, 168, 205, 168);
vertex(185, 156);
endShape();

// Fragments of creature on the right
fill(58, 49, 34);
strokeWeight(1);
stroke(41, 28, 15);
quad(500, 242, 457, 257, 456, 270, 500, 278);

strokeWeight(2);
fill(221, 179, 133);
quad(456, 270, 452, 302, 500, 301, 500, 278);

beginShape();
vertex(500, 326);
vertex(451, 371);
bezierVertex(446, 387, 467, 404, 456, 395);
vertex(473, 402);
vertex(500, 402);
endShape();

beginShape();
vertex(500, 353);
bezierVertex(485, 365, 485, 373, 490, 380);
vertex(500, 393);
endShape();

// Moon
beginShape();
fill(65, 53, 42);
noStroke();
vertex(288, 24);
bezierVertex(316, 18, 301, 23, 308, 18);
vertex(322, 0);
vertex(351, 0);
bezierVertex(351, 17, 332, 31, 341, 24);
bezierVertex(320, 33, 306, 33, 313, 33);
vertex(288, 24);
endShape();

```

ASSIGNMENT 1. Choose a squared fragment from a painting by Paul Klee selected beforehand and reproduce it using the graphic primitives you learned.

Why Paul Klee? First, because we love his work. Second, because his use of geometric shapes and colors make it an excellent candidate for this assignment. Search for the term “Paul Klee paintings” on Google, and you will get an amazing and extensive set of his paintings. Try other surrealist artists—i. e. Max Ernst—, and you will also find a wide array of possibilities. Consider this an opportunity for introducing the selected artist in the context of the class.

Why this exercise? Designing something very specific (an expected outcome or brief) versus the happy accident.

There are different processes involved in developing this activity:

- ▶ Choosing which section to reproduce, and determining the complexity, the amount of content, forms, color, and aesthetic factors. The final solution should resemble the assigned artwork section, making it easy to assess the correspondence between the original and the current representation. It is very important for beginners to learn how to break down a problem into smaller subtasks and developing strategies to achieve a goal.
- ▶ Understanding screen coordinates. Mapping the chosen squared area of the painting to the actual screen 2D coordinate system is a very important step when parametrizing.
- ▶ Abstracting. This skill is required when choosing how to reproduce the shapes and colors of the original painting and which software primitives to use for drawing a section, its position, size, color, and so on.
- ▶ Familiarizing the student with the programming language syntax by repetition.
- ▶ Familiarizing the student with searching the language reference, learning the programming environment, and finding online learning resources.

TAREA N.º 1. Elige un fragmento cuadrado de una pintura de Paul Klee seleccionada y reproducéla utilizando las primitivas gráficas aprendidas.

¿Por qué Paul Klee? Primero, porque nos encanta su trabajo. Segundo, porque su uso de formas geométricas y colores lo convierten en un excelente candidato para este ejercicio. Al buscar en Google los términos “pinturas de Paul Klee” obtendrás un conjunto increíble y extenso de imágenes de sus obras. Prueba con otros artistas del periodo surrealista, como Max Ernst, y también encontrarás una amplia selección de posibilidades. Considera esta una oportunidad para presentar en el contexto de la clase al artista que seleccionaste.

¿Por qué este ejercicio? Diseñar algo muy específico (un resultado esperado o encargo) versus la feliz coincidencia.

Hay diferentes procesos involucrados en el desarrollo de este ejercicio:

- ▶ Elegir qué sección reproducir y determinar la complejidad, la cantidad de contenido, las formas, el color y los factores estéticos. La solución final debe ser muy cercana a la sección de la pintura asignada, para que sea más fácil evaluar la correspondencia entre el original y la representación digital. Esto es muy importante para principiantes: desarrollar estrategias y dividir un problema para lograr una meta.
- ▶ Comprender las coordenadas de la pantalla. Mapear el área cuadrada elegida de la pintura al sistema 2D de la pantalla real es un paso muy importante a la hora de parametrizar.
- ▶ Abstraer. Se requiere a la hora de elegir cómo reproducir las formas y los colores de la pintura original y qué primitivas del lenguaje utilizar para representar una sección, su posición, tamaño o color, entre otras características.
- ▶ Familiarizar al estudiante con la sintaxis del lenguaje de programación mediante la repetición.
- ▶ Familiarizar al estudiante con la investigación en la referencia del lenguaje, el aprendizaje del entorno de programación y la búsqueda de recursos de aprendizaje en línea.

✓ CHECKPOINT

How to know if your students are ready for the next lesson?

Technical component

- ☐ Do the students understand the syntax of the language and how Processing performs syntax highlighting and checks syntax errors?
- ☐ Do the students understand how the `beginShape()`, `vertex()`, and `endShape()` commands work? Can they make decisions about when and how it is appropriate to apply them?
- ☐ Are they able to build and use geometric shapes and color in general?
- ☐ Do the students understand commands related to position, size and color? Can they make decisions about when and how it is appropriate to apply them?

Competencies

- ☐ Can the students develop strategies to achieve a goal? Can they break it down and translate it into instructions that are executed sequentially?
- ☐ Can the students abstract a conceptual model that allows them to build and reuse code to simplify their work?
- ☐ Do the students understand how to consult the language and reference of the Processing libraries and others to overcome obstacles?