PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

FACULTAD DE INGENIERIA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TRABAJO DE TITULACIÓN

PROPUESTA DE TALLER INTEGRAL PARA ARTISTAS VISUALES Y PROGRAMADORES FUNDAMENTADO EN LA PROGRAMACIÓN COMO MEDIO CREATIVO

AUTOR:

ANDREA MISHELL YÁNEZ COELLO

TUTOR:

SUYANA FABIOLA ARCOS VILLAGÓMEZ

QUITO, DICIEMBRE 2023

DEDICATORIA

Dedico este trabajo principalmente a mi familia, quienes han sido sumamente comprensivos conmigo, brindándome su apoyo incondicional tanto en mis aciertos como en mis errores, siempre alentándome a levantarme y seguir adelante en los diferentes aspectos de mi vida. A mis padres por su sabiduría, a mi hermana por la tenacidad que inspira, a mi hermano, de quien soy responsable y es la razón por la que cada día trato de ser un poco mejor, a mi tía que siempre trato de enseñarme aquello que se me dificultaba, y finalmente a mis abuelas, que cuidaron de mi desde la infancia y continúan haciéndolo hasta el día de hoy.

A los amigos que encontré en este camino, sobre todo a mi mejor amiga y su familia, quienes siempre me recibieron cálidamente en su hogar y a ella, que sin su apoyo y amistad no hubiera aprendido ni la mitad de las cosas que he interiorizado durante esta etapa, aceptándome en mis momentos más felices como en los más difíciles.

A mi maestro, que me ayudo incondicionalmente con sus conocimientos y consejos, a pesar de mi inexperiencia, aún me queda mucho por aprender y dominar hasta crear algo con el cual hacerlo sentir orgulloso.

Finalmente, a mi doctora, quien me ha ayudado a entender mi propia mente y cognición para encaminarme a tomar mejores decisiones durante este último año.

AGRADECIMIENTO

A mi institución, la Pontificia Universidad Católica del Ecuador, a los docentes de la Facultad de Ingeniería, quienes constantemente demuestran su capacidad de formación con las nuevas generaciones por sus conocimientos, experiencias, metodologías y apoyo hacia sus alumnos.

A la tutora de esta investigación, la Ing. Suyana Arcos, por la guía que ha supuesto para alinear el desarrollo de este trabajo con las sugerencias con las que me ha provisto, su preocupación, dedicación de tiempo y la paciencia con la que recibió cado uno de los avances.

Al decano y al secretario abogado quienes gustosamente me indicaron el camino para regresar y volver para concluir esta carrera, demostrando una empatía y comprensión que alivio muchas de mis preocupaciones.

Finalmente, agradezco al resto del personal administrativo y encargados que se ocupan del correcto funcionamiento diario de las instalaciones del campus, gracias a su trabajo y sacrifico es que podemos contar un entorno de aprendizaje tan productivo en el que desenvolvernos.

RESUMEN

El presente trabajo se centra en la redacción y recopilación de los componentes necesarios para llevar a cabo un taller integral práctico sobre la aplicación del lenguaje p5.js en la generación dinámica de dibujos a través de la programación, combinando metodologías y recursos de aprendizaje utilizados en el campo de las artes e informática y dirigido a estudiantes de estos mismos campos.

El objetivo del taller es aportar a la falta de literatura que se ha registrado sobre la combinación de estos temas, uniendo los paradigmas de pensamiento y conceptos que históricamente han separado a las personas afines al arte de las personas más técnicas, llevando a cabo actividades en las que unos aprendan de los otros y viceversa; ejercicios de programación, de bosquejo y dibujo creativo, actividades recreativas, introspectivas y recursos digitales de análisis como libros y documentales.

Por último, se llevan a cabo dinámicas de recopilación y socialización de resultados, experiencias y sugerencias sobre el desempeño del taller para mejorar próximas iteraciones o complementarlo.

La fundamentación para construir el esquema del taller recopila sus bases teóricas de libros que precisamente tratan sobre ese formato de aprendizaje, libros sobre historia del arte, referencias y ejemplos de la documentación oficial de la fundación Processing, y el trabajo de investigación de Tega & Golan sobre programación creativa y su aplicación en talleres alrededor del mundo.

Palabras clave: Programación creativa, p5.js, taller integral, dibujo.

ABSTRACT

The following research aims to recompile key elements to build and run a workshop with a practical approach of the p5.js language to draw dynamic concepts trough coding, applying various types of learning styles and resources commonly used among arts and computer engineering students.

What set this work's goal was the lack of workshops or documented classes that combine those fields, their styles of thinking and theorical concepts, at least, on the Spanish language, something that could bring together art like-minded students with tech like-minded ones to learn one from the other through various activities; programing, sketching, drawing, retrospective and introspective activities, etc. Using the digital resources stored in the repository classroom.

At last, in the final stage of the workshop, quizzes and expositions are carried on discussing discoveries, outputs and suggestions for both the exercises and the content of the workshop itself for future iterations.

The workshop's outline compiles its theoretical bases from resources on how to lead, design and run effective workshops, books on art history to complement the catches of information described along the exercises, examples from the official site of the Processing foundation, and Tega & Golan's work about creative programming around the world.

Key words: Creative coding, p5.js, workshop, drawings.

ÍNDICE DE CONTENIDOS

ÍNDICE DE EJERCICIOS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

CAPÍTULO I: DISEÑO Y PLANIFICACIÓN DEL TALLER

1.1. Propuesta del taller

1.1.1. Objetivos

Objetivo General

Crear y estructurar un taller que instruya a estudiantes de arte e informática en la programación con p5.js para crear gráficas interactivas.

Objetivos Específicos

- Explorar conceptos de diseño abstractos, naturales y su significado.
- Organizar los elementos que conformaran el taller y redactarlos para su práctica.
- Aprender a crear dibujos y replicar imágenes fácilmente programando en p5.js
- Entregar una propuesta de taller integrando elementos transformativos y prácticos.

1.1.2. Enfoque

Los recursos para aprender a programar gráficas orientadas las personas no técnicas son escasos incluso en la actualidad, mientras que en los currículos de arte digital se integra a la programación como un módulo innovador lo cierto es que, para la mayoría de los artistas, demanda mucho tiempo aprender a escribir complicadas instrucciones de código con los lenguajes de programación convencionales, pero al mismo tiempo lo necesitan para sus objetivos específicos i.e. eliminar el trabajo redundante o trazar algún patrón iterativo de diseño. Los métodos de enseñanza utilizados para los programadores rara vez funcionan entre estudiantes de arte y diseño pues los ejercicios contenidos en cálculos matemáticos e impresos en texto plano no llaman la misma atención que las composiciones, colores, trazados, fondos o retratos (G. Levin & T. Brain, 2021).

Al no haber material suficiente que aborde la programación como medio artístico, Jhon Maeda, un diseñador, ingeniero y en ese entonces investigador en el MIT, presento en su libro *Design*

by Numbers un pequeño lenguaje de programación que en su mayoría era una reinterpretación de instrucciones de Logo y BASIC para crear figuras geométricas de forma simple a la vez que explicaba conceptos de composición visual con figuras elementales. El objetivo de Maeda era iniciar a gente de otras especialidades, especialmente a dibujantes y diseñadores, dentro de la programación a la vez que proponía una filosofía de impulsarla como un medio creativo en sí mismo (MIT Press, 2023).

Inspirados por su tutor y su lenguaje, sus pupilos Ben Fry y Casey Reas, quienes se encargaron de darle mantenimiento durante años, diseñaron uno nuevo y lo nombraron Processing, su objetivo era programar dibujos tipo *sketch* a través de Java pero más fácil que el lenguaje *DBN*, su comunidad sigue activa y ha sido instruido en planes de estudio de arte y humanidades en todo el mundo, finalmente en 2012 la fundación Processing lanzo su proyecto más importante de la mano de Lauren Lee McCarthy que retroactivamente se basaba en el trabajo de Reas y Fry, el lenguaje p5.js (TRCC, 2022).

Se trata de una librería de JavaScript con la que se puede demostrar una mejor manera de plasmar ideas de diseño fácilmente en código mientras que el esfuerzo real va hacia la recolección de técnicas y estrategias de enseñanza que funcionen para los estudiantes de arte, quienes requieren de un lenguaje fácil de aprender, y que también llame la atención de los programadores a quienes el campo las artes visuales resulta nuevo e incomprensible, además de que esta unión responde a un proceso de establecimiento de puentes de comunicación y democratización de conocimientos, reforzando lazos de expresión, comunión, empatía y conocimiento.

1.1.3. Fundamentos teóricos y prácticos

Llevar a la práctica un taller puede parecer relativamente similar a impartir una clase tradicional pero no por ello es más fácil, se debe diseñar y estructurar con mucha anticipación los componentes que lo conformaran, estos se identifican según los objetivos y el estilo de aprendizaje que mejor se acople a la naturaleza de su tema, a partir de eso también se identifican métodos de evaluación, liderazgo y conducta con los alumnos, *The Workshop Book* explora cada uno de esos elementos y los pasos que dar de modo que la fundamentación y redacción de la propuesta sea adecuado y basado en conceptos pedagógicos.

En cuanto al contenido práctico que respalda el taller, los ejercicios curados de *Code as a Creative Medium* son ideales al no enfocarse tanto en el paradigma tradicional de los ejercicios de programación sino en explorar *que* se puede crear y *por qué* hacerlo, describen patrones de diseño reminiscentes de textiles, colores, ritmos, simetrías, etc. Hasta conceptos más abstractos como el tiempo, las expresiones faciales, la composición del terreno, etc. Golan y Tega (2021) recopilan en su libro más de 30 años de exploración sobre educación de arte y programación a lo largo de varios festivales como la Eyeo Festival, la "Code + Ed" Summit y el Creative Coding Festival, el libro es de los primeros en su tipo en recopilar esos ejercicios para las nuevas generaciones y hay material suficiente como para impartir un curso, taller, módulo o certificado.

Por último, el elemento implícito para sobrellevar esos ejercicios es el propio lenguaje p5.js como librería de JavaScript y reinterpretación de su predecesor Processing, para crear las imágenes descritas en el libro de Golan y Tega el manual de introducción al lenguaje que Lauren Lee McCarthy creo también tiene un papel significativo, se suman también los ejemplos publicados en la página oficial de la fundación Processing en la cual también hay recursos

como referencias, libros y grabaciones de los talleres sobre p5.js impartidos alrededor del mundo de los cuales también se pueden extraer ideas para complementar la propuesta.

1.2. Logística del evento

Para Hamilton (2016) un taller es un trabajo colaborativo orientado hacia un mismo objetivo, como lo es en este caso el impulsar la creatividad a través de la unión de dos campos que en la superficie parecen tan disparejos, durante ese proceso es importante considerar una planificación y preparación previa de modo que dicho objetivo sea cumplido aprovechando al máximo los tiempos establecidos.

1.2.1. Espacio de trabajo

En primer lugar, se establece que las sesiones sean de carácter presencial pues la tecnología remota se agradece, pero da más espacio a la posibilidad de caer en distracciones, si el objetivo es unir a las disciplinas de arte y programación es indispensable que sus estudiantes se encuentren en un mismo lugar para su colaboración, además de que resulta beneficioso alejarse de los contextos y dispositivos que obstruyan la capacidad de auto reflexión y análisis.

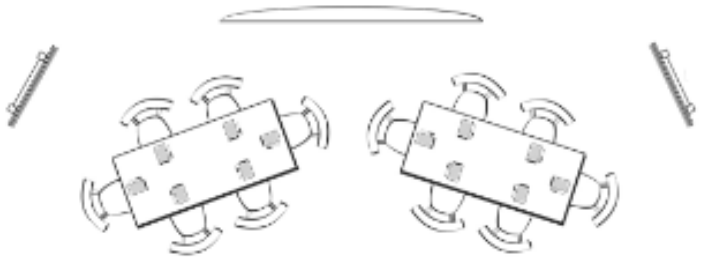
En segundo lugar, está hacer que el lugar de encuentro sea lo menos reminiscente a los espacios convencionales de trabajo o estudio, lugares monótonos como una oficina o aula de clases recuerdan a hábitos recurrentes, Hamilton (2016) recomienda decorar el espacio con elementos reminiscentes al tema del taller pues utilizar la memoria visual en lugar de la verbal hará a la experiencia más inmersiva y memorable:

- Collages de fotos sobre la cotidianeidad.
- Descomposición de objetos en figuras elementales en plantillas.
- Disección de los fundamentos del arte en mapas conceptuales.
- Las construcciones más revolucionarias de la arquitectura descritas en láminas.
- Tener música ambiental de fondo de vez en cuando.

Por otro lado, siempre se debe tener en cuenta al taller como una invitación, por tanto, se debe reservar y preparar la acomodación de los puestos de trabajo por lo menos con una hora de anticipación al inicio de cada sesión, de forma que los estudiantes sean bienvenidos a un lugar limpio y organizado, a excepción de la acomodación de mesas a la cual se le permite un estilo un tanto informal en favor de la movilidad y acercamiento a la pizarra:

Figura 1

Acomodación de mesas estilo cabaré con 4-6 asientos por mesa



Fuente: *The Workshop Book* (p.32) P. Hamilton, 2016, Pearson Business.

- La acomodación cabaré posiciona las mesas cerca de la pizarra y los murales decorados.
- Hay espacio suficiente para desplazarse fácilmente.
- Cabrían entre 4-6 puestos por mesa.

Por último, respecto a los recesos, se recomienda hacer que, en lo posible, la mayoría del grupo se mantenga en la misma sala, aquello hará que los estudiantes tengan más posibilidades de socializar entre sí, además de que alejarse de la sala durante mucho tiempo crea cierta desconexión que es difícil de recuperar al regreso. En conclusión, si bien se requiere algo de presupuesto en la preparación del espacio de trabajo, no es necesario invertir demasiado para construir un ambiente inmersivo, eso depende del ingenio y el correcto uso de los materiales.

1.2.2. Material requerido

Los insumos necesarios que cuentan como parte de la inversión inicial del taller, todos, a excepción de los ordenadores, serán provistos a los participantes:

- **Proyector** para transmitir lo que haga el tutor desde su ordenador hacia la pizarra.
- **Pizarra desplazable** necesaria para ilustrar los contenidos del taller.
- **Marcadores y borradores** de pizarra, los primeros deben tener su tinta lo más fresca posible, contar con varios colores y estar en condiciones para hacer anotaciones que sean visibles para todos.
- **Mesas de trabajo** con tamaño suficiente para ubicar 2 puestos en cada lado y otros 2 puestos a los extremos.
- **Libretas A5** anilladas, con 100 hojas de cuadros, suficientes para cubrir todas las clases del taller, también se puede usar para dibujar los bosquejos de los ejercicios.
- **Lápices y esferos** para apuntar ideas sobre la resolución de ejercicios y las anotaciones sobre las instrucciones del lenguaje.

Por último, cada participante deberá traer su propio ordenador portátil, no es necesario que cuenten con requerimientos técnicos altos mientras tengan instalado y funcional la librería p5.js a través de cualquiera de estos medios:

I. El editor web oficial de p5.js con el que no es necesario instalar nada y programar directamente desde el navegador https://editor.p5js.org/
II. Descargar la librería completa desde https://p5js.org/download/ configurarlo junto a un editor de texto de preferencia i.e. Sublime o Visual Studio Code y ejecutarlo con un servidor local, las instrucciones para esto se encuentran en https://p5js.org/get-started/

1.3. Roles y comportamiento

1.3.1. El líder

El objetivo de todo tallerista es dirigir su clase de forma que impulse la inteligencia colectiva, siendo los participantes capaces de trabajar en equipo para dar con mejores soluciones a las que daría una sola persona por sí misma, según Hamilton (2016) un taller difiere de las clases convencionales al abordar la enseñanza de una forma más práctica y creativa, por lo mismo requiere de una organización y conductas específicas que el líder deberá poner en acción dominando los siguientes rasgos:

- Planificar el uso del tiempo
- Mantener las conductas incluso las insurgentes
- Mantener la atención de los participantes
- Inspirar creatividad a través de dinámicas interesantes
- Motivar una actitud positiva ante la adversidad
- Inspirar la colaboración entre los participantes
- Inspirar confianza sin perder el profesionalismo

Sin embargo, lo más importante que un líder debe poseer es empatía y amabilidad, colocar sus conocimientos al servicio de sus alumnos, ser flexible y aceptar sus sugerencias, además de abrazar tres aspectos críticos:

1) Dirigir un debate constructivo y honesto, que contraste las diferentes opiniones hasta formar uno solo uniendo varios puntos en común, lo cual es especialmente útil cuando se tiene a varios estudiantes especializados en diferentes fundamentos del arte.
2) Dirigir el aprendizaje a través del descubrimiento, experimentación y refinamiento de ideas al momento de probar maneras de mejorar los dibujos de los alumnos.

3) Fomentar la resolución de problemas encapsulando ideas y perspectivas que emerjan de varias direcciones, útil para dibujar un mismo patrón utilizando diferentes instrucciones.

Por último, un aspecto que añade soltura y presencia a la exposición del tallerista es el uso de manierismos tanto físicos como verbales:

- **Desplazarse alrededor de la sala** y acercarse a aquellos que aporten a la discusión de la clase, el permanecer estáticamente en un punto muerto aparta al líder de su grupo.
- **Dar la bienvenida** a todos los participantes durante la sesión 0 y hacer que se introduzcan individualmente, igualmente dirigirse a cada uno por sus nombres.
- Hacer que todos **contribuyan por igual**, alentando a aquellos que siempre alzan su voz, pero aún más a los que les cuesta hacerlo.
- Proveer una **retroalimentación positiva** sobre las ideas que se estén compartiendo entre el grupo, así mismo en lo posible hacer que el grupo reflexione sobre ellas.

1.3.2. Los participantes

Para Hamilton (2016) el número ideal de participantes en un taller oscila entre 12–20 personas, las suficientes como para dividirlos en grupos más pequeños de 3 o 4, por supuesto que el total de participantes puede ser más o menos grande pero lo importante es seguir la regla anterior, por lo general las personas se sienten más cómodas en grupos pequeños y es más fácil la comunicación entre ellos, si bien el tamaño del grupo es la base también lo son otras dinámicas de grupo entre las que se encuentran:

- Formar grupos diversificados con distintas personalidades que aporten varios puntos de vista, sobre todo al hacer una lluvia de ideas sobre lo que desearían dibujar.

- Prestar ayuda a los introvertidos pues hay que asegurar que se sientan integrados, de otra forma perderán el interés, se aburrirán y no les resultara una experiencia agradable.
- Rotar a los integrantes de cada grupo al inicio de cada sesión, así se asegura que todos se conozcan entre sí y den pie a más combinaciones de ideas.
- De vez en cuando realizar ejercicios de forma individual que cada uno deberá exponer.

Pero lo que hace a un taller exitoso no solo es la cantidad sino también la integridad de las personas, la diversidad de género, etnia y culturas marca la diferencia y da forma a la cognición colectiva, incluso si se debe trabajar con un grupo homogéneo, como lo es en este caso aquel conformado por estudiantes de arte y programación, se pueden seguir algunas pautas para agregar esa diversidad al grupo:

- Procurar reunir a varios tipos de programadores tanto juniors como seniors ya que los últimos pueden transmitir sus experiencias a los primeros.
- Procurar reunir a varios tipos de artistas especializados en varios fundamentos del arte como lo son la perspectiva, anatomía, gesticulación, composición, luces y sombras, etc.
- Permitir de vez en cuando la integración de observadores, pueden ser maestros o público general, lo importante es atraer a gente que aprecie o de consejo sobre los resultados que estén obteniendo los estudiantes.
- Instruir el debate moderado contrastando puntos de vista opuestos, aprender a digerirlos y construir sobre ellos, procurar incluir a estudiantes que cumplan estas aptitudes.

1.3.3. Cambios conductuales

La forma en la que nos comportamos define gran parte del desempeño del taller, el tallerista debe estar en capacidad de manejar situaciones difíciles y promover actitudes positivas entre los participantes, el aprendizaje colectivo solo se alcanza a través de un comportamiento apropiado; ceder la palabra, escuchar atentamente, entender los diferentes puntos de vista,

empatizar con los sentimientos del prójimo, entre otras virtudes del respeto, deben establecerse claramente durante la sesión 0 al inicio del taller.

Aún a pesar de que se traten los protocolos básicos de comportamiento a tiempo, la presencia de participantes con actitudes difíciles es bastante común, comprenden actos como la falta de participación, de atención, sobre pasarse con otros integrantes, reusarse a ayudar a otros grupos y, en general, drenar la energía positiva del ambiente, aquellos son factores que previenen el buen desenvolvimiento de las actividades y el alcance de los objetivos, sin embargo, el líder, en primera instancia, debe ser comprensivo con estos estudiantes ya que esos comportamientos muchas veces se deben a razones relacionadas con el miedo:

- Dificultades relacionadas a la temática del taller y sus desafíos
- Sentirse amenazado por otros miembros
- Tener dificultades para trabajar en equipo
- Tener dificultades para ser espontaneo, abrirse a los demás o compartir experiencias
- Tener inseguridades sobre como perciben los demás al individuo

Aun así, se debe preparar un plan para convertir aquellas actitudes difíciles en comportamientos constructivos:

- En cuanto a los estudiantes **reacios a trabajar o colaborar** con los demás, muchas veces tienen malas experiencias pasadas con los métodos de enseñanza tradicionales, se deben escuchar sus preocupaciones/dificultades y reivindicarlos de forma positiva.
- Para los estudiantes **pasivo-agresivos**, muchas veces intentan desestabilizar la autoridad del líder, se debe frenar de inmediato dicho comportamiento reiterando la autoridad sin aislar al alumno y explicándole la importancia de alinearse a las actividades.

- Poner un freno a los estudiantes con un **alto ego**, quienes atienden las sesiones únicamente para demostrar su alto conocimiento frente a los alumnos menos experimentados de forma intencionada.
- Ser muy específicos al momento de **establecer las reglas, los límites y las consecuencias** de no respetarlas, los estudiantes deben tener claro que es lo que se espera de ellos en cuanto a su comportamiento y el impacto que tienen sobre la integridad del taller.

1.4. Metodología de enseñanza

Como se había mencionado antes, los talleres difieren de las clases convencionales al abordar un tema de forma casi enteramente práctica y de libre interpretación, es decir, encontrando nuevas soluciones a lo largo del proceso, pues la única forma de dar con buenas ideas es combinando varias de ellas, incluso si algunas son demasiado grandes o ambiciosas, si se las replantea con el alcance adecuado pueden convertirse en ideas factibles, esto se denomina proceso creativo e involucra cuatro formas de pensamiento:

1. La **divergencia** durante el proceso creativo amplia la diversidad de opciones y opiniones.
2. La **convergencia** de todas las ideas en una misma respuesta.
3. El **análisis** al momento de implementar las instrucciones de código e idear las posibles variaciones al tema propuesto.
4. La **síntesis** de todos los conocimientos adquiridos en un resumen claro, conciso, que pueda ser entendido por el resto de los alumnos.

Para todo tipo de taller, indistintamente del número de participantes, existe un método que agrupa estas cuatro formas de pensamiento en el marco de un proceso creativo, comprende un viaje de búsqueda por todo tipo de respuestas que la imaginación pueda alcanzar si a los equipos

se les permite pensar de forma divergente primero, crear tantas posibilidades sin pensar en las limitantes, y luego convergente, enlazarlas en temas fundamentales:

Tabla 1: Las formas de pensamiento durante las etapas de enseñanza

	Propuesta	**Enfoque**	**Pensamiento**	**Resultado**
1ra etapa: Crear (60% del taller)	Imaginar tantos bosquejos como sea posible.	Aplicar el proceso creativo durante las iteraciones de lluvia de ideas.	Primera etapa de divergencia, espontaneo e impulsivo, imaginar tantas posibilidades como sea posible sin pensar mucho.	Una lista extensa de posibles ideas y direcciones.
2da etapa: Evaluar (10% del taller)	Elegir los mejores candidatos de entre todas las ideas presentadas y los temas inherentes a ellas.	Identificar los temas más interesantes y discutirlos con el equipo antes de escogerlos.	Segunda etapa de convergencia, considerar la factibilidad de las ideas respecto a los objetivos y escoger las más viables para su desarrollo.	Una lista curada de ideas y temas acordados con el equipo.
3ra etapa: Desarrollar (30% del taller)	Tomar aquellas ideas y temas y desglosarlas a gran detalle.	Priorizar las ideas escogidas y organizar los subgrupos que las ejecutaran.	Análisis y síntesis, trabajar sobre las ideas prioritarias para combinar sus mejores elementos.	Una lista final con las mejores ideas, claramente explicadas y delimitadas en pasos e instrucciones.

Fuente: *The Workshop Book* (p.8) por P. Hamilton, 2016, Pearson Business.

1.5. Materiales digitales y aula virtual

Por último, se ha construido un aula virtual para alojar todo el material didáctico y teórico que guiará el desarrollo de las clases, está estructurado por secciones correspondientes a los temas que se irán desarrollando respectivamente, en cado uno se podrá encontrar los siguientes:

- Las presentaciones de los ejercicios guiados en formato diapositiva adjunto a la exposición teórica de sus temas y conceptos.

- La actividad extra por desarrollar e integrar mejor los conceptos adquiridos.
- Un recurso en dominio público que complemente el conocimiento y la actividad extra.
- La compilación de los ejercicios guiados en código por secciones.

Figura 2

Material del taller digitalizado para su alojamiento en un aula virtual de Moodle

El sitio actualmente este alojado en el servicio nube de la plataforma Moodle **creativecode.moodlecloud.com** y se puede configurar el acceso de cualquier alumno a este mediante su inscripción por correo electrónico o en el panel del administrador del sitio que en este caso sería el docente.

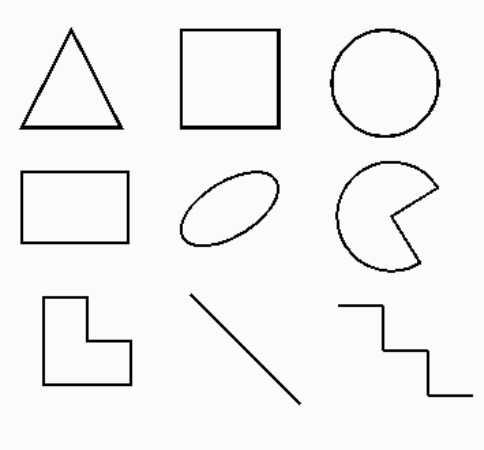
2.1. Gráficas primitivas

2.1.1. Uno con todo

Primera toma de contacto con el conjunto de instrucciones que ofrece p5.js para trazar cada tipo de primitiva que sea posible i.e. un triángulo, rectángulo, círculo, cuadrilátero, elipse, arco, línea, polígono, polilínea, etc. (Golan & Tega, 2021, pg. 152)

Figura 3

Gráfica resultante del ejercicio 2.1.1



Objetivos

- Familiarizarse con las funciones **setup (), draw (), push ()** y **pop ()**

Variaciones

- Implementar instrucciones para el relleno de fondo, el grueso, color del trazo, etc.
- Incluir polígonos más complejos como la curva de Bézier.

Contexto y significado

La apreciación de figuras geométricas fundamentales en la naturaleza se remonta al arte de la prehistoria, en ese entonces la representación de imágenes, animales y signos abstractos se hacía a través de pinturas rupestres y estatuillas de hueso o piedra con las primeras técnicas de escultura y tallado, siendo estas dos últimas estrechamente ligadas a los primeros indicios de arquitectura, que también se considera una rama del arte, porque en ella se exploran aspectos como la proporcionalidad, la conceptualización de objetos en tres dimensiones, la geometría que da armonía y proporcionalidad a las construcciones, etc. Recursos que comparte con el dibujo, la pintura y escultura.

El monumento más famoso de la arquitectura megalítica debe ser el *crómlech* de *Stonehenge*, una estructura geométrica circular de monolitos de piedra de 100 metros de diámetro y dos círculos concéntricos (Caralt & Casal, 2012, pg.13 - 21).

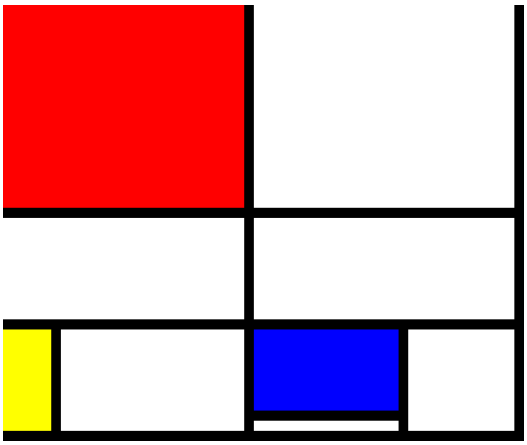
Código del ejercicio adjunto en los anexos

2.1.2. Cuadro Mondrian

Implementa código que reproduzca alguna de las variaciones de la pintura de Piet Mondrian, como ejemplo esta la Composición No. III en rojo, amarillo, azul y negro (1929) útil para entrenar la atención a los detalles. (Golan & Tega, 2021, pg. 152)

Figura 4

Gráfica resultante del ejercicio 2.1.2



Objetivos

- Familiarizarse con las instrucciones **fill ()**, **background ()**, **noStroke ()** y **rect ()**.

Variaciones

- Seguir el mismo procedimiento para reproducir la composición No. II, IV o una composición propia con otros tres colores primarios.

Contexto y significado

Composición con rojo, azul y amarillo es una famosa pintura abstracta por Piet Mondrian, en ella se presentan solo formas geométricas, líneas y colores primarios divididos en cuadrículas rectangulares con líneas negras que definen sus bordes, dispuestos en una composición equilibrada. De tal manera que se abstrae la armonía y el equilibrio entre los bloques rojos, azules y amarillos que celebran la simplicidad y el orden que subyace en toda realidad universal, convirtiéndose en un símbolo icónico del arte moderno.

Este movimiento se llamó *Neoplasticismo* y se contrapuso a otros movimientos como el realismo o el neo clasismo que habían permeado fórmulas verosímiles durante años (Smarthistory, 2016).

Código del ejercicio adjunto en los anexos

2.2. Iteraciones

2.2.1. Progresión geométrica

Generación de elementos visuales cuyas dimensiones exhiban una progresión geométrica, el tamaño de cada elemento debe ser calculado en cada iteración multiplicado por el tamaño de su predecesor a un radio constante, en el ejemplo cada arco es 1.3 veces más largo cada vez. (Golan & Tega, 2021, pg. 154)

Figura 5

Gráfica resultante del ejercicio 2.2.1

Objetivos

- Introducir el uso del bucle **for**, las instrucciones **noFill ()** y **strokeWeight ()**
- Utilizar la primitiva **ellipse ()** con el parámetro de número de sectores radiales.
- Aplicar la variable **frameCount** para animar los arcos.

```
/*Primer ejercicio con bucles*/
function setup() {
  createCanvas(400, 400);
}

function draw() {
  //setea el fondo en negro
  background(255);

  //iteraciones que forman los arcos blancos
  for(var i = 12; i < 800; i*=1.3) {
    noFill();
    //hacer que el grueso del trazo sea más grande cada vez
    strokeWeight(0.05*i)
    //en realidad dibujo un elipse completo pero debido
    //al tamaño del canvas y la posición crea la ilusión
    //de ser un arco
    ellipse(200,400,i);
  }
}
```

2.2.2. Cuadriláteros transicionales

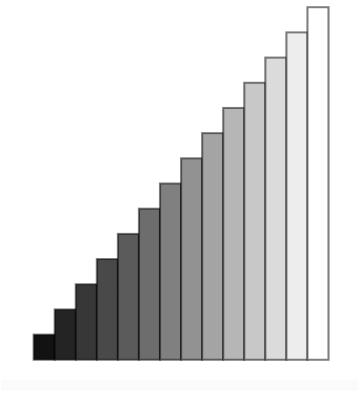
Cualquier instrucción sobre una propiedad visual puede vincularse a un bucle no tan solo funciones de transformación, se utiliza un bucle para generar una serie de rectángulos que gradualmente modifiquen sus propiedades como la posición, altura y relleno. (Golan & Tega, 2021, pg. 153)

Objetivos

- Introducir el uso de la función **map ()** para escalar valores entre un rango determinado.
- Introducir el uso de variables para alterar las propiedades de las gráficas simultáneamente.

Figura 6

Gráfica resultante del ejercicio 2.2.2



```
/* Utilizar iteraciones para formar un patrón visual
creando variaciones de una misma gráfica primitiva */

let col = 0;//saturación del negro en el relleno de la cuadrícula
let spacing = 20;//espacio entre figuras
let margin = 60;//espacio entre las figuras y los margenes del canvas
let rectHeight = 25;//altura de partida

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

function draw() {
  //setea el fondo en blanco
  background(255);
  //setea los bordes del trazo en negro
  stroke(0);
  //setea el grueso del trazo ligero
  strokeWeight(1);

  //bucle para crear 15 cuadrículas rectángulares
  for (let i = 1; i < 15; i++) {
    //escala el valor de saturación entre un rango de 14 - 255
    col = map(i, 0, 14, 0, 255);
    //rellena la forma según el valor de saturación obtenido
    fill(col);
    //genera un rectángulo de altura cada vez menor
    rect(i * spacing + margin, 380, 20, i * -rectHeight);
  }
}
```

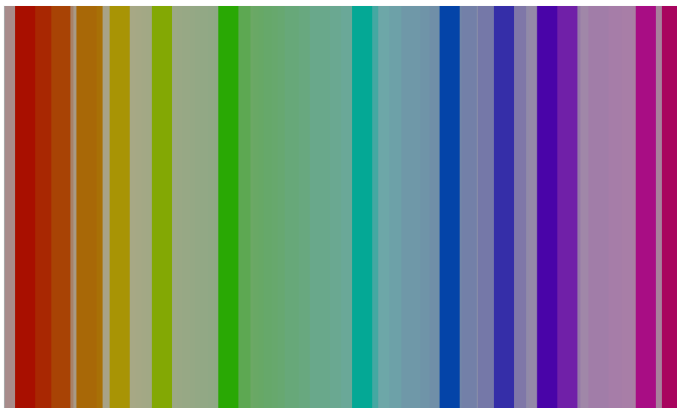
2.3. Color

2.3.1. Insaturación de color

La saturación es la fuerza o pureza del color y representa la cantidad de gris en proporción al tinte, un color ‘saturado’ es puro y uno ‘insaturado’ posee gran porcentaje de gris. Desplazar el cursor verticalmente sobre una gradiente de colores para alterar su saturación. (Golan & Tega, 2021, pg. 155)

Figura 7

Gráfica resultante del ejercicio 2.3.1



Objetivos

- Introducir el uso de las instrucciones **mouseX** y **mouseY** para mapear la posición del cursor en el canvas.

Contexto y significado

Los colores se usan como símbolos (estados de ánimo) o códigos en diferentes circunstancias, una vez instituidos, constituyen un lenguaje más o menos extendido. Sus tres características principales son el **tono, brillo y saturación**, esta última se refiere a la intensidad del color en tono de grises donde los colores más insaturados se usan para sugerir luces nocturnas, falta de luz en interiores o transmitir sensaciones de incertidumbre.

Una muestra de esta técnica se puede apreciar en la pintura Saint-Séverin No. 2 de Robert Delaunay la cual representa el interior de una iglesia a través de un prisma de formas geométricas que evocan sus interiores, arcos y bóvedas en una composición de columnas que se doblan, arquean, expanden o se disuelven en series sucesivas, además de que se saturan en diferentes tonos de azul para crear un efecto de luces y sombras (artsmia.org, 2023).

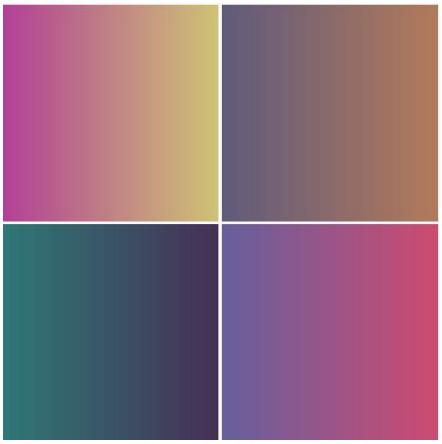
```
/*Gradiente de insaturación por movimiento del cursor*/

const barWidth = 20;
let lastBar = -1;

function setup() {
  createCanvas(720, 400);
  //el modo de color es seteado en el sistema HSB para saturación
  colorMode(HSB, 400, 400, 100);
  noStroke();
}

function draw() {
  //devuelve la coordenada X en 2D
  let detectedBar = mouseX / barWidth;
  //si no ha alcanzado el límite en 2D
  if (detectedBar !== lastBar) {
    //devuelve la posición original
    let barX = detectedBar * barWidth;
    //rellena de insaturación en la posición capturada
    fill(barX, mouseY, 66);
    //genera la barra con dicho valor
    rect(barX, 0, barWidth, height);
    //guarda la última posición saturada
    lastBar = detectedBar;
  }
}
```

2.3.2. Gradiente vertical animada

Generar una barra que represente la transición gradual de un color a otro, para ello implementar un bucle que renderice finas rectas adyacentes y paralelas con ligeros cambios de color. (Golan & Tega, 2021, pg. 155)

Figura 8

Gráfica resultante del ejercicio 2.3.2



Objetivos

- Introducir el uso de la función **noise ()** para calcular valores aleatorios armónicos según el ruido Perlin.
- Utilizar la instrucción **lerpColor** para generar un nuevo color a partir de dos colores interpolados.

Contexto y significado

Los degradados o gradientes son transiciones de color habituales en diseño gráfico, aplicaciones, interfaces y productos, se trata de un rango de colores ordenados linealmente en los que se va produciendo una transición gradual y progresiva entre ellos; se reduce el porcentaje de un color mientras que aumenta el del otro, por lo que se van produciendo nuevas tonalidades en medio de esas fusiones.

A diferencia de los colores planos que transmiten una composición aséptica, los degradados dan profundidad, tonalidades, fusiones, matices y transiciones cercanos a los colores de la naturaleza: puestas de sol, cielos, mar, plantas, frutas, etc. Que enfaticen ciertas partes de un diseño mediante zonas de luz y oscuridad de dichas transiciones (Elementor, 2022).

```
/*Construir una barra de gradientes variables*/
function setup() {
  createCanvas(650, 300);
}

function draw() {
  //siempre se debe dividir la taza de frames por una constante
  //ya que la cantidad de cuadros por segundo de ejecución es
  //muy grande
  const m = 100;

  //se procede a calcular los valores de saturación del espectro
  //RGB en función del ruido Perlin que agrega un factor aleatorio
  //más armonico y natural
  const leftR = 255 * noise(frameCount / m);
  const leftG = 255 * noise(1000 + frameCount / m);
  const leftB = 255 * noise(2000 + frameCount / m);
  //calcula un ruido menor que dara mayor saturación en la sección izquierda
  const rightR = 255 * noise(3000 + frameCount / m);
  const rightG = 255 * noise(4000  + frameCount / m);
  const rightB = 255 * noise(5000 + frameCount / m);
  //calcula ruido menos saturado para la sección derecha

  /*crea los colores que se van a degradar en ambos extremos según
  los niveles de rojo, verde y azul calculados anteriormente*/
  const leftColor = color(leftR, leftG, leftB);
  const rightColor = color(rightR, rightG, rightB);

  //bucle para rellenar la barra horizontal de rectas
  for(let x = 0; x < width; x++)
  {
    /*interpola ambos colores para dar con un tercero que sea
    más o menos cercano a uno u otro según su posición*/
    const lineColor = lerpColor(leftColor, rightColor, x / width);
    //usa el color en el trazado de la recta
    stroke(lineColor);
    //genera una recta horizontal
    line(x, 0, x, height);
  }
}
```

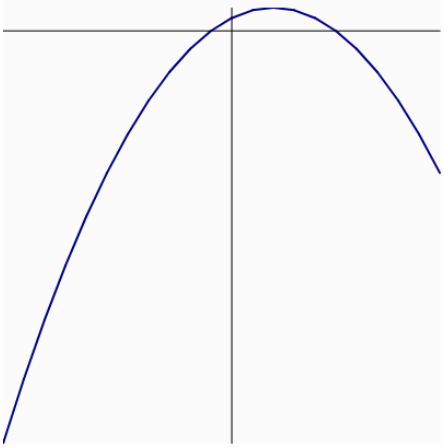
2.4. Curvas

2.4.1. Parábola

Una curva que representa la función cuadrática $f(x) = ax^2 + bx + c,\ a \neq 0$ donde $a, b\ y\ c$ son constantes a elección. Codificar un programa que mapee los pares ordenados resultantes de la función y dibuje la parábola resultante (Golan & Tega, 2021, pg. 164).

Figura 9

Gráfica resultante del ejercicio 2.4.1



Objetivos

- Aprender a abordar un problema con un enfoque modular: una función para calcular pares ordenados, una para trazar rectas que conecten esos puntos y otra para trazar los ejes ordenados.
- Aprender a usar arreglos y objetos para guardar y pasar parámetros compuestos.

Contexto y significado

Los arcos parabólicos han inspirado las arquitecturas más elegantes y estables encontradas en puentes y monumentos desde la antigüedad, pues su curva característica permite el reparto de cargas verticales de distribución uniforme más eficiente de todos. Sus cimientos conforman la base del sistema de bóvedas, contrafuertes y cúpulas de las catedrales góticas del Renacimiento. En el siglo XV Filippo Brunelleschi diseñó la cúpula renacentista octogonal de la Catedral de Santa María del Fiore, en el XVII Christopher Wren hizo lo mismo con la cúpula de Catedral de San Pablo de Londres, mientras se descubría que los contrafuertes de la Capilla del *King's College* de Cambridge cumplían con el mismo principio (Benaim, 2008, pg.507).

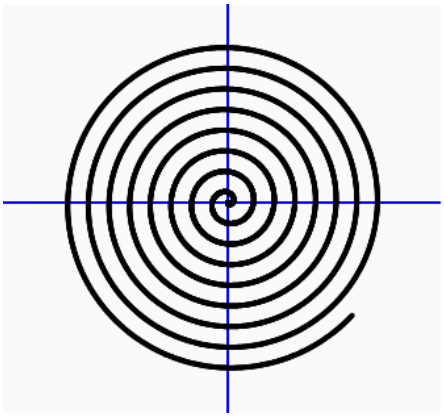
Código del ejercicio adjunto en los anexos

2.4.2. Espiral de Arquímedes

Codificar un programa que dibuje la espiral arquimediana, aquella cuyo radio crece por medio de un movimiento rotacional y rectilíneo de un punto sobre una recta que gira a una velocidad angular constante (Golan & Tega, 2021, pg. 164).

Figura 10

Gráfica resultante del ejercicio 2.4.2



Objetivos

- Introducir las funciones trigonométricas **sin** y **cos** para calcular radios uniformes.

Variaciones

- Construir demás tipos de espirales con técnicas diversas como la espiral logarítmica a través de las ecuaciones polares.

Contexto y significado

Las espirales forman parte de nuestro entorno cotidiano, podemos contemplarlas en todas las escalas posibles, tanto en el espacio como en el tiempo. Las huellas dactilares o el caracol de nuestro oído interno serían ejemplos de espirales que posee nuestro propio organismo, mientras que las olas que culminan enroscándose, las conchas de los caracoles, el movimiento de los ciclones o tornados y las curvas espirales de las galaxias serían resultado de la búsqueda de espirales en la naturaleza. Estas se han usado por diferentes culturas con diferentes significados, i.e. la Venus de Milo fue representada girando sobre sí misma en movimiento ascendente, como si estuviese abandonando el ropaje de la materia en su ascenso en espiral (Cook, 1903, pg. 26).

Código del ejercicio adjunto en los anexos

2.5. Figuras

2.5.1. Estrella

Dibujar los vértices de un polígono regular de 10 lados usando las funciones trigonométricas **sin**() y **cos**() similar a como se usa generalmente para crear circunferencias, incluyendo una rotación en el sentido del reloj (Golan & Tega, 2021, pg. 166).

Figura 11

Gráfica resultante del ejercicio 2.5.1

Objetivos

- Utilizar las funciones trigonométricas **sin** y **cos** para modificar el radio y número de vértices de una estrella.

Variaciones

- Desplazar varias estrellas desde un extremo del canvas hacia todas las direcciones.

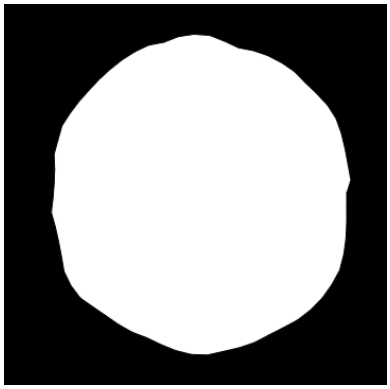
Código del ejercicio adjunto en los anexos

2.5.2. Molécula

Por los medios que se consideren convenientes, dar forma a una molécula en dos dimensiones que sea interactiva de alguna forma i.e. trazar los contornos de entre un conjunto de esferas o a partir de una curva paramétrica (Golan & Tega, 2021, pg. 167).

Figura 12

Gráfica resultante del ejercicio 2.5.2



Variaciones

- Replicar un proceso similar, pero con un bucle de curvas de Bézier.
- Simular una figura similar calculando una elipse Cassini, una crinoide u otra curva paramétrica.

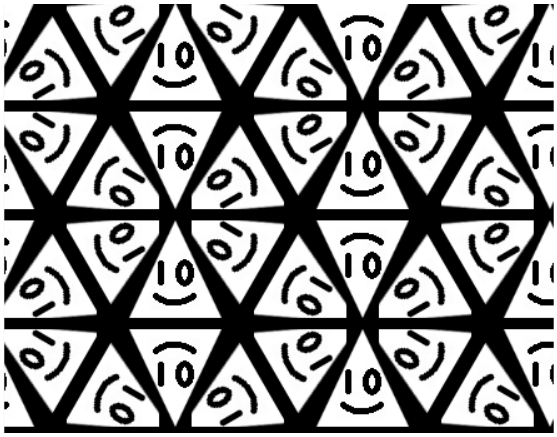
Código del ejercicio adjunto en los anexos

3.1. Caleidoscopio

Idear un motivo gráfico simple y codificar un programa que haga uso del conjunto de instrucciones de transformación de primitivas para trasladar, reflejar y rotar copias de ese motivo similar a como lo haría un caleidoscopio (Golan & Tega, 2021, pg. 152).

Figura 13

Gráfica resultante del ejercicio 3.1



Objetivos

- Aprender a utilizar la función **mask ()** para enmascarar una figura con una imagen.
- Aprender a utilizar la función **image ()** para dibujar un bosquejo enmascarado.
- Aprender a utilizar la función **rotate ()** y **scale ()** para rotar y reflejar una figura.

Contexto y significado

Un caleidoscopio es un cilindro en forma de prisma triangular conformado por espejos en el interior y láminas traslúcidas a los extremos con varios objetos de colores y formas diferentes, cuyas imágenes se ven multiplicadas simétricamente al girar el cilindro mientras se mira por el extremo opuesto. Dichos espejos pueden estar dispuestos en distintos ángulos; a 40° de cada uno se generan ocho imágenes duplicadas, en 60° se observan seis duplicados y en 90° cuatro. Fue inventado en 1816 por el físico escocés David Brewster y por su facilidad de fabricación es uno de los juguetes más conocidos y apreciados por su efecto óptico donde elementos simples son capaces de crear imágenes y efectos dinámicos, armónicos, llenos de luz y color.

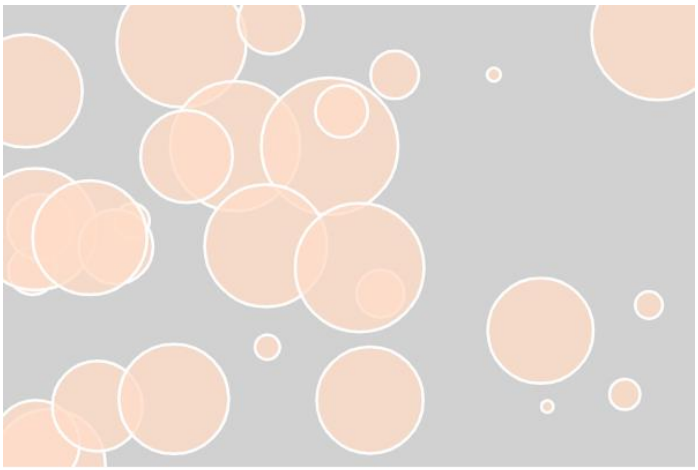
Código del ejercicio adjunto en los anexos

3.2. Elementos estocásticos

Crear imágenes evocativas por medio de un bucle que deposite pequeños elementos en posiciones aleatorias por todo el lienzo, estos elementos pueden parecerse a los cráteres de la luna, baches en la carretera, hormigas, chispas de chocolate, los hoyos en el queso suizo, etc. (Golan & Tega, 2021, pg. 154).

Figura 14

Gráfica resultante del ejercicio 3.2



Objetivos

- Aprender a usar la función **random ()** para generar elementos estocásticos.
- Aprender a usar la función **splice ()** para insertar o remover elementos de un arreglo.
- Aprender a usar la función **dist ()** para calcular la distancia entre dos puntos.
- Usar la instrucción **mouseIsPressed** para detectar la entrada del cursor.

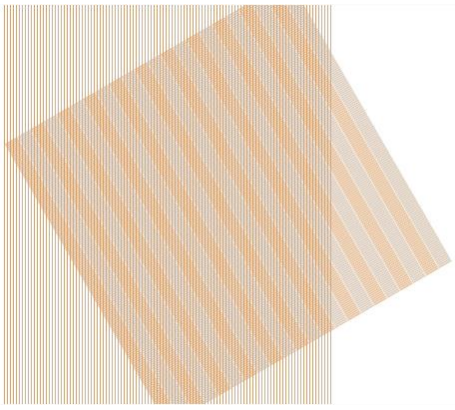
Código del ejercicio adjunto en los anexos

3.3. Patrón Moiré

Generar conjuntos de rectas o curvas paralelas separados en espacios muy estrechos que conformen una especie de “reja”, luego sobreponer una copia de dicha reja sobre la otra con una ligera rotación para crear un efecto Moiré (Golan & Tega, 2021, pg. 154).

Figura 15

Gráfica resultante del ejercicio 3.3



Objetivos

- Aprender a generar rectas paralelas de diferente ancho, largo y posición a través de bucles.

Variaciones

- Replicar cualquiera de los otros patrones Moiré que se pueden encontrar.

Contexto y significado

En pantallas digitales, por ejemplo, si se coloca la cámara de un teléfono móvil apuntando a la pantalla de un monitor, se puede apreciar un cierto efecto resultante de la superposición de las líneas de ambas pantallas. Dichos patrones se denominan moirés o muarés y se forman al combinar dos rejillas de líneas o curvas de tamaño o ángulos diferentes, *Moiré* es un término francés que refiere a un tipo de textil con apariencia ondulante o fluctuante debido a la misma estructura del tejido, comúnmente de seda, lana, algodón o rayón. El efecto se puede apreciar en varias aplicaciones de la física, óptica, impresión, textiles o artes gráficas.

Japón es de los lugares en los que más se pueden apreciar en kimonos, cestas tejidas y persianas de bambú, pues ese tipo de textiles comenzaron a fabricarse en Oriente mucho antes que en Francia (Scientific American, 1963, pg. 54).

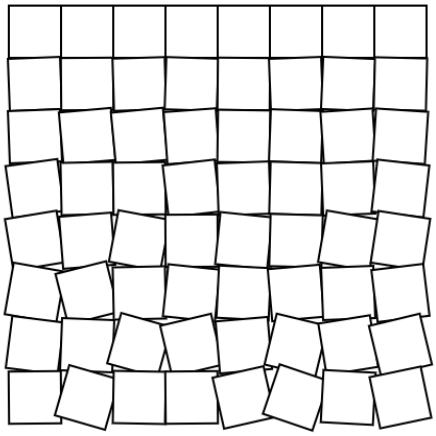
Código del ejercicio adjunto en los anexos

3.4. *Schotter*

El *Schotter* de Georg Ness (1968) es una obra de arte generativo que representa una variación de orden gradual a través de una serie de cuadriculas en una grilla de 12x22, la orientación de las cuadriculas cambia aleatoria pero ligeramente a medida que se recorren más filas en la composición, replicar esta pieza prestando atención a los detalles (Golan & Tega, 2021, pg. 154).

Figura 16

Gráfica resultante del ejercicio 3.4



Objetivos

- Utilizar la función **random ()** para representar un conjunto de formas adyacentes con ligeras variaciones entre sí de forma gradual.

Variaciones

- Continuar con la versión extendida del *Schotter* que incrementa gradualmente la distorsión de espaciado y rotación de las cuadriculas proporcionalmente al número de filas que se añadan al extremo inferior.

Contexto y significado

El Schotter original se compone por 22 filas de 12 cuadrados cada una, dichos cuadrados aumentan gradualmente su aleatoriedad de rotación y posición a medida que se alcanzan las filas inferiores. La ventaja que supone el diseño generativo es que, lo que normalmente tomaría varias horas para dibujar todas esas cajas aleatorias de forma manual, con un simple algoritmo y la función **square** de p5.js se puede añadir miles de cajas más de forma automática y en segundos adaptando el ingreso de parámetros en el código.

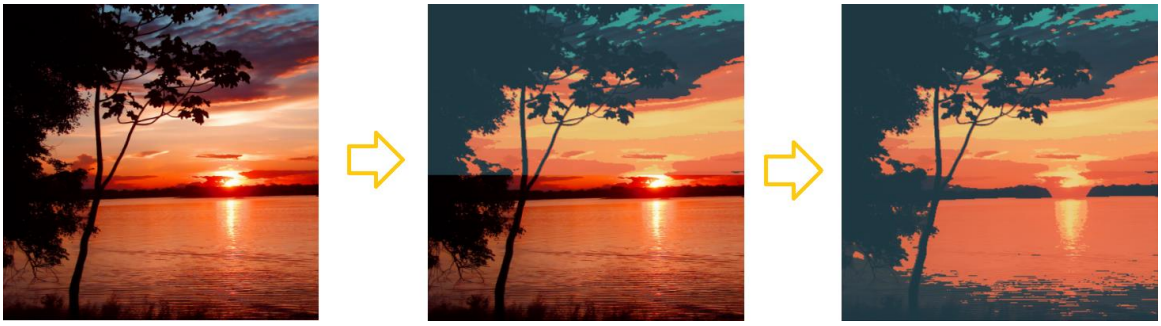
Código del ejercicio adjunto en los anexos

3.5. Filtro de paleta de colores

En este ejercicio se engloban varios de los conceptos repasados hasta ahora sobre saturación de color, iteraciones e interpolación de valores. El objetivo es codificar un programa que cargue una imagen y la reimprima aplicando un filtro con una paleta de colores predeterminada, es decir, analizar cada color compuesto en la imagen y reemplazarlo con el color de la paleta más cercano posible, adicionalmente se debe animar dicha renderización para apreciar el cambio entre la imagen original y la aplicación del filtro.

Figura 17

Gráfica resultante del ejercicio 3.5



Objetivos

- Aprender a dibujar el buffer completo de pixeles de una imagen de forma manual.
- Aprender a cambiar el color de un píxel por otro mediante interpolación entre colores.
- Identificar patrones ocultos en la composición.

Variaciones

- Añadir el cambio de imagen de forma aleatoria entre un conjunto de imágenes precargadas.
- Integrar una paleta de colores más amplia para apreciar distintos resultados.

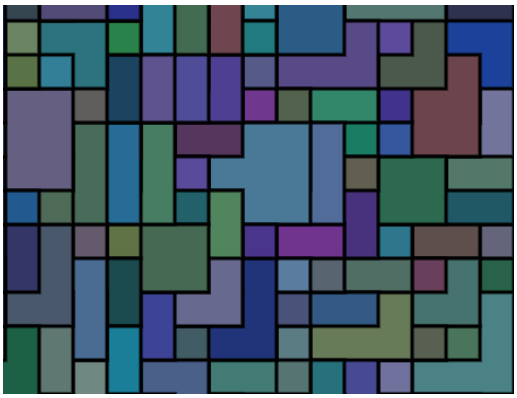
Código del ejercicio adjunto en los anexos

4.1. Patrón iterativo en el diseño de textiles

Codificar un programa que genere un patrón de mosaico o una textura compuesta, similares a los que se aprecian en papeles tapices o textiles, considerando cuestiones estéticas como la simetría, el ritmo, los colores, las escalas, las formas y el balance de todos esos elementos. El patrón debe ser diseñado de tal forma que el mosaico sea extenso y transmita cierto grado de uniformidad para ser apreciado en paredes, pisos o incluso en la propia ropa. Exportar el patrón resultante en un formato de alta resolución e imprimirlo en una gran lámina para su exposición en la clase. Como sugerencia, dibujar el patrón que se tenga en mente primero (Golan & Tega, 2021, pg.18).

Figura 18

Gráfica resultante del ejercicio 4.1



Objetivos

- Utilizar el sistema cartesiano de coordenadas y las instrucciones de trazo primitivas para crear diseños armónicos.
- Usar la abstracción para encapsular las funciones encargadas del diseño modular de los elementos.

Contexto y significado

Un patrón es el punto de inicio desde el cual se percibe y se impone orden en el mundo, ejemplos del diseño de patrones funcionales, decorativos y expresivos datan de tiempos ancestrales en forma de mosaicos, calendarios, tapicería, joyería, caligrafía, acolchados, muebles y arquitectura.

Siempre ha existido una inherente conexión entre el diseño de patrones, la geometría y los algoritmos matemáticos e iterativos, este ejercicio invita a interiorizar el entendimiento de esa relación en términos formales. Sin embargo, la parte más crucial del proceso debe ser liberar los diseños a la realidad por medios físicos ya sea una imprenta, material de fábrica o en lotes a gran escala, la fase de síntesis se logra a través de la creación de algo físico (Golan & Tega, 2021, pg. 18).

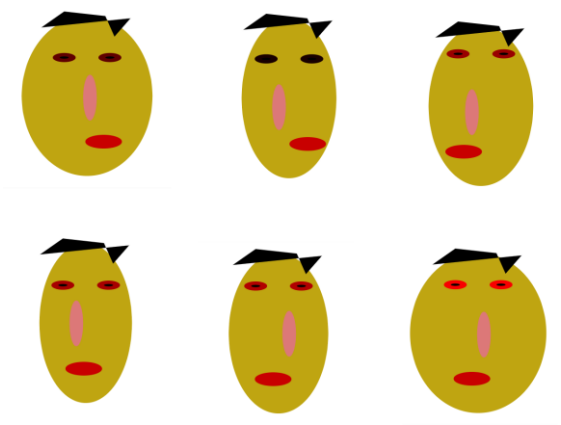
Código del ejercicio adjunto en los anexos

4.2. Rango de expresiones en el rostro

Codificar un programa que parametrice la imagen de un rostro con al menos tres variables i.e. el tamaño, posición, color o alguna otra propiedad visual de los ojos, nariz, boca, etc. Dichas variaciones podrán usarse para construir varias expresiones tales como alegre, triste, confundido, etc. Una identidad ya sea hombre, mujer, macho o hembra, o una especie como felino, sabueso, simio, reptil, etc. Prestar consideración al contorno de las fisonomías como la forma de la nariz, mentón, orejas, quijada, etc. Así como también otras características como el color de piel, cabello, barba, distancia entre los ojos, etc. El programa deberá generar un rostro nuevo aleatorio en cada ejecución (Golan & Tega, 2021, pg.24).

Figura 19

Gráfica resultante del ejercicio 4.2



Objetivos

- Utilizar el sistema de primitivas para dibujar formas paramétricas.
- Aplicar los principios de diseño generativo en expresiones faciales.
- Implementar un sistema que diseñe elementos derivados de un conjunto de parámetros retroactivamente.

Contexto y significado

Los humanos poseen una extensa sensibilidad y percepción sobre las expresiones y el lenguaje corporal, desde la infancia quedan grabados en la mente gran variedad de rostros con cambios sutiles entre ellos, haciendo reconocible el más mínimo estado de ánimo con una transparencia imposible de alcanzar incluso para los dispositivos, pues reconocer rasgos familiares o a un conocido entre la multitud con tanta precisión es gracias a nuestra cognición, mientras que la falta de percepción de rostros o la sobre posición de estos donde no las hay se consideran desordenes cognitivos como *prosopagnosia* y *pareidolia* respectivamente. Una técnica de visualización de datos multivariables a través de la representación de rostros son las caras de Chernoff, en ella rasgos como los ojos, narices, orejas y boca permiten hacer asociaciones y detectar diferencias dependiendo de su forma, tamaño, posición y orientación, en concreto Herman Chernoff codifico 18 variables para sintetizar un rostro (Golan & Tega, 2021, pg. 24).

Código del ejercicio adjunto en los anexos

4.3. Representación del tiempo

Diseñar y graficar un reloj que represente el tiempo de una forma poco convencional, para ello se procede a indagar sobre las diferentes concepciones que ha tenido el tiempo a lo largo de la historia, no todos necesariamente han medido dicha magnitud en números, existe la cronobiología (tiempo biológico), los ciclos lunares y solares, el tiempo sidéreo, decimal, métrico, etc. También está el tiempo que abarca épocas como el geológico, histórico, psicológico e individual. Documentar el diseño con su historia, dispositivos y su uso en sociedad (Golan & Tega, 2021, pg. 30).

Objetivos

- Investigar los diferentes sistemas de cronometraje en la historia.
- Divisar esquemas gráficos que midan y presenten el tiempo de formas distintas.

- Diseñar las formas del reloj, sus colores y movimientos a través de instrucciones.

Variaciones

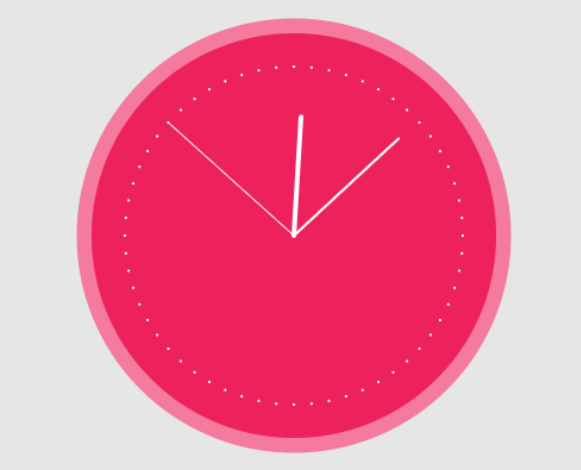
- Evitar usar números romanos, arábicos o chinos, en su lugar hacer que el tiempo sea legible por otros medios como patrones de bits numéricos o iteraciones contables y que opere en una escala de tiempo más lenta, cambiando cada mes, temporada o año.

Contexto y significado

Los diversos intentos de medir el tiempo empezaron hace miles de años con inventos astronómicos como los gnomons, relojes de agua, calendarios lunares y relojes de sol, siendo la representación estándar mundial con horas y minutos divididos en 60 unidades una herencia directa de estos últimos, los cuales fueron los primeros en implementar un sistema de conteo sexagesimal. La evolución de la cronometría ha sido históricamente impulsada por la necesidad de la economía y la milicia por una mayor precisión, exactitud y sincronización en sus mediciones, cada uno de esos avances ha tenido repercusiones profundas en la forma en la que trabaja la ciencia, agricultura, navegación, comunicaciones, armamentística, etc. (Golan & Tega, 2021, pg. 30)

Figura 20

Gráfica resultante del ejercicio 4.3



Código del ejercicio adjunto en los anexos

4.4. Alfabeto modular

Diseñar un tipo de letra, simulando una existente u otra completamente nueva, creándola de cero a través de gráficas primitivas o haciendo que todas sus letras sigan una estructura visual con parámetros similares, por ejemplo, un alfabeto en el que cada letra se componga de curvas (modificando varios puntos de la curva de Bézier) o cuadriculas (pixelado) exclusivamente. Primero hacer un bosquejo del diseño de las letras en papel, componer todo el alfabeto a través de las instrucciones de p5.js y desplegarlas todas de forma ordenada en un mismo marco (Golan & Tega, 2021, pg. 56).

Objetivos

- Aprender a graficar un alfabeto modular ya sea desde cero con gráficas primitivas o desde una fuente preconstruida.
- Concebir varios conceptos gráficos para crear una tipografía dinámica.
- Usar parámetros para modificar o manipular las formas de letras.
- Utilizar estructuras de datos para almacenar datos geométricos.

Variaciones

- Almacenar los parámetros de modificación de las letras en un arreglo o estructura de objetos que luego lo utilizara una función para renderizar cualquier letra requerida.

Figura 21

Gráfica resultante del ejercicio 4.4

Contexto y significado

Desde la tipografía *Univers* de Adrian Frutiger (1954) pasando por el lenguaje *METAFONT* de Donald Knuth (1977) y llegando al primer conjunto de fuentes digitales *Multiple Máster* de Adobe System (1994), se ha vuelto una práctica común en el diseño gráfico el idear sistemas de símbolos y letras altamente adaptables que van más allá de los rígidos límites de las tipografías estáticas. Hasta antes de la invención de *Univers*, los diseñadores tipográficos ya se habían interesado en las propiedades inherentes en las letras y sus similitudes, *Univers* fue la primera tipografía no solo en intentar diseñar cada letra de la forma más única posible sino también en tratar de comprender las diferentes relaciones que existen entre varios grupos de formas y figuras en las que se componen los alfabetos (Golan & Tega, 2021, pg. 56)

```
//alfabeto modular
let pixelFont;
//recorremos este arreglo dentro de una función
//aplicando la fuente cargada previamente
let letters = ['A', 'B', 'D', 'E', 'F', 'G',
               'H', 'I', 'J', 'K', 'L', 'M',
               'N', 'O', 'P', 'Q', 'R', 'S',
               'T', 'U', 'V', 'W', 'X', 'Y',
               'Z'];

function preload() {
  pixelFont = loadFont('https://happycoding.io/fonts/happycoding/happycoding.ttf');
}

function setup() {
  createCanvas(650, 470);
  textSize(125);
  //setea la fuente para dibujar texto en el canvas
  textFont(pixelFont);
}

function draw() {
  background(32);
  fill(255);
  let count = 0;

  //ocupa todo el largo del canvas
  for(let y = 100; y <= height; y += 120)
  {
    //ocupa todo el ancho del canvas
    for(let x = 20; x <= width; x += 80)
    {
      if(count < letters.length)
      {
        //dibuja la letra correspondiente en esta posición
        text(letters[count], x, y);
        count++;
      }
      else
        break;
    }
  }
}
```

CAPÍTULO V: CONCLUSIONES Y EVALUACIÓN FINAL DEL TALLER

5.1. Métodos de evaluación y retroalimentación del taller

Como el objetivo perseguido en este taller es poco frecuente en la literatura, al tratarse sobre dos contextos de aprendizaje distintos combinados, es importante que durante la conclusión sea documentada la evaluación de resultados y la retroalimentación sobre los métodos de enseñanza aplicados, de esa forma se podrán crear mejores formas de llevar a cabo la dinámica en el futuro. Sin embargo, eso requiere saber escoger los métodos que mejor se adapten al formato del taller de forma que ni los alumnos ni los docentes sean sobrepasados con más información de la necesaria, es un trabajo colaborativo para sondear tanto el alcance de lo aprendido como de los vacíos que se deben llenar a través de la socialización de experiencias y resultados:

- Llevar a cabo entrevistas, tanto a los internos del taller como a otros agentes externos especialistas sobre el tema en cuestión, por ejemplo, docentes de diseño y de programación por igual.
- Hacer que cada alumno comparta y comente su trabajo con otro compañero y viceversa.
- Invitar a consultores externos relacionados al desarrollo de otros talleres a la socialización de resultados con el objetivo de comentar críticas constructivas.
- Que cada alumno se encargue de revisar su propio trabajo revisando otras referencias, buscando ejercicios similares o comparándolo con la de sus compañeros.
- Mandar a redactar varios informes sobre los temas consultados para que cada persona los lea en detalle, asegurar que cada uno sea revisado por más de una persona.

Se pueden idear muchas otras formas de recabar retroinformación, sugerencias y comentarios, pero la más efectiva siempre será hacer entrevistas directas con los alumnos, la parte difícil de ello es formular las preguntas correctas y que estas sean precisas con lo que nos interesa saber, para redactarlas se debe tomar en cuenta lo siguientes criterios:

- Las necesidades de las personas.
- Las oportunidades y fortalezas del proyecto respecto al mercado.
- Los desafíos del proyecto respecto a sus objetivos.
- Las tendencias del mercado que se alineen a los objetivos del proyecto.
- El estado del arte existente sobre los contenidos del proyecto.
- Los niveles de satisfacción del público sobre el desempeño del proyecto.
- Los niveles de satisfacción del público sobre los métodos de enseñanza aplicados.

Tomando en cuenta los anteriores criterios se pueden redactar todo tipo de preguntas, sin embargo, como ejemplo se ha formulado el siguiente cuestionario para su publicación y recolección de comentarios junto a los docentes y, más importante, los alumnos:

Tabla 2: Cuestionario de evaluación de desempeño del taller

Nro.	Cuestionario
1	¿Qué tan diferentes resultaron ser los métodos de enseñanza de programación para artistas y diseñadores en contraste con los utilizados tradicionalmente en un contexto técnico-informático?
2	¿Cómo se manejó la clase para sincronizar a los estudiantes primerizos y experimentados por igual?
3	¿Cómo se impulsa el proceso creativo, la búsqueda de símbolos y mensajes, el pensamiento crítico, la perspectiva y el alma en la educación técnica formal?
4	¿Cómo se manejó el primer día de clases?
5	¿Cuál de todos los ejercicios o temas tratados le resulto el más interesante o su favorito?
6	¿Hubo alguna ocasión o día que haya considerado su peor experiencia durante el taller?
7	¿Cuál considera usted que haya sido la solución o creación más memorable que haya apreciado durante la clase?
8	¿Cuál sería su sugerencia para los educadores que impartan a los artistas y programadores por igual?

5.2. Conclusiones

Enseñar arte y diseño computacional siempre contara con altos y bajos sobre todo al tratarse de armar un silabo de desempeño inmediato como lo puede ser un taller, en formatos más longevos pueden abordarse aún más temas con distintos enfoques, sin embargo, la

demostración de su utilidad queda patente al observar cómo, mientras en los libros de programación técnicos tradicionales se instruye el *cómo* escribir instrucciones, acá se discute el *que* se puede crear y *porque* hacerlo, en ese sentido la redacción de este proyecto ha sido una forma de capturar ese vacío y darle un enfoque de entre todos los otros posibles que se le puedan agregar, un compendio, por un lado, para artistas y diseñadores que ya se encuentren explorando la programación como medio creativo o tengan curiosidad por empezar, y por el otro, informáticos desarrolladores de software e ingenieros que buscan formas más abiertas, expresivas y flexibles de demostrar sus habilidades.

Adicionalmente, también significa un espacio de exploración para los educadores en todas esas áreas, pues no es sencillo estructurar el plan de acción al mismo tiempo que se guía a los estudiantes en esta nueva forma de relacionar la computación con el diseño, se necesita de una amplia investigación en materia de pedagogía, tipos de cognición y métodos de enseñanza para preparar el material necesario que se ajuste a los objetivos de ese paradigma, esta forma de abordar la enseñanza de programación es alineado a las necesidades de una nueva literatura computacional en el siglo XXI, una época en la que ya no se puede concebir a la programación como una habilidad limitada a los herméticos círculos de ingeniería y administración de negocios, una época en la cual se aprecia su aplicabilidad en el diseño, arquitectura, música, humanidades, periodismo, activismo y demás campos creativos.

Una muestra de ello es el grado de extensión a los que se pueden adaptar estas ideas en diferentes interdisciplinas, después de todo, p5.js es un derivado de Processing y como tal, las habilidades técnicas adquiridas sobre el control de patrones y formas visuales son extensibles a otros entornos como Tracery, MSP, Jitter, Arduiono, Unity, etc. Cada uno enfocado en diferentes estilos de trabajo e inteligencias como el aspecto visoespacial, musical, lingüístico, kinestésico y lógico, al tiempo que se procura siempre abordarlos con herramientas de código

abierto y software libre que se alineen a la filosofía de democratización del aprendizaje de programación.

La otra parte fundamental son los módulos compuestos, ejercicios más complejos que a diferencia de los técnicamente enfocados poseen más de una capa de abstracción, la mayoría de ellos han sido propuestas recurrentes en los talleres de p5.js impartidos a lo largo del mundo alrededor en las últimas décadas, esto ha permitido encontrar énfasis en aspectos más cercanos al diseño modular y visual; el patrón iterativo, generador de rostros y alfabeto modular en lo generativo, la representación del tiempo y elementos estocásticos en lo interactivo, el caleidoscopio y el filtro de colores en la trans medianidad, etc. Crenado nuevos conceptos prácticos y estéticos en el medio de la computación.

Finalmente, queda recalcar que el objetivo principal del proyecto siempre será unir a dos campos que superficialmente serian irreconciliables a través de experiencias compartidas y de la exploración de la historia, algo que es poco común para los programadores pero que les permitirá adquirir nuevas perspectivas, por supuesto que, ninguna de las interpretaciones de las obras es definitiva, pues hacer una investigación exhaustiva de la historia del arte y sus orígenes es difícil incluso para los estudiantes de arte, pero construir de a poco sus bases es un desafío que vale la pena para los educadores del futuro, todo aquello sumado a la exploración del formato, las metodologías de enseñanza, las formas de aprendizaje y la exploración de ideas abrirán el camino a nuevas formas de pensamiento.

5.3. Recomendaciones técnicas y pedagógicas

Finalmente, el proceso de síntesis de aprendizaje concluye con la recopilación de posibles formas de mejoramiento de la calidad e integridad técnica de los ejercicios de programación propuestos por un lado y, por el otro, maneras de mejorar la metodología de impartición de las clases que lo conforman, basados en parte por el libro de Hamilton, ya que realmente no es

algo en lo que se profundizo en las secciones anteriores por el enfoque técnico, en el caso de los ejercicios, o teórico, en el caso del plan de acción del taller y sus elementos.

5.3.1. Integración del lenguaje DBN

Llevar a cabo los mismos ejercicios o similares, pero siguiendo la metodología descrita en el libro *Design by Numbers* con el lenguaje del mismo nombre ya que explora la misma filosofía de acercamiento al diseño y no tanto a los complicados cálculos matemáticos de la programación grafica tradicional, además de ser la inspiración directa del lenguaje *Procesing* y de p5.js, cuenta con su propio entorno de edición de código, comandos reminiscentes de Logo, Java y Basic, y arte algorítmico.

5.3.2. Integración de lenguajes de alto nivel

Comúnmente se aprenden los conceptos básicos de programación en lenguajes de alto nivel como Python, Java o JavaScript, en ese sentido, si bien es más complicado, también es posible y conveniente crear diseños abstractos enfocados a la codificación y el arte por igual a través de esos mismos lenguajes, tópicos como el color, trazado, paisajes y retratos se pueden replicar, pero interiorizando los fundamentos de otros paradigmas muy comunes como la orientación a objetos.

5.3.3. Agregar un enfoque metodológico

Normalmente en la programación o desarrollo de proyectos se siguen ciertos pasos estandarizados dentro de una metodología de desarrollo de software, sería ideal instruir dicho enfoque dentro del taller así sea en los pequeños, complejos o grupales ejercicios y al tratarse en su mayoría de prototipar diseños primero se puede adaptar un ciclo en espiral para el refinamiento de ideas, codificación, metas alcanzadas y aproximación de resultados.

5.3.4. Desarrollar propuestas interactivas

En lo que contempla el taller se han explorado ejercicios sobre entrada y salida de datos, transiciones dinámicas y cambios de estado, lo ideal sería integrar todo eso en una sola

aplicación, como un mini videojuego, que cuente con gráficas, interfaces y controles responsivos a un solo botón, dicho proyecto requerirá que se exploren nuevos patrones de diseño, como las máquinas de estado finitas o el patrón de instancia única.

5.3.5. Consumir y materializar datos

P5.js cuenta con los suficientes recursos para implementar un acercamiento a las redes neuronales y métodos de análisis de datos adaptables a sus elementos visuales que son de utilidad en robots de conversaciones automáticos, paisajes o criptografías generativas, mezclas de sonidos, trazo de gráficos estadísticos, etc. Lo importante es diseñar un código que sea capaz de segregar datos y transformar la información resultante en forma gráfica.

5.3.6. Extracción de experiencias personales

Llevar al plano tangible los objetos y tópicos inherentes a los ejercicios explorados hasta ahora es especialmente útil al ser algunos de estos reminiscentes de texturas, patrones, pinturas y teoría del color, temas relacionados a los fundamentos de las artes visuales y las artes plásticas con las que se puede experimentar y aplicar de primera mano el proceso creativo entre los alumnos, incluso integrando sus propias experiencias personales, en lugar de limitarse a la réplica digital únicamente.

El objetivo es inspirar una conexión física y emocional con los temas del taller, los cuales, aparte de crear imágenes complejas a través de herramientas digitales, están enfocados a explorar el trasfondo de sus fundamentos, ampliando los horizontes del pensamiento y apreciación de las artes visuales. Se puede pensar en todo tipo de actividades para lograrlo, he aquí algunas de las ideas más comunes:

- **Crear un corto animado** a través de las herramientas que dispone p5.js para renderizar una o varias imágenes al mismo tiempo.
- **Trazar diagramas, pegar posters o tomar fotografías** de cualquier cosa que los que los alumnos deseen expresar, que les recuerde a los tópicos explorados en los ejercicios

o algún otro tema derivado que les haya llamado la atención a partir de un punto en específico.

- **Planificar una expedición** hacia un lugar relacionado con el tema, como lo puede ser en nuestro caso un museo, un monasterio, una biblioteca, una galería, etc.
- **Traer objetos multi sensoriales** que los alumnos puedan sentir en sus 5 sentidos, por ejemplo, tocar varias telas de diferentes texturas y bordados, pinturas al óleo o en acuarela, esculturas de barro o tallados en madera, etc.

5.3.7. Lluvia de ideas

Idear entre todos un conjunto amplio de ideas espontaneas en cortos periodos de tiempo que potencialmente podrían funcionar resulta útil para dar más espacio a los alumnos a explorar aquellas cosas que les interese ver particularmente y no aburrirlos al estar siempre sujetos estrictamente a los ejercicios del plan del taller.

El objetivo de la actividad es reintegrar dos de los métodos de enseñanza vistos al inicio en el proceso creativo; la divergencia se da mejor en grupos que activamente colaboran y comparten entre sí para dar con las mejores ideas, mientras que la convergencia se da después al momento de filtrar y discutir aquellas con más potencial de llevarse a cabo dentro de las expectativas previstas.

Si bien el proceso del pensamiento colectivo e individual es complejo, muchas investigaciones señalan que una combinación común entre los procesos creativos exitosos es la mezcla entre la experiencia personal, las habilidades especiales y el contexto cultural, en nuestro caso, los programadores por lo general tenderán a crear ideas basadas en lo que ya conocen para desafiar sus habilidades y no tanto en lo creativo, caso contrario a como pasa con los artistas que usan sus experiencias personales para dar una respuesta subjetiva al tema que estén tratando, al poner a ambos tipos de estudiantes en colaboración se creara una dinámica balanceada para traer lo mejor de ambos contextos.

5.3.8. Construcción retroactiva

Se refiere a trabajar en nuevas ideas en base a otras que ya han tenido éxito, es decir, tomar varios de los pequeños ejercicios en los que se ha trabajado hasta el momento y tomar lo aprendido de cada uno para crear algo grande que combine varios de esos elementos. En ese sentido, el líder del taller siempre deberá buscar formas de inspirar nuevas ideas y una de ellas es hacer retrospectiva de los logros conseguidos para diseñar soluciones que van más allá de lo convencional, lo cual no es fácil ya que, por lo general, es más seguro hacer cosas que ya han funcionado antes con instrucciones específicas, sin embargo, crear nuevas soluciones en base a poca o nula documentación siempre es más intimidante.

El objetivo es incentivar la habilidad de visionar y enfocar soluciones hacia el futuro, mediante la reflexión tanto de las ideas exitosas como de aquellas que han fallado, pues otro punto importante es combatir el rechazo inherente hacia la derrota, algo completamente natural durante cualquier proceso creativo. A partir de allí, existen algunos lineamientos para alinear dicho objetivo con la discusión de proyectos; reconocer los proyectos exitosos por sus ideas y su equipo, más no por conveniencias externas o la participación de individuo más extraordinario que su equipo, discutir posibles factores externos determinantes para el éxito o fracaso como los competidores, el mercado o los servicios.

Si existen fallas, discutirlas internamente con el equipo para calibrarlas, mejorar las formas de trabajo, comunicación y desempeño, no reservar conclusiones importantes al equipo, evitar caer en situaciones redundantes o ideas ajenas, recordar que en situaciones comprometedoras es cuando más se necesita acudir a la retroalimentación de otros equipos y practicar entre todos el ejercicio de convergencia colectiva.

5.3.9. Saber complementar ideas

Todas las ideas se complementan entre sí y estas a su vez provienen de las que les precedieron, lo importante al combinarlas es no perder la esencia de la idea original y el objetivo a conquistar, después de todo, si existe una idea es muy probable que haya muchas formas de mejorarla sin deshacer la investigación realizada sobre esta, es importante recalcar que hay una diferencia entre converger nuevas ideas, lo cual es más fácil, a mejorar otras ya existentes, a lo cual los programadores, por lo general, tienden a mirar con ojo crítico proyectos que les son presentados señalando sus debilidades y por ello prefieren trabajar en sus propias ideas a tomar inspiración de otros autores o movimientos, caso completamente contrario al de los artistas.

Para mejorar una idea se necesita identificar las razones que satisface y de qué forma lo hace, esto es analizar a las personas, sus necesidades y que impacto supone llevar dicha idea a cabo en sociedad, la socialización de mejorías se realiza a través de votación durante la cual se tienen en cuenta solo aquellas modificaciones que sean más factibles en ejecución y que no se despeguen tanto de la idea original, siempre manteniendo un tono constructivo y positivo.

ANEXOS

Códigos completos de los ejercicios que no pudieron ser adjuntos anteriormente junto a la teoría por disposición del espacio, aquellos ejercicios con una anotación con asterisco se encuentran aquí.

Ejercicio 2.1.1

```
/*Instrucciones básicas para graficar las primeras primitivas simples*/

function setup() {
  //monta el tamaño del canvas
  createCanvas(400, 400);
}

function draw() {
  //color del trazo
  stroke("Black");
  //no rellena el color de fondo
  noFill();

  push();//setea el estilo de dibujo y transformaciones actuales
  scale(1.3);//setea el tamaño para todas las primitivas

  //triángulo
  triangle(20.5, 75, 48.5, 20, 76, 75);

  //rectángulo
  rect(110, 20, 55, 55);

  //círculo
  circle(225, 50, 60);

  //cuadrilátero
  quad(20, 105, 80, 105, 80, 145, 20, 145);
  pop();//resetea a la configuración inicial

  //elipse
  push(); //vamos a aplicar una trasnofrmación de rotación
  rotate(45); //solo para la elipse ya que esa instrucción
  ellipse(232, -65, 40, 80); //no cuenta con un parametro para ello
  pop();
```

```
  //arco
  push();
  arc(295, 165, 80, 80, 0 + QUARTER_PI, 180 + HALF_PI, PIE);

  //recta
  line(150, 220, 230, 300);
  pop();

  //polígono
  push();
  //en figuras complejas, utilizamos instrucciones de traslación
  //para desplazar la figura a nivel general, así solo se centra
  //en la construcción de los puntos que lo conforman
  translate(10, 190);
  scale(1.6);
  beginShape();
  vertex(20, 20);
  vertex(40, 20);
  vertex(40, 40);
  vertex(60, 40);
  vertex(60, 60);
  vertex(20, 60);
  endShape(CLOSE);
  pop();

  //polilínea
  push();
  scale(0.6);
  translate(400, 360);
  beginShape();
  vertex(30, 20);
  vertex(85, 20);
  vertex(85, 75);
  vertex(140, 75);
  vertex(140, 130);
  vertex(195, 130);
  endShape();
  pop();
}
```

Ejercicio 2.1.2

```
/* Recreación de la Composición No. III de Piet Mondrian */
function setup() {
  createCanvas(540, 430);
  noStroke();//sin marcos para el canvas
}

function draw() {
  //rellena el color de fondo en negro
  background(0);

  //sección en rojo
  push();
  fill("red");//setea el color para rellenar la figura
  rect(0, 0, 250, 200);

  //sección en blanco
  fill("white");
  rect(0, 210, 250, 100);
  rect(260, 210, 270, 100);
  rect(60, 320, 190, 100);
  rect(260, 0, 270, 200);
  rect(260, 410, 150, 10);
  rect(420, 320, 110, 100);

  //sección en amarillo
  fill("yellow");
  rect(0, 320, 50, 100);

  //sección en azul
  fill("blue");
  rect(260, 320, 150, 80);
  pop();
}
```

Ejercicio 2.4.1

```
/*Traza la curva que pasa por una función cuadrática*/
function setup() {
  createCanvas(400, 400);
  noLoop();
}

function draw() {
  //evalua la función cuadrática entre los valores -11 y 10
  graphFunction(-11, 10);
}

function graphFunction(x1, x2) {
  //retorna el arreglo de pares ordenados, el punto
  //más alto y el más bajo de la parabola, todos en un objeto
  let obj = getValues(x1, x2);
  //separa el arreglo del objeto obtenido
  let points = obj.values;
  let y1 = obj.min;//separa el punto mínimo del objeto
  let y2 = obj.max;//separa el punto máximo del objeto
  //con los datos anteriores proccede a dibujar la curva
  drawGraph(points, x1, x2, y1, y2);
  //adicionalmente traza los ejes ordenados para apreciar mejor la curva
  drawAxes(x1, x2, y1, y2);
}

function getValues(x1, x2){
  let arr = [];
  let min = Infinity;
  let max = -Infinity;

  const a = -1; //no puede ser igual a cero
  const b = 4; //afecta la ubicación del vertice
  const c = 5; //corte de la curva con los ejes

  //iteración que evalua la función ax^2+bx+c en todos los puntos
  for(let x = x1; x <= x2; x++)
  {
    let y = a * pow(x, 2) + b * x + c;
    arr.push({x, y});//guarda el par ordenado en el arreglo
```

```
    //guarda la ordenada Y más alta y baja de la curva
    if(y < min)
      min = y;
    if(y > max)
      max = y;
  }

  //devuelve el arreglo de pares ordenados y los vértices
  return {values: arr, min, max};
}

function drawGraph(arr, x1, x2, y1, y2){
  push();
  stroke("navy");
  strokeWeight(2);
  for(let i = 0; i < arr.length - 1; i++)
  {
    //se deben mapear los puntos para que se ajusten al canvas
    let x = arr[i].x;
    let y = arr[i].y;
    //mapeamos el primer punto contiguo
    let screenX = map(x, x1, x2, 0, width - 1);
    let screenY = map(y, y1, y2, height - 1, 0);
    //mapeamos el segundo punto contiguo
    let xNext = arr[i + 1].x;
    let yNext = arr[i + 1].y;

    let screenXNext = map(xNext, x1, x2, 0, width - 1);
    let screenYNext = map(yNext, y1, y2, height - 1, 0);
    //traza una pequeña recta por cada par de coordenadas
    line(screenX, screenY, screenXNext, screenYNext);
  }
  pop();
}

function drawAxes(x1, x2, y1, y2){
  //escala el valor 0 a un valor en el rango entre la altura
  //max y min de la curva, y la altura del canvas
  let axesX = map(0, y1, y2, height - 1, 0);
  line(0, axesX, width - 1, axesX);
  //escala el valor 0 a un valor en el rango entre la apertura
  //max y min de la curva, y el ancho del canvas
  let axesY = map(0, x1, x2, 0, width - 1);
  line(axesY, 0, axesY, height - 1);
}
```

Ejercicio 2.4.2

```
//variable sobre la que se calcula el angulo del punto que gira
let angle = 0
//constante sobre la que el angulo se recalcula progresivamente
const angularSpeed = 0.01
//variable sobre la que se calcula el radio del punto que
//gira en un ángulo determinado
let distance = 0
//constante sobre la que dicho radio se recalcula progresivamente
const distanceSpeed = 0.025

function setup() {
  createCanvas(600, 450);
  background(250);
  frameRate(30)

  //dibuja los ejes que dividen el canvas en una cuadrícula
  stroke("blue");
  strokeWeight(2)
  line(0, height/2, width, height/2)
  line(width/2, 0, width/2, height)
}

function draw() {
  //grueso de la espiral
  strokeWeight(4)
  //color del trazo
  stroke('black')
  //ubica el origen de la espiral en el centro de la cuadrícula
  translate(width/2, height/2)

  //entre más grande sea este valor más aumentara la
  //velocidad angular en el bucle
  const drawingSpeed = 70
  //para almacenar las coordenadas de partida en cada iteración
  let prevX = 0, prevY = 0
```

```
  for (let i = 0; i < drawingSpeed; i++)
  {
    //continua con la rotación hasta que el radio
    //supere la mitad del ancho del canvas
    if (distance > width/2) noLoop()

    //continua girando el punto a una velocidad angular constante
    angle += angularSpeed
    distance += distanceSpeed

    let x = sin(angle) * distance
    let y = cos(angle) * distance

    line(prevX, prevY, x, y)
    prevX = x
    prevY = y
  }
}
```

Ejercicio 2.5.1

```
/*Trazo de un polígono regular de 5 vértices*/
function setup() {
  createCanvas(500, 400);
}

function draw() {
  background(102);
  //posiciona la figura en el medio
  translate(width * 0.5, height * 0.5);
  //si el ángulo es positivo gira en el sentido del reloj
  rotate(frameCount / 100.0);
  //crea una estrella de 5 vértices, con tamaño 70 y anchura 30
  star(0, 0, 30, 70, 5);
}

function star(x, y, radius1, radius2, npoints) {
  //divide 360° sobre el número de vertices para calcular
  //el ángulo de las puntas externas
  let angle = TWO_PI / npoints;
  //calcula el ángulo necesario para las bases internas
  let halfAngle = angle / 2.0;

  beginShape();
  for (let a = 0; a < TWO_PI; a += angle)
  {
    //primero calcula la distancia entre los puntas externas
    //es proporcional al radio2
    let sx = x + cos(a) * radius2;
    let sy = y + sin(a) * radius2;
    vertex(sx, sy);
    //proporciona la abertura entre los vertices
    //es proporcional al radio1
    sx = x + cos(a + halfAngle) * radius1;
    sy = y + sin(a + halfAngle) * radius1;
    vertex(sx, sy);
  }
  endShape(CLOSE);
}
```

Ejercicio 2.5.2

```
/*Molécula animada*/

//extremo superior para el tamaño de la molécula
//debe estar fuera de la función draw para que no se resetee
//cada vez que termine la ronda de circunferencias laterales
let yoff = 0.0;

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

function draw() {
  //coloca un fondo negro para apreciar la forma de la molécula
  background(0);
  //posiciona la figura en la mitad del canvas
  translate(width / 2, height / 2);
  //el radio para todas las circunferencias generadas debe ser
  //constante para mantener la forma de la molécula integra
  var radius = 150;

  beginShape();
  //extremo inferior en el rango de valores
  let xoff = 0;
  for (var a = 0; a < TWO_PI; a += 0.1)
  {
    //escala un valor aleatorio uniforma a traves del ruido perlin
    let offset = map(noise(xoff, yoff), 0, 1, -25, 25);
    //genera las circunferencias a partir del radio constante
    //pero que sean más o menos variadas según el valor escalado
    let r = radius + offset;
    let x = r * cos(a);
    let y = r * sin(a);
    vertex(x, y);
    //crea otra circunferencia un poco desplazada hacia al lado
    xoff += 0.1;
  }
  endShape();
  //crea otra ronda de circunferencias desplazadas hacia abajo
  yoff += 0.01;
}
```

Ejercicio 3.1

```
/*Caleidoscopio*/

let face; //para la imagen
let shape; //para la primitiva
let hex;

//cargamos la imagen como preparación
function preload() {
  //almacena la imagen cargada desde los archivos del proyecto
  face = loadImage("face.jpg");
}

function setup() {
  createCanvas(650, 480);

  //creamos una figura nativa de p5 con una altura y ancho
  //determinado y la almacenamos en una variable
  shape = createGraphics(100, 87);
  shape.noStroke();
  shape.beginShape();
  //esto crea un triángulo
  shape.vertex(0, shape.height);
  shape.vertex(shape.width / 2, 0);
  shape.vertex(shape.width, shape.height);
  shape.endShape(CLOSE);

  //enmarca la figura creada con la imagen cargada
  face.mask(shape);

  //creamos otra figura en forma de hexagono por lo que
  //se necesitara el doble de las dimensiones del triángulo
  hex = createGraphics(shape.width * 2, shape.height * 2)
  //renderiza los 6 triangulos del hexagono considerando su posición, rotación y reflejo
  hex.image(hexSide(face, false, 0), 0, 0);
  hex.image(hexSide(face, true, 1), shape.width / 2, 0);
  hex.image(hexSide(face, false, 2), shape.width, 0);
  hex.image(hexSide(face, true, 3), 0, shape.height);
  hex.image(hexSide(face, false, 4), shape.width / 2, shape.height);
  hex.image(hexSide(face, true, 5), shape.width, shape.height);
}

function draw() {
  background(0);
  noLoop();

  let isOddRow = false;
  let xOffset = 0;

  for (let y = -hex.height / 2; y < height; y += hex.height / 2)
  {
    if (isOddRow)
      xOffset = shape.width * 1.5;
    else
      xOffset = 0;
```

```
    for (let x = -shape.width / 2; x < width; x += hex.width + shape.width)
    {
      image(hex, x + xOffset, y);
    }
    isOddRow = !isOddRow;
  }
}

//toma la imagen proyectada en el triangulo y lo rota reflejandose
function hexSide(img, reflect, rotations) {

  let g = createGraphics(img.width, img.height);
  let a = 50 * tan(radians(25));
  let b = img.height - a;
  let c = b - img.height / 2;

  g.imageMode(CENTER);
  g.translate(g.width / 2, b);

  if (rotations % 2 === 1)
    g.translate(0, img.height - 2 * b);

  //la unica forma de hacer un efecto espejo de una figura es
  //escalandolo a la inversa
  if (reflect)
    g.scale(-1, 1);

  //rota la figura desplazada y reflejada en 60°
  g.rotate(radians(rotations * 60));
  g.image(img, 0, -c);
  return g;
}
```

Ejercicio 3.2

```
/*Pinta un lienzo lleno de burbujas que explotan al hacer click*/
var bubbles = [];

function setup() {
  createCanvas(600, 400);

  //toma una cantidad aleatoria de burbujas
  let numBubbles = random(20, 40);
  //genera los pares ordenados en cualquier parte del canvas
  //además serán de tamaños variables, guarda las características
  //de cada burbuja en un arreglo
  for(var i = 0; i < numBubbles; i++)
  {
    bubbles.push({
      x: random(width),
      y: random(height),
      radius: random(5, 60)
    });
  }
}

function draw() {
  //coloca un fondo lila
  background(209);
  //grueso y contorno de las burbujas
  strokeWeight(2.5);
  stroke("white");

  //renderiza todas las burbujas
  for (var i = 0; i < bubbles.length; i++)
  {
  var bubble = bubbles[i];
    //si el cursor esta por encima de la burbuja
    if (dist(mouseX, mouseY, bubble.x, bubble.y) < bubble.radius)
    {
      if (mouseIsPressed)
        bubbles.splice(i, 1);//al hacer click desaparece
  }
    else
      //deja la burbuja tal como esta y coloca el relleno rosa
      fill(255, 220, 200, 200);
    ellipse(bubble.x, bubble.y, bubble.radius*2);
  }
}
```

Ejercicio 3.3

```
//Patrón Moiré
//espacio entre los conjuntos de líneas
let step = 30;

function setup() {
  createCanvas(650, 400);
  //trazar líneas muy ligeras
  strokeWeight(1);
}

function draw() {
  background(255);

  //posiciona la primera reja en la mitad
  translate(width/ 4, 0);
  //primera rejilla de líneas paralelas en 90 grados
  for(let x = 0; x ⩽ width / 2; x += step)
    setOfLines(4, x, "#d99e64");
  for(let x = 15; x ⩽ width / 2; x += step)
    setOfLines(4, x, "#c9b097");

  //posiciona la segunda reja sobre la primera
  translate(width / 4, 0);
  //rotación de la segunda rejilla
  rotate(PI / 3);
  //segunda rejilla de líneas paralelas en horizontal
  for(let y = -100; y ⩽ height / 2; y += step)
    setOfCroppedLines(y, "#d99e64");
  for(let y = -115; y ⩽ height / 2; y += step)
    setOfCroppedLines(y, "#c9b097");
}
```

```
function setOfLines(numLines, pos, colorCode){
  for(let cont = 0; cont ⩽ numLines; cont++)
  {
    stroke(colorCode);
    line(pos, 0, pos, height);
    pos += 3;
  }
}

function setOfCroppedLines(pos, colorCode){
  for(let x = 0; x ⩽ width / 2; x+= 2)
  {
    stroke(colorCode);
    line(x, pos, x, pos + 13);
  }
}
```

Ejercicio 3.4

```
//Schotter por Georg Ness (1968)
let angle = 0;
let size;

function setup() {
  createCanvas(500, 500);
  noLoop();
}

function draw() {
  background(255);
  stroke(0);
  strokeWeight(2);

  size = min(width / 10, height / 10);

  for (let y = size; y < height - size; y += size)
  {
    for (let x = size; x < width - size; x += size)
    {
      push();
      translate(x + size / 2, y + size / 2);
      rotateAmount = random(-angle, angle);
      rotate(rotateAmount);
      square(-size / 2, -size / 2, size);
      pop();
    }
    angle += 0.05;
  }
}
```

Ejercicio 3.5

```
//Filtro de paleta de colores sobre una imagen
let img;
let palette;
let y = 0;

function preload(){
  //carga la imagen en una variable
  img = loadImage('sunset.png');
}

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

  /*si la imagen tiene dimensiones superiores a las del canvas
  re ajusta el tamaño de la misma para que quepa en el canvas*/
  img.resize(width, height);

  /*Arreglo que representa la paleta de colores que conformaran
  el filtro, entre más colores se agreguen más preciso será el
  filtro que reemplace los colores originales con otros muy
  parecidos

  En este caso se ha tomado ciertos códigos hexadecimales
  de colores similares a los de un atardecer*/
  palette = [
    '#264653', '#2a9d8f',
    '#e9c46a', '#f4a261',
    '#e76f51'
  ];
  /*seteamos la imagen original de fondo desde el principio
  de esa manera da la ilusión de ver como se aplica el filtro
  en tiempo real*/
  image(img, 0, 0);
}
```

```
function draw() {
  /*al saber que la imagen esta a la par con el canvas podemos
  usar sus dimensiones para renderizar la imagen recorriendo
  pixel por pixel*/
  for(let x = 0; x < width; x++)
  {
    //obtiene el color del pixel ubicado en esta posición
    const pixelColor = img.get(x, y);
    //obtiene el color de la paleta más cercano al color anterior
    const paletteColor = getClosestColor(pixelColor);
    //dibuja dicho pixel manualmente con el color obtenido
    stroke(paletteColor);
    point(x, y);
  }
  //renderiza una fila del buffer de pixeles en cada frame
  y++;
  if(y ⩾ height)
    noLoop();
}

//determina el color de la paleta que sea más parecido al
//color del pixel analizado
function getClosestColor(pixelColor){
  //extrae los valores de rojo, verde y azul del pixel
  const pixelR = red(pixelColor);
  const pixelG = green(pixelColor);
  const pixelB = blue(pixelColor);

  /*distancia mínima que determinara el color de la paleta
  más cercano al color del pixel, lo seteamos con una
  grande de primeras pero luego se ira actualizando*/
  let minDistance = 999999;
  let targetColor;
```

```
  //recorre todos los colores de la paleta
  for(const c of palette)
  {
    //extrae los valores de rojo, verde y azul del color en turno
    const paletteR = red(c);
    const paletteG = green(c);
    const paletteB = blue(c);

    /*calcula la distancia que hay entre los valores RGB del pixel
    y del color de la paleta como si fuera la distancia entre
    dos pares ordenados*/
    const colorDistance =
          dist(pixelR, pixelG, pixelB,
               paletteR, paletteG, paletteB);

    /*si la distancia calculada para este color de la paleta
    es menor al antecesor entonces se ha encontrado el
    nuevo color más cercano al pixel original*/
    if(colorDistance < minDistance)
    {
      targetColor = c;
      minDistance = colorDistance;
    }
  }
  return targetColor;
}
```

Ejercicio 4.1

```
//Textura de cuadriculas de colores
size = 30;
widthMult = 4;
heightMult = 4;

function setup() {
  createCanvas(700, 700);
  angleMode(DEGREES);
  noLoop();
}

function draw(){
  strokeWeight(3);

  startR = random(165);
  startG = random(165);
  startB = random(165);

  background(startR + 45, startG + 45, startB + 45);

  for (x = width; x > -size * widthMult; x -= size)
  {
    for (y = height; y > -size * heightMult; y -= size)
    {
      fill(
        random(startR, startR + 90),
        random(startG, startG + 90),
        random(startB, startB + 90)
      );
      push();
      translate(x + size / 2, y + size / 2);
      rect(0,0,
        size * floor(random(1, widthMult)),
        size * floor(random(1, heightMult))
      );
      pop();
    }
  }
}
```

Ejercicio 4.2

```
/*Genera una nueva cara aleatorizando algunos rasgos
faciales en cada ejecución*/

//almacena el conjunto de valores aleatorios en un solo objeto
let obj;

function setup() {
  createCanvas(400, 400);
  noStroke();
  /*no se puede usar la palabra reservada random a
  secas para la función*/
  obj = random1();
}

function draw() {
  background(255);
  //construye el rostro recuperando los parametros obtenidos
  head(obj.headWidth);
  eyes(obj.eyeY, obj.eyeRedness);
  hair();
  nose(obj.noseX, obj.noseY);
  mouth(obj.mouthX);
}

//obtiene valores aleatorios para parametrizar la posición
//de los ojos, boca y nariz, ancho de cabeza, y la
//pigmentación de las cuencas de los ojos
function random1() {
  let headWidth; //ancho de cabeza
  let eyeY; //solo la posición vertical de los ojos varia
  let eyeRedness; //el contorno de ojos es en varios tonos de rojo
  let mouthX; //desplaza la boca hacia los lados
  //la posición de la nariz si es aleatoria en ambos ejes
  let noseX;
  let noseY;

  eyeY = random(80, 120);
  noseX = random(175, 225);
  noseY = random(200, 250);
  headWidth = random(200, 300);
  eyeRedness = random(0, 255);
  mouthX = random(150, 250);
```

```
  return {eyeY, noseX, noseY, headWidth, eyeRedness, mouthX};
}

function hair() {
  //el cabello es constante
  beginShape();
  vertex(100, 50);
  vertex(150, 15);
  vertex(240, 25);
  vertex(260, 70);
  vertex(295, 30);
  endShape(CLOSE);
}

function eyes(eyeY, eyeRedness) {
  //cuencas de los ojos
  fill(eyeRedness, 0, 0);
  ellipse(150, eyeY, 50, 20);
  ellipse(250, eyeY, 50, 20);

  //pupilas
  fill(0);
  ellipse(150, eyeY, 20, 5);
  ellipse(250, eyeY, 20, 5);
}

function head(headWidth) {
  //el color de piel es constante
  fill(191, 165, 17);
  //la forma de la cabeza es una elipse
  ellipse(200, 200, headWidth, 350);
}

function mouth(mouthX) {
  //la boca siempre es roja
  fill(200, 0, 0);
  ellipse(mouthX, 300, 80, 30);
}

function nose(noseX, noseY) {
  //la nariz siempre es rosa
  fill(220, 120, 120);
  ellipse(noseX, noseY, 30, 100);
}
```

Ejercicio 4.3

```
//Reloj en tiempo real
let cx, cy;
let secondsRadius;
let minutesRadius;
let hoursRadius;
let clockDiameter;

function setup() {
  createCanvas(660, 400);
  stroke(255);

  //calcula el mínimo valor entre las dimensiones del canvas
  //de ese modo el diametro del reloj cabra en la mitad del canvas
  let radius = min(width, height) / 2;
  //radio del segundero y la circunferencia de puntos
  secondsRadius = radius * 0.71;
  //radio del minutero
  minutesRadius = radius * 0.6;
  //radio del horero
  hoursRadius = radius * 0.5;
  //diametro del reloj
  clockDiameter = radius * 1.7;
  //para el ancho y alto del reloj
  cx = width / 2;
  cy = height / 2;
}

function draw() {
  background(230);

  //Dibuja el fondo del reloj
  noStroke();
  fill(244, 122, 158);
  ellipse(cx, cy, clockDiameter + 25, clockDiameter + 25);
  fill(237, 34, 93);
  ellipse(cx, cy, clockDiameter, clockDiameter);
```

```
  //angulos para las funciones sin() y cos() empiezan a las 03:00
  let s = map(second(), 0, 60, 0, TWO_PI) - HALF_PI;
  let m = map(minute() + norm(second(), 0, 60), 0, 60, 0, TWO_PI) - HALF_PI;
  let h = map(hour() + norm(minute(), 0, 60), 0, 24, 0, TWO_PI * 2) - HALF_PI;

  //Dibuja el segundero
  stroke(255);
  strokeWeight(1);
  line(cx, cy, cx + cos(s) * secondsRadius, cy + sin(s) * secondsRadius);
  //dibuja el minutero
  strokeWeight(2);
  line(cx, cy, cx + cos(m) * minutesRadius, cy + sin(m) * minutesRadius);
  //dibuja el horero
  strokeWeight(4);
  line(cx, cy, cx + cos(h) * hoursRadius, cy + sin(h) * hoursRadius);

  //Traza los ticks para los segundos
  strokeWeight(2);
  beginShape(POINTS);
  for (let a = 0; a < 360; a += 6) {
    let angle = radians(a);
    let x = cx + cos(angle) * secondsRadius;
    let y = cy + sin(angle) * secondsRadius;
    vertex(x, y);
  }
  endShape();
}
```

BIBLIOGRAFÍA

Benaim, R. (2008). *The Design of Prestressed Concrete Bridges.* Londres: CRC Press.

Caralt, M., & Casal, F. (2012). *La Historia del Arte: Explicada a los jóvenes.* España: Paidós.

Chadwick, S. (27 de Abril de 2016). *Mondrian, Composition with Red, Blue, and Yellow.* Obtenido de Smarthistory: https://smarthistory.org/mondrian-composition-ii-in-red-blue-and-yellow/

Cook, T. A. (1903). *Spirals in nature and art.* Obtenido de Internet Archive: https://archive.org/details/spiralsinnaturea00cook/page/n9/mode/2up

Delaunay, R. (1909). *Saint-Séverin No. 2: Study for 'The City' with the Eiffel Tower*. Obtenido de Arts MIA Collection: https://collections.artsmia.org/art/772/saint-severin-no-2-robert-delaunay

Hamilton, P. (2016). *The Workshop Book: How to design and lead successful workshops.* Reino Unido: Pearson Business.

Levin, G., & Brain, T. (2021). *Code as Creative Medium: A handbook for computational art and design.* Massachusetts: The MIT Press.

McCarhty, L., Reas, C., & Fry, B. (2017). *Introducción a p5.js.* Estados Unidos: Processing Foundation.

McCarthy, L., & Qianqian, Y. (2023). *Ejemplos prácticos en p5.js*. Obtenido de p5.js org: https://p5js.org/es/examples/

MIT Press. (24 de Agosto de 2001). Obtenido de Design By Numbers: https://mitpress.mit.edu/9780262632447/design-by-numbers/

Oster, G., & Nishijima, Y. (1963). Moiré Patterns. *Scientific American Vol. 208, No. 5*, 54-63.

Rodenbröker, T. (Diciembre de 2022). *Tim Rodenbröker Creative Coding*. Obtenido de A brief history of Processing and p5.js: https://timrodenbroeker.de/history-processing-p5js/

Vaisman, S. (09 de Mayo de 2022). *Elementor Blog*. Obtenido de How to Use Gradients in Web Design: Trends & Examples: https://elementor.com/blog/gradients-in-web-design/

Workman, K. (2019). *p5.js Tutorials*. Obtenido de Happy Coding: https://happycoding.io/tutorials/p5js/