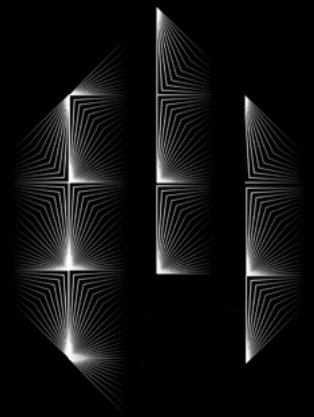
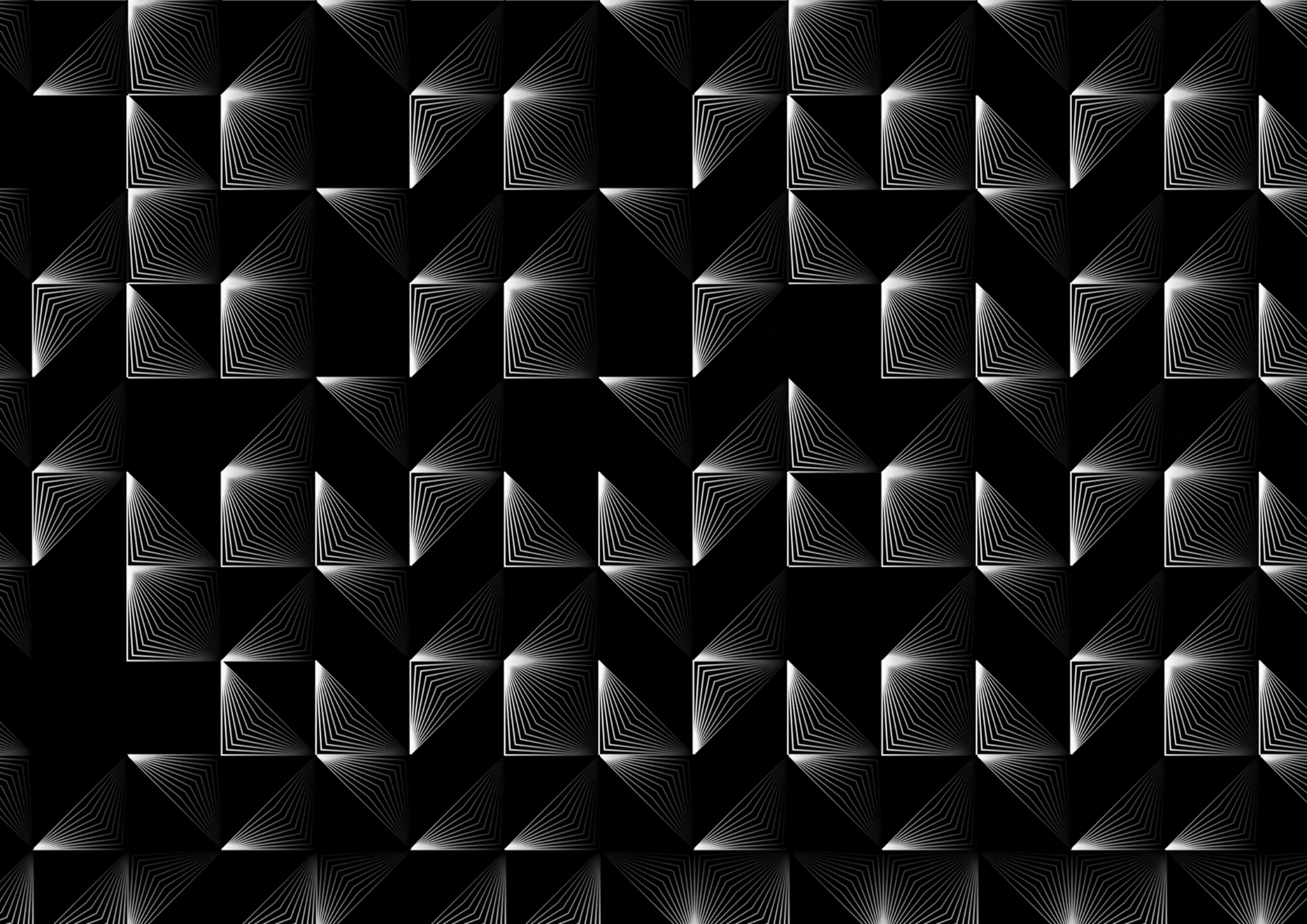
DISEÑO DE IDENTIDADES VISUALES FLEXIBLES



basadas en la modularidad

Una aplicación del código creativo a la creación de marca

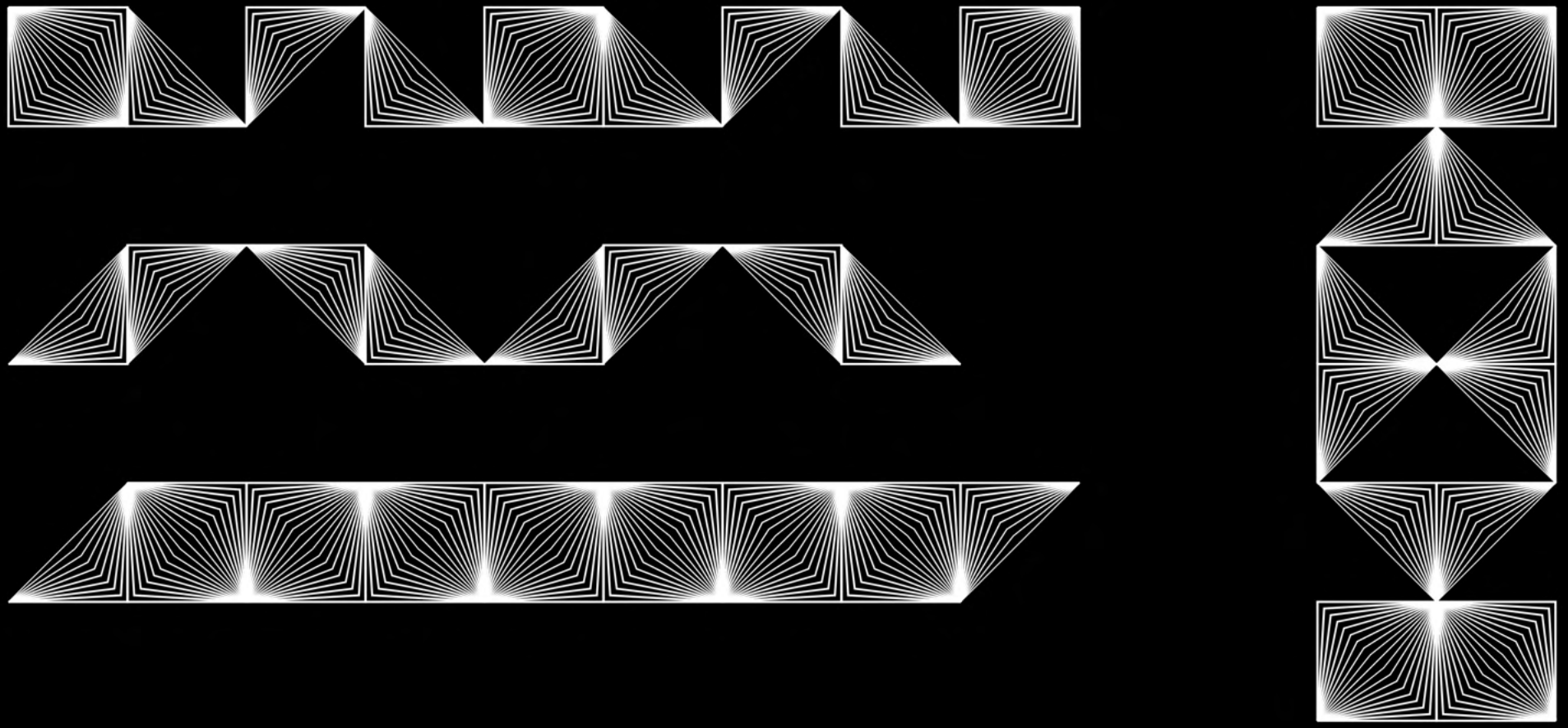


Alumna
Paula Martín Gómez de la Serna

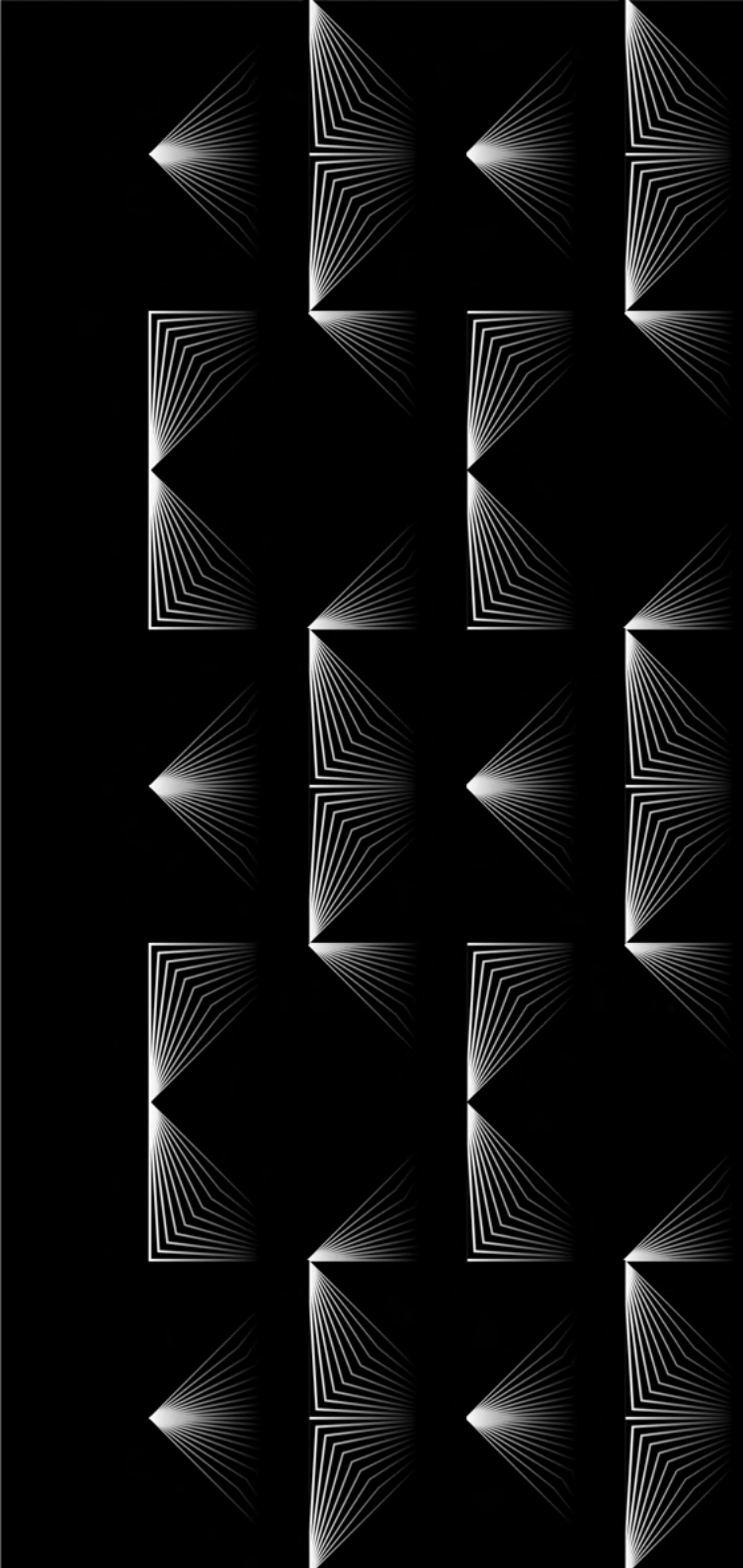
Tutor académico
Alfonso Ruiz Rallo

Trabajo de Fin de Grado
Grado en Diseño
Facultad de Bellas Artes
Universidad de La Laguna

Curso académico 2021-2022
Septiembre 2022



“Art and Technology—A New Unity”
Walter Gropius, 1923



RESUMEN

El diseño gráfico está cambiando cada vez más de lo estático a lo dinámico. Involucrar el pensamiento sistémico en el proceso de diseño conduce no solo a resultados novedosos y muy interesantes, sino también a procesos mucho más eficientes.
La necesidad de crear sistemas visuales flexibles capaces de adecuarse a diferentes contextos, audiencias o medios sin perder su identidad es cada vez mayor.
La modularidad permite crear identidades visuales cambiantes basadas en lenguajes y no únicamente en símbolos. Con este trabajo de final de grado se pretende demostrar la versatilidad de la modularidad combinada con el código creativo diseñando una propuesta elástica y adaptativa de identidad visual, utilizando la herramienta Processing, un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java.

Palabras clave

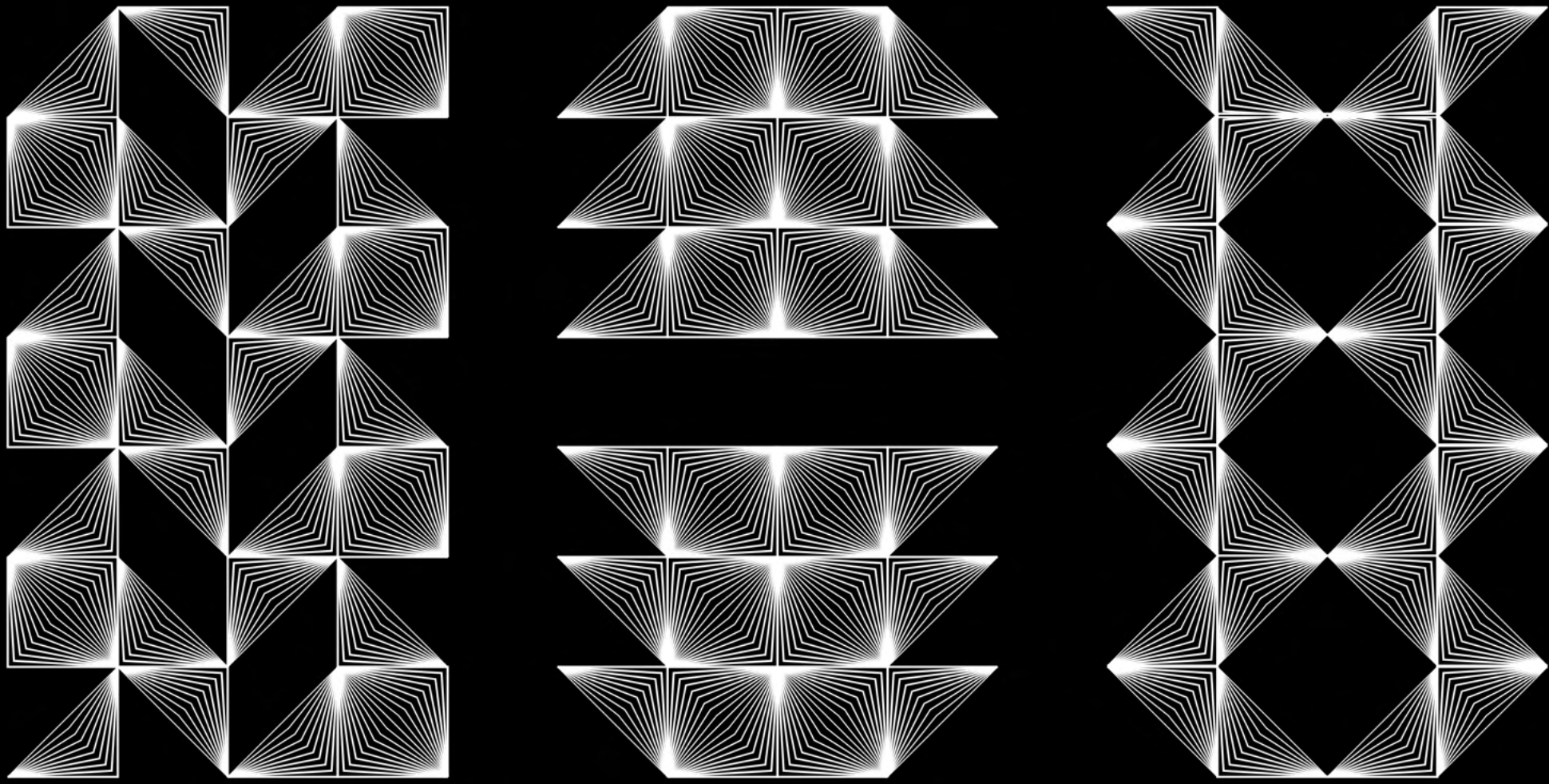
Sistemas flexibles, identidades visuales contemporáneas, diseño generativo, modularidad, Processing, código creativo, tipografía modular.

ABSTRACT

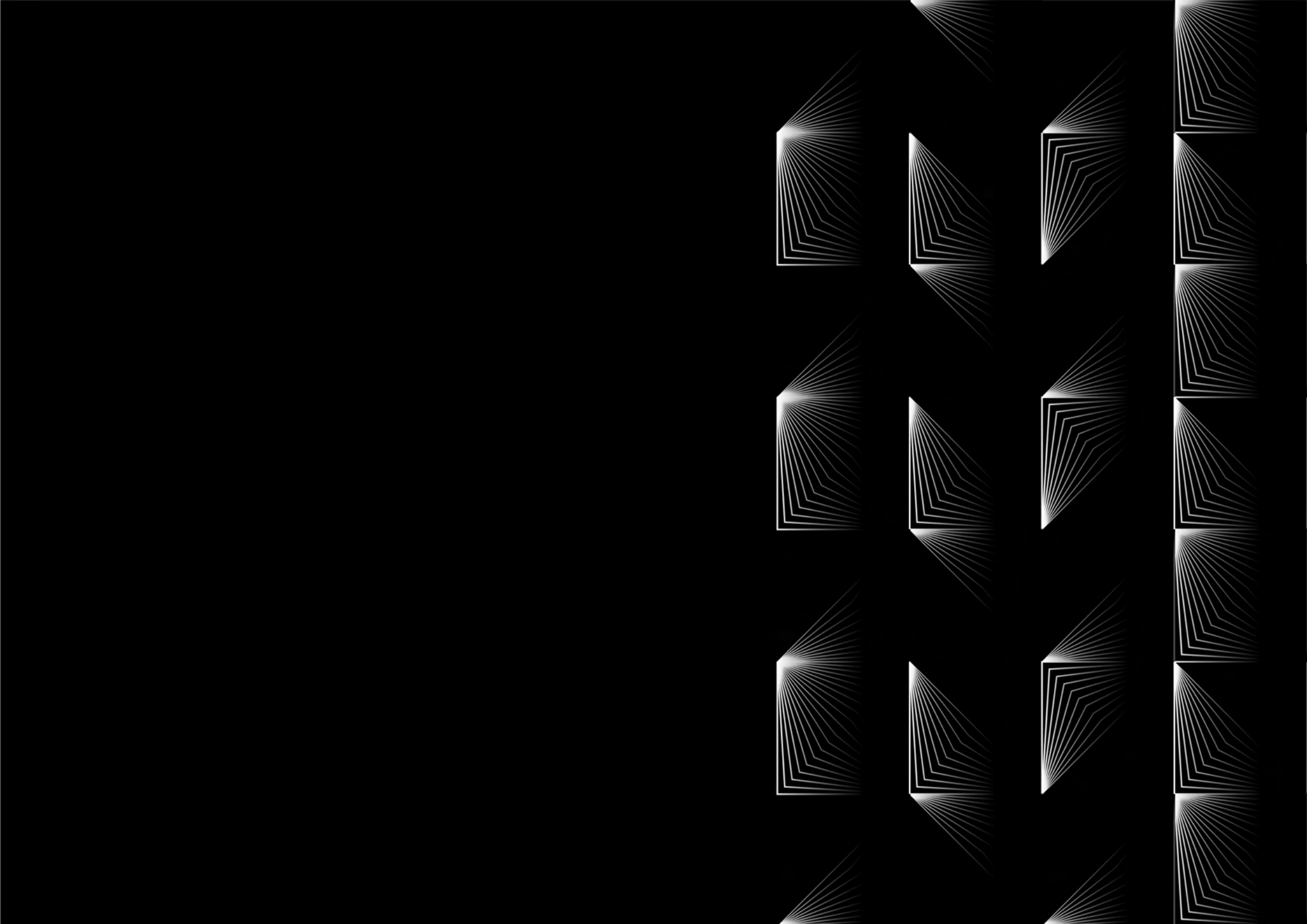
Graphic design is changing more and more from the static to the dynamic. Involving systemic thinking in the design process leads not only to novel and very interesting results, but also to much more efficient processes.
The need to create flexible visual systems capable of adapting to different contexts, audiences or media without losing their identity is increasing.
Modularity allows creating changing visual identities based on languages and not only on symbols. The aim of this final project is to demonstrate the versatility of modularity combined with creative code, elaborating an elastic and adaptive visual identity proposal using Processing as a tool, a programming language and integrated development environment. Java-based open source.

Keywords

Flexible systems, contemporary visual identities, generative design, modularity, Processing, creative coding, modular typography.



ÍNDICE



INTRODUCCIÓN

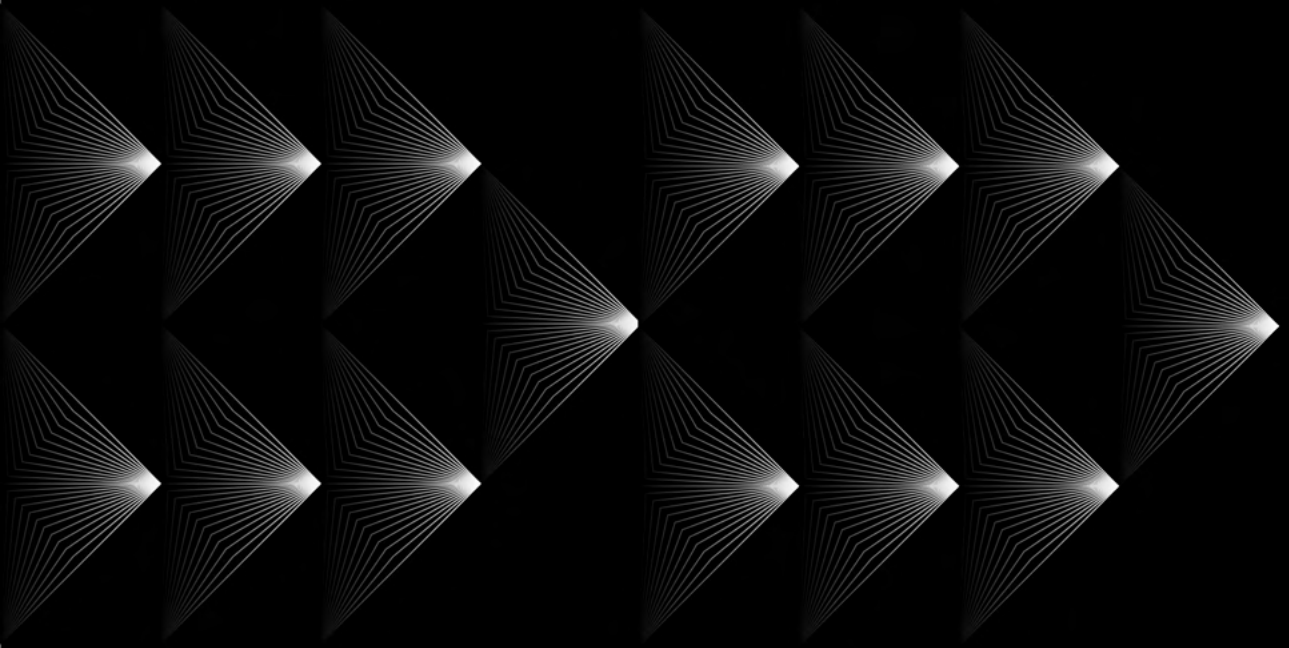
Vivimos en unos tiempos en los que el diseño gráfico viene regido por los cambios sociales y económicos que estamos viviendo, donde las nuevas tecnologías y las redes sociales transforman nuestra manera de comunicarnos. Todo ello promueve cambios en la profesión del diseñador gráfico, donde se necesitan nuevas especializaciones y herramientas que nos permitan reaccionar con agilidad a la nueva realidad.

El diseño de sistemas visuales flexibles se opone a la identidad visual basada únicamente en un logotipo inmutable. Se crean identidades dinámicas, líquidas o fluidas. Se trata de una visión que se aleja mucho de la manufactura artesanal y personalizada del proyecto, donde la empresa que crea y diseña es diferente de la que aplica y ejecuta. Como bien dijo Martin Lorenz, "la versatilidad de la comunicación hace la necesidad de los sistemas flexibles evidente" (Lorenz, 2021, pág. 17).

Actualmente, la comunicación visual sucede mayoritariamente a través de pantallas, lo que nos ofrece una infinidad de opciones de elaboración de propuestas interactivas y adaptables automáticamente al formato, dispositivo, contenido y usuario.

A su vez, la codificación creativa es un método que utiliza la programación informática para la expresión artística. En este método, el objetivo no está predefinido y el proceso se basa en el descubrimiento, la variación y la exploración de resultados en su mayoría inesperados. Es un método que no está vinculado a ningún medio. Esto es lo que la hace tan especial y lo que nos descubre su enorme potencial. Según el lenguaje de programación utilizado, el espectro de salidas es infinito.

La idea de este proyecto es sugerir una forma de pensamiento y forma de trabajo que está inspirada por la realidad de comunicar en una era digital.



OBJETIVOS

Generales

Aplicar a nuestro proceso de diseño el pensamiento sistémico basado en la modularidad.

Lograr una propuesta de diseño flexible y adaptativa a todos los formatos digitales de la actualidad y crear una herramienta de diseño haciendo uso de la codificación creativa.

Específicos

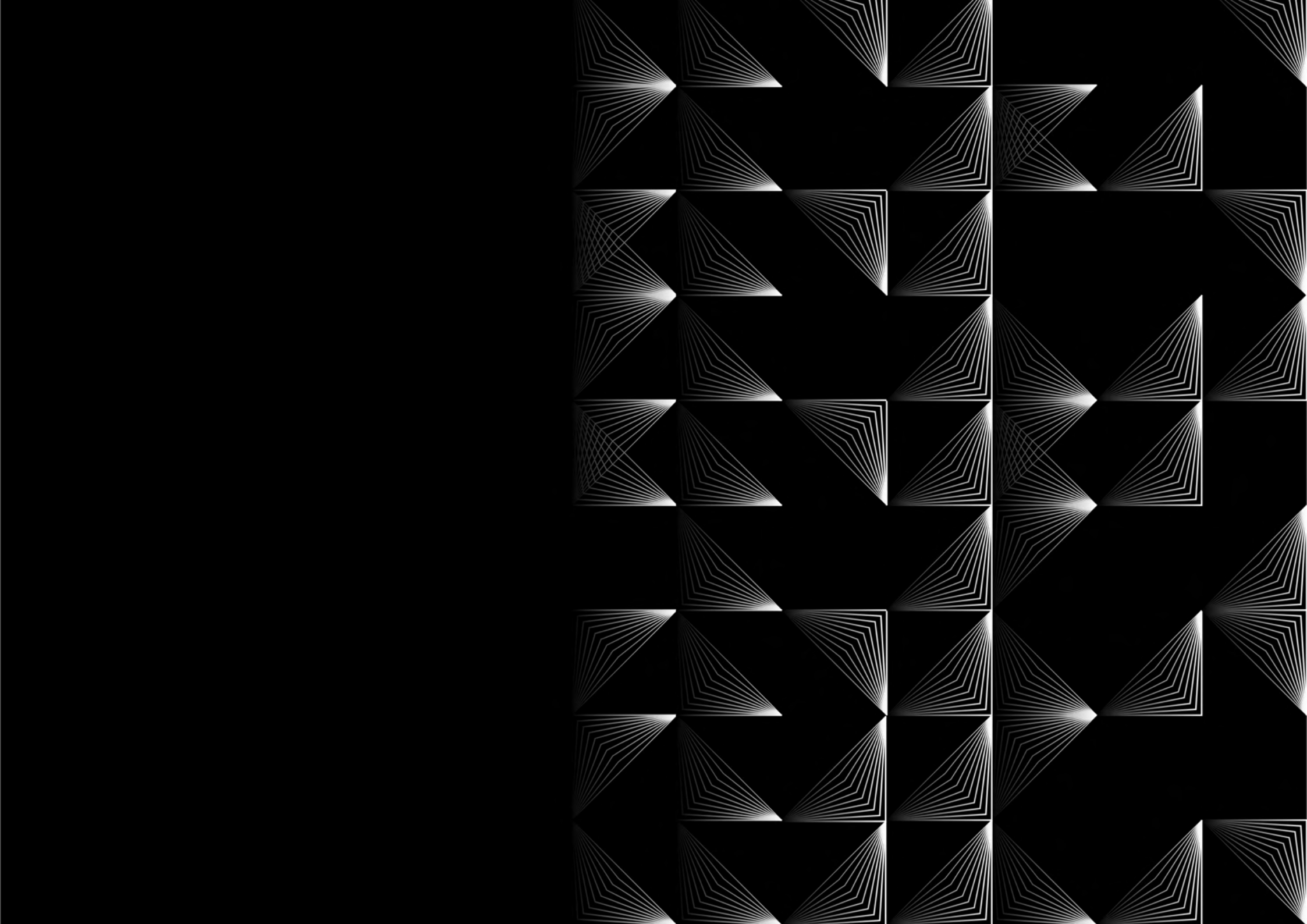
Diseñar de toda la identidad visual del I Festival de Diseño y Tecnologia de Canarias.

Crear contenido promocional flexible tanto en analógico como en formatos digitales.

Crear animaciones haciendo uso del código creativo.

Diseñar una herramienta de diseño haciendo uso del código creativo que nos permita demostrar la infinidad de posibilidades que ofrece el diseño de identidades flexible y modular.

Lograr que nuestro diseño sea interactivo haciendo uso del código creativo.



METODOLOGÍA

La metodología de trabajo seguida durante el desarrollo de este proyecto se estructura en las siguientes fases:

1. Investigación y anlálisis de los nuevos métodos de trabajo. Modularidad, identidades flexibles y código creativo.

2. Estudio de referentes en el sector.

3. Estudiar de que manera podríamos aplicar todos los contenidos previamente analizados

4. Primeras propuestas de identidad visual flexible mediante el estudio de módulos y creación de activos.

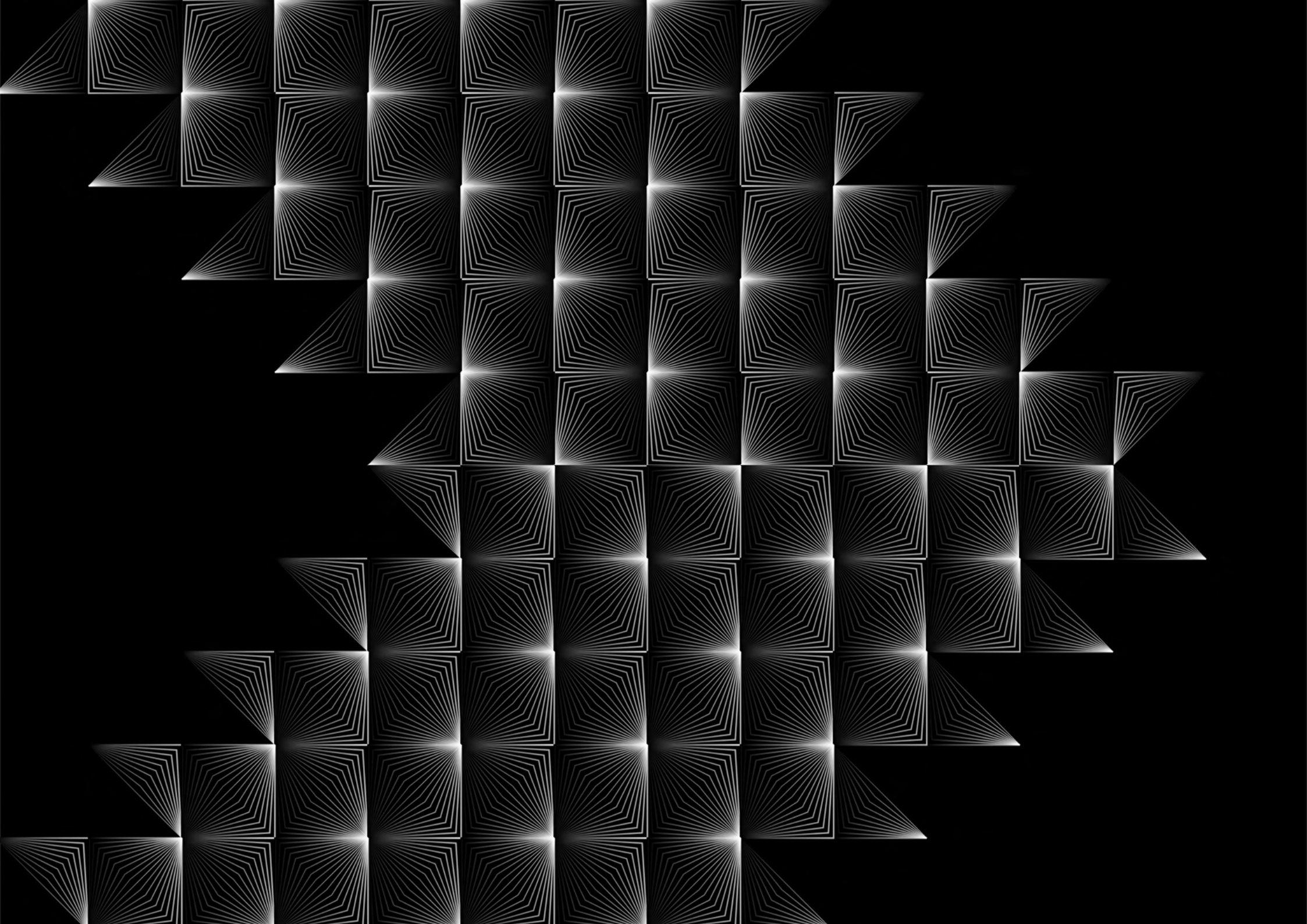
5. Propuesta final de módulo y creación de activos finales

6. Propuestas Analógicas y digitales de material promocional de la identidad.

7. Estudio del programa Processing

8. Primeras propuestas de código creativo con Processing (Animaciones)

9. Propuesta final de diseño interactivo con Processing



FASE DE INVESTIGACIÓN

1. MARCO TEÓRICO

1.1 Comienzos de la geometrización de las formas

El interés por la geometrización de las formas en el mundo gráfico, comenzó durante el Renacimiento, cuando artistas y matemáticos se dedicaron a aplicar el legado geométrico de la Grecia clásica a la pintura, la escultura, la arquitectura y además, al diseño tipográfico.

La invención y difusión de la imprenta convirtió el diseño de tipos en una necesidad donde la proporción y la geometría actuaron de guías. Artistas matemáticos como Luca Pacioli, Leonardo da Vinci o Alberto Durero incluyeron en sus obras descripciones precisas para el diseño de letras, siguiendo la vieja tradición geométrica griega de dibujar solo con regla y compás.
El matemático Luca Pacioli, se preocupó en 1509, durante la impresión de su tratado De divina proportione, ilustrado por Da Vinci, por las proporciones armónicas ideales entre la masa impresa de tinta y la superficie del papel en blanco, dando instrucciones precisas para la elaboración de un alfabeto, que denominó “alphabeto dignissimo antico”, donde usaba básicamente cuadrados y círculos.

A su vez, el maestro grabador Alberto Durero, dedicó en 1525 un capítulo de su famosa obra Los cuatro libros de la medida a cómo aplicar reglas geométricas al diseño del alfabeto latino y gótico. Durero insertaba sus letras en cuadrados en los que determinaba lo que consideraba sus proporciones armónicas.

Las instrucciones que dictaron Durero y Pacioli para dibujar sus letras latinas y góticas pueden considerarse el inicio de la tipografía matemática[1].

Figura 01. Alfabeto gótico de Alberto Durero, 1525.

1. Luque, B. (2017, julio). Tipografía digital | Investigación y Ciencia. Investigación y Ciencia. Recuperado12 marzo, 2022, de https://www.investigacionyciencia.es/revistas/investigacion-y-ciencia/el-origen-de-la-tecnologa-709/tipografa-digital-15399

Posterior a esto, fue durante el reinado del rey Luis XIV en Francia, que se continuó con esta matematización y geometrización de la letra, cuando en 1692 ordenó la creación de un comité de eruditos para el diseño de un nuevo tipo creado bajo principios científicos.

El resultado fue la familia tipográfica Romain du Roi (romano del rey) y fue diseñada para el uso exclusivo de la Imprimiere Royale (una de las imprentas más grandes e importantes de Francia que aún existe). Racionalista, con letras que mantienen un eje vertical perfecto y serifas horizontales simétricas, ignorando los modelos caligráficos en favor de principios analíticos y matemáticos[2].

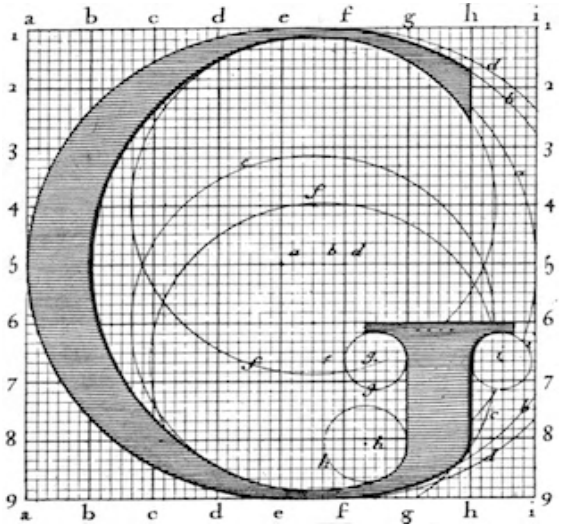


Figura 02. Romain Du Roi, 1692.

2. Britannica, T. Editors of Encyclopaedia (1998, julio). Romain du Roi. Encyclopedia Britannica. https://www.britannica.com/topic/Romain-du-Roi

Esta geometrización se volvió a dar ya en el siglo XX con el tipo de letra New Sans de Edward Johnston's Railway , que apareció por primera vez en un póster de julio de 1916. Inspirado en las proporciones de las letras romanas clásicas, basado en formas cuadradas y circulares.

Se ha utilizado, casi sin cambios en esencia, de forma continua y atemporal en señalización, carteles y publicidad durante casi un siglo. con que se propone reducir las formas a sus constituyentes esenciales. Esta tipografía fue la precursora de la Gill Sans (geométrica con rasgos humanistas 1928)[3].

Figura 03. Diseño tipográfico de Edward Johnston Railway, 1916.

3. Johnston's Underground Type. (2022, julio). Luc Devroye. Recuperado 24 abril, 2022, de http://luc.devroye.org/fonts-66504.html

Dos años más tarde, en 1919, Walter Gropius, crea la escuela de arquitectura, diseño, artesanía y fotografía, Bauhaus, la cual sentó las bases normativas y patrones de lo que hoy conocemos como diseño gráfico.

Desde su fundación, adoptó el principio funcionalista formulado por Louis Sullivan a finales del Siglo XIX “la forma sigue a la función”, y es que se abogaba por la simplicidad y funcionalidad de las formas, reduciendo al máximo cualquier tipo de adorno en busca de la geometrización.

La Bauhaus abrazó el nuevo estilo tipográfico formulado por autores como Jan Tishold y se emplearon casi exclusivamente las tipografías sans-serif y fue así como Josef Albers en 1923 o Herbert Bayer en 1925 , ambos profesores de la escuela, siguiendo el espíritu del minimalismo reductivo crearon las tipografías Kombinationsschrift y Universal respectivamente.

Por un lado, La Kombinationsschrift de Josef Albers, se basa en la combinación de un cuadrado, un círculo y un cuarto de círculo. Fue diseñada y producida en un momento en que los impresores hacían también composición.

Con el fin de imprimir una palabra tenían que componerla con las letras individuales. La Kombinationsschrift de Albers pretendía una mejora con la impresión con tipos móviles[4].

Figura 04. Kombinationsschrift, Josef Albers. Combinado de Metallglas-Aktiengesellschaft Offenburg Baden.

4. Moliz, A. (2016, octubre). TpStulle, la nueva tipografía de TwoPoints acompañada por una historia. Gráffica.info. Recuperado 24 abril , 2022, de https://graffica.info/tpstulle-tipografia-de-twopoints/

Por otro lado, Herbert Bayer, dos años más tarde, en 1925 diseñó su fuente tipográfica más conocida, la Universal, cuya versión digital recibe el nombre de Architype Bayer. Esta tipo pretendía, en inicio, usar sólo la geometría más pura, aunque después se fue flexibilizando para hacerla más funcional y aplicable[5].

Figura 05. UNIVERSAL TYPE / BREIT HALBFETT. Maqueta en tinta y gouache. Circa, 1925.

Siguiendo cronológicamente, en 1926, la tipografía Erbar-Grotesk por Jakob Erbar fue otro de los primeros sans serif geométricos, sirviendo de inspiración para la Futura de Paul Renner, lanzada un año después. Se trata de un tipo de letra sans-serif en estilo geométrico.

El objetivo del diseñador Jakob Erbar era diseñar un tipo de impresión que estuviera libre de todas las características individuales, poseyera formas de letras completamente legibles y fuera una creación puramente tipográfica.

Su conclusión fue que esto solo podía funcionar si la forma tipo se desarrollaba a partir de un elemento fundamental, el círculo.

Figura 06. Espécimen de la familia tipográfica Erbar.

5. Guayabero, Ó. (2019, enero). Herbert Bayer.ESDESIGN. Recuperado Abril 20, 2022, de https://www.esdesignbarcelona.com/actualidad/diseno-grafico/herbert-bayer

En 1927 Paul Renner presenta **Futura**, un tipo de letra sans-serif geométrico diseñado como una contribución al proyecto New Frankfurt. Se basa en formas geométricas, especialmente el círculo, similar en espíritu al estilo de diseño Bauhaus de la época

Aunque Renner no estaba asociado con la Bauhaus, compartía muchos de sus modismos y creía que un tipo de letra moderno debería expresar modelos modernos. El diseño de Renner se basó en formas geométricas simples: círculos casi perfectos, triángulos y cuadrados.

Figura 0.7 Espécimen de la familia tipográfica Futura.

Su versatilidad probablemente se vea favorecida por las diferentes fuentes y pesos de Futura, incluidos Futura Condensed, Demibold, Display, Black, Steile Futura y Futura Inline.

Lanzada por primera vez en 1929, la **Futura Black** es un diseño alternativo que utiliza formas de letras de estarcido que se inspira en la anterior mencionada Erbar grotesc (1926). Podemos diferenciar claramente el uso de módulos geométricos[6].

Figura 08. Futura black, Paul Renner, 1929.

6. Keung, L. (2020, agosto). All About the Futura Font and Its History. Design Tutsplus. Recuperado 26 abril, 2022, de https://design.tutsplus.com/articles/all-about-futura-font-and-its-history--cms-35382

Ese mismo año, 1929, AM Cassandre, maestro del diseño publicitario francés, creó la tipografía Bifur, patrocinada por Charles Peignot, director de la fundición tipográfica francesa Deberny & Peignot. Se trataba de una compleja mezcla de líneas gruesas y delgadas y barras transversales. Fue un shock para el mundo tipográfico.

El exceso de Bifur puede funcionar bien tanto en el diseño maximalista como en el Art Deco. Es generalmente reconocible como un ícono de su tiempo y en términos de tipografía, desafía las convenciones[7].

De nuevo, la geometría y la creación modular están presentes, pero esta vez de una forma particular.

Figura 09. Bifur , AJM Cassandre, 1929

7. Heller, S. (2019, marzo). Cassandre's Most Eclectic Typeface – PRINT Magazine. PRINT Magazine. Recuperado 3 mayo, 2022, de https://www.printmag.com/daily-heller/bifur-cassandre-typeface/

En 1930 se publica Patrona Grotesk, una ingeniosa contribución checa al proyecto modernista temprano en el que el alfabeto se concebía como un sistema de partes que podían organizarse como componentes variables en diseños regulados por la geometría.

Fue diseñado por V. Kánský alrededor de 1931. Patrona Grotesk se basó en el desglose de formas de letras en secciones inconexas que podían organizarse para construir una gama completa de formas de letras similares a plantillas en varios idiomas, así como una amplia gama de bordes y patrones novedosos[8].

Figura 10. Patrona. V Kansky,1931.

Otra interpretación de construcción tipográfica como sistema de partes podría ser la Fregio Mecano ("adorno mecánico"). Es un conjunto de 20 formas geométricas para la construcción de letras e imágenes. Se fabricó en Italia probablemente a mediados de la década de 1930. Se desconoce el diseñador.

Figura 11. Fregio Mecano, 1930.

8. Nawrot, K., & Peško, R. (2017, septiembre). The Visual History of Type author Paul McNeil selects and dissects his six favourite faces. It's Nice That. Recuperado 4 mayo, 2022, de https://www.itsnicethat.com/news/the-visual-history-of-type-paul-mcneil-graphic-design-publication-110917

Con los últimos ecos de la Bauhaus y los movimientos de vanguardia, se establecieron finalmente en Europa las tipografías sans-serif, la racionalización del diseño, las composiciones simétricas y las estructuras formales basadas en la geometría.

Mientras tanto, en España, Esteban Trochut, que dirige una famosa imprenta en Barcelona, y su hijo Joan editaron ADAM (Archivo de Documentación de Arte Moderno) , una publicación cuyo propósito era proponer la tipografía como elementos creativos de la imprenta.

Tras la Guerra Civil, retomaron sus actividades con una nueva versión, esta vez denominada NOVADAM.

En el segundo número, publicado en 1942, se presentaba **Super Fast Type,** un sistema tipográfico ideado por Joan Trochut, compuesto por tres juegos modulares y tres complementos adicionales, con posibilidades de combinación casi ilimitadas, que permitían no sólo la creación de letras, pero también dibujos o ilustraciones[9].

Figura 12. Páginas de los especímenes compuestos por Trochut Blanchart para promocionar su tipografía.

9. Gamonal, R., Gamonal, R., Balius, A., Trochut, A., & Gil, E. (2017, marzo). Super Tipo Veloz: la tipografía supercalifragilística (I). Pioneros Gráficos. Recuperado 17 mayo, 2022, de https://pionerosgraficos.com/la-tipografia-supercalifragilistica/

Las innovaciones científicas de los años sesenta tuvieron un reflejo inmediato sobre el mundo gráfico. Los avances tecnológicos siempre han afectado a la comunicación visual, sin embargo, lo que provocó realmente un antes y un después fue la aparición de los ordenadores, que cambiaron todo hasta nuestros días.

En la infancia de la tipografía digital, en 1967 Willem Hendrik "Wim" Crouwel, diseñador gráfico y tipógrafo holandés, diseñó el tipo de letra New Alphabet, un diseño que abarca las limitaciones de la tecnología de tubo de rayos catódicos utilizada por las primeras pantallas de visualización de datos y equipos de fotocomposición, por lo que solo contiene trazos horizontales y verticales[10].

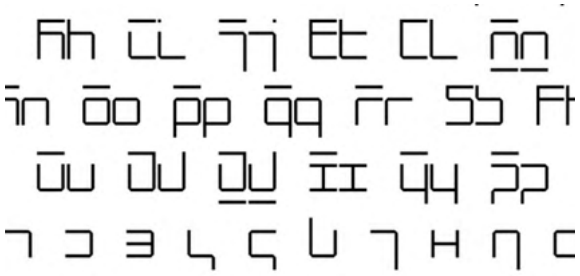


Figura 13. New Alphabet. Wim Crouwel, 1967.

De nuevo, el uso de módulos está presente en la creación tipográfica incluso cuando la composición con tipos móviles se deja atrás.

En 1969, dos años más tarde, en el Laboratorio Nacional de Física del Reino Unido, siguiendo el ejemplo de New Alphet, Timothy Epps y Christopher Evans crearon su propio alfabeto en respuesta al desafío de superar la ilegibilidad humana de los tipos de letra legibles por máquinas.

Figura 14. Chris Evans and Timothy Epps, 1969.

10. Wim Crouwel. New Alphabet. 1967. (n.d.). MoMA. Recuperado 17 mayo, 2022, from https://www.moma.org/collection/works/139322

La venta de los primeros Apple Macintosh, en 1982, mostraron un sistema operativo en los que se podían escoger diferentes tipos de letra para su aplicación en hojas de texto.

A partir de aquel momento, los ordenadores animaron a los diseñadores a pedir más y más tipos de letras para cubrir sus aspiraciones creativas y los nuevos programas de diseño profesionalizaron la tipografía digital.

Es así como de nuevo la construcción modular continúa en el plano del diseño tipográfico, de la mano de Zuzana Licko con la familia de fuentes Lo-Res, una tipografía de tipo pixel.

La familia Lo-Res consiste en una síntesis de diseños pixelados, que reemplaza las familias Emigre, Emperor, Oakland y Universal preexistentes y agrupa estos diseños de mapas de bits relacionados en uno[11].

Como vemos, la tipografía con base geométrica-modular es una opción para el diseñador de fuentes y que sus posibilidades son infinitas.

Figura 15. Lo-Res. Zuzana Licko, 1985/2001.

11. Licko, Z. (n.d.). Emigre: Lo-Res Font Family. Emigre: Fonts. Recuperado 22 abril, 2022, de https://www.emigre.com/Fonts/Lo-Res

1.2 Modularidad como tendencia estilística y metodológica

Es a partir del S.XXI que la modularidad da un salto del ámbito fundamentalmente tipográfico a ser casi un principio básico a tener en cuenta a la hora de abordar cualquier proyecto de diseño.

Fue a finales del siglo XX , el diseñador Jurriaan Schrofer, que mostró la funcionalidad de la modularidad más allá del diseño tipográfico, diseñando una serie de patrones e incluso identidades, basadas en la geometrización y la modularidad.

Hizo unas serigrafías donde el patrón se basa en elementos simplificados geométricamente que podrían combinarse para hacer glifos en una escritura de alfabeto latino.

Planeó todas las permutaciones posibles de los elementos en un mosaico, haciendo un diagrama de cada permutación y dibujando todo ello manualmente[12].

Esto nos da un adelanto de cómo lo que antiguamente costaba una infinidad en realizarse, en la actualidad con tan solo un click lo tendríamos.

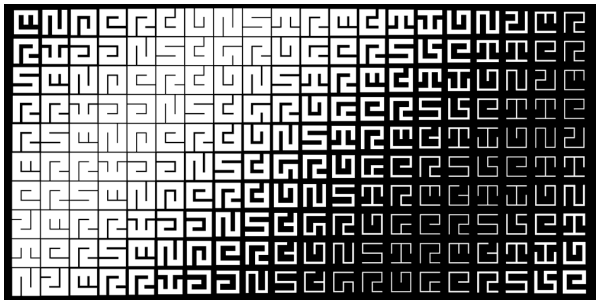


Figura 16. Imagen de la biografía de Schrofer de Frederike Huygen, 2012/2013.

12. Meilleur, M. (n.d.). schrofer_screenprint — mauricemeilleur \ design. MauriceMeilleur. Recuperado 2 mayo, 2022, de https://mauricemeilleur.net/schrofer_screenprint

Otro ejemplo del salto de la modularidad del diseño tipográfico a otros ámbitos del diseño nos los ofrece Hamish Muir, cofundador del estudio de diseño gráfico 8vo con sede en Londres (1985–2001) y coeditor de Octavo, revista de tipografía (1986–92).

Las actividades de Muir McNeil se centran en la exploración de métodos sistemáticos y algorítmicos en el diseño tipográfico, el diseño gráfico y la imagen en movimiento, basados en la modularidad.

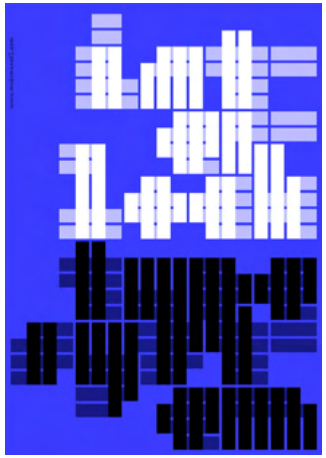


Figura 17. Póster para la semana del diseño de San Francisco 2019 'CommUNITY'

En los últimos 15 años, ha habido una eclosión del diseño tipográfico en general y modular en particular.

La modularidad ha pasado al ámbito sistemático y algorítmico, lo cual nos ha abierto las puertas a un mundo que nos ofrece infinitas posibilidades de creación, descubriéndonos el diseño de sistemas flexibles y adaptativos a cualquier formato.

Exploraremos esto mismo definiendo antes tres conceptos relevantes para nuestro proyecto.

1.3 Modularidad, sistemas flexibles y código creativo

Modularidad y sistemas flexibles

La definición de modular, según el diccionario, es construido con unidades o dimensiones estandarizadas para permitir flexibilidad y variedad de uso.

En el mundo del diseño, se trata básicamente de una manera de construir cosas a partir de partes independientes con interfaces estándar que se pueden apilar, reorganizar, personalizar, reutilizar, etc. El objetivo final es poder cambiar o ajustar elementos singulares sin reemplazar todo el sistema, principalmente al diseñar una identidad visual.

Las redes sociales han hecho que la co municación sea más compleja. No solo crearon nuevos canales de comunicación, sino también nuevas formas de comunicarse. Si bien los destinatarios de las comunicaciones solían ser pasivos, ahora son parte activa del proceso. Las audiencias no solo comparten, califican y comentan, sino que también influyen en otras audiencias.

Cualquier cosa que deba comunicar algo, necesita un lenguaje visual consistente.

Hubo un tiempo en el que el diseño de identidades visuales era sinónimo de diseño de logos, y en alguna ocasión aún puede serlo, hasta cierto punto.

Hoy en día, una identidad visual basada en un logo, no es adecuada para comunicar de manera efectiva. No se adapta al entorno, no es capaz de formular el mensaje óptimo. Es por esto, que necesitamos identidades visuales capaces de resolver problemas comunicativos contemporáneos[13].

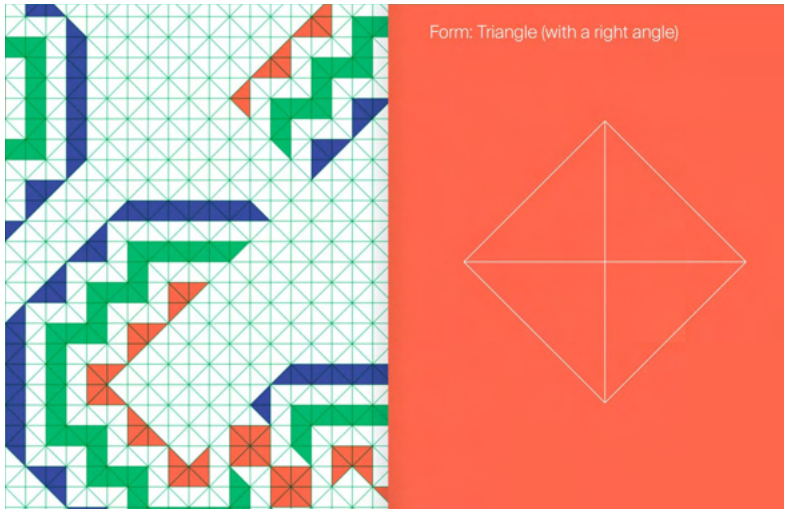
Una identidad visual con base sistémica, es capaz de articular diferentes mensa

13. Lorenz, M. (2021). Flexible Visual Systems: The Design Manual for Contemporary Visual Identities (J. Kahl, L. Harmsen, & Slanted, Eds.). Karlsruhe: Slanted Publishers.

jes de diferentes maneras a diferentes audiencias en diferentes circunstancias. Por el contrario, una identidad basada en un logo, repetirá el mismo mensaje una y otra vez.

Estas identidades con base sistémica son en realidad sistemas visuales flexibles, cuya principal función es hacer que la institución, evento, producto, corporación.. sea identificativa, pasando de lo estático a lo flexible, y de lo flexible a lo multisensorial y multidimensional.

La modularidad es una forma de construir un sistema flexible, pero hay muchas más. Los sistemas basados en formas geométricas están en auge, porque reducir las posibilidades de diseño agiliza el proceso de diseño, simplifica la estética, y hace que los elementos individuales sean más fáciles de reconocer y por tanto, aumenta la identificabilidad de la identidad visual. Incluso cuando estos elementos individuales son reubi-



Figuras 18/19. Flexible visual systems. The design manual for contemporary visual identities. Martin Lorenz, 2021.

cados, el sistema permanece estable.

El término “programa” fue introducido por primera vez en el mundo del diseño por el diseñador gráfico suizo Karls Gerstner. Ya en 1950 Gernster rechazó las identidades visuales basadas en logos, y optó por sistemas visuales flexibles, a lo que él se refería como “programas”.

Desde la perspectiva actual se podría pensar que Gerstner se referiría a programas informáticos, aunque vendrían décadas después, pero la definición de programa de Gerstner no se limitaba a una herramienta concreta, ni a la época.

Gerstner dijo en una entrevista con Ulrike Felsing en 2007 “Un día me di cuenta de que no tiene sentido diseñar sellos y luego colocarlos en algún lugar. El diseño en sí debe tomar el lugar del logotipo”
Las identidades visuales que Gerstner diseñó para Boîte à Musique, Blech Electronic Centre y Holzäpfel ilustran esta declaración. Son reconocibles fácilmente sin necesidad de un logo. La ausencia de logo parece hacerlas incluso más flexibles y consistentes[14].

Los logotipos flexibles fueron una característica común en el diseño de identidades visuales en los 2000, sin embargo sólo fueron una fase transicional entre los sistemas visuales basados en logos y los sistemas visuales basados en sistemas flexibles.

Poco a poco nuestra mentalidad va cambiando, y sobre todo por la digitalización de la comunicación. Difícilmente una identidad visual puede evitar ser aplicada a los diferentes canales de comunicación digital.

Nos acostumbramos a establecer reglas de diseño flexibles que funcionan igual de bien en diferentes formatos y dispositivos. ¿Cómo se ve nuestro diseño cuan-

14. Lorenz, M. (2021). Flexible Visual Systems: The Design Manual for Contemporary Visual Identities (J. Kahl, L. Harmsen, & Slanted, Eds.). Karlsruhe: Slanted Publishers.

do la ventana del navegador se hace más pequeña o más grande, el sitio web se ve en un ordenador de escritorio, tableta o teléfono inteligente, cuando el dispositivo se sostiene horizontal o verticalmente o qué usuario navega a través de nuestro sitio web y cómo?

En este entorno flexible, el logotipo como forma estática e inmutable se convierte puramente en una imagen de perfil que desaparece con el primer scroll o swipe[15].

Modularidad y código creativo

A su vez, la modularidad es esencial para el desarrollo de software. Sin él, los grandes sistemas de software simplemente no podrían realizarse. Los diseñadores generalmente se esfuerzan por lograr un alto grado de modularidad separando diferentes preocupaciones sobre diferentes módulos, un proceso llamado modularización.

La artista parisina Vera Molnar, hacia 1960, fue una de las primeras personas en el mundo en utilizar las primeras versiones de la computadora como herramienta artística. Hoy se la considera una pionera del arte computarizado y la codificación creativa.

La mayoría de las definiciones acerca de la codificación creativa fallan porque intentan explicar el término sobre la base de un medio. La entrada en Wikipedia es un buen ejemplo, porque enumera directamente ejemplos de uso, lo que estrecha mucho la vista del lector y, por lo tanto, solo permite una visión muy limitada de la tecnología en sí.

Mark C. Mitchell y Oliver Bown (2013) la describen de una manera más acertada. Según ellos “se trata de un proceso basado en la exploración, la iteración, la reflexión y el descubrimiento, donde el código se utiliza como medio principal para crear una amplia gama de artefactos de

15. Lorenz, M. (2021). Flexible Visual Systems: The Design Manual for Contemporary Visual Identities (J. Kahl, L. Harmsen, & Slanted, Eds.). Karlsruhe: Slanted Publishers.

medios.”
La codificación creativa y pensamiento computacional en los ámbitos del diseño gráfico nos ofrecen una amplísima gama de oportunidades de creación de sistemas basados en lenguajes inteligentes, que a su vez son capaces de generar sistemas visuales flexibles en cuestión de segundos.

En cierto sentido, la codificación creativa es un método que permite a los diseñadores formular sus propios pensamientos e ideas de una forma nueva. Así, paso a paso, se les abre la entrada a un nuevo mundo del diseño sin un medio previamente definido.

Según el lenguaje de programación utilizado, la gama de resultados posibles varía: algunas herramientas son adecuadas para desarrollar aplicaciones web interactivas, otras lo son más para representar imágenes en movimiento y otras sirven para aplicaciones de realidad virtual. Un único lenguaje de programación permite animaciones ³D e ilustraciones imprimibles, así como divertidas interacciones en la web.[16].

Eso es exactamente lo que hace que la codificación creativa sea tan especial y tan esquiva: es un enfoque que no está limitado por un solo medio. Aquí es donde radica el enorme potencial: una sola herramienta (lenguaje de programación) se puede utilizar para una amplia gama de aplicaciones.

Como escribió TimRodenbröeker (2022) en su blog sobre codificación creativa, “el diseño inductivo e involuntario se subestima enormemente y el paradigma del diseño como una herramienta para resolver los problemas existentes es un obstáculo enorme para los jóvenes diseñadores que quieren aprender a codificar creativamente. Desafiar esta suposición básica es ciertamente tedioso, ¡pero definitivamente vale la pena!”

16. What is Creative Coding? · tim rodenbröker creative coding. (2022, junio). Tim Rodenbröker. Recuperado 4 mayo, 2022, de https://timrodenbroeker.de/what-is-creative-coding/

¿Qué ocurre cuando unimos la codificación creativa y la modularidad para diseñar un sistema visual flexible?

Lo que obtenemos es una forma de trabajo específica que nos facilita todo el proceso de planificación de nuestro proyecto y además nos ofrece, de nuevo, una infinidad de posibilidades y resultados, donde no solo la imagen final es la que cuenta, si no todo el proceso creativo hasta llegar al resultado final, que forma parte de este sistema flexible y es esto precisamente, lo que lo convertirá en identificativo.

2. TRABAJO DE CAMPO

2.1 Referentes en el sector

¿Para que se están usando los sistemas flexibles, la modularidad y el código cretivo, por qué y por quién?

Combinado el código creativo y la modularidad, está Vanesa Zúñiga Tinizaray, más conocida como "AMUKI" . Es una diseñadora nacida en Ecuador, especializada en tipografía modular, diseño de patrones y tipografías en movimiento. Investiga los signos visuales de las culturas originarias de latinoamérica para compartir estos saberes ancestrales a través del diseño, proponiendo interpretaciones visuales en constante evolución.

Trabaja con la herramienta Processing, un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, el mismo con el que desarrollaremos parte de este proyecto.

Los patrones Tinkuy, es uno de sus trabajos que más me han inspirado. El sistema Tinkuy Patterns se divide en seis archivos que contienen un total de más de 2650 módulos que se pueden combinar entre sí creando un abanico infinito de posibilidades. A partir de estos patrones, creó un alfabeto. Estos patrones y alfabeto están elaborados primero en adobe illustrator, y posteriormente combinados en Processing.

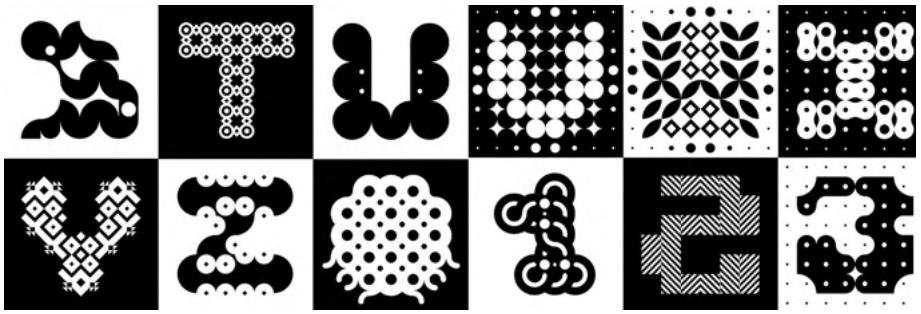


Figura 20. Tinkuy Patterns, MODULAR TYPOGRAPHY VOL.2, AMUKI, 2022.

Por otro lado, el trabajo de **Tim Rodenbröker** es realmente fascinante. Tim es tecnólogo creativo y educador independiente en Alemania. Es el fundador de trcc, una plataforma de aprendizaje en línea con una comunidad global asociada para la codificación creativa.

Su proyecto „Poster 2.0“ es otra funete de inspiración para este proyecto, y surge mientras prepara un curso de codificación creativa para la facultad de diseño de Rhine-Waal, donde nuevamente fusiona el diseño gráfico con la tecnología.

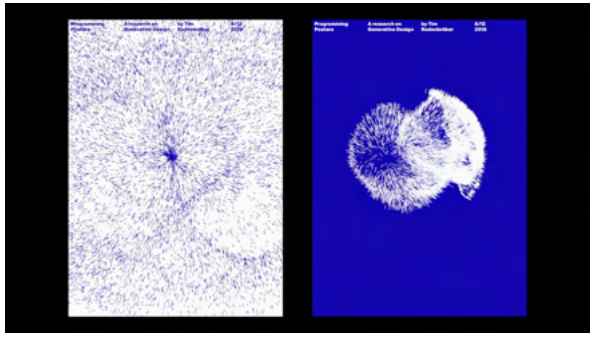


Figura 21. Programming Posters, Tim Rodenbröker, 2018.

Al igual que AMUKI, su herramienta de trabajo es Processing. Según Tim, en comparación con otras metodologías basadas en código, Processing se adapta perfectamente a las necesidades de los diseñadores gráficos y principiantes a través de comandos intuitivos, una amplia gama de tutoriales, una muy buena referencia y una gran comunidad.

Tim decidió restringir a sus alumnos a

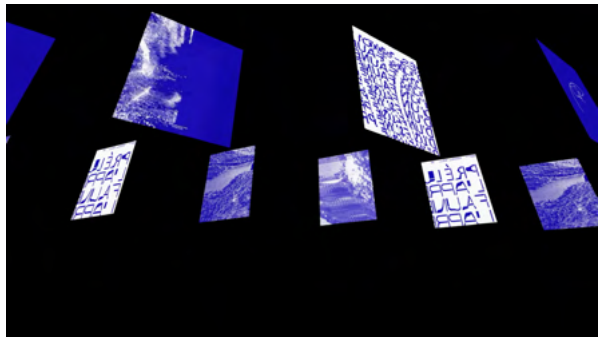


Figura 22. Programming Posters, Tim Rodenbröker, 2018

que trabajaran exclusivamente con dos colores y una familia de fuentes definida, lo cual aseguró su máximo enfoque en el lugar correcto: las posibilidades creativas de la codificación.

Para visualizar las restricciones del proyecto, hizo uso de una técnica de creatividad llamada “El Triángulo Mágico”: cada borde de la forma representa una regla o restricción. Los tres bordes juntos construyen un marco alrededor del espacio creativo, que es literalmente el área donde existen las ideas.

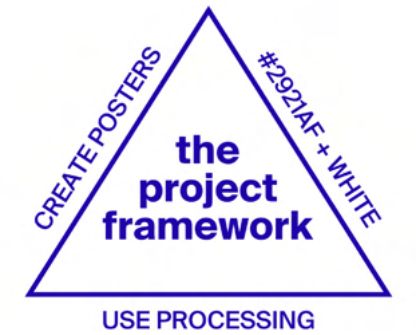


Figura 23. “The Magic Triangle”, Tim Rodenbröker, 2018.

El resultado, es más que un póster con una superficie con texto, colores e imágenes: es una aplicación interactiva, animada, audible, basada en datos e intermediaria. Involucra todos los sentidos y

disciplinas.
Martin Lorenz, a su vez, autor del libro Flexible visual systems (The design manual for contemporary visual identities) es otro referente en el desarrollo de este proyecto.

Lorenz es diseñador actualmente en TwoPoints.Net con Lupi Asensio y además profesor en la ELISAVA, Barcelona.

TwoPoints.Net es un estudio de diseño especializado en sistemas flexibles para identidades visuales y proyectos editoriales.

Entre sus clientes se encuentran museos como el Museo Picasso y el CCCB, instituciones como Helsinki Design Lab, Nesta y Barcelona Cultura, universidades como RISD, MIT y Harvard, editoriales como Actar, Gestalten, Viction:ary, Phaidon y Routledge, empresas como NIKE, Bershka e IBM, y revistas como ESPN, ICON y Quaderns.

Su proyecto “Center for Complexity” es uno de sus proyectos que de nuevo ha inspirado a este trabajo de final de grado, donde mezclan modularidad y código creativo para crear un sistema visual flexible.

El concepto visual es muy simple. Diseñan una fuente cuyos módulos se pueden llenar con una mezcla heterogénea de imágenes complejas. Las imágenes visualizan la perspectiva holística sobre un tema y la fuente es la herramienta

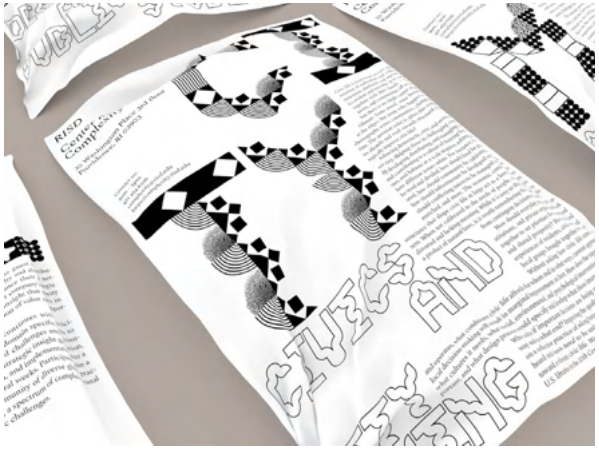


Figura 24. Center for Complexity, TwoPoints.net, 2020.

para construir la comunicación.
Por otro lado están Nigel Cottier y Janis Maroscheck, ambos centrados en la creación y variación tipográfica a partir de lenguajes informáticos.

Nigel cottier es el autor del libro Letterform Variations, publicado por Slanted, que se trata de un estudio lúdico sobre la construcción de formas de letras utilizando sistemas básicos basados en cuadrículas y formas, y su potencial para generar grandes cantidades de resultados

Figura 25. Letterform Variations, Nigel Cottier, 2021.

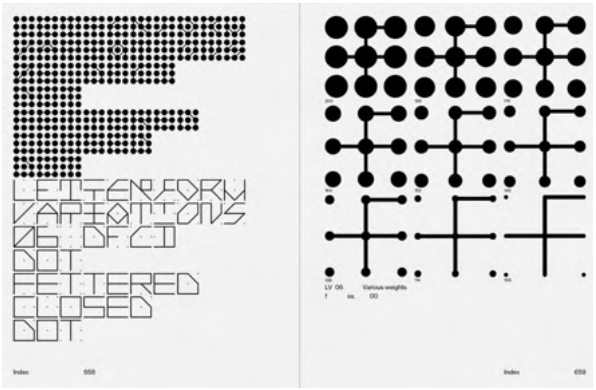


Figura 26. Letterform Variations, Nigel Cottier, 2021.

alfabéticos variables.
Janis Maroscheck, por otro lado, se centra en la síntesis de las formas e imágenes.

Es autor del libro Shape grammars, publicado también por slanted, estudia cómo producir en serie piezas únicas, o cómo un ordenador puede hacerse cargo y apoyar el trabajo creativo.

Jannis, en este caso, ha diseñado y programado sus propios sistemas de producción que pueden dibujar un número ilimitado de formas gráficas individuales. El resultado es un catálogo sistemático, una especie de diccionario de formas, para navegar y explorar sistemas geométricos, en el que siempre se puede descubrir algo nuevo.

Shape Grammars pretende ser un manual de diseño que, además de 150.000 formas generadas, muestra algunos potenciales y limitaciones del diseño gene-

Figura 27. Shape Grammars, Jannis Maroschek, 2020.

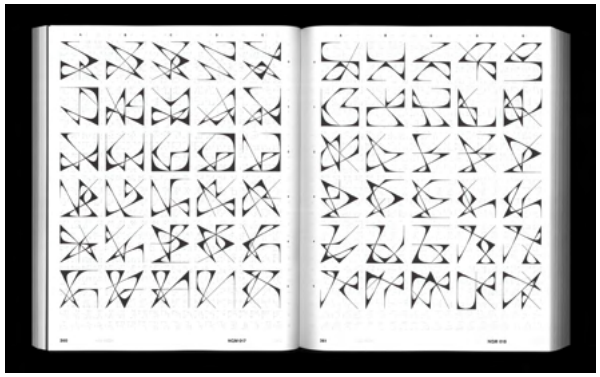


Figura 28. Shape Grammars, Jannis Maroschek, 2020.

2.2 Entrevistas

2.21 En conversación con Tim Rodenbröker. Codificación creativa.

Entrevista traducida del inglés por la autora.

El formato de la entrevista fue a través de unas preguntas ya formuladas que se enviaron via Gmail esperando su respuesta. Previamente se acordó hacerlo de esta manera debido al poco tiempo del que disponía Tim, y por la facilidad y comodidad que era poder responder a las preguntas cuando él encontrara su momento y poderles dedicar el tiempo necesario sin ningú tipo de presión.

En un principio se le preguntó por su bagaje académico y si tenía algún tipo de relación con el código creativo, a lo que nos respondió que no, que no tenía experiencia en este campo, al menos cuando empezó. Estudió diseño de comunicación y luego se involucró en el tema por puro interés, llegando a realizar un máster sobre ello, pero mucho más adelante.

Según él, su curiosidad y la necesidad por crear cosas fue lo que le dirigió directamente al código creativo, ya que le permite crear una infinidad de cosas con una sola herramienta. Con el tiempo descubrió que le gustaría centrarse en este tema de manera específica y ahí fue cuando comenzó a experimentar.

Nos comentó que definitivamente el código creativo le ha ayudado a desarrollar sus habilidades como diseñador, pues en el mundo de hoy, todo se mueve infinitamente rápido y se necesita mucha flexibilidad en el pensamiento para seguir siendo capaces de actuar. Aprender a pensar es de lo que se trata.

Además, hablamos de las posibiliddes del código creativo. Son muchas las cosas que te permite realizar la codificación creativa que ni siquiera son posibles con el software de Adobe. Dos ejemplos: la

automatización del diseño, que es un tema muy amplio, o las herramientas de diseño personalizadas para proyectos específicos.

Se habló se los pros y los contras del diseño a partir de código creativo. Según Tim, la principal supuesta desventaja de la codificación creativa es el hecho de que lleva mucho tiempo y esfuerzo entrar en ella. Se necesita bastante tiempo para llegar al punto en el que pueda usarlo de manera efectiva. Las herramientas de diseño tradicionales son mucho más fáciles de aprender y, por lo tanto, son tentadoras para los jóvenes que buscan su camino en el mundo del diseño. Al mismo tiempo, existen muchos prejuicios y miedos asociados a la programación. Sin embargo, vale la pena entrar en él, porque hay mucho que aprender y descubrir.

También nos confirmó que en el futuro, el diseño gráfico y la tecnolgía irán unidos de la mano, y que según John Maeda, nada ha tenido tanta influencia en el diseño como el desarrollo de la computadora. La influencia de la tecnología en el diseño probablemente no se puede sobreestimar.

A nivel personal, esta entrevista fue un completo descubrimiento y definitivamente motivante. Como futura diseñadora, nos anima a involucrarnos en este mundo sin ningún miedo al fallo o al error, porque es precisamente ahí donde está el valor de todo el proceso de creación.

2.22 En conversación con Martin Lorenz. Diseño de Sistemas Visuales flexibles.

Entrevista traducida del inglés por la autora.

Esta entrevista se realizó en el mismo formato que la anterior, por las mismas razones. La poca disponibilidad de Martin y la facilidad que le suponía realizarla de manera escrita y online.

Con Martin se también se comenzó de su relación académica con el código creativo, a lo cual nos respondió que no había ninguna relación, sin embargo, tuvo un maestro que trató de explicarle un poco de Python en su día. Peter van Blokland. Según él, si querías aprender más, tenías que hacerlo en tu tiempo libre.

Se le preguntó cómo había descubierto esta nueva forma de diseño, como ajuste a los tiempos tecnológicos o por necesidad. Nos respondió que él no es tan racional. Está constantemente buscando cosas en las que es malo y trata de aprenderlas. Aprender es lo que más disfruta en la vida. Todas las nuevas herramientas que surgen dan forma al diseño actual. Aprender las nuevas herramientas nos hace pensar con las manos.

Lo que descubrió fue que nos dirigíamos hacia un enfoque sistémico. No solo en diseño. El cambio de perspectiva estaba ocurriendo en todas partes y en todas las disciplinas. Vemos y hacemos todo en relación con otra cosa. Para resolver un problema, debe ver cómo se ve afectado y tiene un efecto en otros sistemas.

Martin cree que las ventajas de trabajar diseñando sistemas visuales flexibles son la eficiencia, porque la aplicación se puede automatizar, la durabilidad, porque los componentes se pueden optimizar sin tener que revisar todo el sistema, el fomento del trabajo en equipo, ya que se basa en reglas objetivamente comprensibles.

Pensar en sistemas crea un sentido de responsabilidad, ya que su trabajo afecta tanto al sistema como a todos y todo lo involucrado. Pensar en sistemas lleva a la oportunidad de criticar los sistemas porque te das cuenta de que el sistema puede ser el problema, no sus componentes. No trabajar ni pensar en sistemas no se corresponde ni con nuestras redes de comunicación de hoy ni con los comportamientos comunicativos contemporáneos.

Hablamos de la modularidad y sus formas de construir sistemas flexibles. De su gran eficacia porque son fáciles de entender y aplicar.

En su caso, además de crear sistemas visuales flexibles con código creativo, también los diseña sin él. Con papel y fotografía. Según Martin, hace uso del código siempre que lo necesita, pero hay cosas que no puede hacer con código y no quiere estar limitado a una sola herramienta.

“Si solo tienes un martillo, todo será un clavo para ti. Quiero tener diferentes soluciones para diferentes problemas.”

Esta última reflexión es lo que resalto de esta entrevista. Como diseñador, debemos ser capaces de estar a la altura de la contemporaneidad y tener suficientes posibilidades de resolución a la hora de afrontar un problema de diseño.

2.3 Processing

¿qué es? ¿cómo funciona? ¿posibilidades?

Ben Fry y Casey Reas comenzaron Processing en la primavera de 2001 y continúan trabajando obsesivamente en él. En 2012, comenzaron Processing Foundation junto con Dan Shiffman, quien se unió formalmente como tercer líder del proyecto.

Processing es un cuaderno de bocetos de software flexible y un lenguaje para aprender a codificar. Desde sus inicios ha promovido la alfabetización de software dentro de las artes visuales y la alfabetización visual dentro de la tecnología. Hay decenas de miles de estudiantes, artistas, diseñadores, investigadores y aficionados que utilizan Processing para aprender y crear prototipos.

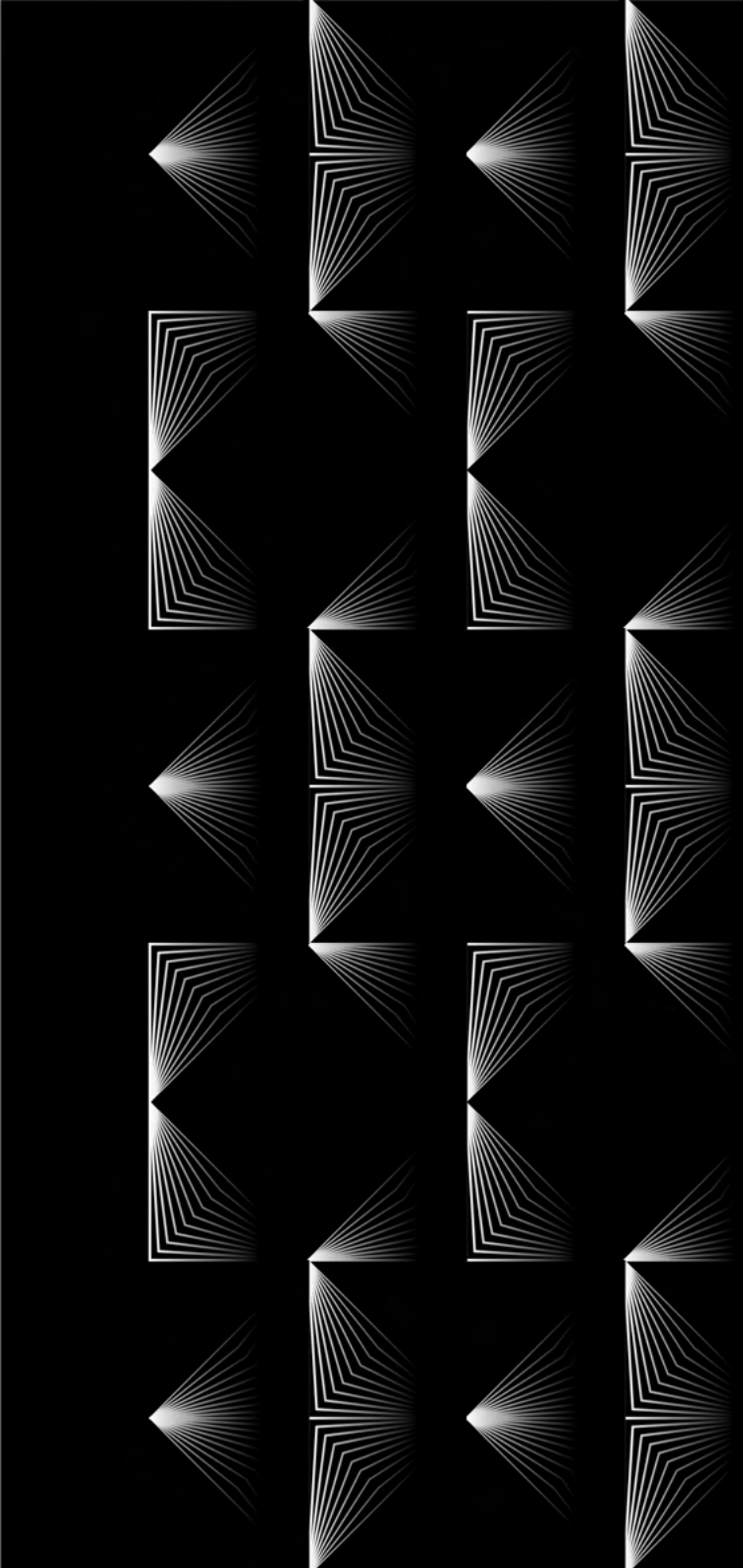
Sigue siendo una alternativa a las herramientas de software propietario con licencias costosas y restrictivas, lo que lo hace accesible para escuelas y estudiantes individuales. Su estado gratuito, libre y de código abierto fomenta la participación y colaboración de la comunidad que es vital para su crecimiento.

Los colaboradores comparten programas, contribuyen con código y crean bibliotecas, herramientas y modos para ampliar las posibilidades del software. La comunidad de Processing ha escrito más de cien bibliotecas para visión artificial, visualización de datos, composición musical, redes, exportación de archivos 3D y electrónica de programación.

Desde el principio, Processing fue diseñado para ser lo más simple posible para los principiantes, sabiendo que su simplicidad también beneficiaría a los usuarios más experimentados. Se inspiró en la inmediatez de lenguajes anteriores como BASIC y Logo.

Con su enfoque en la creación de medios visuales e interactivos, los estudiantes nuevos en la programación encuentran satisfactorio hacer que algo aparezca en su pantalla al momento de usar el software.

Este plan de estudios motivador ha demostrado ser exitoso para guiar a los estudiantes de diseño, arte y arquitectura hacia la programación y para involucrar al cuerpo estudiantil en general en las clases generales de informática.



FASE DE IDEACIÓN

1. DEFINICIÓN DE PROPUESTA DE PROYECTO

1.1 Planteamiento de la idea.

La necesidad de divulgar sobre tecnología en el ámbito de las artes y el diseño urge cada vez más, y es justamente esto lo que se pretende con este TFG.

Es muy bajo el número de jornadas, eventos, festivales que la Universidad de La Laguna ha acogido en torno a este tema, y es por esto que la idea de este proyecto es que la ULL (cliente ficticio) nos encarga el diseño toda la identidad visual del primer festival creado por la Facultad de Bellas artes en torno al Diseño y la Tecnología. “VISUAL BEINGS 2023”.

1.2 Análisis de antecedentes

En la isla de Tenerife desde 2010 se han realizado jornadas, eventos, charlas o reuniones de divulgación tecnológica hasta la actualidad.

Algunas son el primer Encuentro sobre Innovación y Sociedad del Conocimiento: Tecnológica Santa Cruz (2010), las Jornadas sobre Tecnología y Nuevas Emergencias (2012), las jornadas Ciencia y Tecnología REDiri"s (2015) o la Feria de la ciencia (2017) en la Orotava.

Más actuales destacan la Jornada Laboratorio de Digitalización (2021) o la I Jornada de Innovación Empresarial en Canarias (2022).

La implantación de la tecnología en el ámbito empresarial se ha convertido en un tema candente en la isla, sin embargo, muy poco se habla de la influencia de la tecnología en el ámbito creativo.

En 2022 surge Technarte, un proyecto conjunto de Technarte, TEA e ITER, para el desarrollo de una residencia artística en la Isla de Tenerife que vincule arte, cultura y tecnología. A pesar de el gran potencial del proyecto sigue sin ser especialmente divulgativo ya que es más formato residencia.

A su vez, la Universidad de La Laguna son pocos los talleres, charlas o eventos que ha acogido en relación a estos temas. En mayo de 2022 fue sede de la 14ª edición del festival NumaCircuit en la facultad de Bellas Artes. Esta última edición combinó experimentación, performance, creación, visuales, arte sonoro y talleres creativos, pero de nuevo, el diseño gráfico queda fuera de la ecuación.

Es precisamente esta realidad lo que refleja la necesidad de un espacio en la isla, especialmente en la universidad, que permita difundir y divulgar sobre las nuevas tecnologías y los nuevos medios en el ámbito creativo.

1.3 Briefing

Cliente : Facultad de Bellas Artes, ULL

Contenido/producto: Desarrollo de un sistema flexible para la identidad visual de “Visual Beings 2023” Primera edición del Festival de Diseño y Tecnología de Canarias.

Grupo Objetivo: Estudiantes (Arte, diseño, informática, ingenierías...) además personal no universitario. Profesionales del sector, no profesionales, interesados en el tema.

Competencia: Festival Numacircuit.

Palabras clave: Diseño inteligente, tecnología, eficacia.

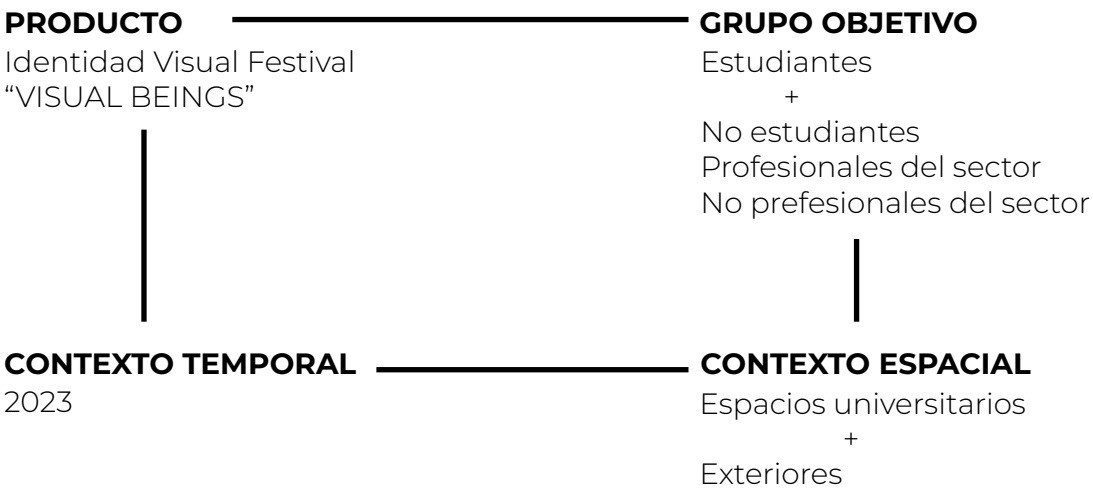
Contexto espacial: Espacios universitarios (interior) y espacios exteriores.

Contexto temporal: El proyecto está planteado para su inauguración en el año 2023.

Formatos: Web, redes sociales, carteleria y papelería, vinilos, camisetas, tote bags, bolígrafos y PINS.

Objetivo de comunicación: Diseñar una identidad visual con base modular que permita la flexibilidad y adaptación a cualquier formato del diseño, que sea atemporal y única. Que el usuario sea capaz de interactuar con ella en formatos digitales, que sea capaz de mostrarse en diferentes dimensiones (2d y 3d)

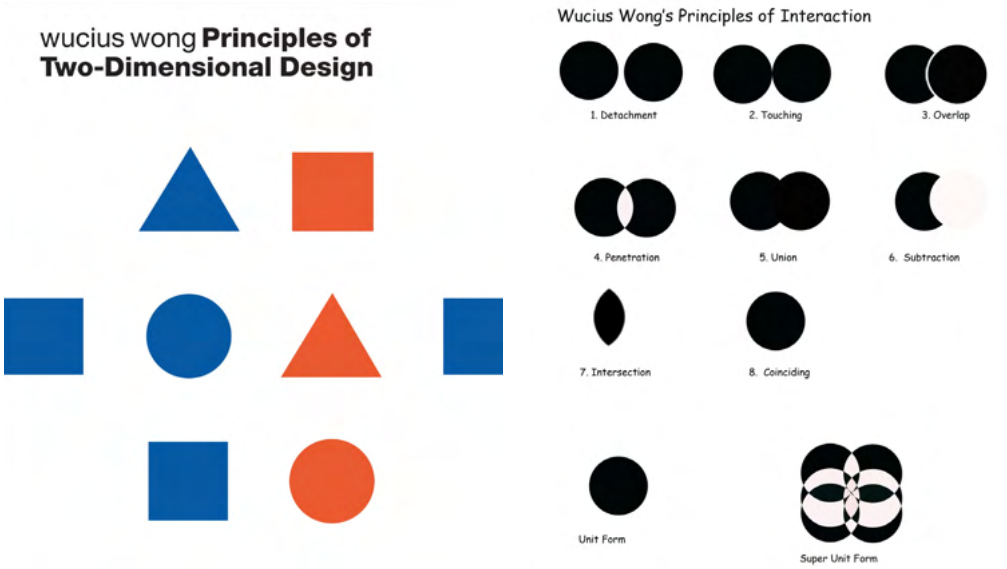
1.4 Parámetros de comunicación



1.5 Referencias visuales

Salir del ámbito gráfico

Figuras 29/30. Estructura experimental usando impresión 3D. Zaha Hadid Architects, 2017.



Figuras 31/32. Principles of two-dimentional design. Wucious Wong, 1972.

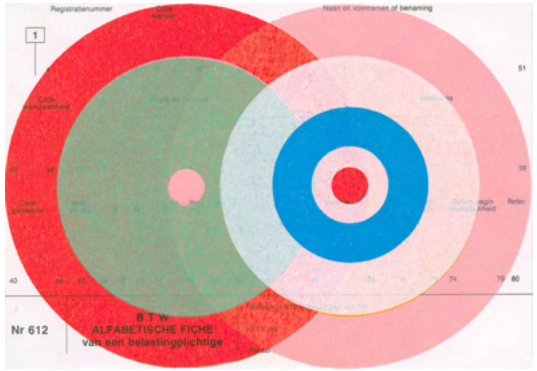


Figura 33. Untitled. Karel Martens, 2015.

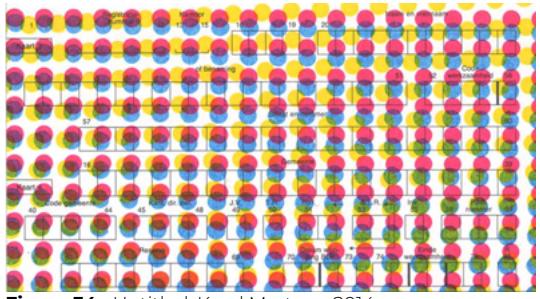
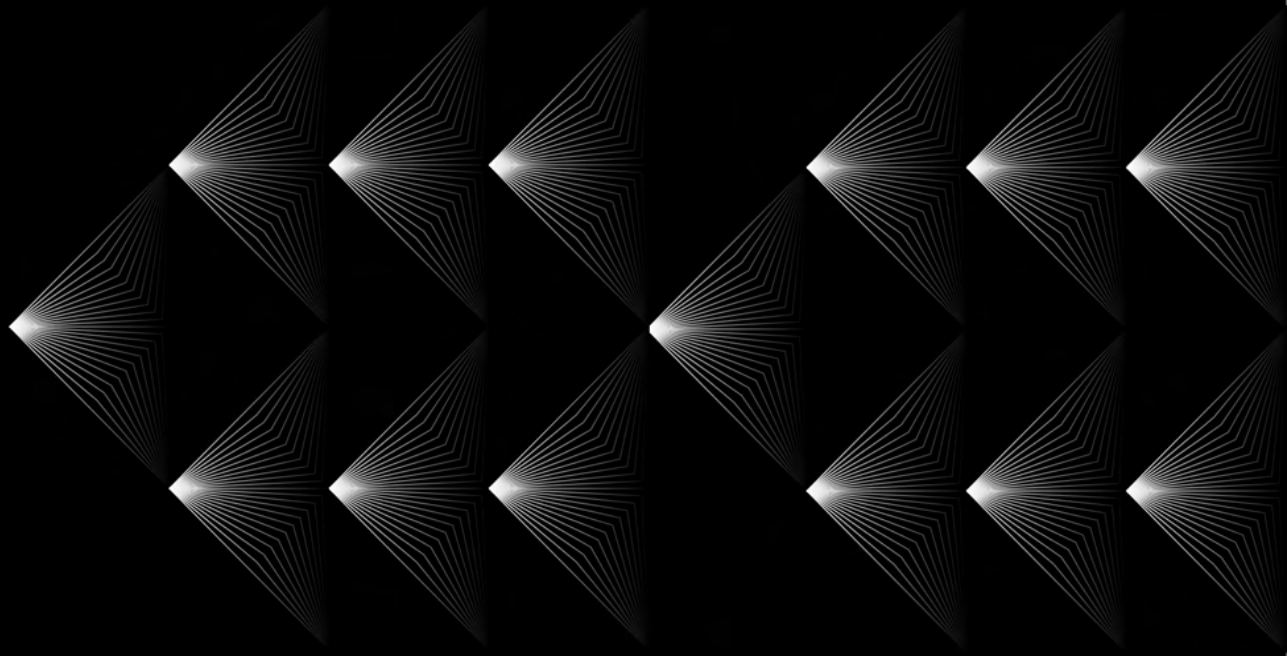


Figura 34. Untitled. Karel Martens, 2014.

Figura 35. Twopoints.net. Center for Complexity, Rhode Island School of Design, 2020.

Figura 36. Twopoints.net. Center for Complexity, Rhode Island School of Design, 2020.



FASE DE DESARROLLO

1. ¿CÓMO DISEÑAR UN SISTEMA FLEXIBLE?

1.1 Cambio de paradigma en el diseño de comunicación

Con la llegada de las redes sociales, no solo se crearon nuevos canales de comunicación, sino también nuevas formas de comunicarse, en las que los destinatarios del mensaje forman parte del proceso, haciendo necesario que el lenguaje visual de nuestros diseños sea consistente y flexible.

Hubo un tiempo en el que el diseño de identidades visuales era sinónimo de diseño de logos, y en alguna ocasión aún puede serlo, hasta cierto punto.

Hoy en día, una identidad visual basada en un logo, no es adecuada para comunicar de manera efectiva. No se adapta al entorno, no es capaz de formular el mensaje óptimo. Es por esto, que necesitamos identidades visuales capaces de resolver problemas comunicativos contemporáneos.

Al final, lo que logramos es que una marca sea reconocible por todo el lenguaje modular que creemos para la marca y no por un simple logo que se acabará utilizando únicamente como foto de perfil en redes, o como sello en productos.

Las ventajas de esta nueva forma de diseño es la flexiblidad de nuestro sistema, y las infinitas posibilidades de combinación que nos ofrece.

Esta imagen a la derecha muestra un ejemplo de cómo sin la necesidad de un logo, sabemos que estas propuestas todas pertenecen a la misma marca y el lenguaje creado es lo que la hace reconocible.

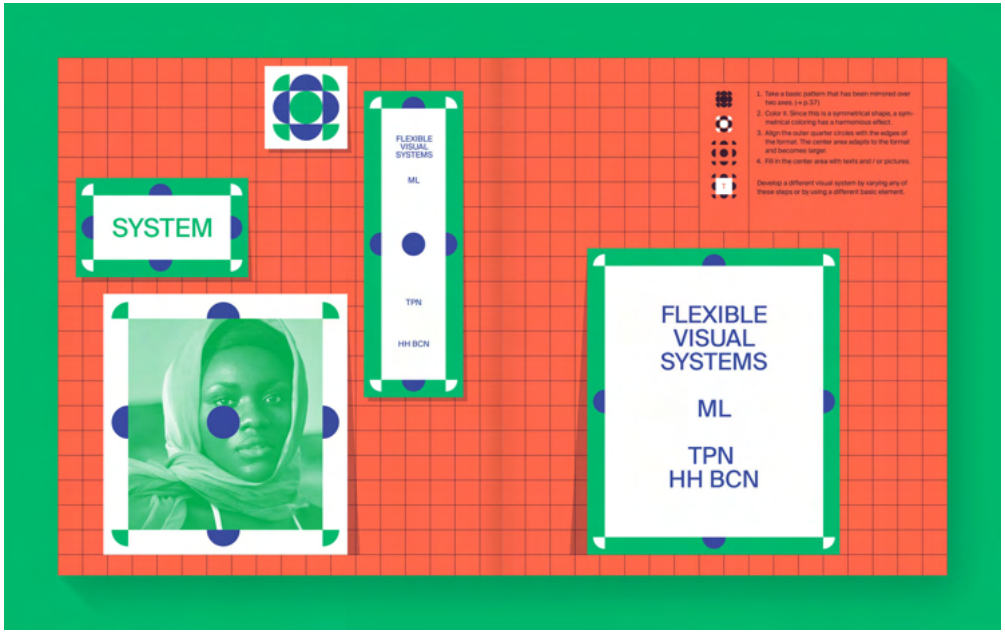


Figura 37. Flexible Visual Systems. Martin Lorenz, 2021.

1.2 Creación de activos a partir de componentes

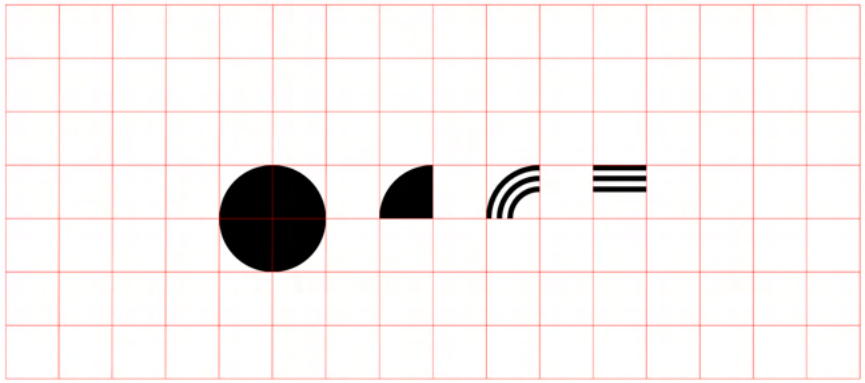
El punto de partida de este proyecto es la experimentación con componentes en cuadrículas de Adobe Ilustrator y el desarrollo de diferentes activos modulares.

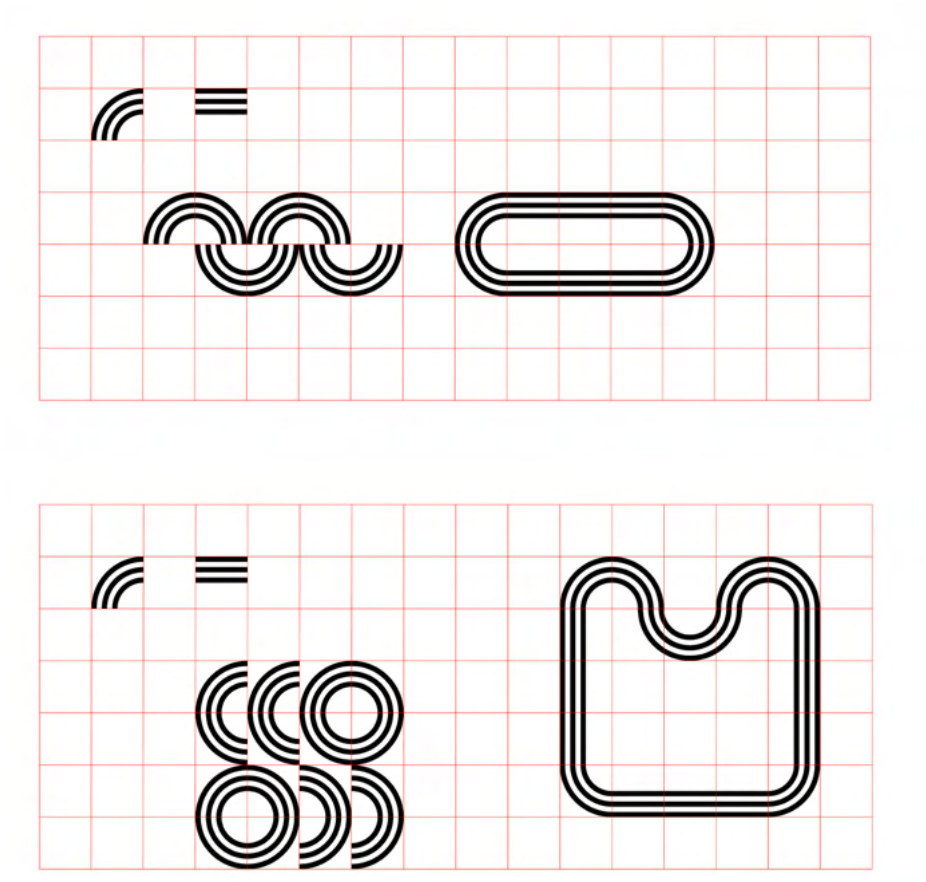
1. Definimos dos palabras que definirán la estética de nuestra identidad en base al nombre del festival “VISUAL BEINGS”

Estas son **orgánico** (BEINGS) y **tecnológico** (VISUAL)

2. Definimos una cuadrícula y las formas geométricas base con las que comenzaremos a diseñar.

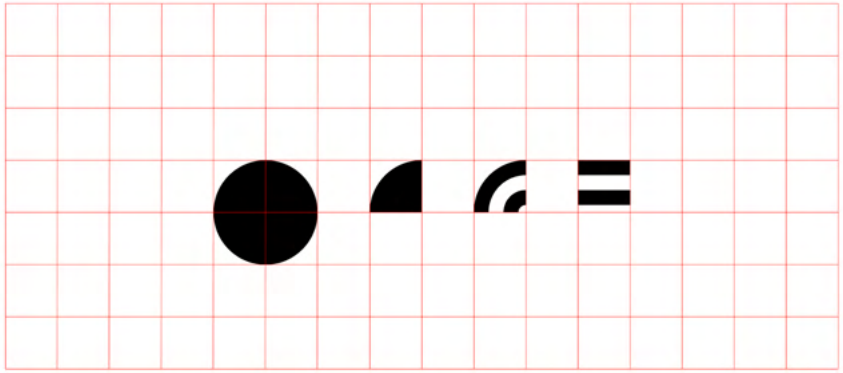
Forma base: Círculo / cuarto de circulo

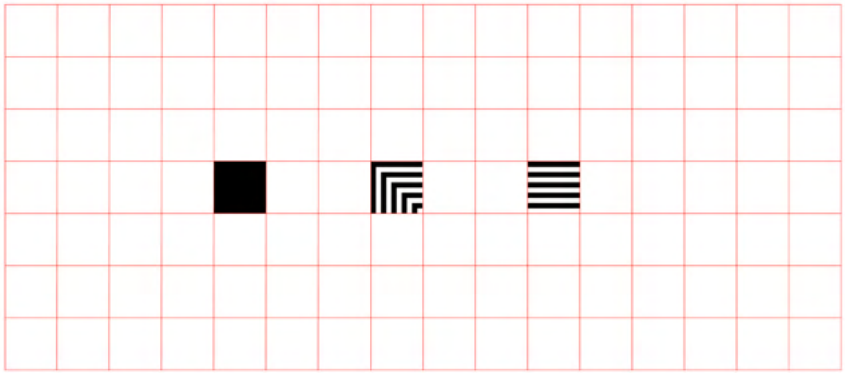


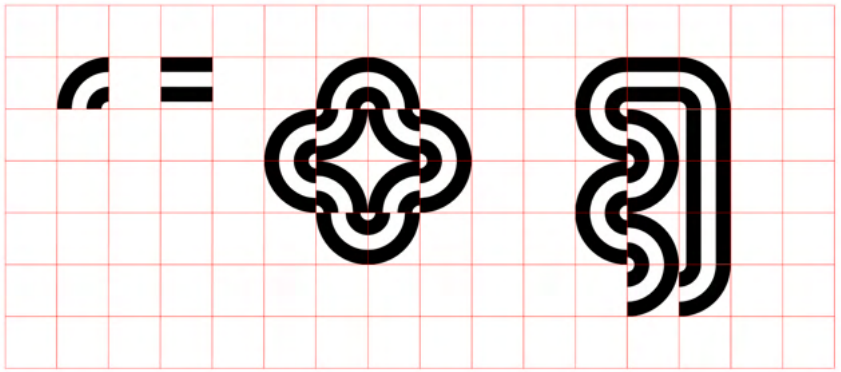


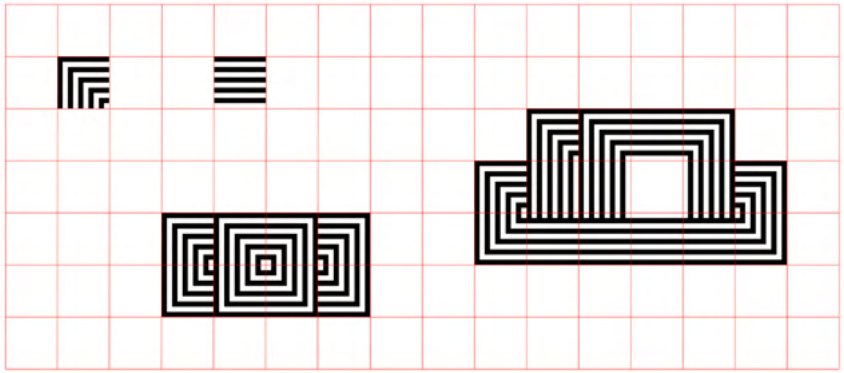
La idea es, a partir de los componentes base, en este caso el cuarto de círculo, crear una serie de diferentes activos, tanto horizontales, como verticales, para posteriormente ver cómo lo usaremos.

Simplemente, explorar las posibilidades que este componente nos ofrezca.









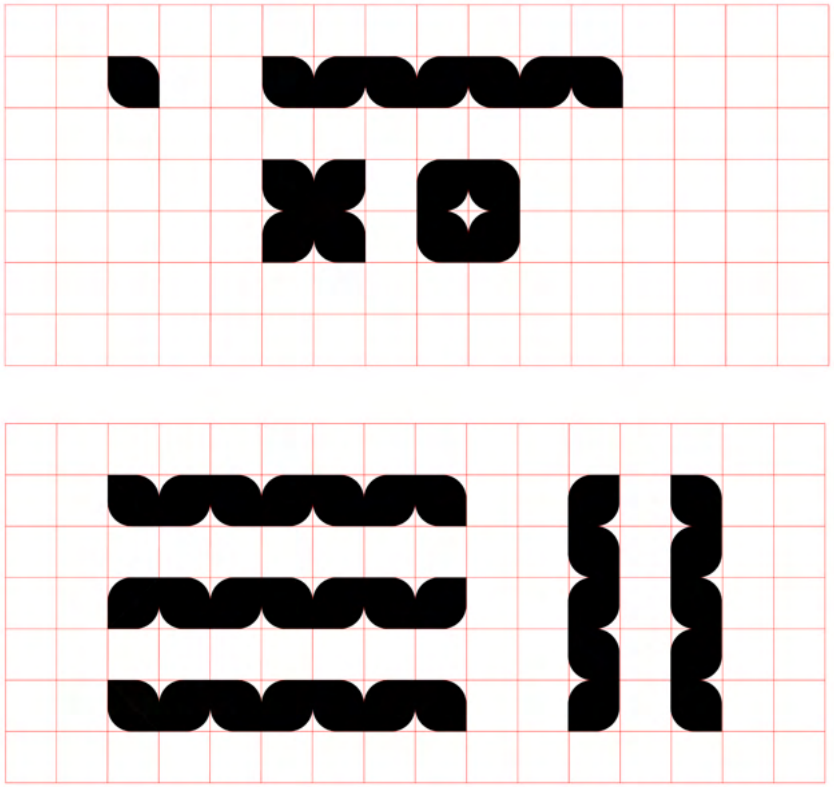
1.3 Primeras propuestas

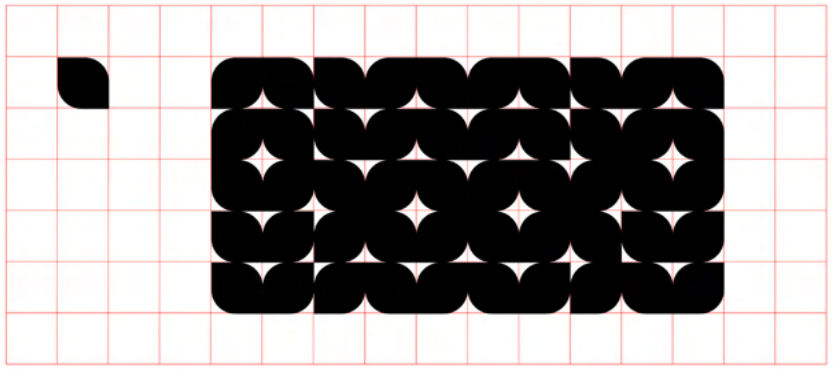
Una vez explorada la modularidad, comencé a centrarme más en las palabras que definirían la estética en nuestro proyecto. Orgánico y tecnológico.

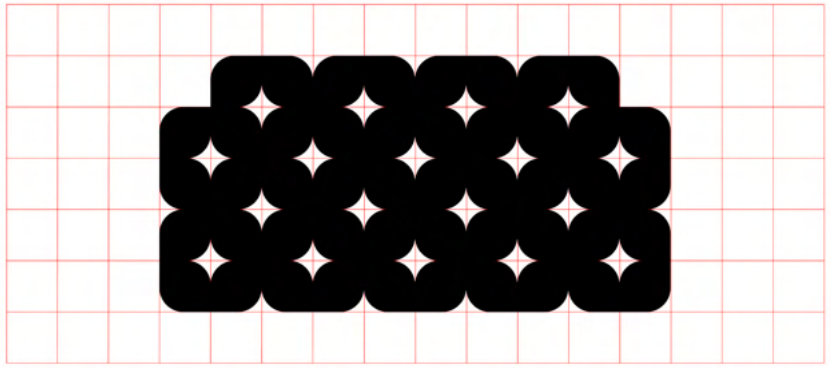
La idea era encontrar un componente base redondeado que diera esa sensación de orgánico o vivo, mientras que combinándolo obtuvieramos resultados visualmente modernos y “tecnológicos”

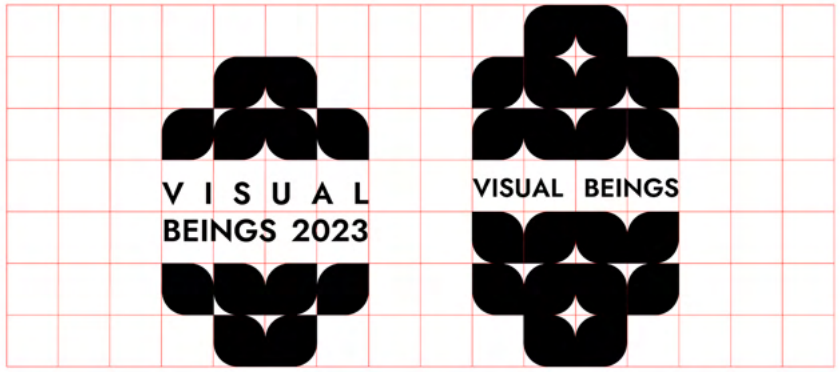
Creamos símbolos, líneas y recursos horizontales y verticales, además de algún patrón.

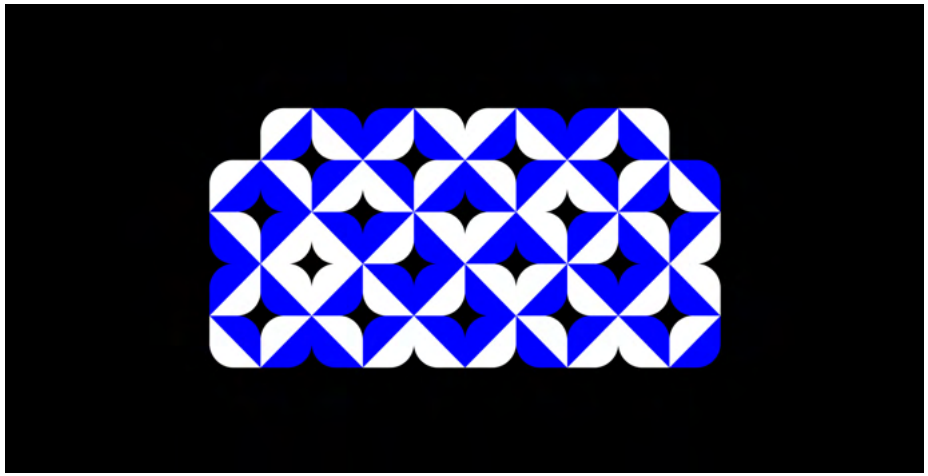
Finalmente les añadiremos color, a ver qué posibilidades nos ofrece.

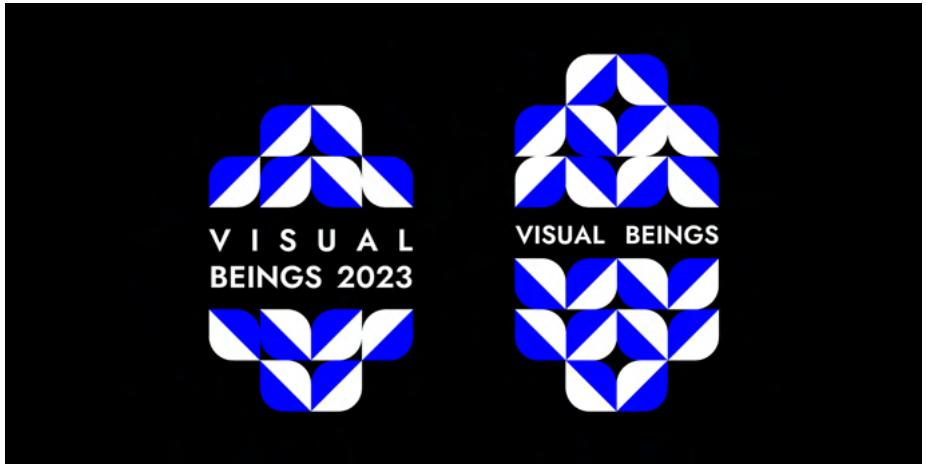




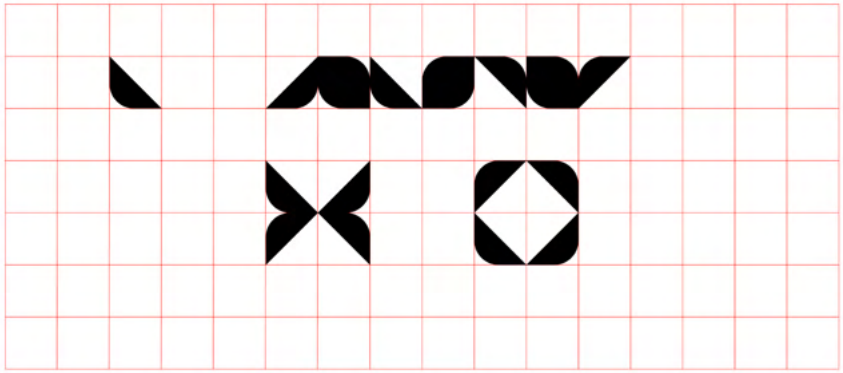


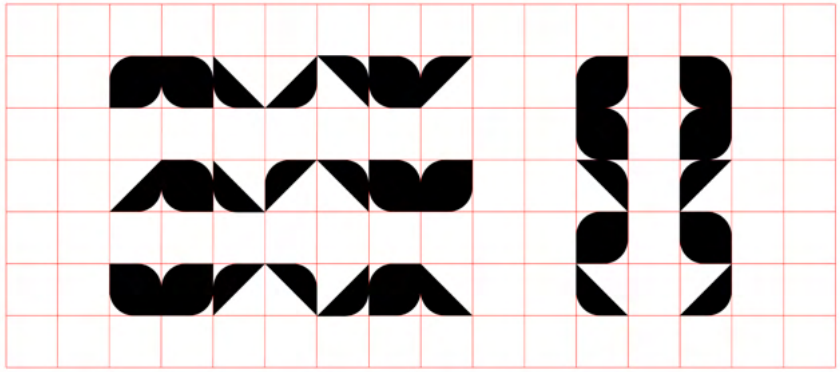

VISUAL
BEINGS 2023
VISUAL BEINGS

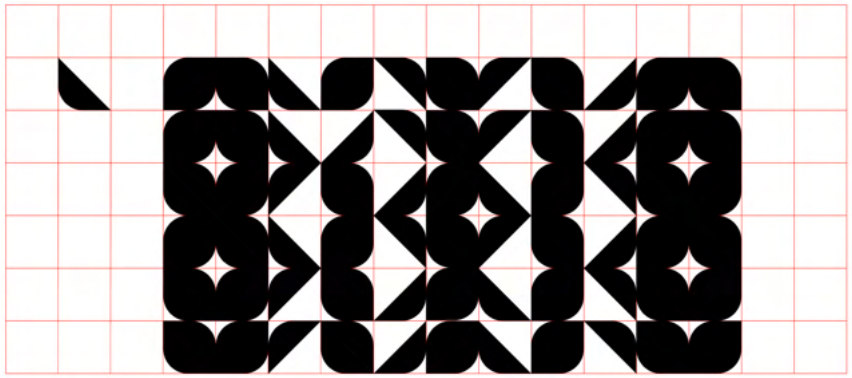


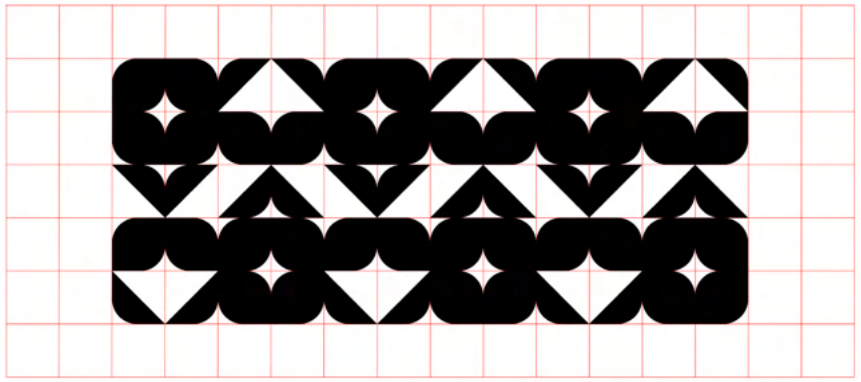

VISUAL
BEINGS 2023
VISUAL BEINGS

Tras crear varios diseños con el componente anterior, decidí usar la mitad del mismo componente, para crear otra propuesta de “0”. Los resultados son completamente distintos. Esto solo demuestra la infinidad de posibilidades que nos ofrece el diseño modular.

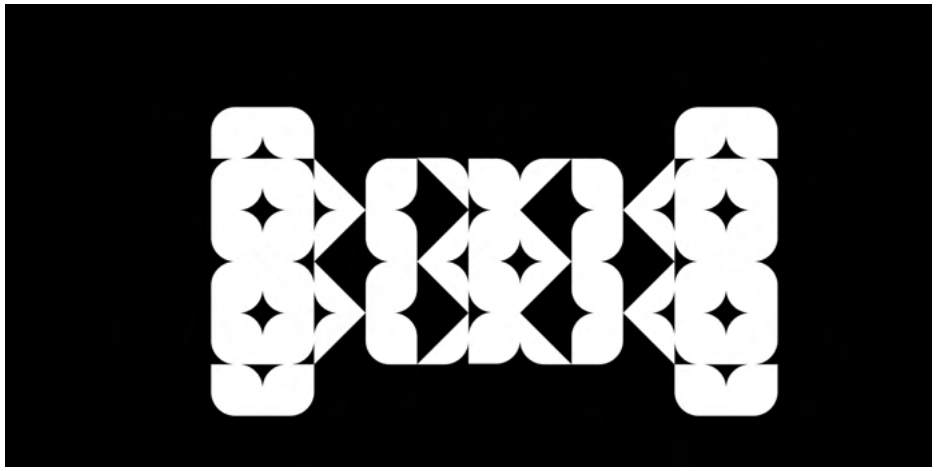


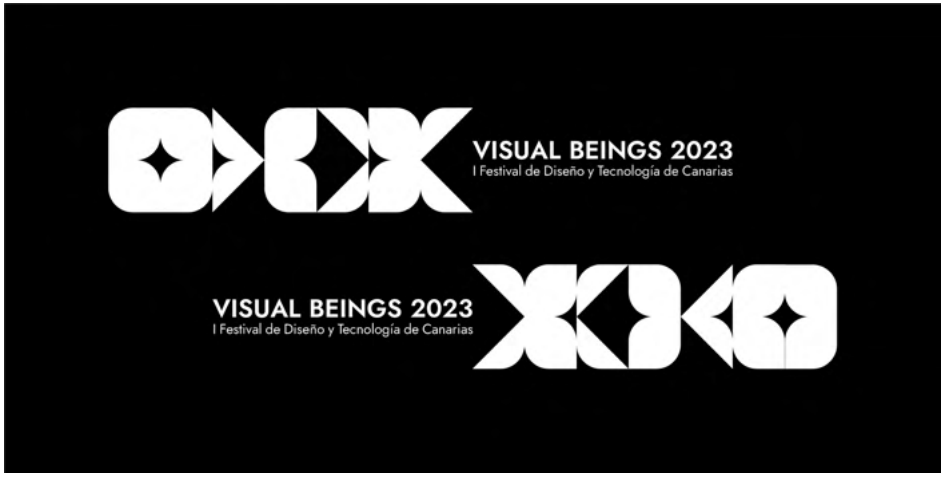







VISUAL BEINGS 2023
I Festival de Diseño y Tecnología de Canarias
VISUAL BEINGS 2023
I Festival de Diseño y Tecnología de Canarias

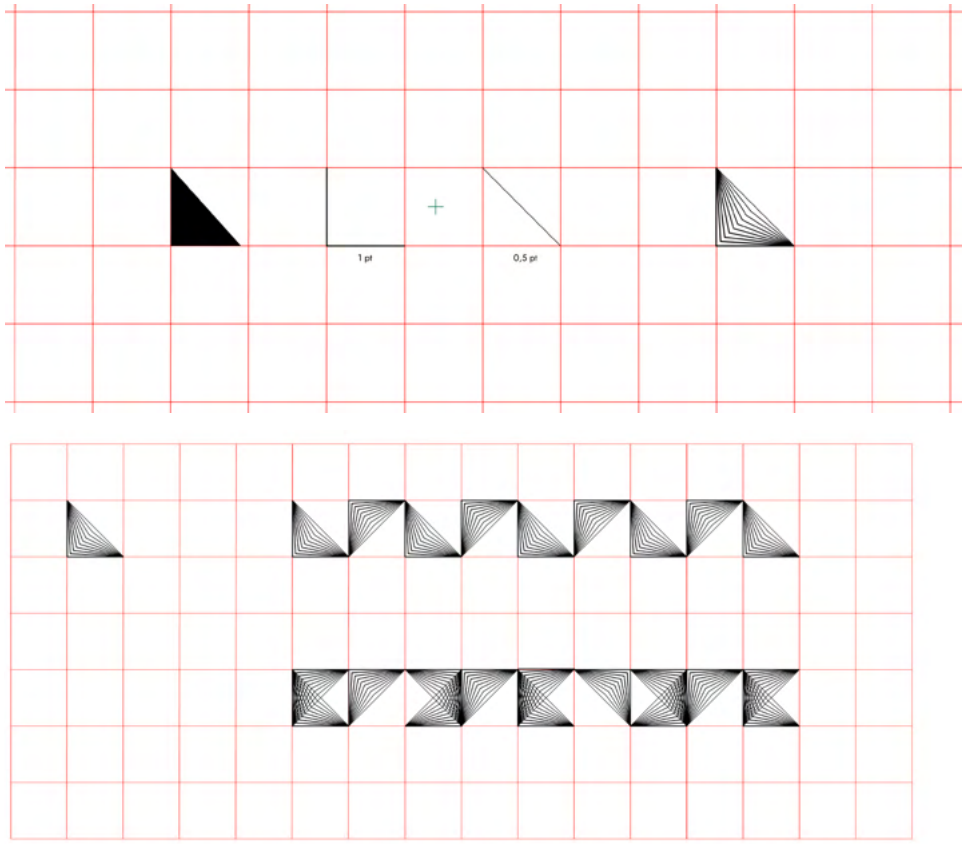


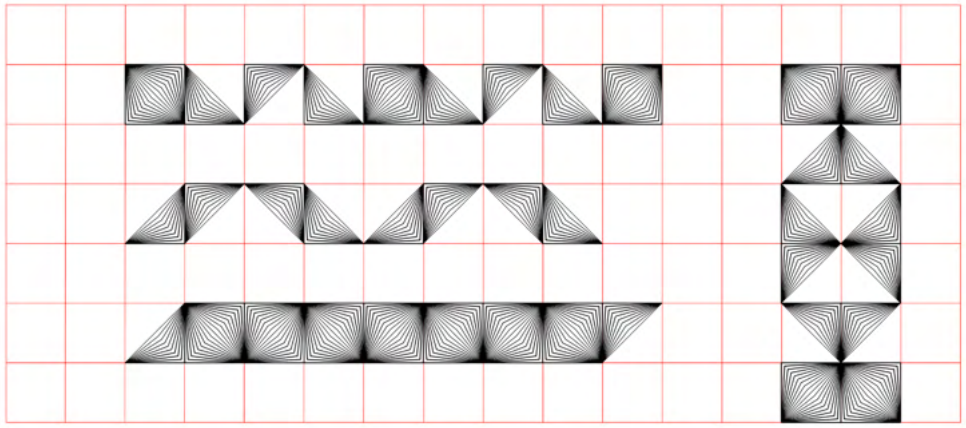

VISUAL BEINGS 2023
I Festival de Diseño y Tecnología de Canarias
VISUAL BEINGS 2023
I Festival de Diseño y Tecnología de Canarias

1.4 Propuesta final

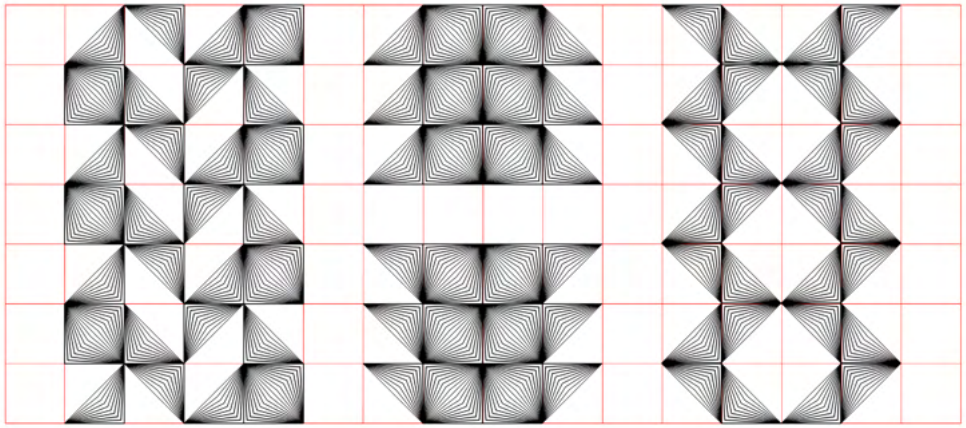
Finalmente, decidí escoger un nuevo componente que de verdad llevara intrínsecos los conceptos de orgánico y tecnológico.

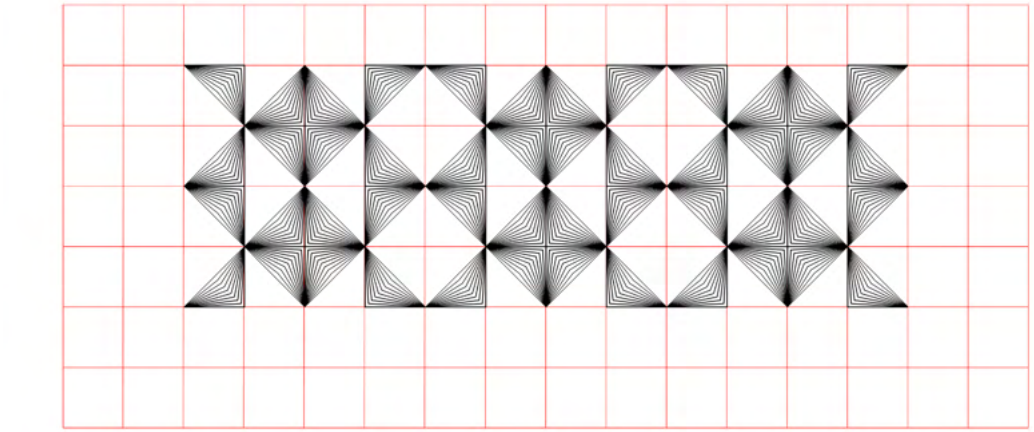
Comencé con la forma de un triángulo, pero decidí aplicarle un degradado lineal, que visualmente aporta mucho significado en relación a lo tecnológico y además, crea movimiento en la imagen que refleja lo orgánico/vivo.



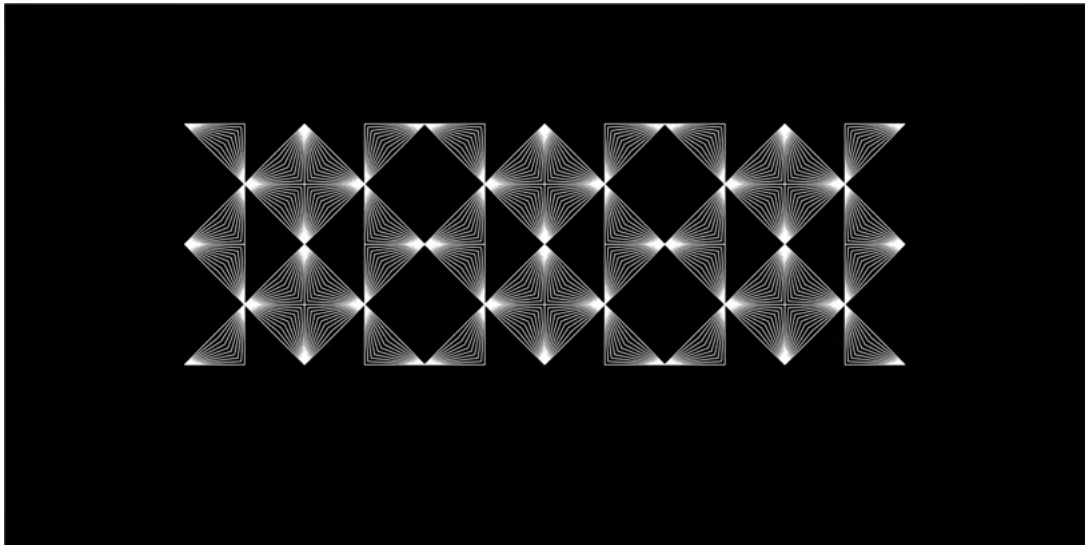


Probé en un principio el diseño de líneas tanto horizontales como verticales, y luego recursos verticales en base a estas líneas y con nuevas combinaciones.





Luego realicé recursos horizontales que pueden servir para vinilo, cartelería, flyer, etc...



1.41 Elección tipográfica

Una vez sentadas las bases de nuestro sistema, procedí a elegir una tipografía para comenzar a combinar con nuestros activos.

La elección de una tipografía de palo seco fue por la necesidad de reflejar lo orgánico, el futuro, lo moderno...

La fuente SemiBold será utilizada únicamente para el nombre del festival "VISUAL BEINGS 2023" y la Regular, para el texto que va describiendo el festival "I Festival de Diseño y Tecnologia de canarias"

Los tamaños en los que se usará estas fuentes variarán según los formatos y no tiene ningún tipo de restricciones en este sentido.

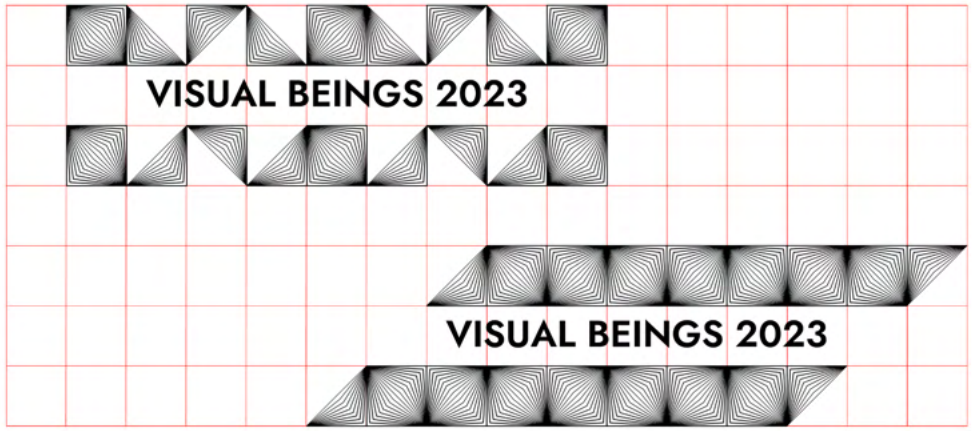
Una vez realizado este paso, comenzamos a componer con nuestros activos y la tipografía.

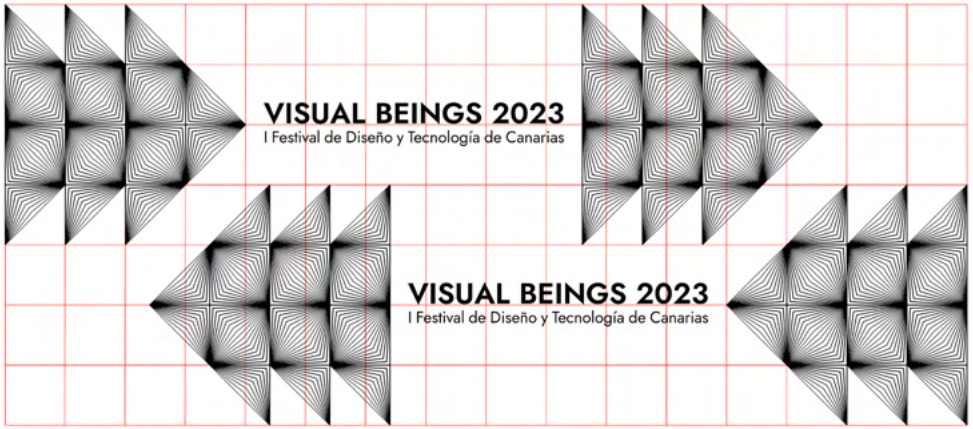
JOST SemiBold

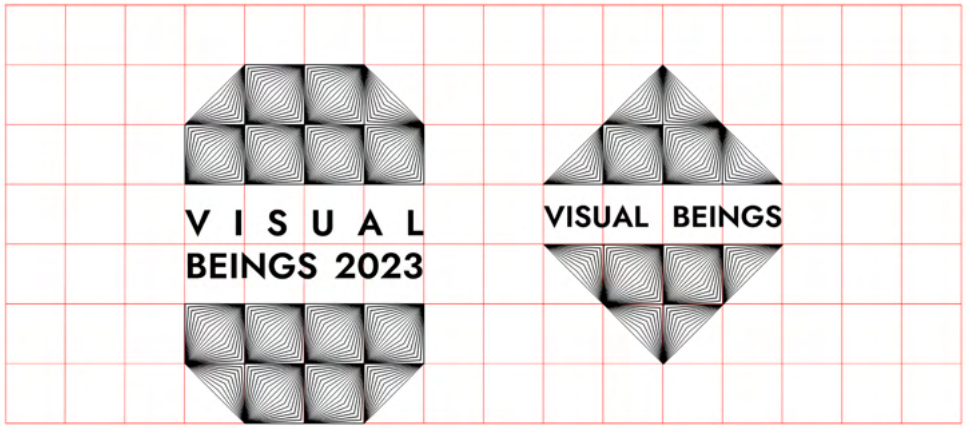
JOST Regular

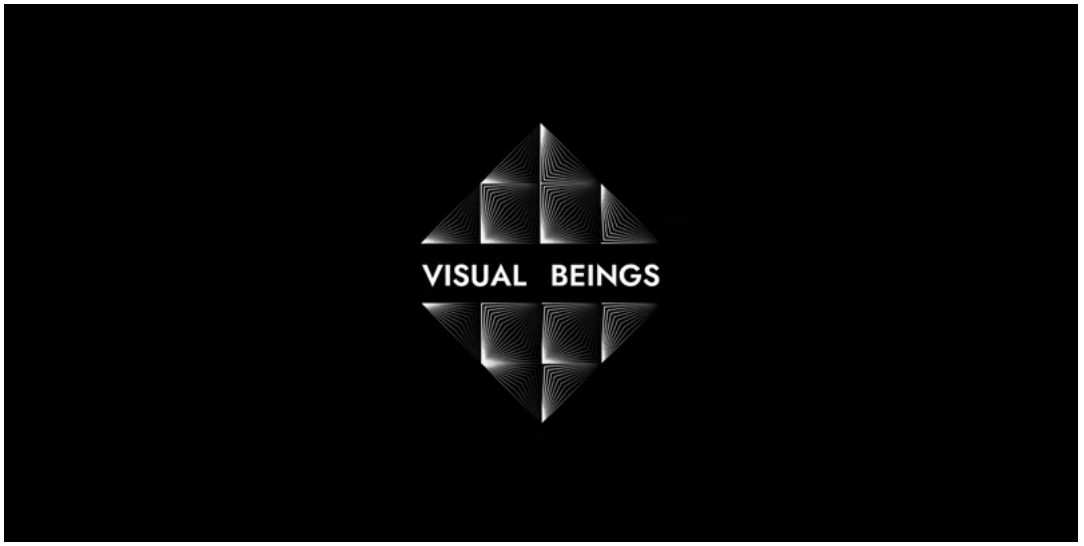
VISUAL BEINGS 2023

I Festival de Diseño y Tecnología de Canarias 2023


VISUAL BEINGS 2023
VISUAL BEINGS 2023


VISUAL BEINGS 2023
I Festival de Diseño y Tecnología de Canarias
VISUAL BEINGS 2023
I Festival de Diseño y Tecnología de Canarias


VISUAL
BEINGS 2023
VISUAL BEINGS


VISUAL BEINGS

1.42 Rueda de color

La rueda de color es una elección bastante relevante porque afecta a el número de medios en los que el diseño se podrá enseñar, ya que es necesario que los colores elegidos puedan representarse en todos los modos de color.

Se emplea una paleta compuesta por cuatro colores. Negro puro, blanco, y un verde frío.

La idea es sumar al degradado lineal de nuestro módulo base, un degradado de color, que funda el azul en el naranja.

CMYK 0,0,0,100
RGB 0,0,0
HEX #000000

CMYK 0,0,0,0
RGB 255,255,255
HEX #ffffff

CMYK 84,14,93,2
RGB 0,168,69
HEX #009245

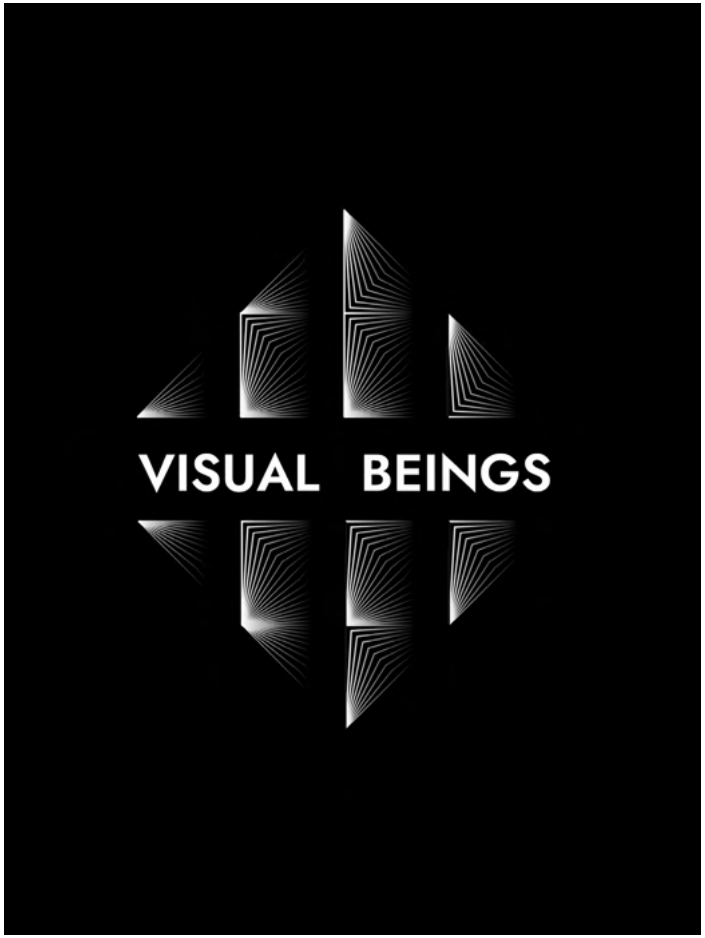
1.5 Definición de activos finales y aplicación tipográfica y de color.

Una vez definidas la tipografía, la rueda de color de nuestro proyecto, y nuestros componentes base y posibles combinaciones, procedemos a realizar los diseños finales.

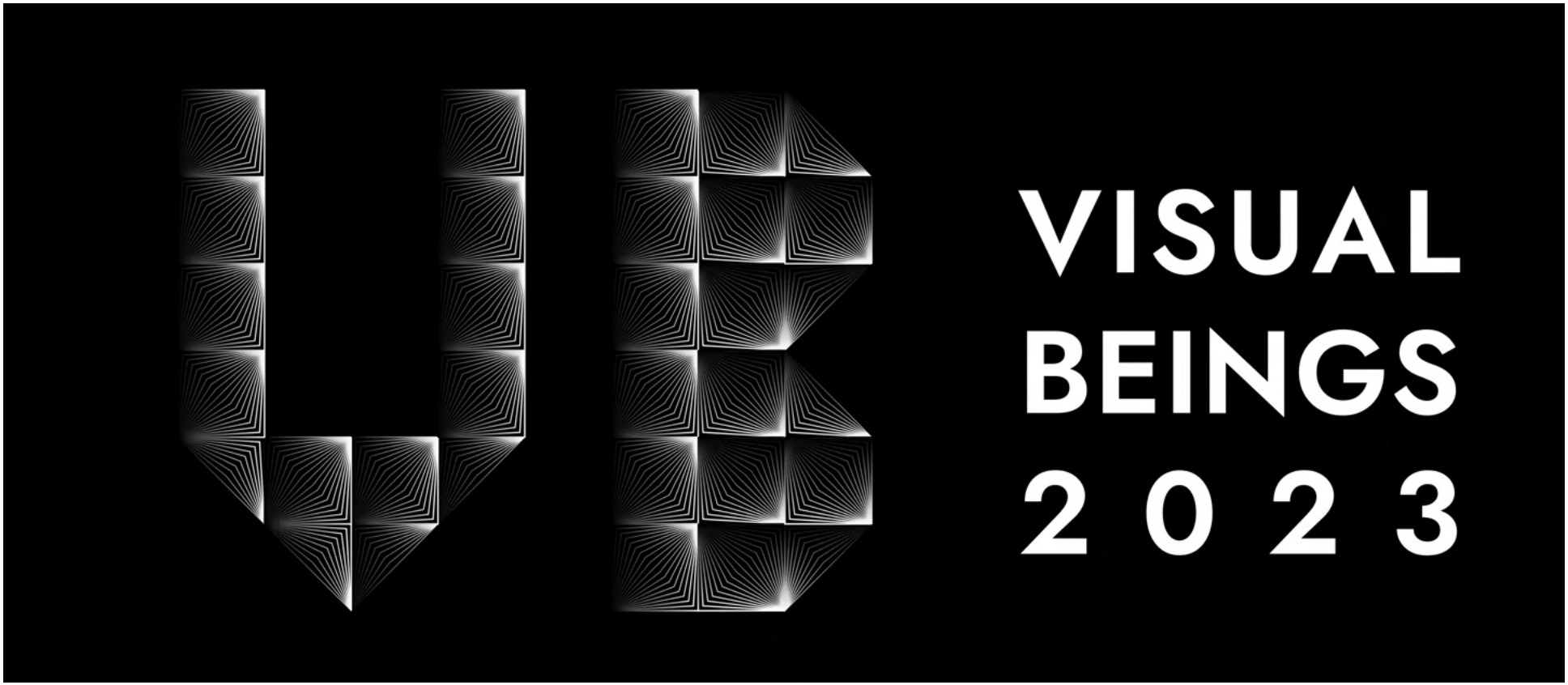
El sistema modular ha sido utilizado únicamente para la elaboración de nuestros activos. Como vemos, la composición tipográfica no tiene ninguna restricción de este tipo.

Esta composición reticular podría haberse utilizado para cuadrar tipografías y componer todo el diseño de manera modular, sin embargo, la idea de este TFG es la libertad de creación en este sentido.

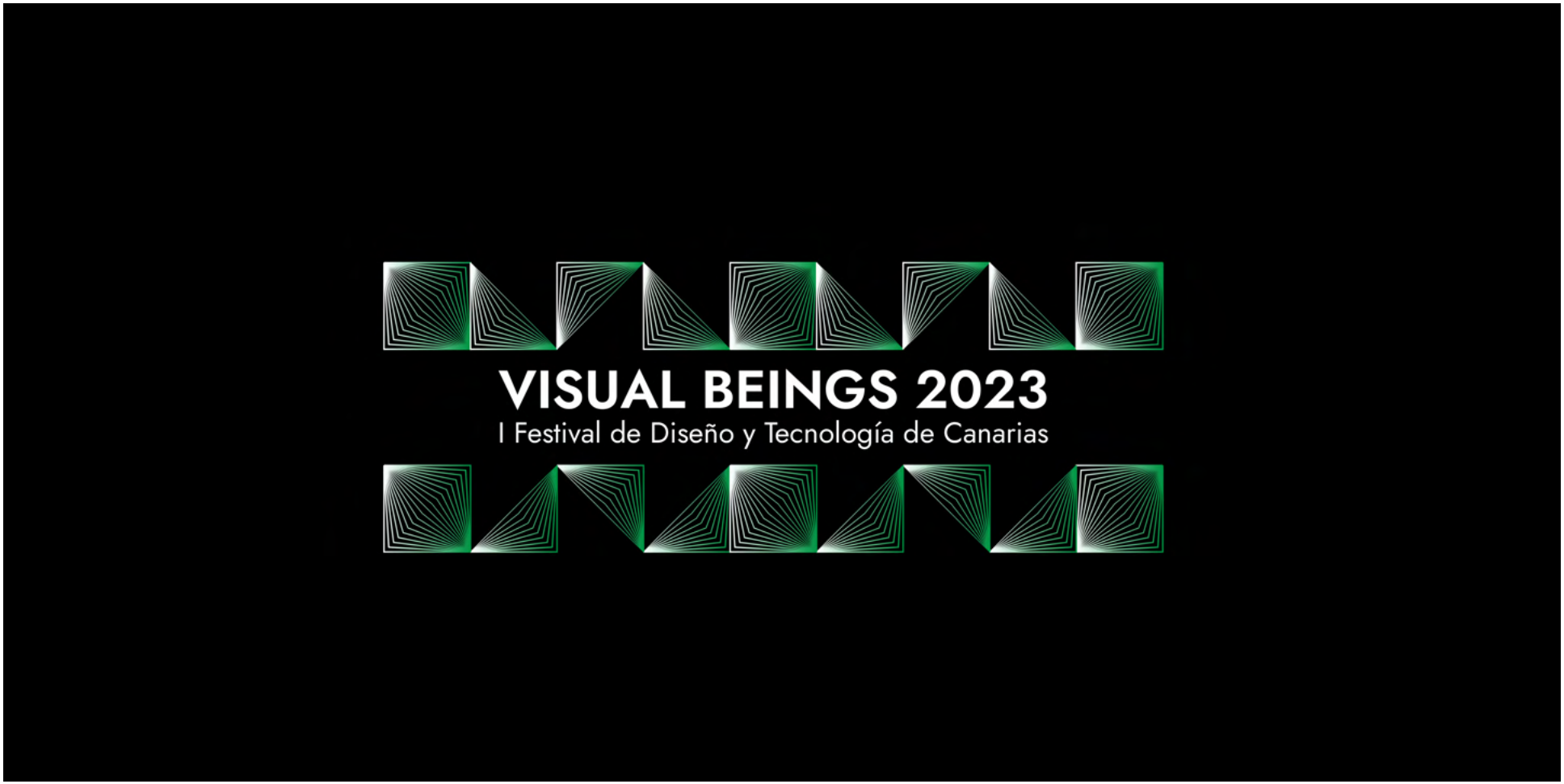
La modularidad abarca únicamente el diseño ilustrativo, que sí tiene algún tipo de restricción que se especificará en el manual de identidad.



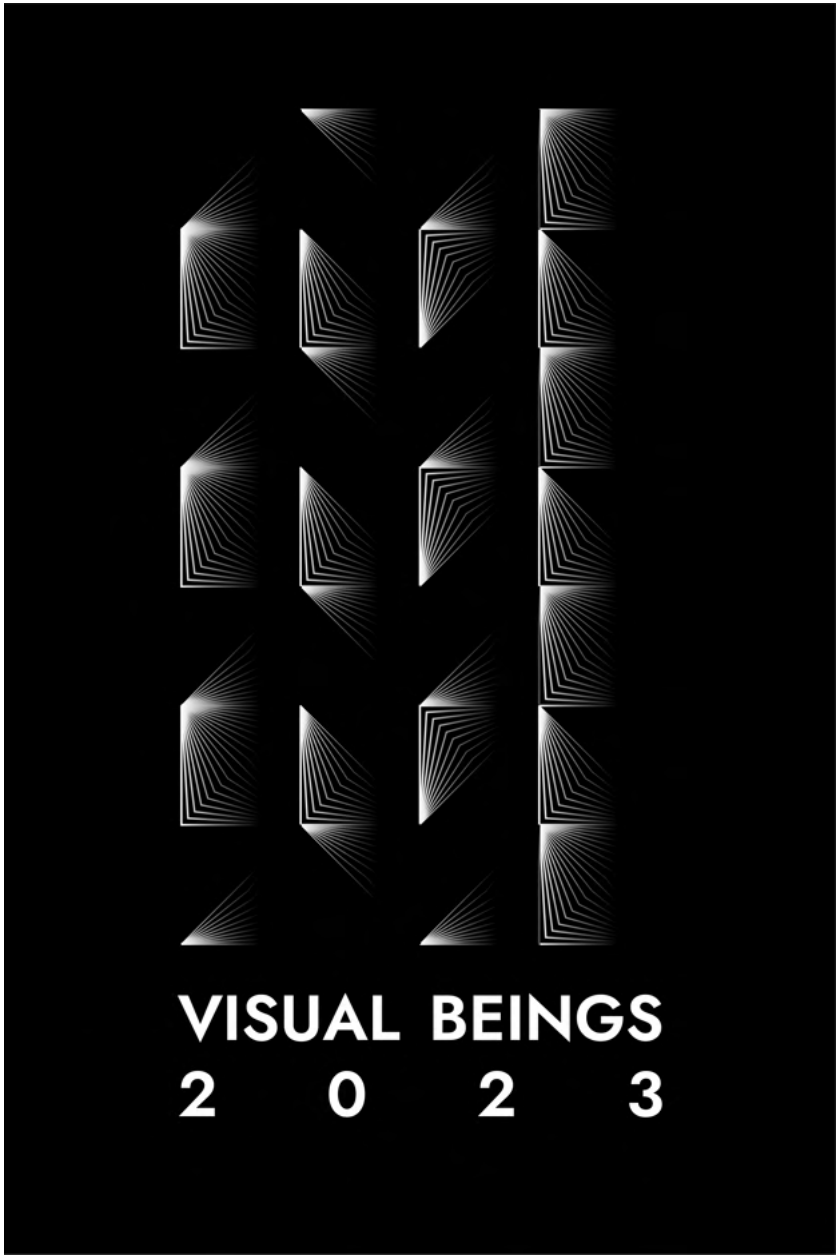
Símbolo básico + Naming



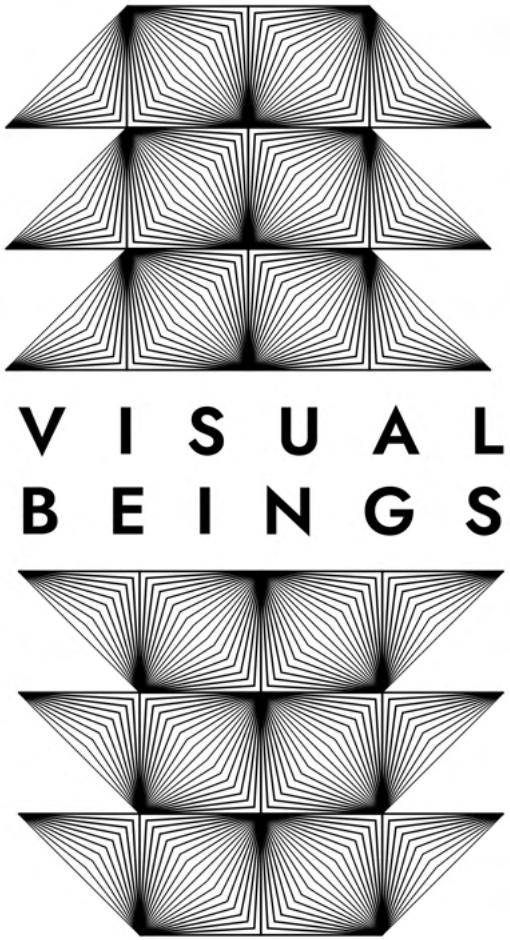
Construcción tipográfica modular + Naming



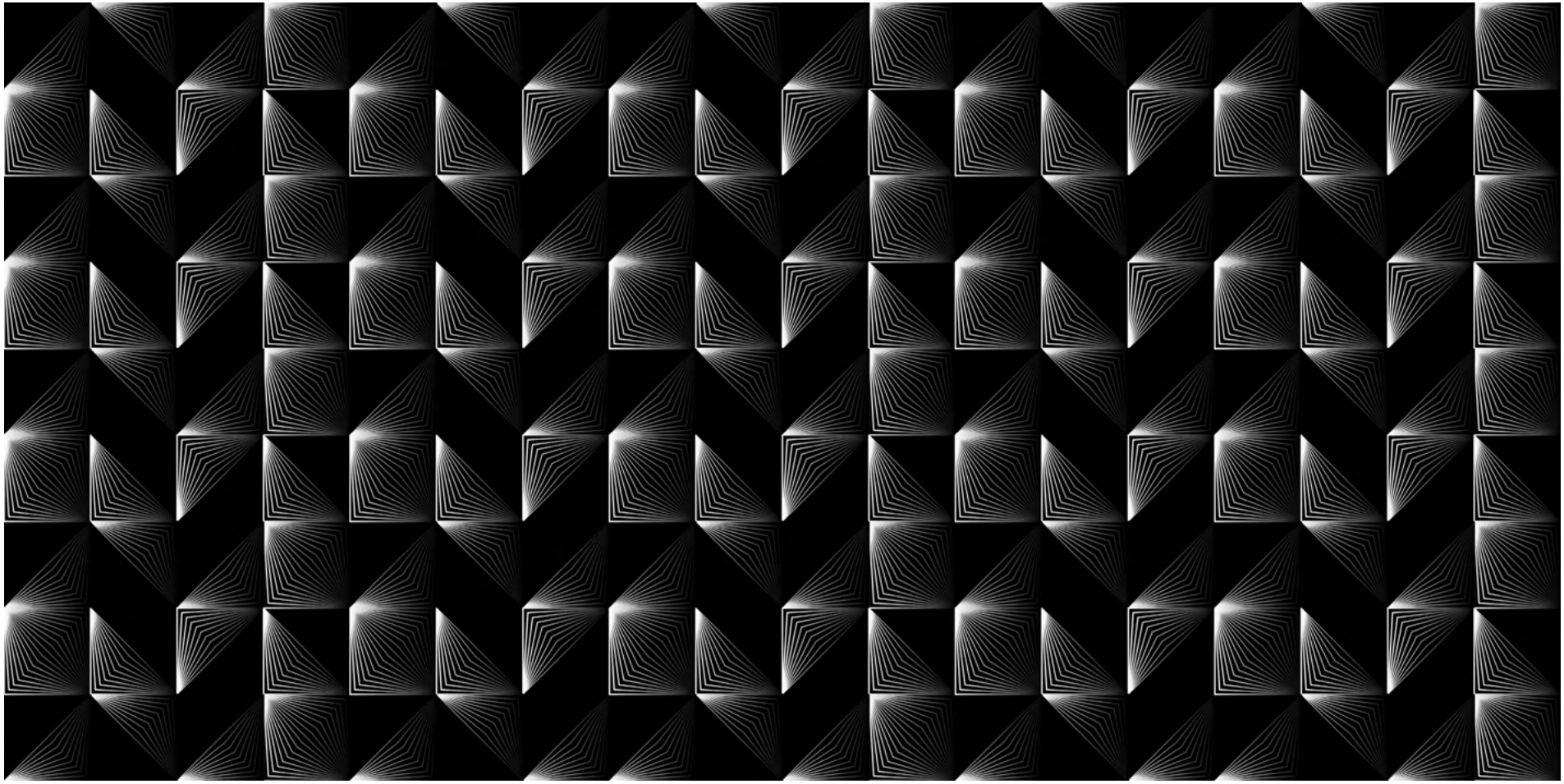
Aplicación de color en el degradado lineal de los activos + Naming



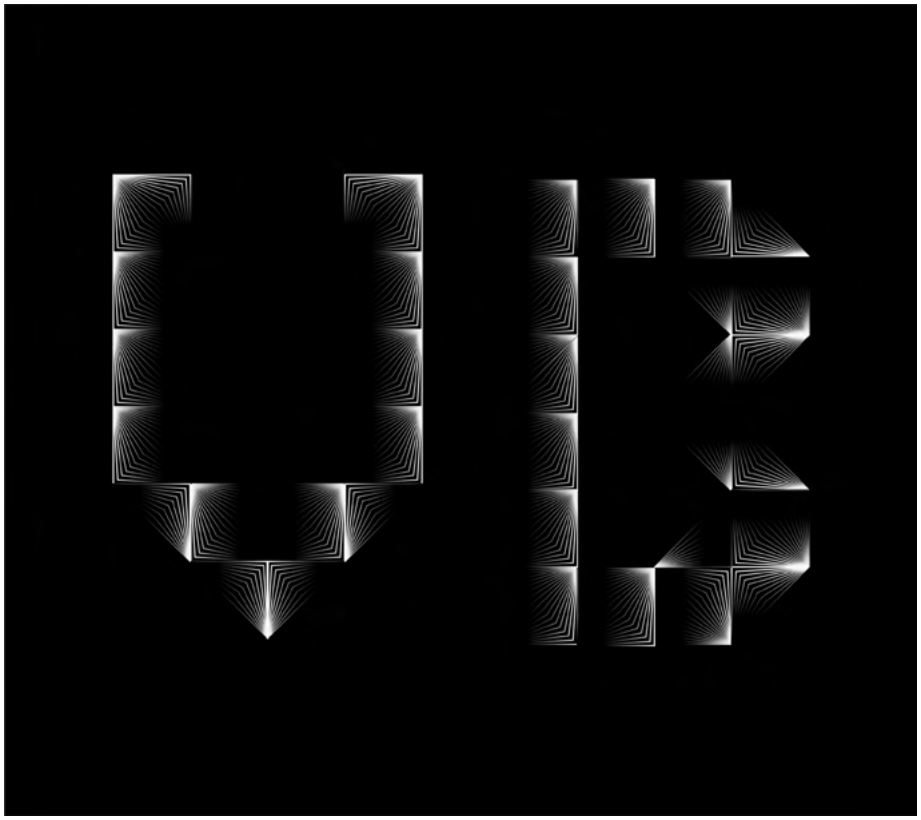
Recurso vertical + Naming



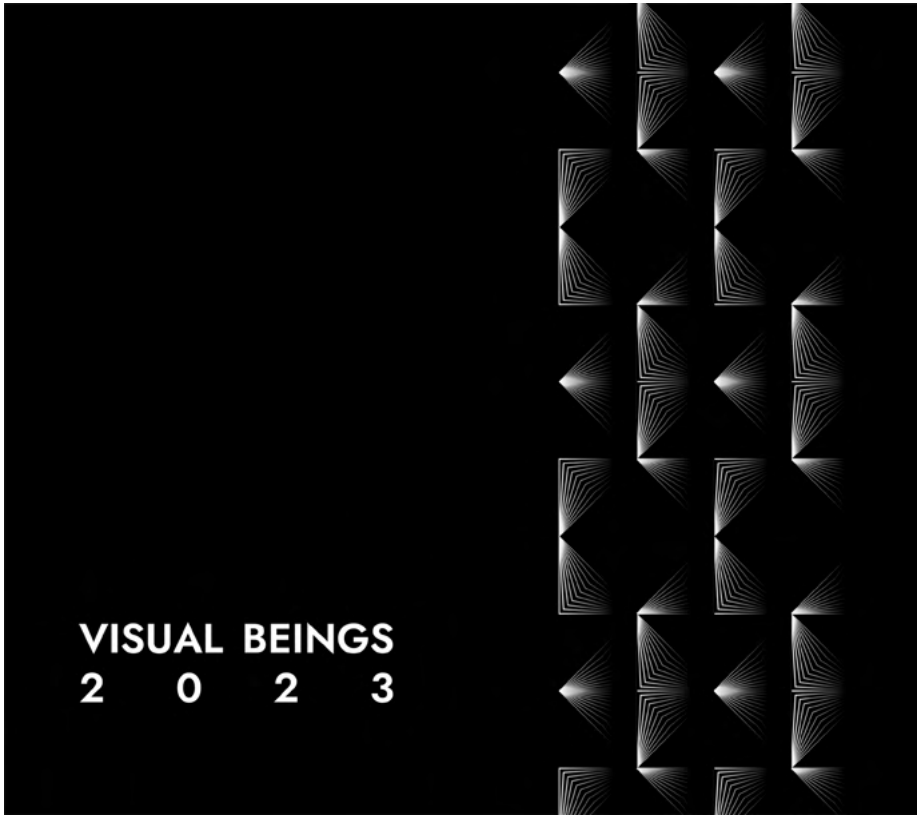
Recurso vertical + Naming



Diseño de patrón



Variación tipográfica



Recurso vertical + Naming

2. MATERIAL PROMOCIONAL

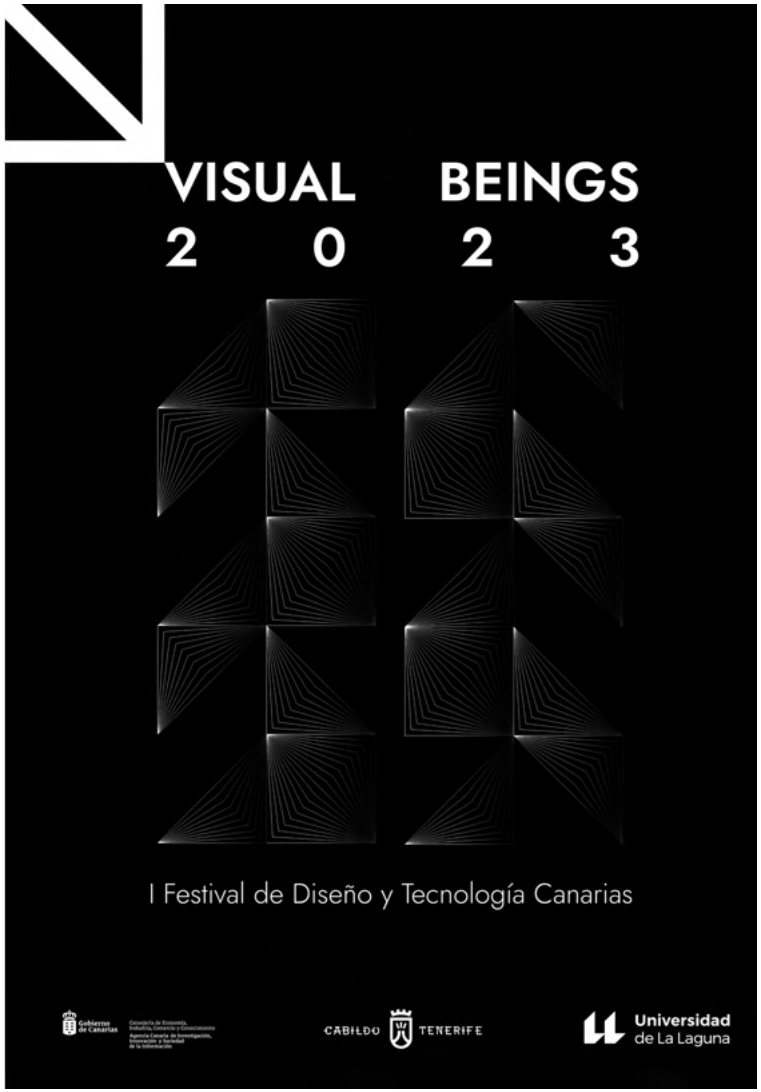
2.1 Analógico

A partir de los activos creados para el festival, se comenzó a crear material gráfco de carácter promocional.

Se realizaron propuestas de cartelería, de vinilo, de señalización de interior, y varios banners publicitarios.

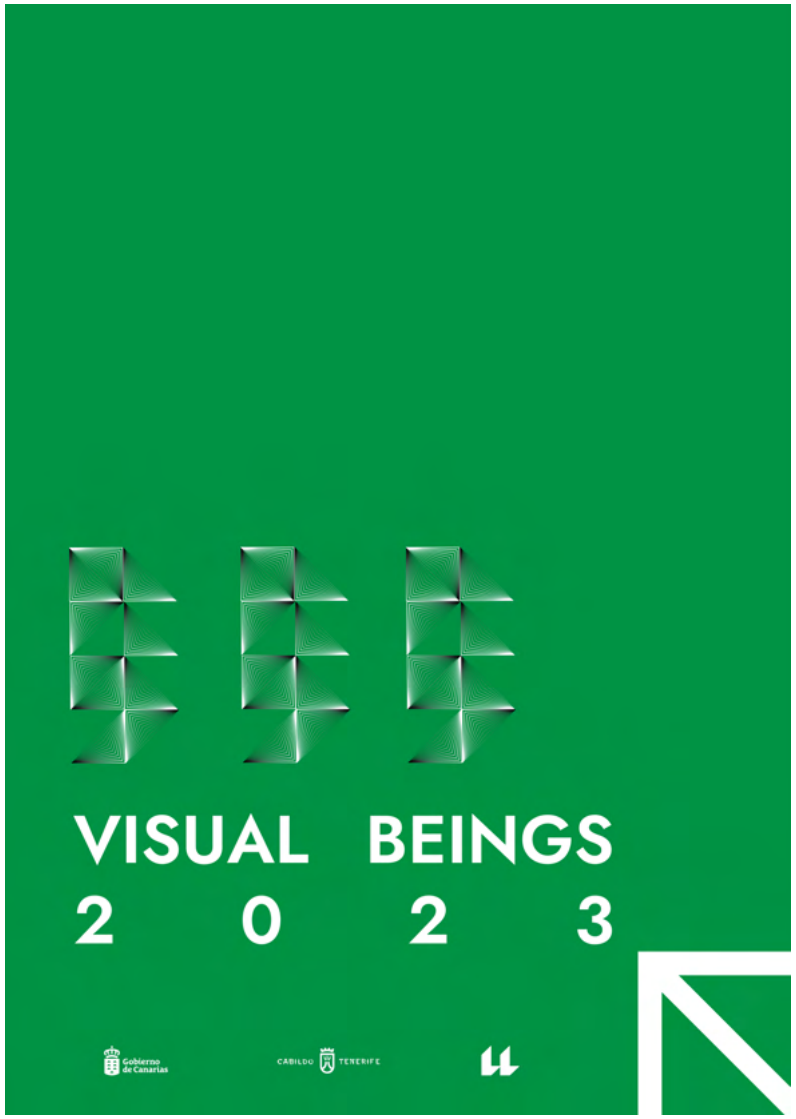
Como observamos, no hay guías a la hora de crear nuestras propuestas porque no se trata de una identidad fija. Al ser una identidad visual flexible, hay muy pocas restricciones que especificaremos posteriormente en el manual de identidad.

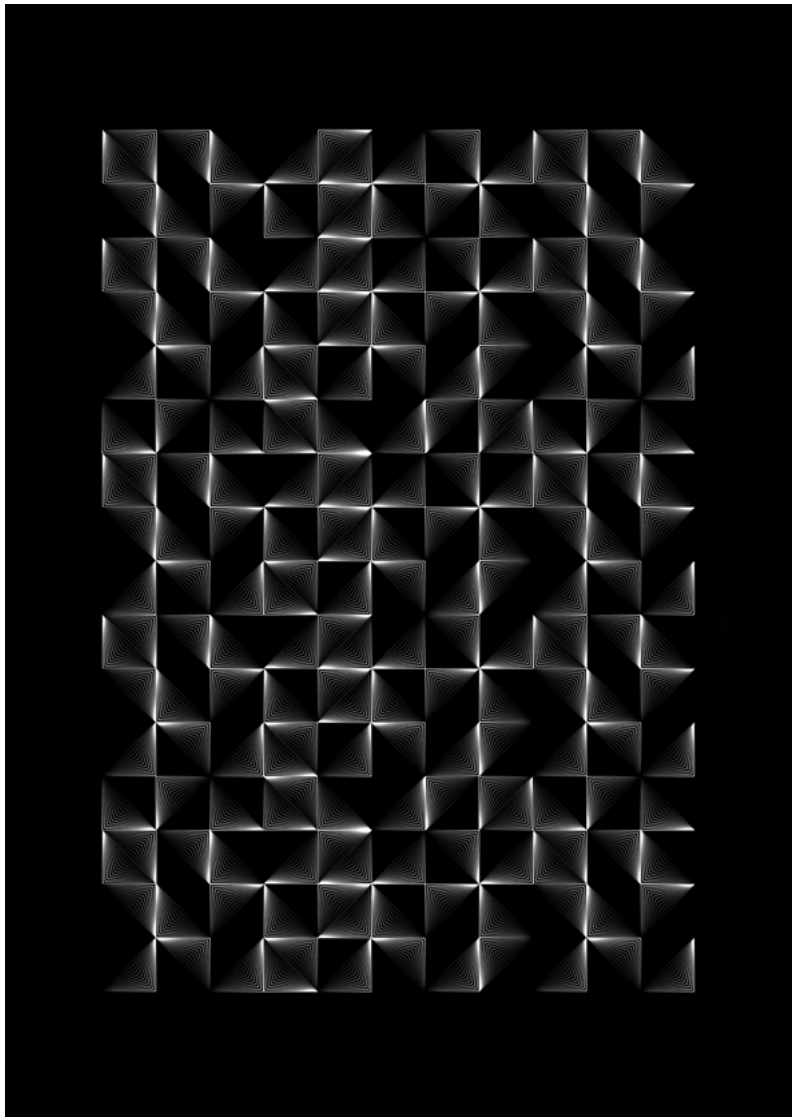
Los módulos son tratados como ilustración y el diseño en cuadrícula ocurre únicamente en el diseño de las ilustraciones. Una vez esto es aplicado a un cartel o cualquier otro soporte, el resto de la composicion es competamente libre respetando únicamente las tipografías y los colores correspondientes a la identidad.



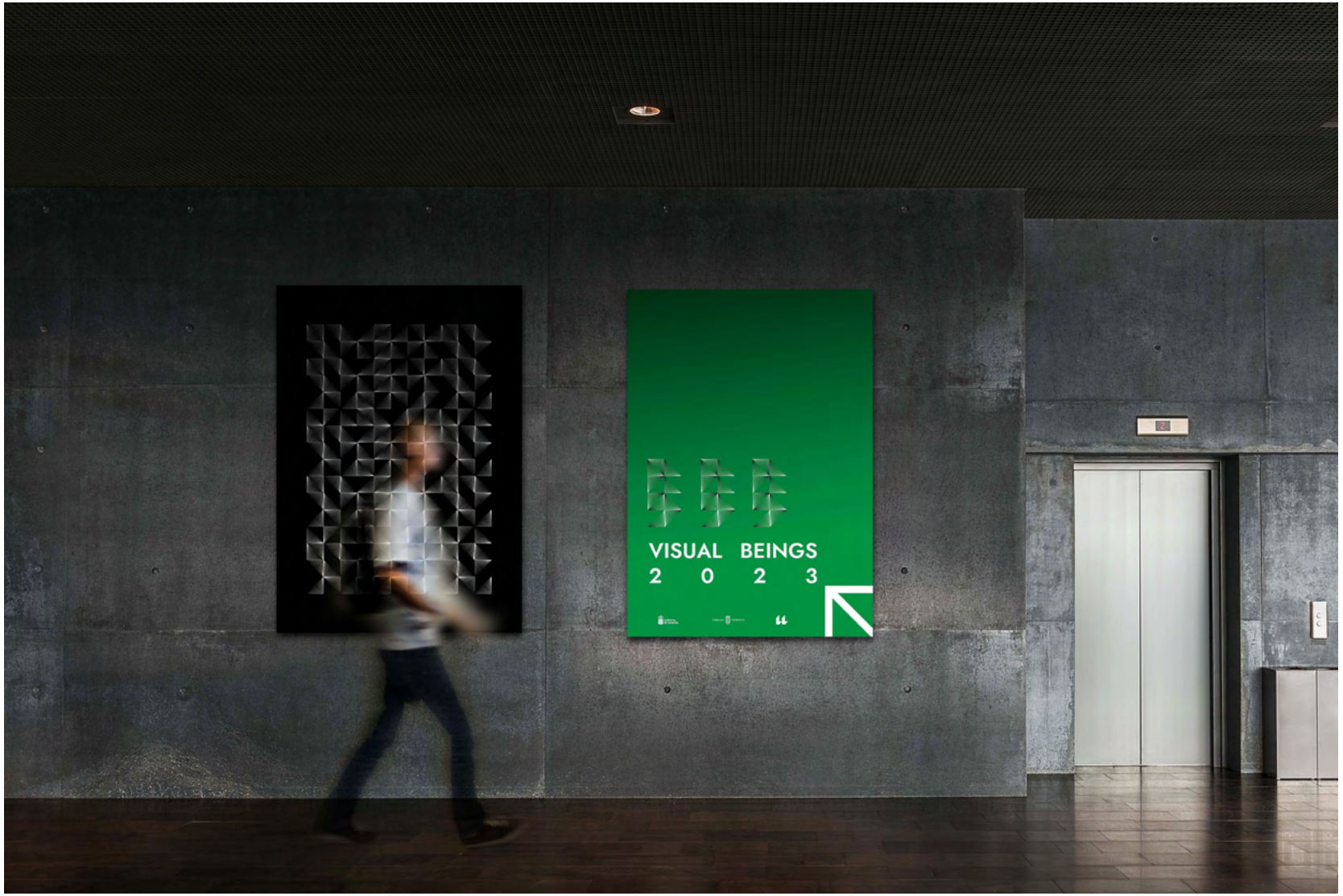
Primera propuesta de póster (Exterior)

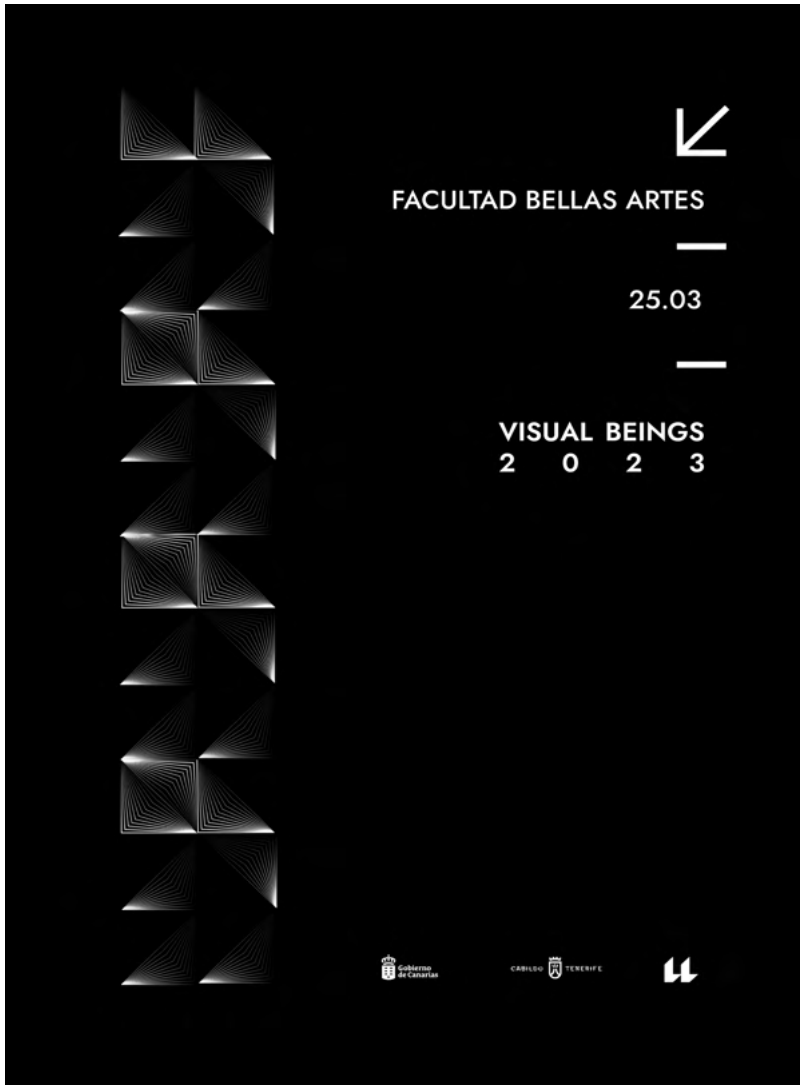

VISUAL BEINGS
2 0 2 3
I Festival de Diseño y Tecnología Canarias
CABILDO TENERIFE
Universidad de La Laguna





Segundas propuestas de póster. (Interior)
(Diseñados para estar juntos)


VISUAL BEINGS
2 0 2 3



Propuesta carteleria en Mupi (Exterior)

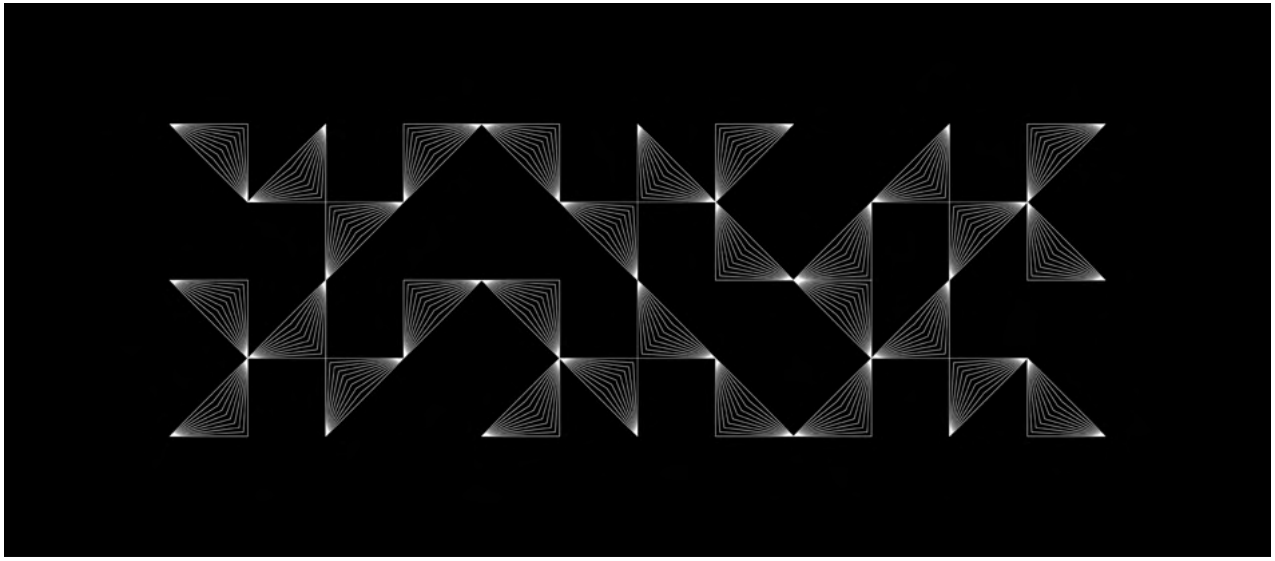

FACULTAD BELLAS ARTES
25.03
VISUAL BEINGS
2 0 2 3

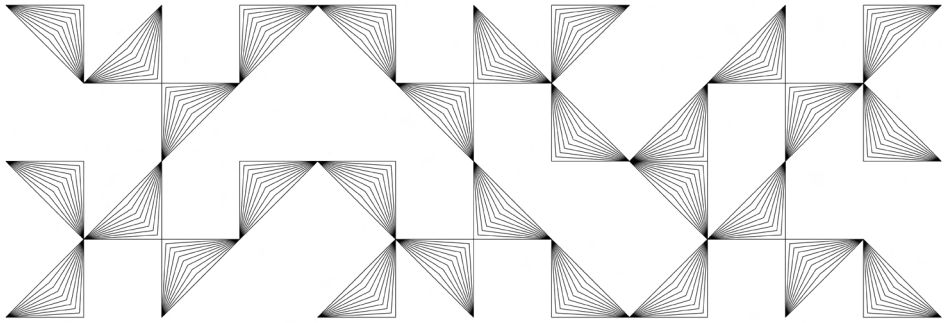
Propuesta vinilo horizontal.


VISUAL BEINGS
2 0 2 3
I Festival de Diseño
y Tecnología de
Canarias

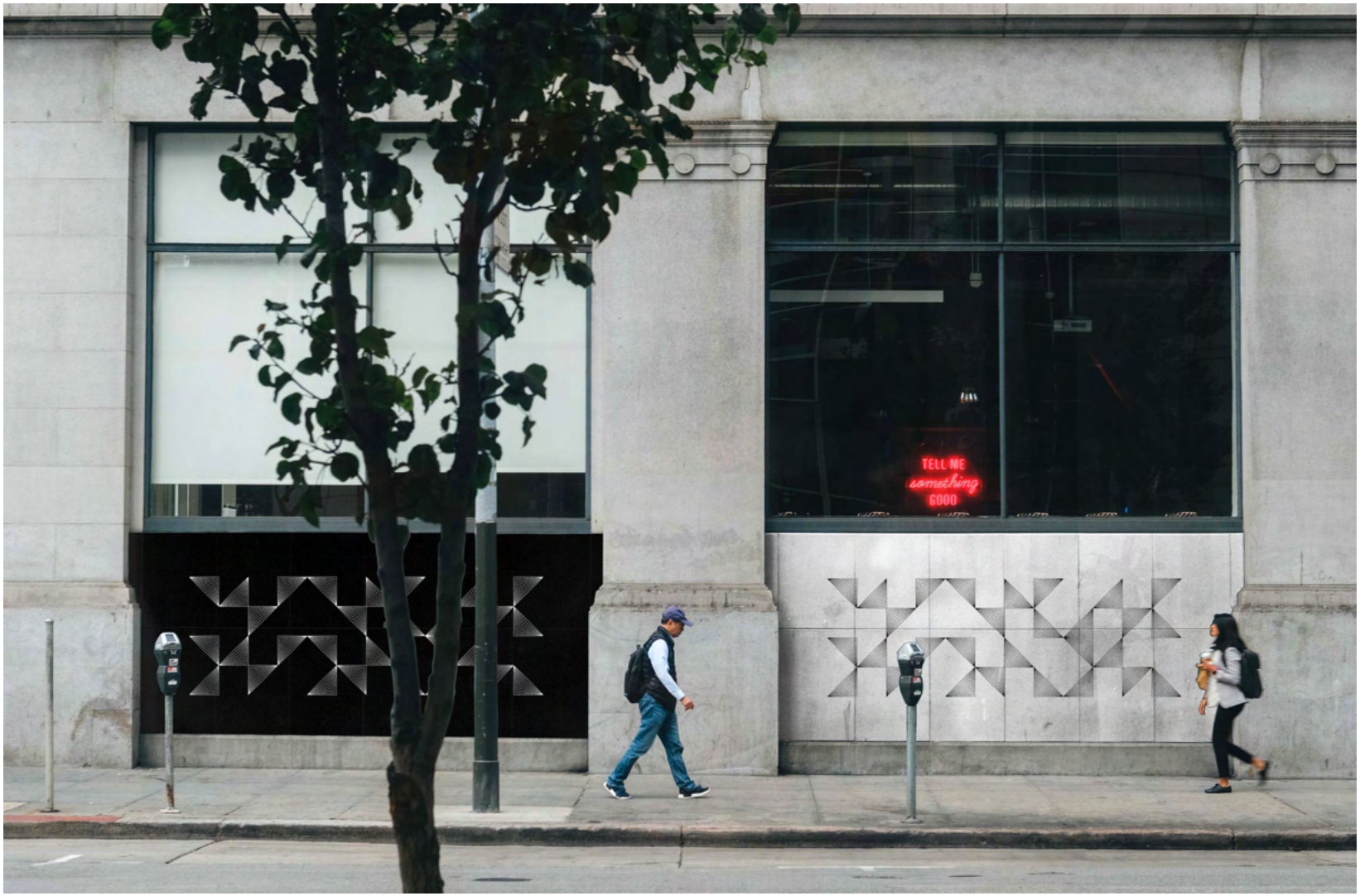
Propuesta Indicaciones (Interior)

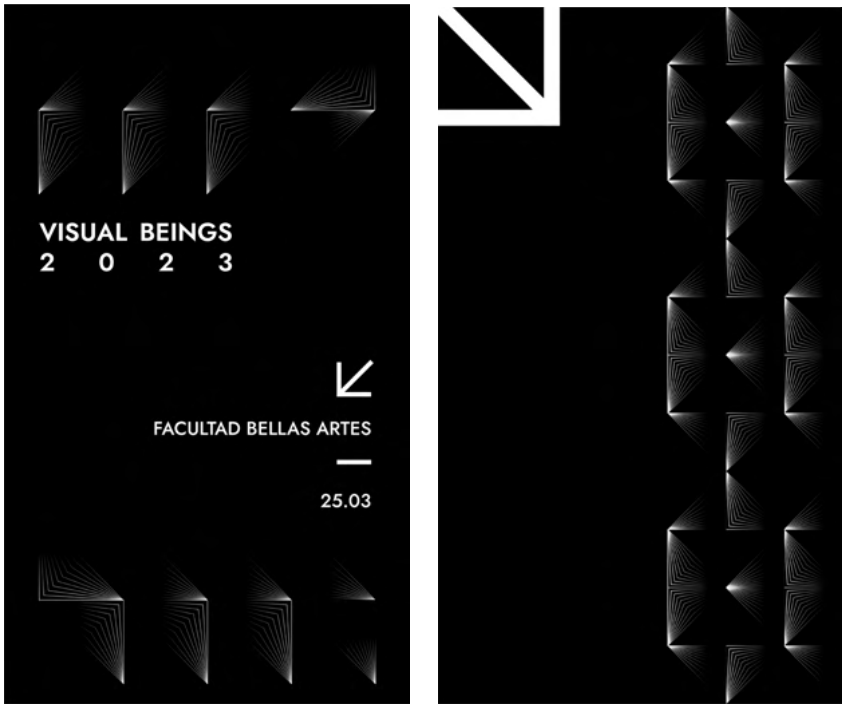

03 ↗
—
VISUAL
BEINGS



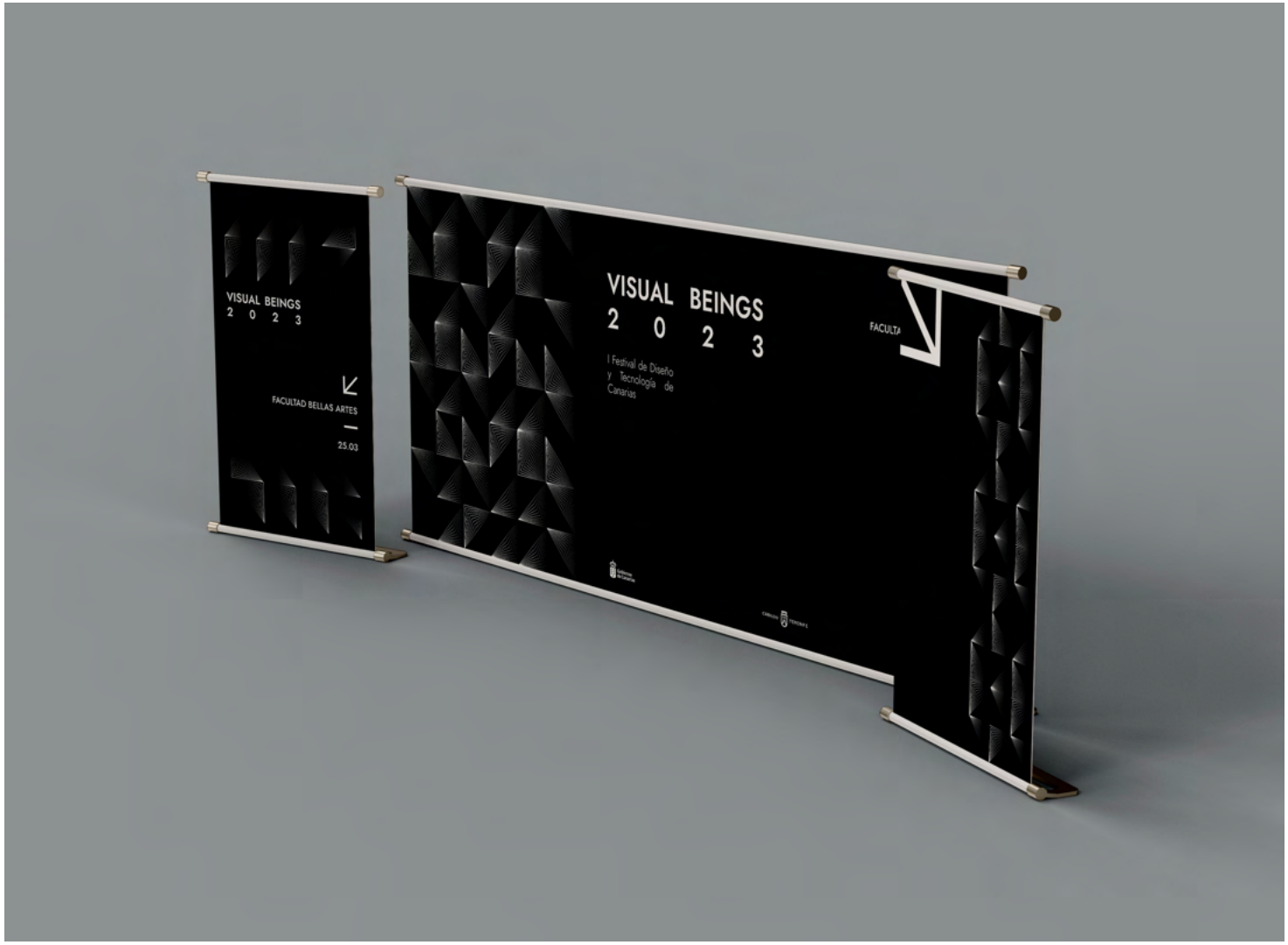


Propuesta cartelería. (Exterior)


TELL ME
something
GOOD



Propuesta de banners verticales y horizontales.
(Interior)


VISUAL BEINGS
2 0 2 3
FACULTAD BELLAS ARTES
25.03
VISUAL BEINGS
2 0 2 3
I Festival de Diseño y Tecnología de Canarias
FACULTA

2.2 Digital

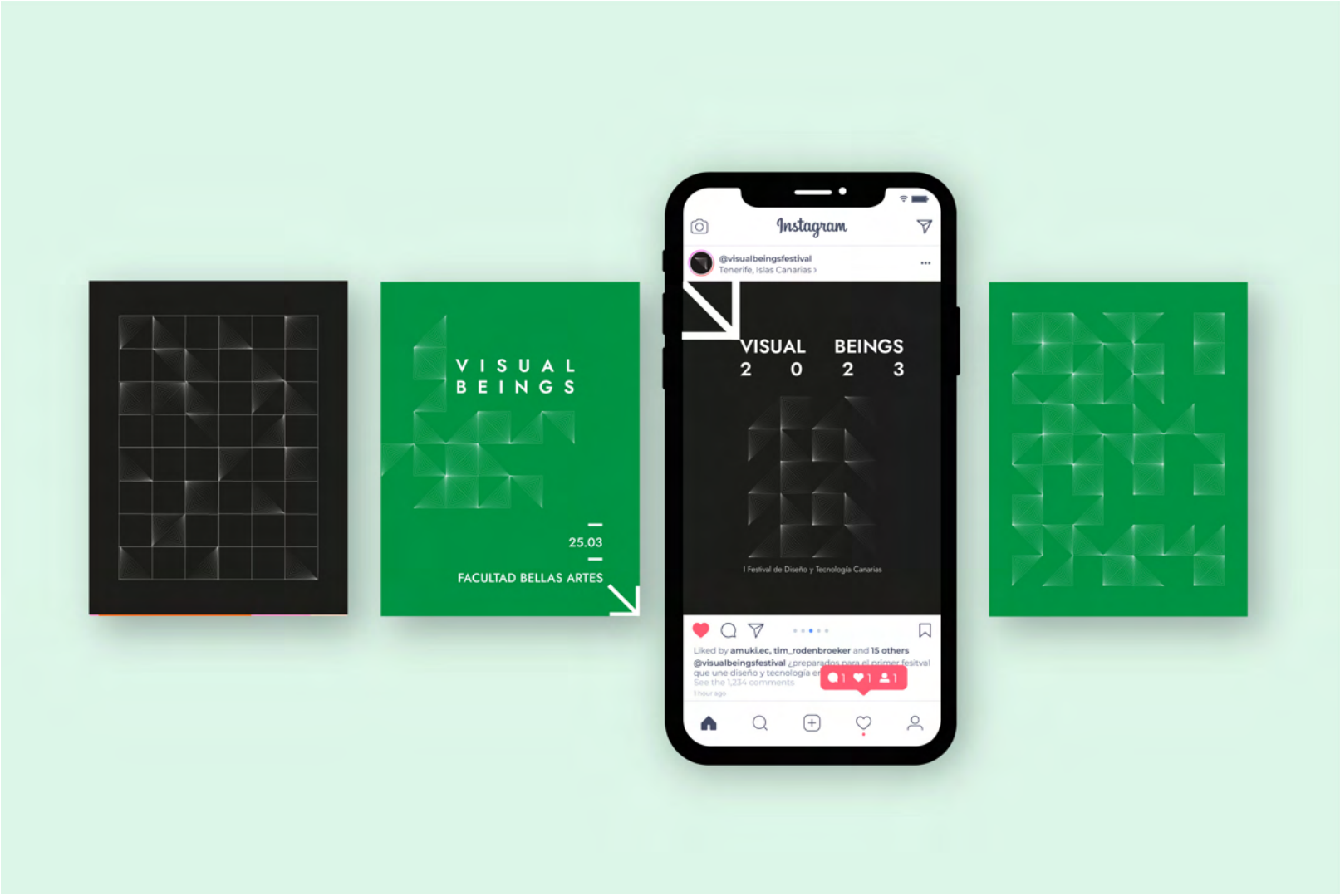
Además de la promoción en formato analógico, se han diseñado contenidos digitales también derivados de esos activos y componentes modulares iniciales.

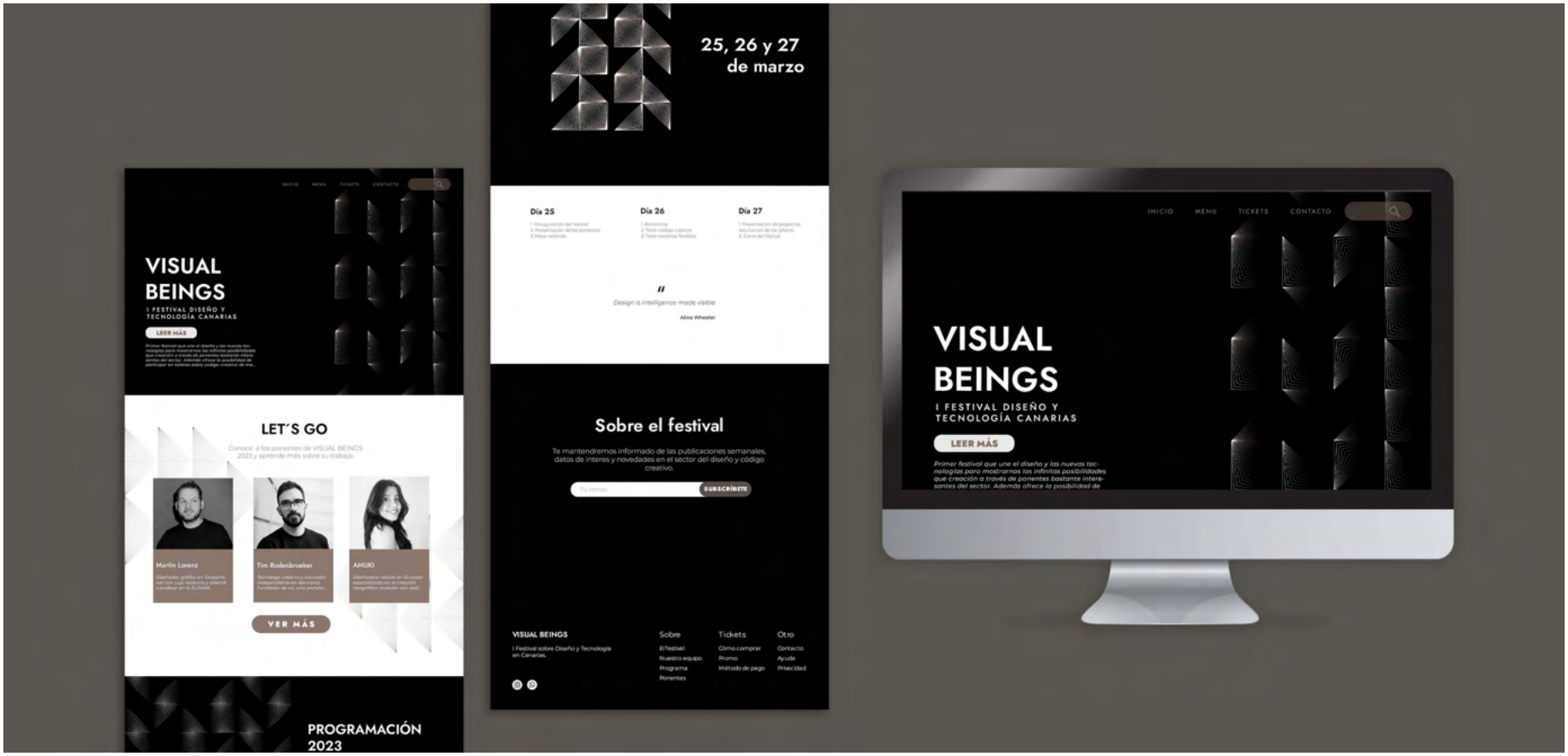
Todas estas propuestas son nuevamente “cero restrictivas” y ofrecen una infinidad de combinaciones.

Primero se creó un perfil en Instagram ,ya que la mayoría de nuestro público objetivo son estudiantes , y además porque es la red social casi por excelencia entre la juventud.

Se diseñaron publicaciones informativas, pero también animaciones realizadas con código creativo que muestra las posibilidades del diseño modular.

18. Los vídeos realizados para el feed de instagram se podrán ver en el siguiente enlace:
- https://drive.google.com/drive/folders/1HqTMHELZi-5fHEZggKOZEqpaEiH1J-bM-V?usp=sharing


VISUAL
BEINGS
25.03
FACULTAD BELLAS ARTES
Instagram
@visualbeingsfestival
Tenerife, Islas Canarias
VISUAL BEINGS
2 0 2 3
I Festival de Diseño y Tecnología Canarias
Liked by amuki.ec, tim_rodenbroeker and 15 others
@visualbeingsfestival ¿preparados para el primer fesitval
que une diseño y tecnología
See the 1,234 comments
1 hour ago



Además de la página de instagram se elaboró una web, donde se puede encontrar información del festival más detallada, información sobre los ponentes, y más animaciones e ilustraciones hechas específicamente para la web.

3. DISEÑO DE UNA HERRAMIENTA INTERACTIVA CON PROCESSING

3.1 Estudio del programa

La parte más interesante de este trabajo de fin de grado comienza con el estudio de este programa.

Previamente hemos puesto en práctica la modularidad diseñando una identidad flexible para nuestro festival. Es ahora cuando nos adentraremos en las posibilidades que nos ofrece el código creativo.

Processing, como en un principio analizamos en la investigación del trabajo, se trata de un programa que funciona como cuaderno de bocetos de software flexible basado en el lenguaje Java Script que nos permite diseñar a partir de código.

Mi proceso de aprendizaje fue fijar un objetivo, y comenzar a buscar formas de lograrlo.

En este caso el objetivo era la creación de una herramienta que permitiera que nuestra identidad fuera interactiva.

Me subscribí en Patreon, donde Tim Rodenbröeker, uno de los referentes en este ámbito, al que también hago referencia en la investigación, comparte sus conocimientos sobre este programa de forma didáctica y ofrece cursos sobre los esenciales de Processing.

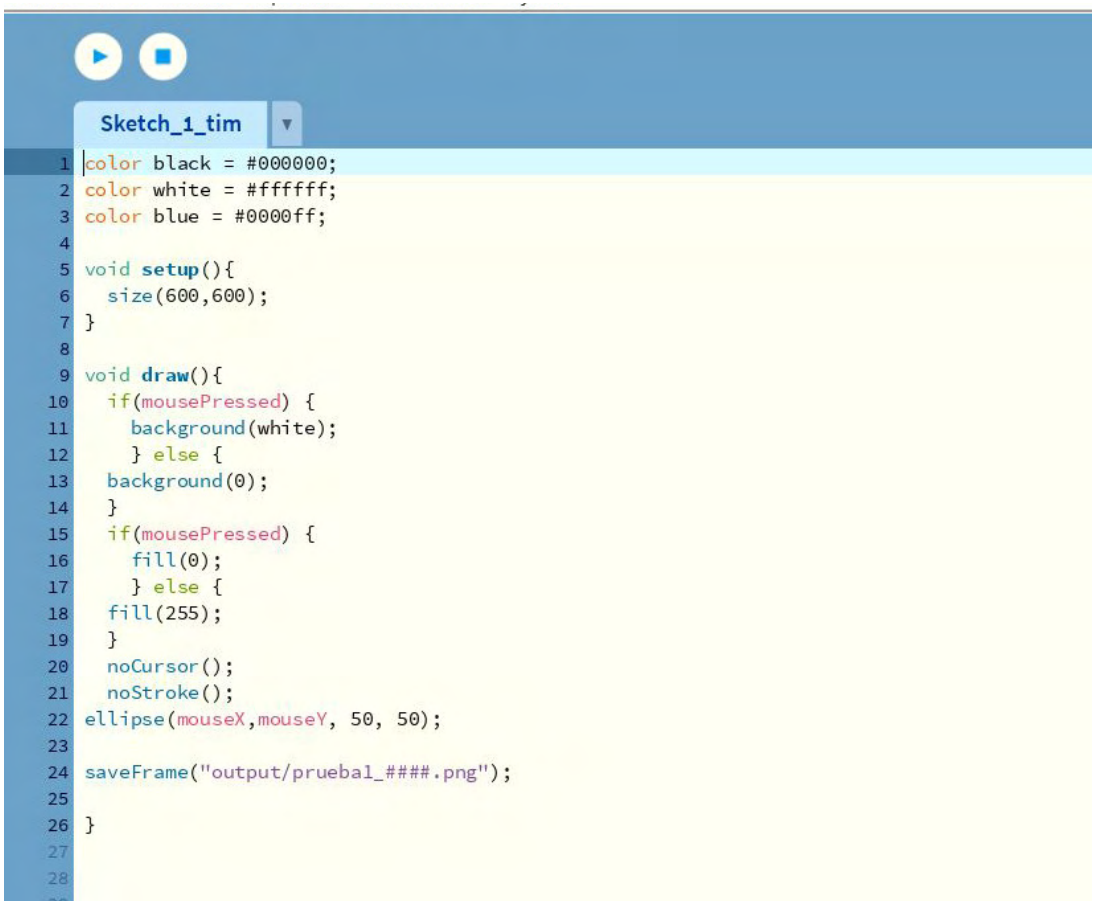
Comencé experimentando en el programa con módulos más sencillos que el mio. Estos fueron un circulo y posteriormente un cuadrado.

Processing funciona de la siguiente manera: cualquier Sketch debe contener la funcion de *"voidsetup()"* y *"void draw()"*

En *void setup()* es donde se escriben todos los tecnicismos de nuestro sketch (proyecto). Es decir, las dimensiones etc
En *void draw()* escribiremos todo lo que queremos que el programa diseñe.

Cuando ejecutamos nuestro sketch, el programa ejecuta nuestro código una y otra vez hasta que lo detengamos. Es por esto que el orden en el que escribamos nuestro código es importante y además si no queremos que el programa nos dibuje una cosa sobre otra, debemos añadir un *background()* al principio de nuestro *void draw() loop*.

A la hora de crear nuestra geometría, debemos antes setear la función *fill()* y posteriormente crear nuestra forma, por ejemplo *ellipse (x1, y1, sizeX, sizeY)* para que sea visible.



También podemos quitarle el trazo a las formas escribiendo *noStroke()* antes de escribir nuestra geometría.

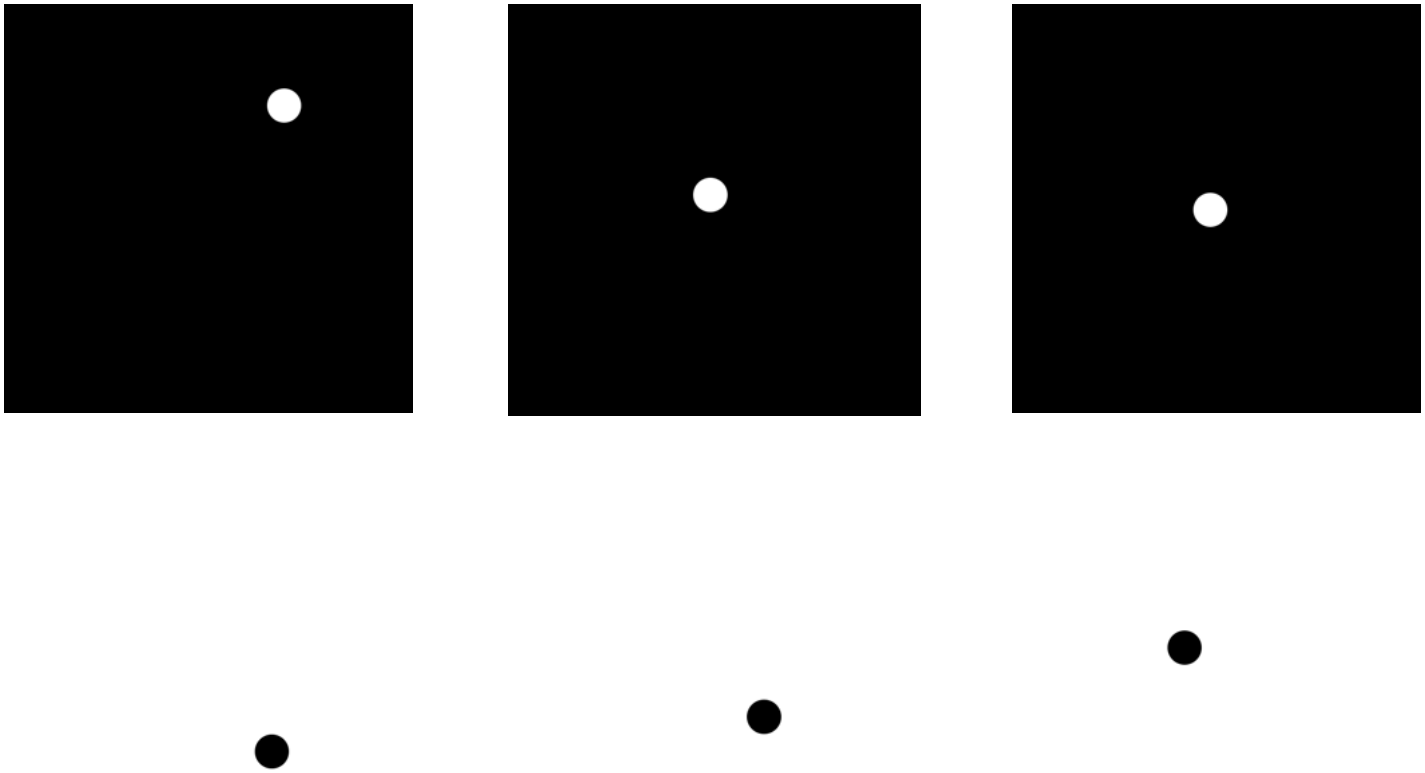
Fuera de nuestras funciones de *void setup()* y *void draw()* podemos definir variables que nos puedan ser útiles para facilitar la compresión del código o el buen funcionamiento, como en este ejemplo están definidos los colores.

Con esta primera información comencé a probar el programa.

En este primer ejemplo, diseñé una elipse blanca de 50 x 50 px (sin trazo), que se moviera siguiendo el movimiento del ratón sobre un fondo negro, y que además, cuando el usuario presionara el ratón, los colores de estos se invirtieran.

Para ello escribí dos condicionales que definían uno, que cuando se estuviera presionando el ratón el fondo fuera blanco, y en caso contrario, fuera siempre negro, y el otro que cuando se presionara el ratón, nuestra elipse fuera siempre negra, y en el caso contrario, blanca.

Para guardar esta animación interactiva, creé esa función de *saveFrame ()* que básicamente desde que el código se ejecuta, el programa cada vez que lee el código nos guarda un .png de lo que ocurre. De esta forma, hasta que no detenemos la grabación, el programa nos guarda cada frame de lo que ocurre, que nosotros posteriormente uniremos con la herramienta “creador de películas” de el propio programa y obtendremos un archivo .GIFF o .mp4



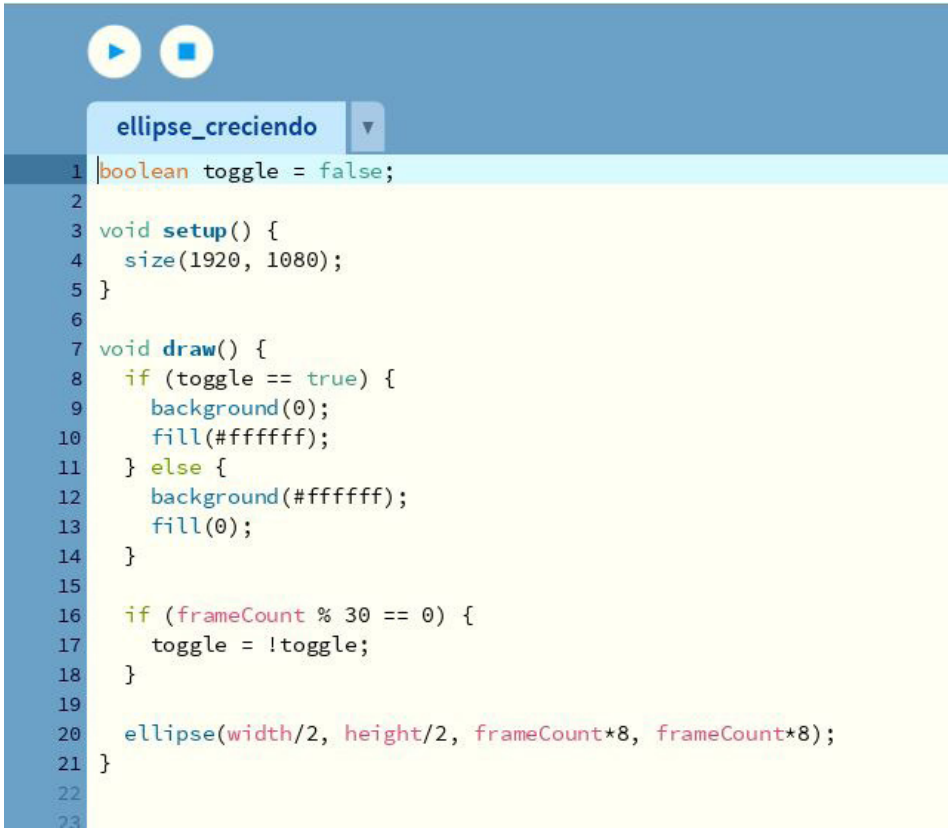
19. Esta primera animación se podrá ver en el siguiente enlace bajo el nombre “prueba 1”
- https://drive.google.com/file/d/1MwP37BCAU575V-DFxgGvm6DJv2Ln5MknF/view?usp=sharing

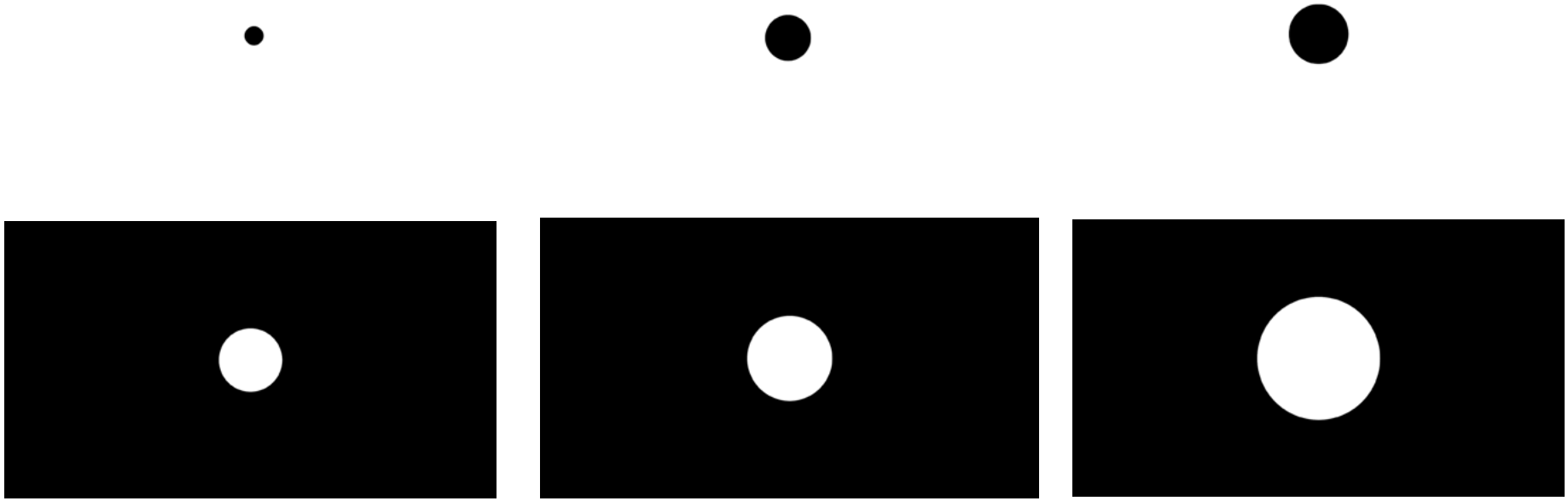
Continué escribiendo nuevos códigos muy simples y probando cosas nuevas.

Esta vez, usando también una elipse, cree una animación donde esta elipse va creciendo y además cambiando de color de blanco a negro. Además, si la elipse es blanca el fondo se vuelve negro y si la elipse es negra, el fondo blanco.

En este caso hemos hecho uso de *booleanas*, que son variables de verdadero o falso. Con esta variable (que en ejemplo hemos llamado *toggle*) hemos creado un condicional que pasados *x frames* de ejecución del programa va cambiando el valor de nuestra variable *toogle* de verdadero a falso.

Partiendo de este valor cambiante de nuestra variable *booleana*, creamos un condicional en el que si *toogle == true* el fondo será negro y la elipse blanca, y por el contrario si *toggle == false*, los colores se invertirán.



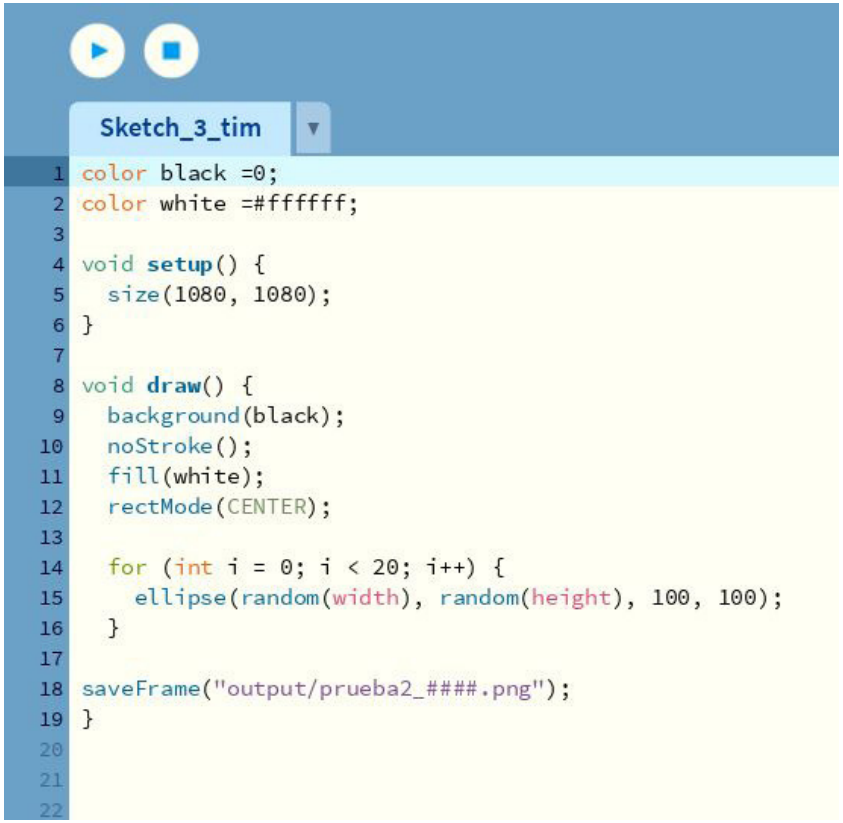


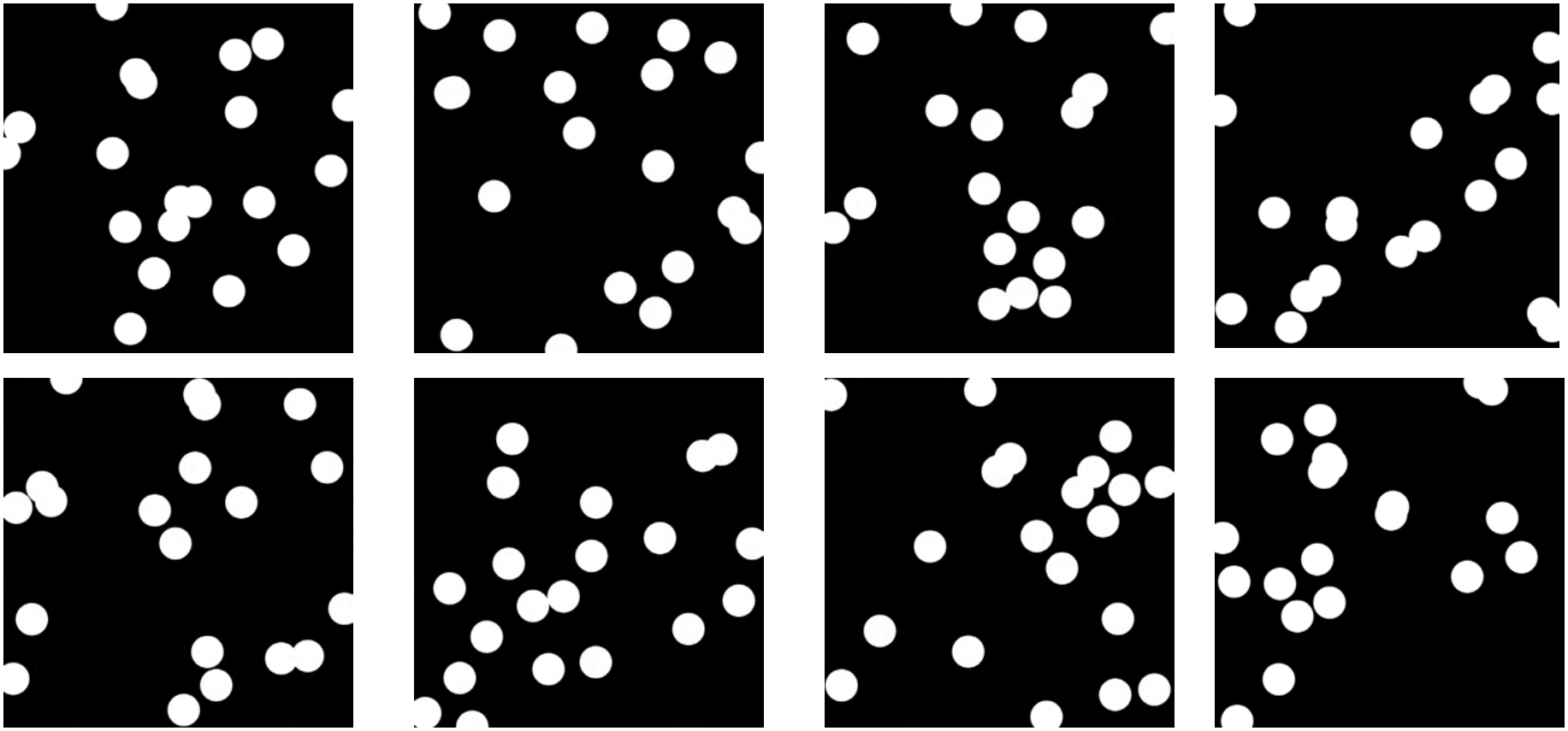
20. Esta animación se podrá ver en el siguiente enlace bajo el nombre “prueba 2”
- https://drive.google.com/file/d/1NI9k5LPnfFqbmr-BUVemP_K2uCEQJO1yC/view?usp=sharing

Mi siguiente prueba fue el uso de los *"for Loops"* que se escriben cuando queremos crear dos o más figuras iguales. Básicamente, controla una secuencia de repeticiones.

Una estructura básica de loop tiene tres partes: inicialización, prueba y actualización. Cada parte debe estar separada por un punto y coma (;). El ciclo continúa hasta que la prueba se evalúa como falsa.

En este caso, la variable i de nuestro *for loop* comienza inicializada en 0 (inicialización), en el centro, con *i<20* indicamos que el *loop* se va a seguir repitiendo hasta que la variable i sea igual o mayor que 20. (prueba) Con *i++* hacemos que cada vez que vuelva a empezar el *loop* se sume uno a la variable i, que en este caso irá creciendo hasta llegar a 20, que es el número de elipses que queremos crear.





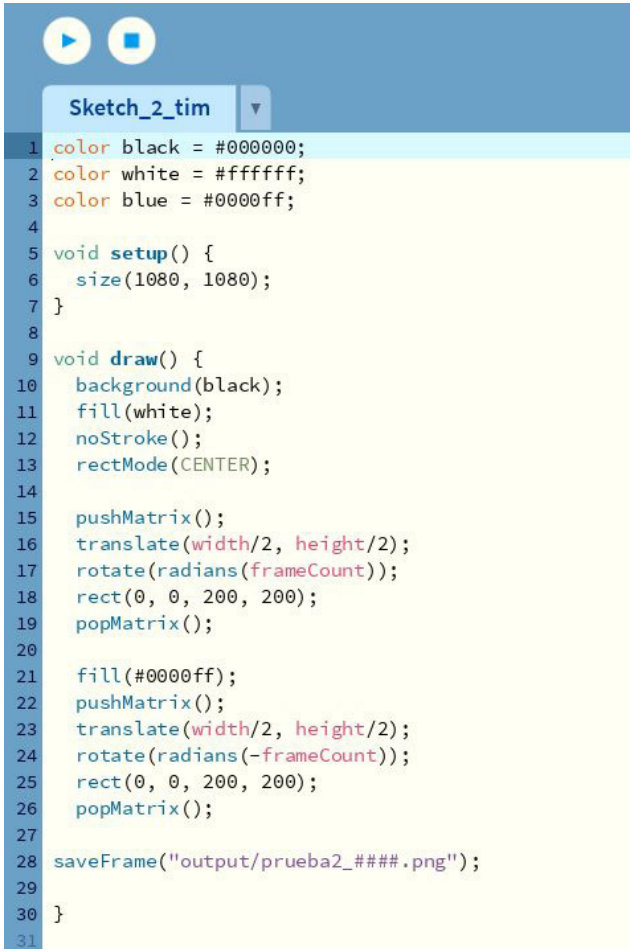
21. Esta animación se podrá ver en el siguiente enlace bajo el nombre “prueba 3”
- https://drive.google.com/file/d/19tSv5h5TzDiD-2T51Wka197ozQFyK1RlX/view?usp=sharing

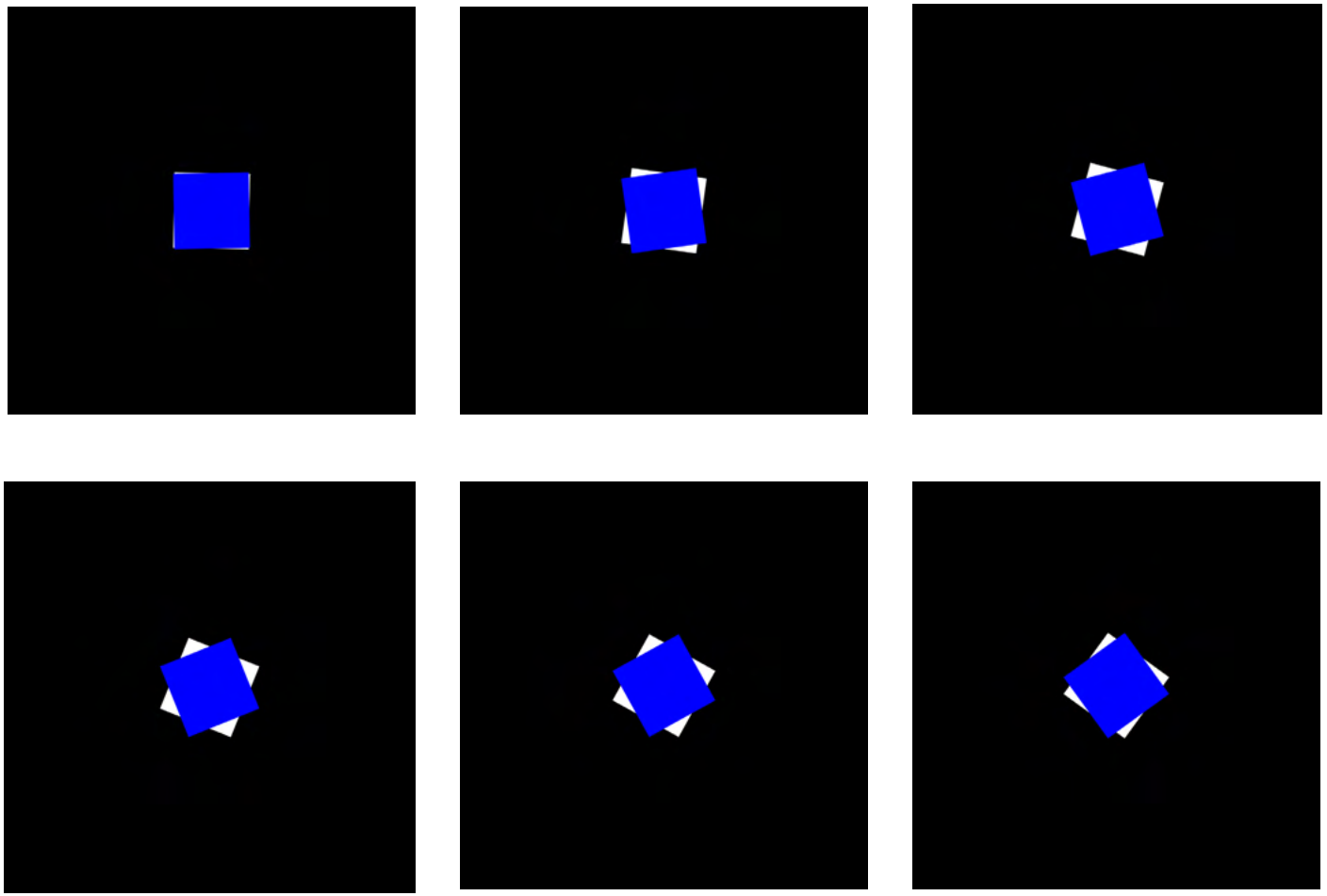
Para la siguiente prueba decidí cambiar de módulo y utilizar un cuadrado.

Quise aprender a utilizar las matrices *pushMatrix() y popMatrix()* que entran en juego cuando queremos realizar más de una transformación en nuestro sketch, para aislarlas.

En este caso, mi idea era generar dos cuadrados, y rotar uno en el sentido de las agujas del reloj, y el otro, que estaría colocado justo encima del anterior, hacia el otro lado.

Este código es bastante simple pero si se trabaja con las matrices de manera más extensa nos permite generar animaciones bastante complejas.





22. Esta animación se podrá ver en el siguiente enlace bajo el nombre “prueba 4”
- https://drive.google.com/file/d/1BWq05DNC-flHFL-DzlNOUYELURYjghtFh/view?usp=sharing

Mi siguiente prueba fue el intento de diseño de un patrón circular, creando elipses sin relleno pero con trazo bastante grueso, con diámetros en aumento. De manera que el resultado final fuera un "patrón" circular.

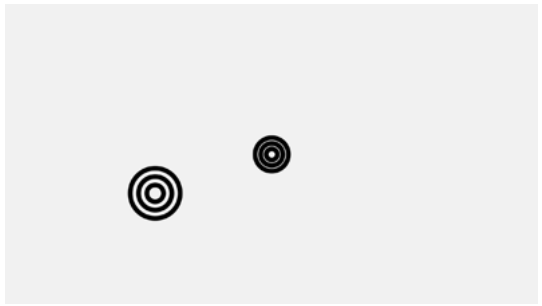
Estos "patrones" irán aumentando hasta llegar a x tamaño (que viene definido por frameCount 24), y además, uno de ellos será interactivo, moviéndose según el ratón. El otro estaría colocado justo en el centro de nuestro sketch, creciendo de igual manera pero estático y además con un diámetro menor.

Para esto use de nuevo los *for loops()* La idea era obtener esa repetición de elipses que configuraran un patrón final.

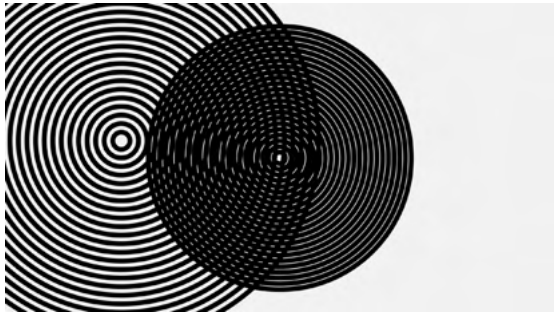
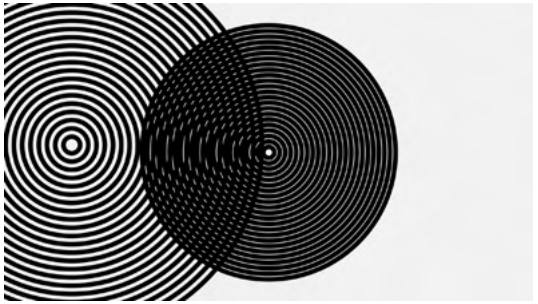
Hay que tener en cuenta que al ser diseñadores y no informáticos, lo importante en estos procesos de escribir código es que entiendas la funcionalidad y el cómo sucede, pues existen repositorios infinitos de códigos ya diseñados que podemos modificar a nuestro gusto sin necesidad de entender en profundidad los mismos, y es justo esta la ventaja de Processing.

Patron_circulos

```
1  color fg=0;
2  color bg=#f1f1f1;
3
4  void setup() {
5    size(1920, 1080);
6  }
7
8  void draw() {
9    background(bg);
10   stroke(fg);
11   noFill();
12   strokeWeight(17);
13   rectMode(CENTER);
14
15   for (int i = 0; i < constrain(frameCount, 1, 24); i++) {
16     ellipse(width/2, height/2, i*40, i*40);
17     ellipse(mouseX, mouseY, i*60, i*60);
18   }
19   saveFrame("output/prueba2_####.png");
20 }
```



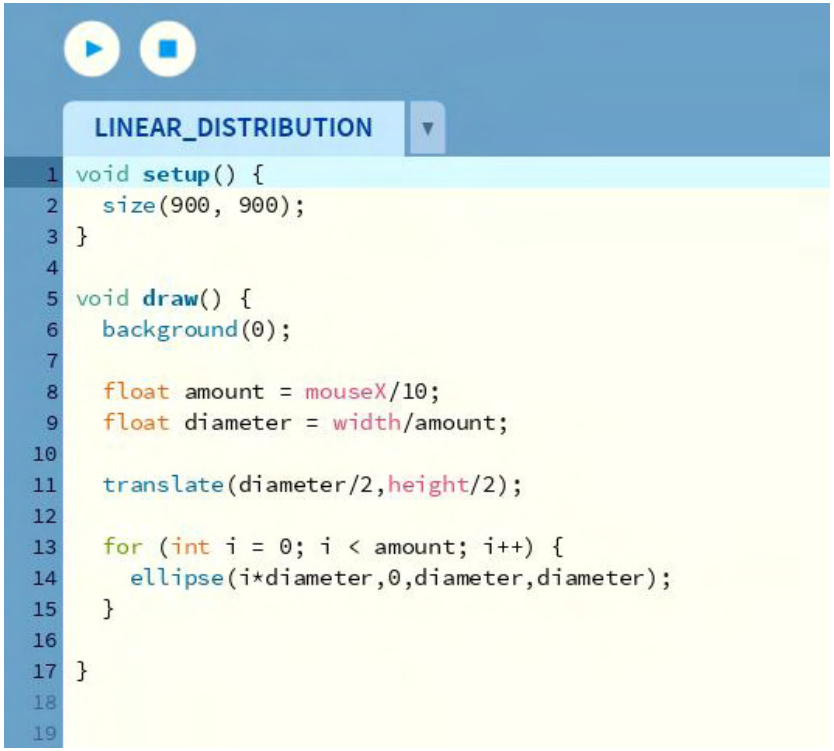


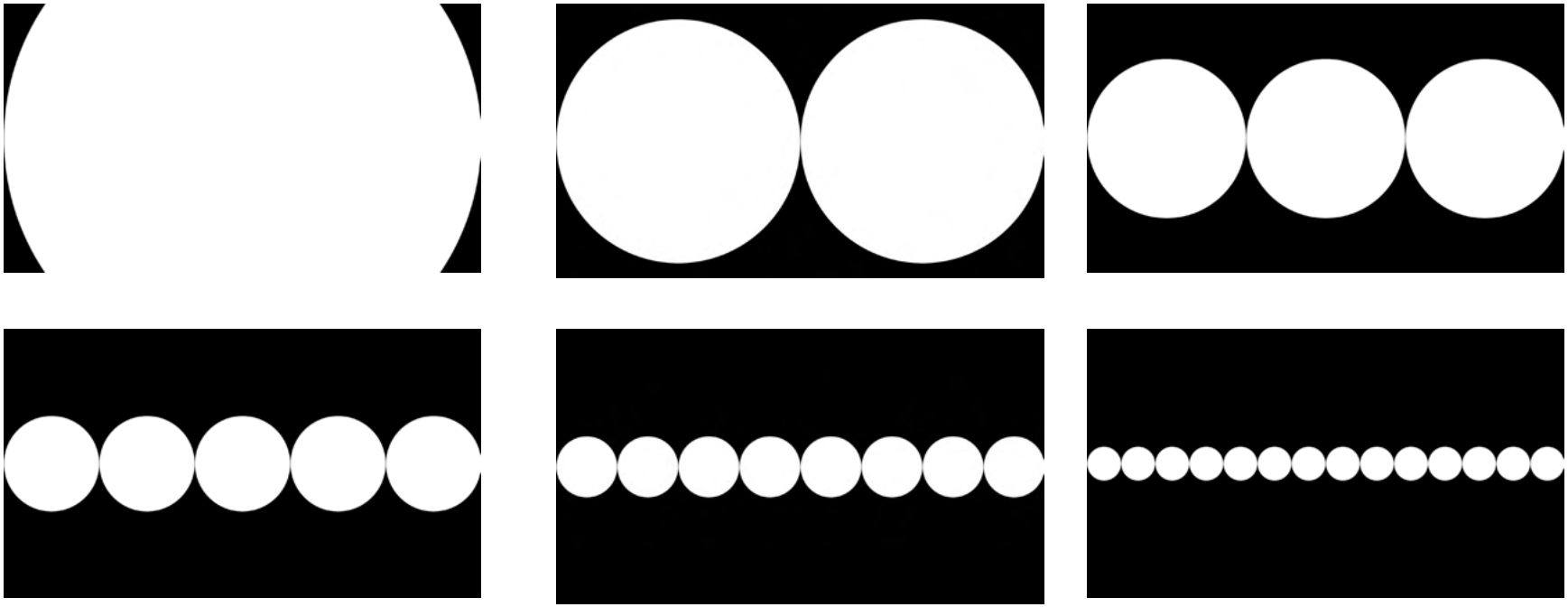



23. Esta animación se podrá ver en el siguiente enlace bajo el nombre “prueba 6”
- https://drive.google.com/file/d/11LDsqf7Mut-Fuuol7wDY_TB4hc4xb4bDC/view?usp=sharing

Avancé hacia construcciones más complejas visualmente, aunque el código sigue siendo bastante sencillo.

En este caso, quise hacer una distribución linear de elipses a lo largo del eje X que fuera de ancho a ancho, y que el número de elipses se adaptara según el movimiento del ratón en el eje X también, de manera que si mi ratón está más cerca del lado izquierdo de mi sketch, hay menos elipses, y si se mueve hacia el lado derecho, hay más, todas ellas distribuidas una junto a la otra en línea en el eje X.





24. Esta animación se podrá ver en el siguiente enlace bajo el nombre “prueba 7”
- https://drive.google.com/file/d/1h3i7IoJKYIKown6a-RIt-vWPeMzMtiV_O/view?usp=sharing

Diseñé más propuestas con código un poco más complejo basados en sistemas de cuadrículas, pero como dije anteriormente, lo importante como diseñadores es entender de que manera podemos modificar nosotros códigos ya existentes, y además entender sobre todo la funcionalidad de los mismos.

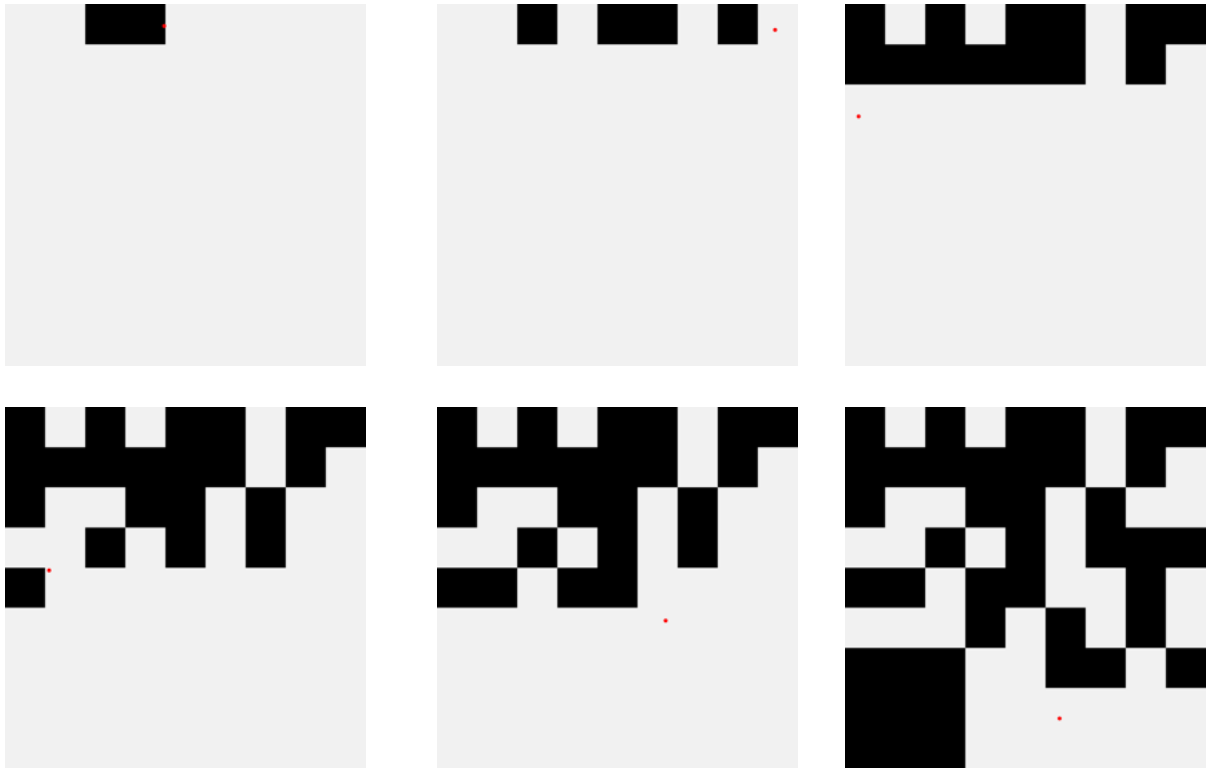
Al final, este programa es de aprendizaje autodidacta, basado en el ensayo error.

Este último ejemplo de animación es el resultado de esto mismo que comento anteriormente. No se realizó escribiendo el código de 0, si no modificando las variables.

Se trata de una aplicación interactiva de construcción de cuadrícula a partir de cuadrados que se mueven según el movimiento del ratón.
.

grid2

```
boolean[][] state;
color FG = #F1F1f1;
color BG = #000000;
int mx, my;

void setup() {
  size(900, 900);
  state = new boolean[9][9];
  for (int x = 0; x < state.length; x++) {
    for (int y = 0; y < state[x].length; y++) {
      state[x][y] = false;
    }
  }
}

void draw() {
  background(BG);
  noStroke();

  mx = int(map(mouseX, 0, width, 0, state.length));
  my = int(map(mouseY, 0, height, 0, state[0].length));

  for (int x = 0; x < state.length; x++) {
    for (int y = 0; y < state[0].length; y++) {
      if (state[x][y] == false) {
        fill(FG);
      } else {
        fill(BG);
      }
      rect(x*100, y*100, 100, 100);
    }
  }

  fill(#FF0000);
  ellipse(mouseX, mouseY, 10, 10);
}

void mouseReleased() {
  state[mx][my] = !state[mx][my];

  saveFrame("output/prueba2_####.png");
}
```



25. Esta animación se podrá ver en el siguiente enlace bajo el nombre “prueba 8”
- https://drive.google.com/file/d/1uffV4wM5XV2lr-5nw4h3mY8o-Y-0KmsO6/view?usp=sharing

3.2 Método AMUKI

Una vez realizadas todas estas pruebas siguiendo las directrices de Tim Rodenbroeker, llegué a la conclusión de que para crear una herramienta interactiva que mostrara la versatilidad de mi módulo, que era nuestro objetivo inicial, no eran quizás las más ideales, sobre todo porque los códigos no estuvieron pensados en un principio para mi módulo.

Continué investigando y descubrí un código escrito por AMUKI, otra de las referentes de las que hablo en mi investigación.

Ella también es diseñadora, especializada en el diseño tipográfico modular, y ofrece cursos de código creativo con Processing para enseñar esto mismo.

El código que conseguí, es uno creado para diseño tipográfico modular e ilustración basado en un sistema de cuadrículas.

Este código es bastante interesante porque nos permite “dibujar” sobre un lienzo donde se muestra además una cuadrícula.

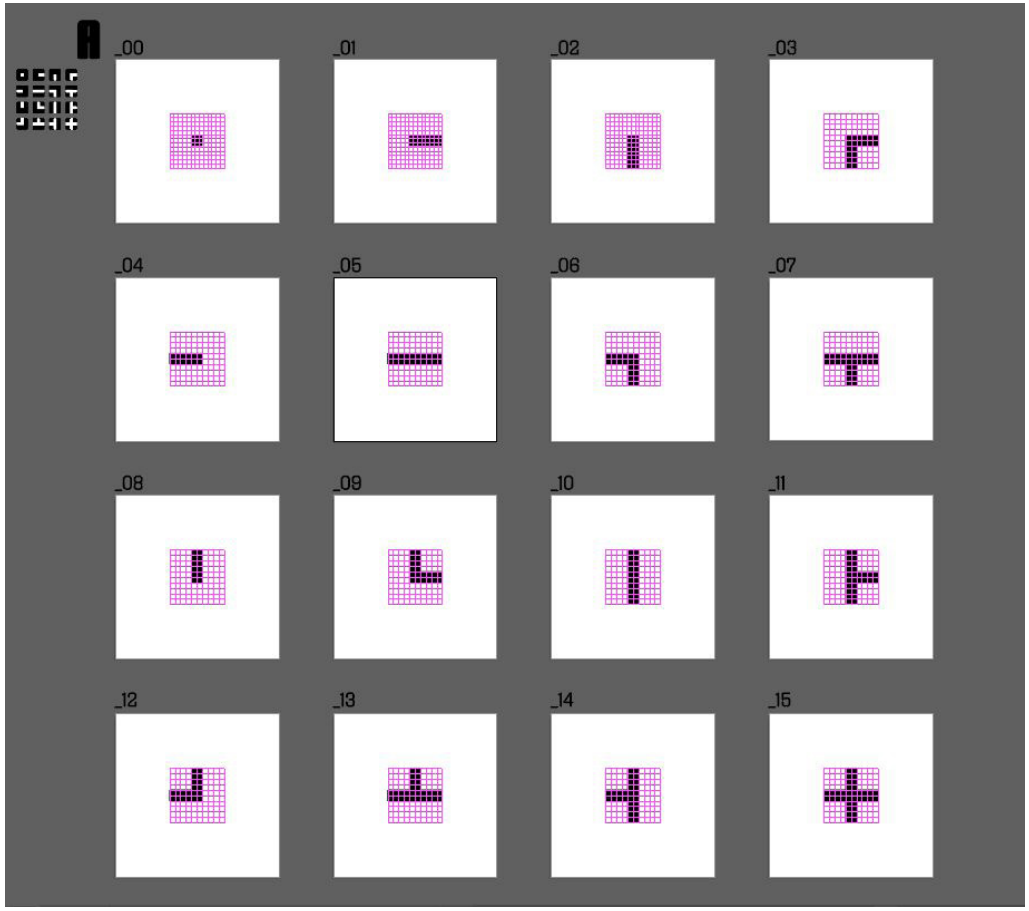
Decidí probar el código de AMUKI tal y como había sido diseñado, de nuevo con módulos más simples que el mio, para lograr entender la funcionalidad del mismo y después poder adaptarla a mi propuesta.

Este código requiere un trabajo previo de Adobe Illustrator, donde se diseñan ocho plantillas de módulos con 15 variaciones de cada módulo por plantilla.

Esto es así porque el código de AMUKI está diseñado para diseño tipográfico, por lo que a la hora de dibujar en el sketch, cada espacio de la cuadrícula está condicionado por lo que se dibuje a su alrededor, para así poder conformar letras.

La imagen que vemos a la derecha es una de las ocho plantillas previamente diseñadas en ai.

Estas plantillas van numeradas de la A a la H. La imagen muestra los módulos que diseñé para la plantilla A.

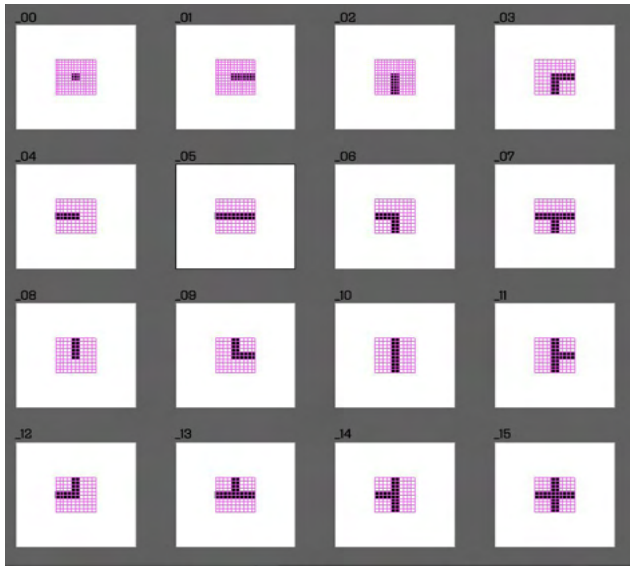


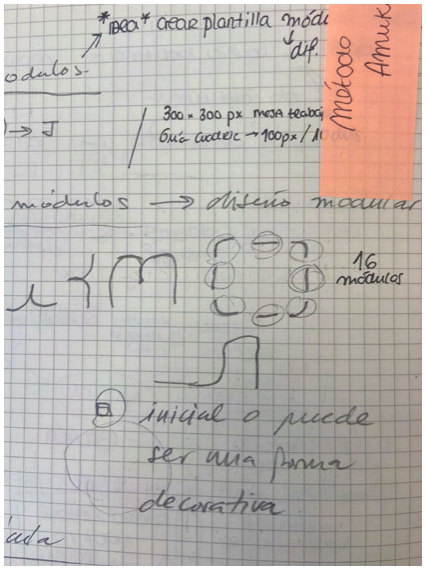
En estas plantillas deben haber siempre las mismas variaciones de módulos, es decir, debe haber siempre un módulo horizontal, otro vertical, otro curvo hacia la derecha, otro curvo hacia la izquierda, y así hasta conformar las 15 variaciones necesarias para poder diseñar tipografía o ilustración.

En la imagen de la derecha vemos un pequeño esquema de lo anterior comentado.

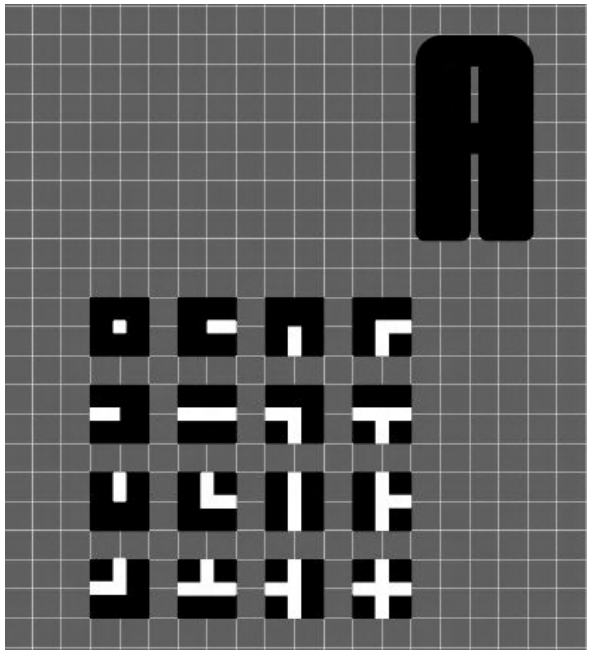
Mostraré tres de las ocho plantillas que diseñé para probar el método AMUKI. La plantillas A, B y C.

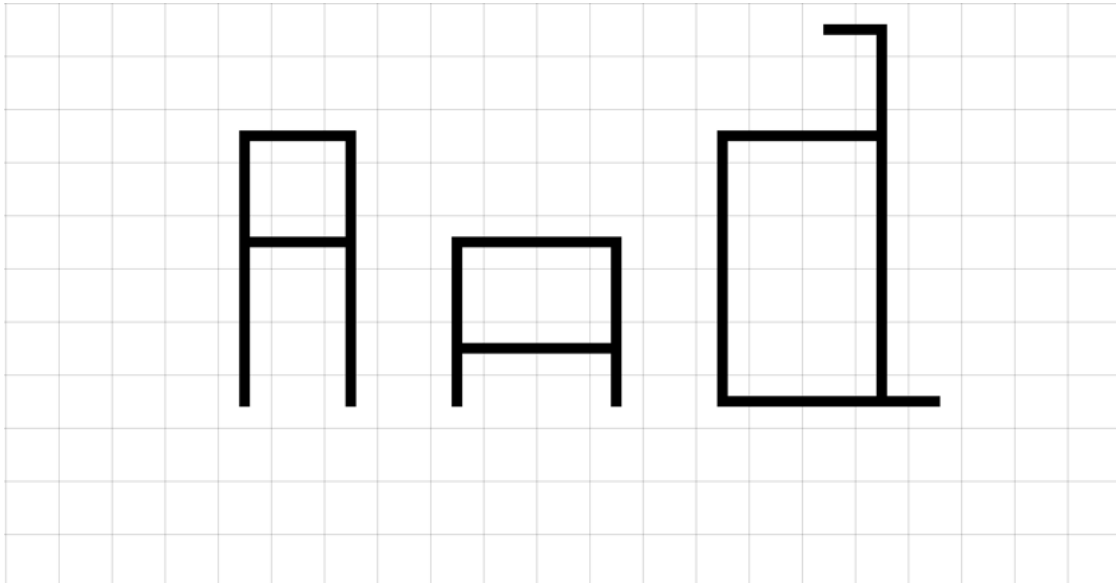
Son plantillas bastante sencillas cuya finalidad es conseguir entender este método a la perfección.





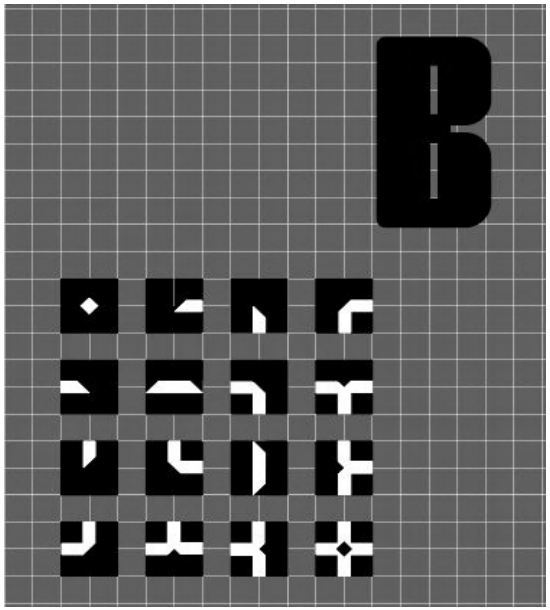
La plantilla A es una variación de un módulo bastante simple que es el cuadrado. A partir de la repetición de un cuadrado bastante pequeño obtenemos esta apariencia de trazo recto.

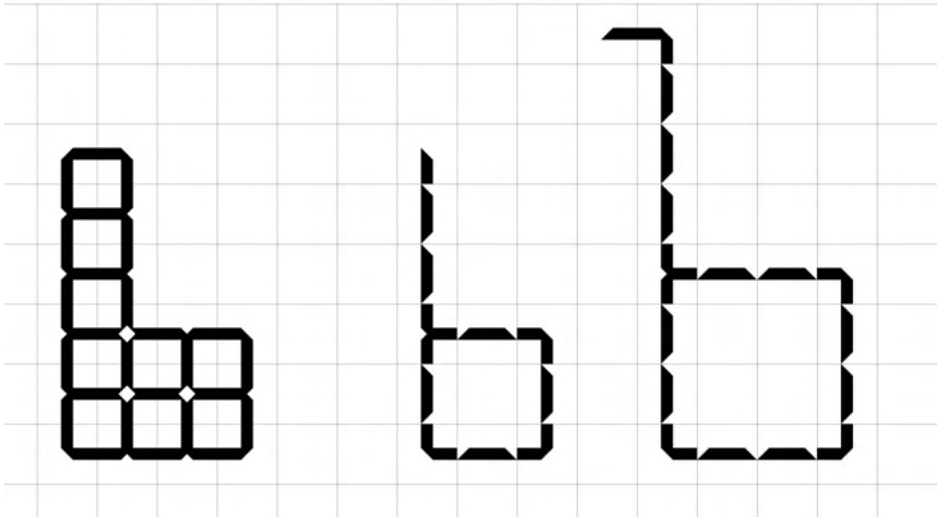




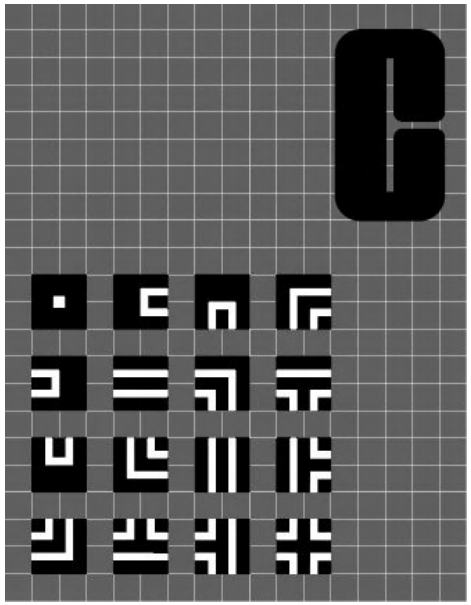
La plantilla B parte del mismo cuadrado que la plantilla A, pero las terminaciones acaban siempre con medio cuadrado.

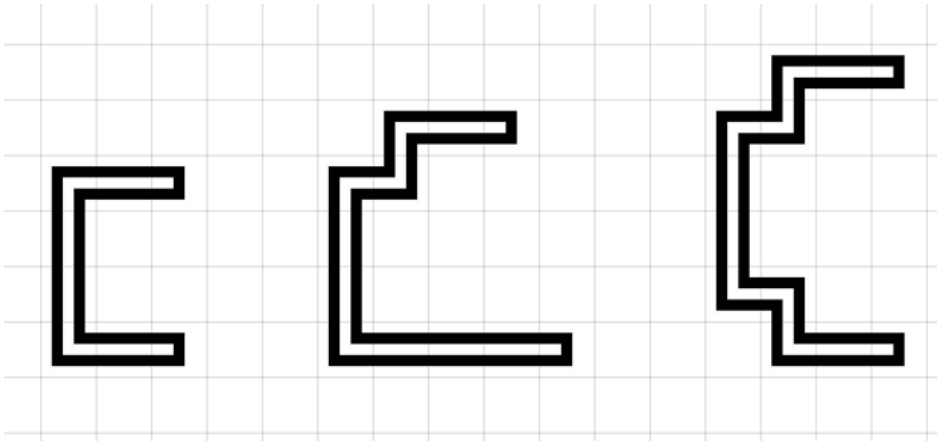
De esta manera obtenemos terminaciones puntiagudas.





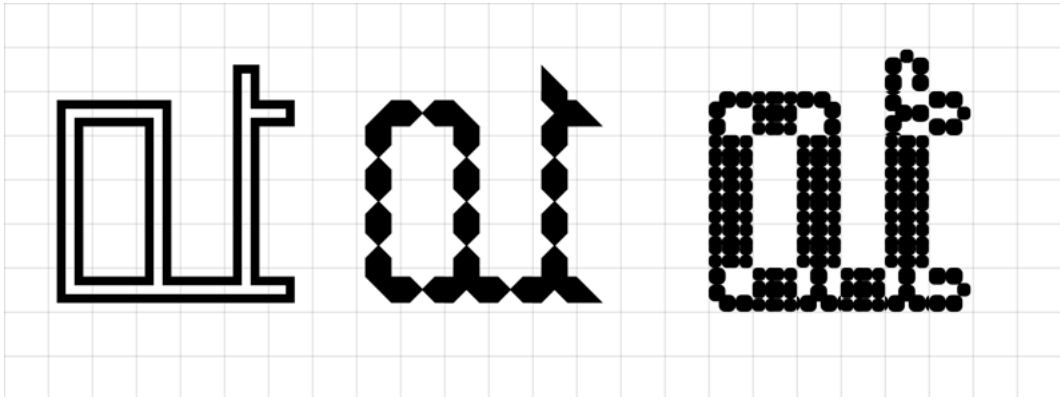
La plantilla C se trata una composición doble ambas con el mismo grosor de cuadrado que la plantilla A y B, enfrentándolas y creando un espacio entre ambos trazos.

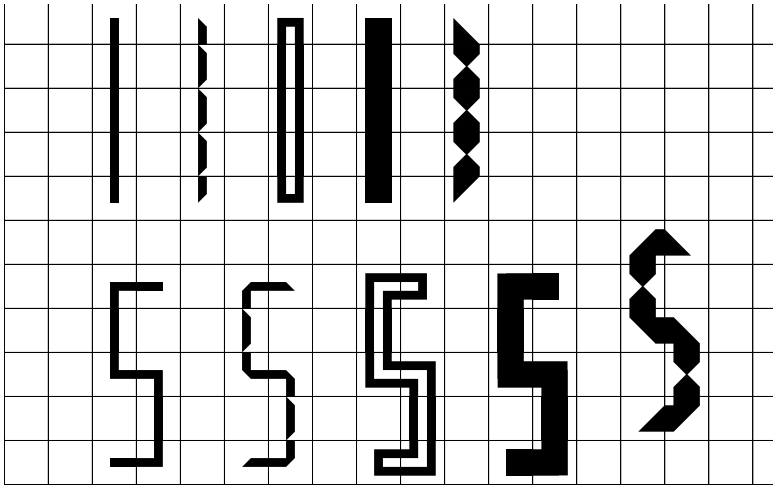




Lo interesante de que el código cargue ocho plantillas diferentes a la vez, no es únicamente el poder ilustrar con cada una de ellas individualmente, sino poder combinar plantillas para obtener resultados mixtos.

Pulsando las teclas del 1 al 8 podemos intercambiar estas plantillas mientras nuestro sketch está ejecutado.





3.3 Creación de mi propia herramienta interactiva basada en el método AMUKI

Una vez explorado el método AMUKI con módulos distintos al mío, decidí que éste era el código indicado para ser modificado y así poder desarrollar mi propia herramienta interactiva.

El sistema de cuadrículas del código de AMUKI es ideal para poder mostrar la versatilidad de mi módulo, ya que toda mi identidad visual también está basada en los sistemas de cuadrículas.

La idea inicial fue crear una herramienta que se pudiera instalar en tablet u otro dispositivos en el festival, donde la gente pudiera interactuar con mi módulo y crear sus propios diseños modulares.

Esta propuesta requiere un código bastante más sencillo que el del método AMUKI, ya que necesitamos menos módulos y estos no van a depender de sus vecinos en la cuadrícula porque no se trata de diseño tipográfico.

Dicho esto, mi primer paso fue simplificar el código sin que perdiera sentido.

Lo primero que modifiqué fue el número de módulos cargados por el código de AMUKI, que en este caso eran ocho plantillas de módulo con quince variaciones de cada módulo en cada plantilla. Mi idea era simplemente cargar mi módulo en ocho variaciones distintas, por lo que aproveché las ocho plantillas ya creadas en el código de AMUKI y en cada una de ellas reduje el número de módulos a uno.

CÓDIGO AMUKI

```
// load svg modules
modulesA = new PShape[16];
modulesB = new PShape[16];
modulesD = new PShape[16];
modulesC = new PShape[16];
modulesE = new PShape[16];
modulesF = new PShape[16];
modulesG = new PShape[16];
modulesH = new PShape[16];
```

CÓDIGO NUEVO

```
modulesA = new PShape[1];
modulesB = new PShape[1];
modulesD = new PShape[1];
modulesC = new PShape[1];
modulesE = new PShape[1];
modulesF = new PShape[1];
modulesG = new PShape[1];
modulesH = new PShape[1];
```

Pero esto requería más cambios en el código, ya que a la hora de cargar los .svg, en el código de AMUKI utilizaban un *for loop* para cargarlos en los 15 slots de cada plantilla con la siguiente fórmula:

"A_"+nf(i,2)+".svg"

Esto consigue que en la primera vuelta del *loop* cargue el archivo **"A_01.svg"**, en la segunda vuelta el **"A_02.svg"** y así sucesivamente.

Todo esto era innecesario en mi código, ya que yo solo tengo un slot por plantilla, así que lo simplifiqué de la siguiente manera:

CÓDIGO AMUKI

```
for (int i=0; i< modulesA.length; i++) {
  modulesA[i] = loadShape("A_"+nf(i,2)+".svg");
  modulesB[i] = loadShape("B_"+nf(i,2)+".svg");
  modulesC[i] = loadShape("C_"+nf(i,2)+".svg");
  modulesD[i] = loadShape("D_"+nf(i,2)+".svg");
  modulesE[i] = loadShape("E_"+nf(i,2)+".svg");
  modulesF[i] = loadShape("F_"+nf(i,2)+".svg");
  modulesG[i] = loadShape("J_"+nf(i,2)+".svg");
  modulesH[i] = loadShape("K_"+nf(i,2)+".svg");
```

CÓDIGO NUEVO

```
modulesA[0] = loadShape("A_00.svg");
modulesB[0] = loadShape("B_00.svg");
modulesC[0] = loadShape("C_00.svg");
modulesD[0] = loadShape("D_00.svg");
modulesE[0] = loadShape("E_00.svg");
modulesF[0] = loadShape("F_00.svg");
modulesG[0] = loadShape("J_00.svg");
modulesH[0] = loadShape("K_00.svg");
```

A lo largo del código de AMUKI se crean las siguientes funciones:

initTiles(): Función que inicializa la pantalla con todos los cuadrantes de la cuadrícula vacíos (sin módulos). Esto se utilizará para cuando se inicie el programa y también para vaciar la pantalla cuando ya tengamos módulos pintados y queramos empezar a diseñar de cero. Esto lo haremos presionando la tecla de retroceso, lo cual definiremos en el código más adelante.
setTile(): Función que define dónde se va a pintar el módulo cuando hagamos click izquierdo con el ratón.
unsetTile(): Función que define qué módulo vamos a borrar cuando hagamos click derecho con el ratón.
drawGrid(): Función que nos permitirá ver la cuadrícula. En esta función hice un pequeño cambio, donde si el fondo es negro, el color de la cuadrícula cambiará a blanco, para que lo podamos seguir viendo.

Todas estas funciones son útiles para mi herramienta y no tienen líneas extra de código que no necesite, por lo que las he dejado prácticamente igual que en el código de AMUKI.

Sin embargo, más adelante aparece la función drawModuls(), en la cual se define qué módulo aparece en pantalla cuando pintamos. Esto, en el código AMUKI, va a estar determinado por la plantilla que tengamos seleccionada (la cual escogeremos con las teclas del 1 al 8) y por el vecino de cuadrícula que ya haya sido pintado.

A nosotros solo nos interesa la primera parte de esta función, donde escogeremos, tecleando del 1 al 8, una de las ocho variaciones de mi módulo. Sin embargo, toda la funcionalidad de los vecinos de cuadrícula no la necesitamos, así que simplificaremos el código de la siguiente manera:

CÓDIGO AMUKI

```
void drawModuls() {
  if(randomMode)activeModulsSet = modules[int(random(modules.length))];
  shapeMode(CENTER);
  for (int gridY=1; gridY< gridResolutionY-1; gridY++) {
    for (int gridX=1; gridX< gridResolutionX-1; gridX++) {
      // use only active tiles
      char currentTile = tiles[gridX][gridY];
      if (currentTile != '0') {
        String binaryResult = "";
        // check the four neighbours. is it active (not '0')?
        // create a binary result out of it, eg. 1011
        // binaryResult = north + west + south + east;
        // north
        if (tiles[gridX][gridY-1] != '0') binaryResult = "1";
        else binaryResult = "0";
        // west
        if (tiles[gridX-1][gridY] != '0') binaryResult += "1";
        else binaryResult += "0";
        // south
        if (tiles[gridX][gridY+1] != '0') binaryResult += "1";
        else binaryResult += "0";
        // east
        if (tiles[gridX+1][gridY] != '0') binaryResult += "1";
        else binaryResult += "0";

        // convert the binary string to a decimal value from 0-15
        int decimalResult = unbinary(binaryResult);
        float posX = tileSize*gridX - tileSize/2;
        float posY = tileSize*gridY - tileSize/2;

        fill(tileColors[gridX][gridY]);
        noStroke();
```

CÓDIGO NUEVO

```
//CREAMOS LA FUNCION "DRAWMODULS" DONDE DEFINIMOS QUÉ MÓDULO APARECE EN PANTALLA CUANDO PINTAMOS
void drawModuls() {
  //SI LA FUNCION "RANDOM" ESTA ACTIVADA, SELECCIONARA UN MODULO RANDOM
  if(randomMode)activeModulsSet = modules[int(random(modules.length))];
  shapeMode(CENTER);
  for (int gridY=1; gridY< gridResolutionY-1; gridY++) {

    for (int gridX=1; gridX< gridResolutionX-1; gridX++) {
      // use only active tiles
      char currentTile = tiles[gridX][gridY];

      if (currentTile != '0') {
        float posX = tileSize*gridX - tileSize/2;
        float posY = tileSize*gridY - tileSize/2;

        fill(tileColors[gridX][gridY]);
        noStroke();
```

Ahora el programa no está pendiente de los vecinos de cuadrícula de cada módulo, sino que simplemente recorre constantemente la cuadrícula esperando a que haya algún cuadrante activo para pintar un módulo. Y para saber qué módulo va a pintar, utilizo las líneas de código que vemos en la imagen de la derecha.

De esta forma, el programa antes de pintar, primero comprueba qué plantilla tenemos seleccionada, para saber el módulo que va a aparecer cuando hagamos click en el ratón.

Toda esta función de *drawModuls()* es más compleja de entender porque trabaja con variables como *tiles[x][y]* que vienen definidas en la función *setTile()*. Pero la explicación a grosso modo sería que mediante *setTile()* definimos dónde se va a pintar, y con *drawModuls()* definimos qué módulo se va a pintar.

```
switch(currentTile) {
case 'A':
  shape(modulesA[0],posX, posY, tileSize, tileSize);
  break;
case 'B':
  shape(modulesB[0],posX, posY, tileSize, tileSize);
  break;
case 'C':
  shape(modulesC[0],posX, posY, tileSize, tileSize);
  break;
case 'D':
  shape(modulesD[0],posX, posY, tileSize, tileSize);
  break;
case 'E':
  shape(modulesE[0],posX, posY, tileSize, tileSize);
  break;
case 'F':
  shape(modulesF[0],posX, posY, tileSize, tileSize);
  break;
case 'G':
  shape(modulesG[0],posX, posY, tileSize, tileSize);
  break;
case 'H':
  shape(modulesH[0],posX, posY, tileSize, tileSize);
  break;
}
```

Además al final de esta función representada en la imagen anterior, definimos que si la variable *debugMode* está activada, veremos las coordenadas de cada módulo que haya sido pintado:

```
  if (debugMode) {
    fill(150);
    text(currentTile+"\n"+posX+"\n"+posY,posX, posY);
  }
}
```

Para terminar tenemos que definir qué teclas hacen qué, y para eso utilizaremos la función *keyReleased()*. En esta parte del código le hemos dado funcionalidad a las siguientes teclas:

Tecla S: Función añadida que le cambia el valor a una variable booleana que hemos llamado *'rec'*. La cual si está activada empezará a grabar los frames del programa en ejecución y en cuanto la desactivemos, terminará de grabar. De esta manera podremos crear animaciones de nuestro módulo.

Suprimir o Retroceso: llama a la función *initTiles()* para vaciar toda la pantalla, borrando todos los módulos.

Tecla G: Le cambia el valor a la variable booleana *drawGrid*, la cual si está activada, llamará a la función *drawGrid()* para que haga visible el Grid. Mientras que si está desactivada, no lo veremos.

Tecla D: Lo mismo que la anterior tecla pero con la variable debugMode para que podamos ver o no las coordenadas de cada cuadrícula.

Tecla R: Activa o desactiva el modo *random*. En caso de que el modo random esté activado, dentro de la función *drawModuls()* se escogerá una variación aleatoria del módulo antes de pintar.

Teclas del 1 al 8: Seleccionamos la variación del módulo que queramos antes de pintar.

El código de estas especificaciones se enseñará en la siguiente página.

```
void keyReleased() {
  //AL PUSAR LA TECLA "S" ACTIVAMOS O DESACTIVAMOS LA VARIABLE "REC" QUE ANTES DEFINIMOS PARA GRABAR LA PANTALLA
  if (key == 's' || key == 'S') rec = !rec;

  //SI PULSAMOS LA TECLA DE BORRADO SE BORRARAN TODOS LOS MODULOS
  if (key == DELETE || key == BACKSPACE) initTiles();

  //SI PULSAMOS "G" SE DIBUJARA EL GRID
  if (key == 'g' || key == 'G') drawGrid = !drawGrid;

  //SI PULSAMOS "D" VEREMOS LAS COORDENADAS DE LOS MODULOS
  if (key == 'd' || key == 'D') debugMode = !debugMode;

  //SI PULSAMOS "R" SE ACTIVARA EL MODO RANDOM
  if (key == 'r' || key == 'R') randomMode = !randomMode;

  //DEL 1 AL 8 ESCOGEREMOS EL MODULO QUE QUERAMOS PINTAR
  if (key == '1') activeModulsSet = 'A';
  if (key == '2') activeModulsSet = 'B';
  if (key == '3') activeModulsSet = 'C';
  if (key == '4') activeModulsSet = 'D';
  if (key == '5') activeModulsSet = 'E';
  if (key == '6') activeModulsSet = 'F';
  if (key == '7') activeModulsSet = 'G';
  if (key == '8') activeModulsSet = 'H';

  //CON LAS SIGUIENTES TECLAS ESCOGEREMOS LOS DISTINTOS COLORES
  if (key == 'y' || key == 'N') activeTileColor = color(0);
  if (key == 'v' || key == 'V') activeTileColor = color(#009245);
  if (key == 'b' || key == 'B') activeTileColor = color(#ffffff);

  //AL PUSAR LA TECLA "S" ACTIVAMOS O DESACTIVAMOS LA VARIABLE "REC" QUE ANTES DEFINIMOS PARA GRABAR LA PANTALLA
  if (key == 'f' || key == 'F') black = !black;
}
```

26. En el siguiente enlace podrán verse pruebas realizadas con mi herramienta interactiva finalizada en la carpeta "VISUAL BEINGS MOVIES"
- https://drive.google.com/drive/folders/1pKGnYt9yTt-l7A_VgAx0Rau7ZAMhmN-d1l?usp=sharing

Hasta aquí llegan los tecnicismos de mi herramienta interactiva.

Ahora, se mostrarán usos y posibles aplicaciones para el festival.

La idea es que una vez se acceda al espacio del festival, se encuentren unos monitores disponibles (pantallas) con un teclado y un ratón conectados para que el usuario pueda experimentar con las variaciones de mi módulo y crear sus propios activos e ilustraciones.

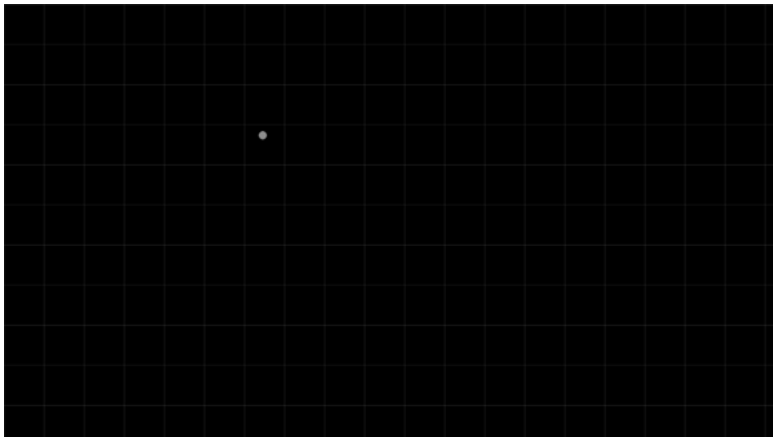
Al lado de estos dispositivos estarán unas instrucciones de uso.

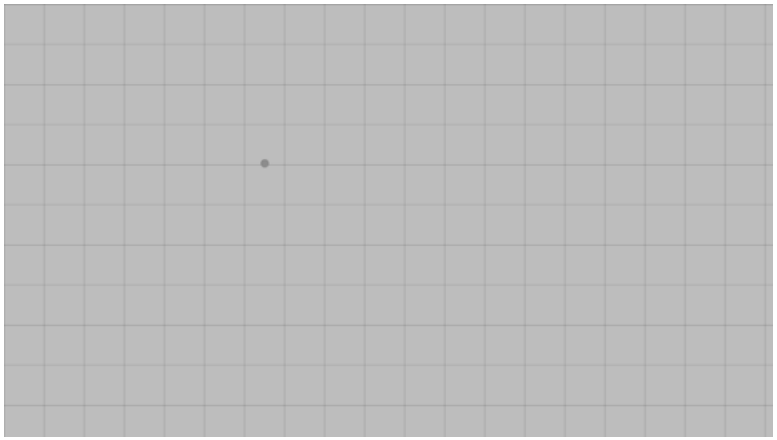
Cada ilustración que se cree en estos dispositivos, será proyectada en banners digitales de gran tamaño que habrán colocados por el espacio del festival.

El diseño con el que se encontrará el usuario es la “pantalla de inicio” de nuestra herramienta, es decir, nuestro código estará siempre ejecutado y nunca se detendrá.

Cuando algún usuario acabe de usar la herramienta, podrá irse y dejar sus diseños visibles o podrá borrar pulsando suprimir.

En caso de que no lo borre, el siguiente usuario, borrará lo anterior presionando suprimir y comenzará a diseñar él / ella.



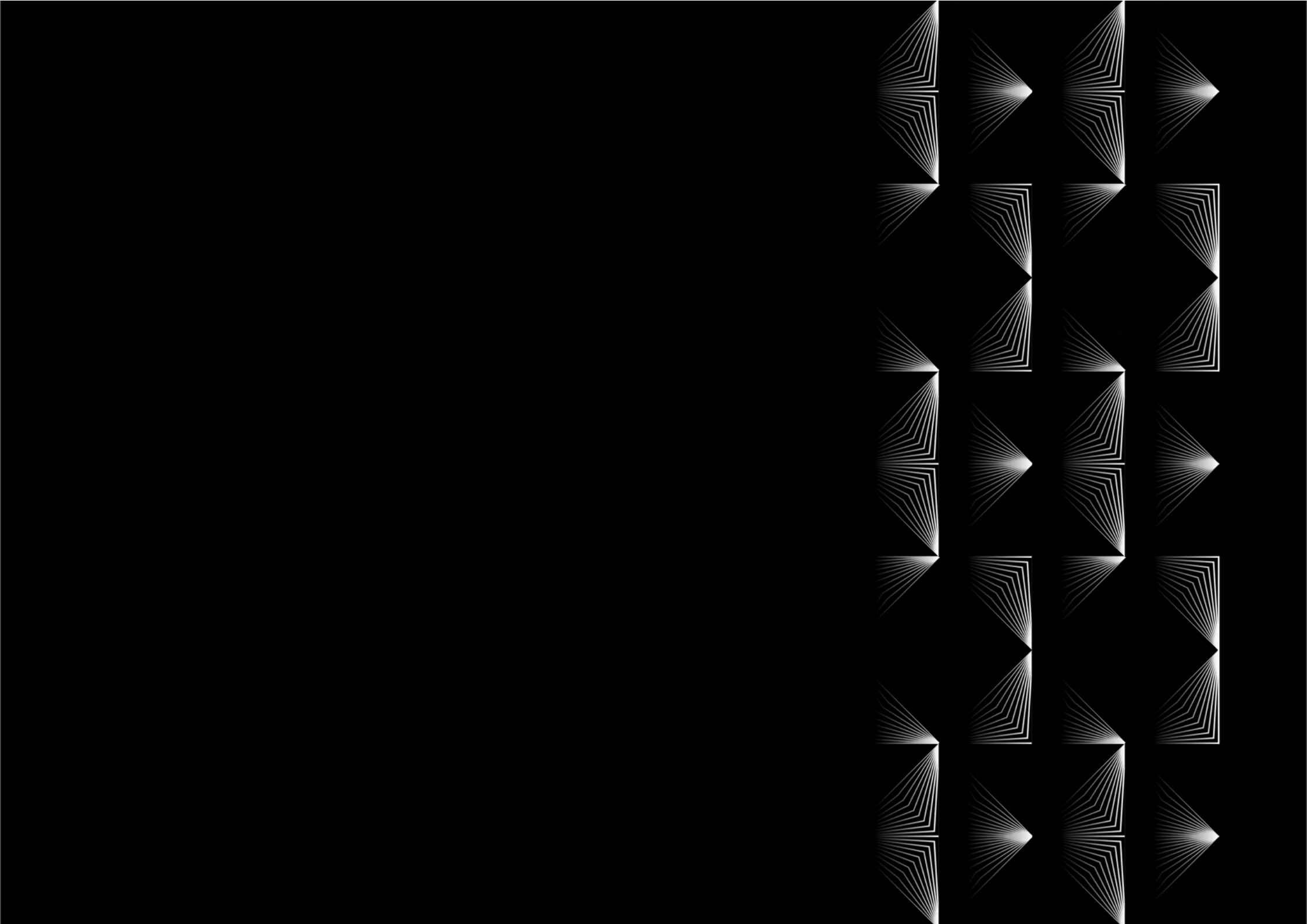


Los monitores serán de esta manera, y además tendrían un teclado y un ratón conectados.

A lado de cada monitor, irán unas instrucciones de uso.

Estas instrucciones en formato A4 irán pegadas o colocadas al lado de cada monitor disponible en el festival.

Con estas instrucciones, el usuario podrá crear una infinidad de diseños y además guardarlos pulsando la tecla S , para que así sus creaciones queden guardadas en el repositorio del festival y puedan ser mostradas en las pantallas digitales los días posteriores.



CONCLUSIÓN

El proyecto ha servido para demostrar la versatilidad del diseño modular, y las infinitas aplicaciones que nos ofrece el código creativo.

Este trabajo de final de grado ha supuesto un gran reto personal por el elevado carácter de investigación y aprendizaje que trae consigo.

En un principio, la idea de estudiar los sistemas flexibles, la modularidad, y el código creativo estaban claras, sin embargo, el cómo hacer un trabajo de final de grado de ello, no tanto.

Logrando centrar la idea finalmente conseguimos diseñar una identidad visual flexible que cumple con todos los objetivos propuestos al inicio del TFG.

Hemos logrado aplicar el pensamiento sistémico en nuestro proceso de diseño que fue el primer objetivo general que se marcó. La idea de la creación modular hace que las formas de trabajo se sistematice n reduciendo la dificultad y el tiempo a la hora de diseñar y además logró que nuestro segundo objetivo general se cumpliera, que fue el diseño de una propuesta flexible y adaptativa a todos los formatos.

Los objetivos específicos por otro lado, los fuimos cumpliendo uno a uno a medida que avanzaba el trabajo, y cada uno hizo posible que el siguiente se pudiera cumplir.

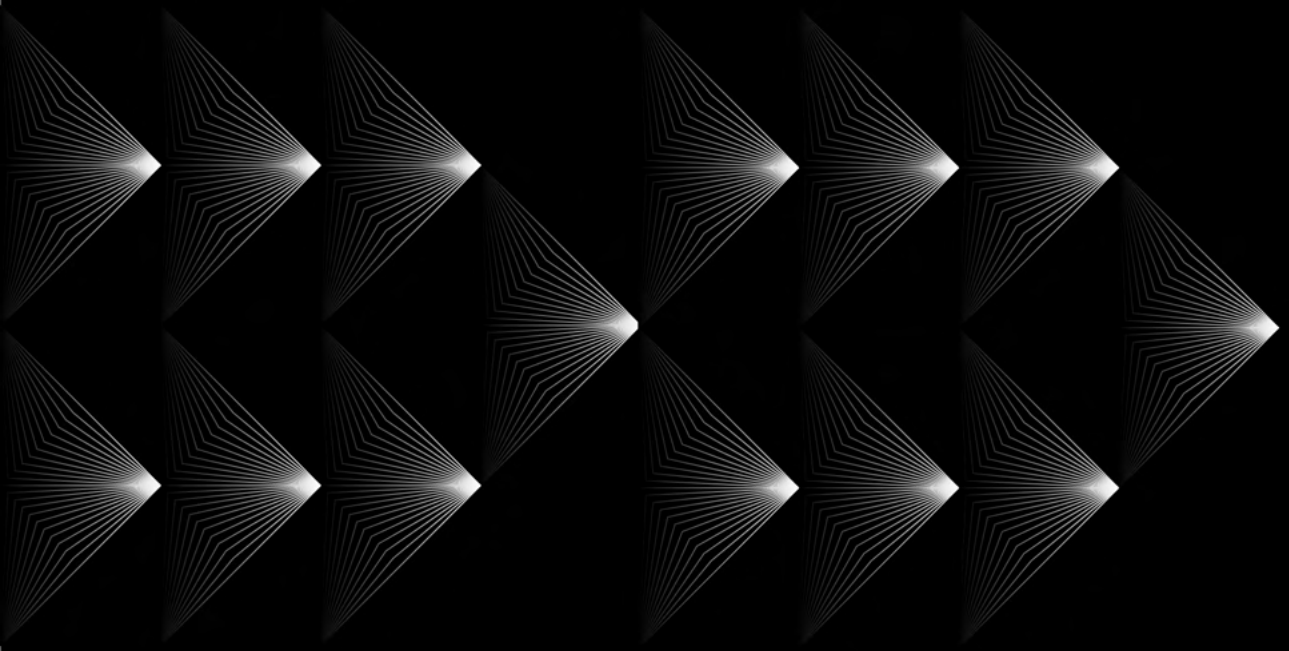
El contenido promocional flexible gracias al diseño modular, la creación de animaciones gracias al desarrollo de nuestra herramienta interactiva con código creativo...

Este trabajo servirá a futuros diseñadores para acercarse un mínimo a el diseño creativo y el diseño modular, y darse cuenta de que no es tan complicado como parece, y que además es muy

divertido.

El ensayo error al final se trata de un método de aprendizaje bastante completo, en el sentido que aprendes de tus errores y posteriormente no los vuelves a cometer.

Como decíamos al principio, debemos adaptarnos a los nuevos tiempos y a el nuevo paradigma de la comunicación, donde el usuario forma parte de todo el proceso de diseño. Hagamos que los usuarios no solo formen parte del proceso, si no que sean capaces de diseñar ellos mismos un futuro inteligente.



BIBLIOGRAFÍA

Britannica, T. Editors of Encyclopaedia (1998, julio). Romain du Roi. Encyclopedia Britannica. https://www.britannica.com/topic/Romain-du-Roi

Gamonal, R., Gamonal, R., Balius, A., Trochut, A., & Gil, E. (2017, marzo). Super Tipo Veloz: la tipografía supercalifragilística (I). Pioneros Gráficos. Recuperado 17 mayo, 2022, de https://pionerosgraficos.com/la-tipografia-supercalifragilistica/

Guayabero, Ó. (2019, enero). Herbert Bayer.ESDESIGN. Recuperado Abril 20, 2022, de https://www.esdesignbarcelona.com/actualidad/diseno-grafico/herbert-bayer

Heller, S. (2019, marzo). Cassandre's Most Eclectic Typeface – PRINT Magazine. PRINT Magazine. Recuperado 3 mayo, 2022, de https://www.printmag.com/daily-heller/bifur-cassandre-typeface/

Johnston's Underground Type. (2022, julio). Luc Devroye. Recuperado 24 abril, 2022, de http://luc.devroye.org/fonts-66504.html

Keung, L. (2020, agosto). All About the Futura Font and Its History. Design Tutsplus. Recuperado 26 abril, 2022, de https://design.tutsplus.com/articles/all-about-futura-font-and-its-history--cms-35382

Licko, Z. (n.d.). Emigre: Lo-Res Font Family. Emigre: Fonts. Recuperado 22 abril, 2022, de https://www.emigre.com/Fonts/Lo-Res

Lorenz, M. (2021). Flexible Visual Systems: The Design Manual for Contemporary Visual Identities (J. Kahl, L. Harmsen, & Slanted, Eds.). Karlsruhe: Slanted Publishers.

Luque, B. (2017, julio). Tipografía digital | Investigación y Ciencia. Investigación y Ciencia. Recuperado12 marzo, 2022, de https://www.investigacionyciencia.es/revistas/investigacion-y-ciencia/el-origen-de-la-tecnologa-709/tipografa-digital-15399

Meilleur, M. (n.d.). schrofer_screenprint — mauricemeilleur \ design. MauriceMeilleur. Recuperado 2 mayo, 2022, de https://mauricemeilleur.net/schrofer_screenprint

Moliz, A. (2016, octubre). TpStulle, la nueva tipografía de TwoPoints acompañada por una historia. Gráffica.info. Recuperado 24 abril , 2022, de https://graffica.info/tpstulle-tipografia-de-twopoints/

Nawrot, K., & Peško, R. (2017, septiembre). The Visual History of Type author Paul McNeil selects and dissects his six favourite faces. It's Nice That. Recuperado 4 mayo, 2022, de https://www.itsnicethat.com/news/the-visual-history-of-type-paul-mcneil-graphic-design-publication-110917

What is Creative Coding? · tim rodenbröker creative coding. (2022, junio). Tim Rodenbröker. Recuperado 4 mayo, 2022, de https://timrodenbroeker.de/what-is-creative-coding/

Wim Crouwel. New Alphabet. 1967. (n.d.). MoMA. Recuperado 17 mayo, 2022, from https://www.moma.org/collection/works/139322

ÍNDICE DE IMÁGENES

Figura 18. Flexible visual systems. The design manual for contemporary visual identities. Martin Lorenz, 2021. Recuperado de https://www.kickstarter.com/projects/martinlorenz/flexible-visual-systems

Figura 19. Flexible visual systems. The design manual for contemporary visual identities. Martin Lorenz, 2021. Recuperado de https://www.kickstarter.com/projects/martinlorenz/flexible-visual-systems

Figura 20. Tinkuy Patterns, MODULAR TYPOGRAPHY VOL.2, AMUKI, 2022. Recuperado de https://www.behance.net/gallery/142518729/Tinkuy-Patterns-Modular-Typography-Vol2

Figura 21. Programming Posters, Tim Rodenbröker, 2018. Recuperado de https://timrodenbroeker.de/programming-posters/

Figura 22. Programming Posters, Tim Rodenbröker, 2018. Recuperado de https://timrodenbroeker.de/programming-posters/

Figura 23. “The Magic Triangle”, Tim Rodenbröker, 2018. Recuperado de https://timrodenbroeker.de/programming-posters/

Figura 24. Center for Complexity, TwoPoints.net, 2020. Recuperado de https://new.twopoints.net/

Figura 25.Letterform Variations, Nigel Cottier, 2021. Recuperado de https://letterformvariations.com/

Figura 26. Letterform Variations, Nigel Cottier, 2021. Recuperado

de https://letterformvariations.com/

Figura 27. Shape Grammars, Jannis Maroschek, 2020. Recuperado de https://maroscheck.de/shapegrammars

Figura 28. Shape Grammars, Jannis Maroschek, 2020. Recuperado de https://maroscheck.de/shapegrammars

Figuras 29/30. Estructura experimental usando impresión 3D. Zaha Hadid Architects, 2017. Recuperado de https://www.archdaily.mx/mx/872087/zaha-hadid-architects-presenta-estructura-experimental-utilizando-la-impresion-3d

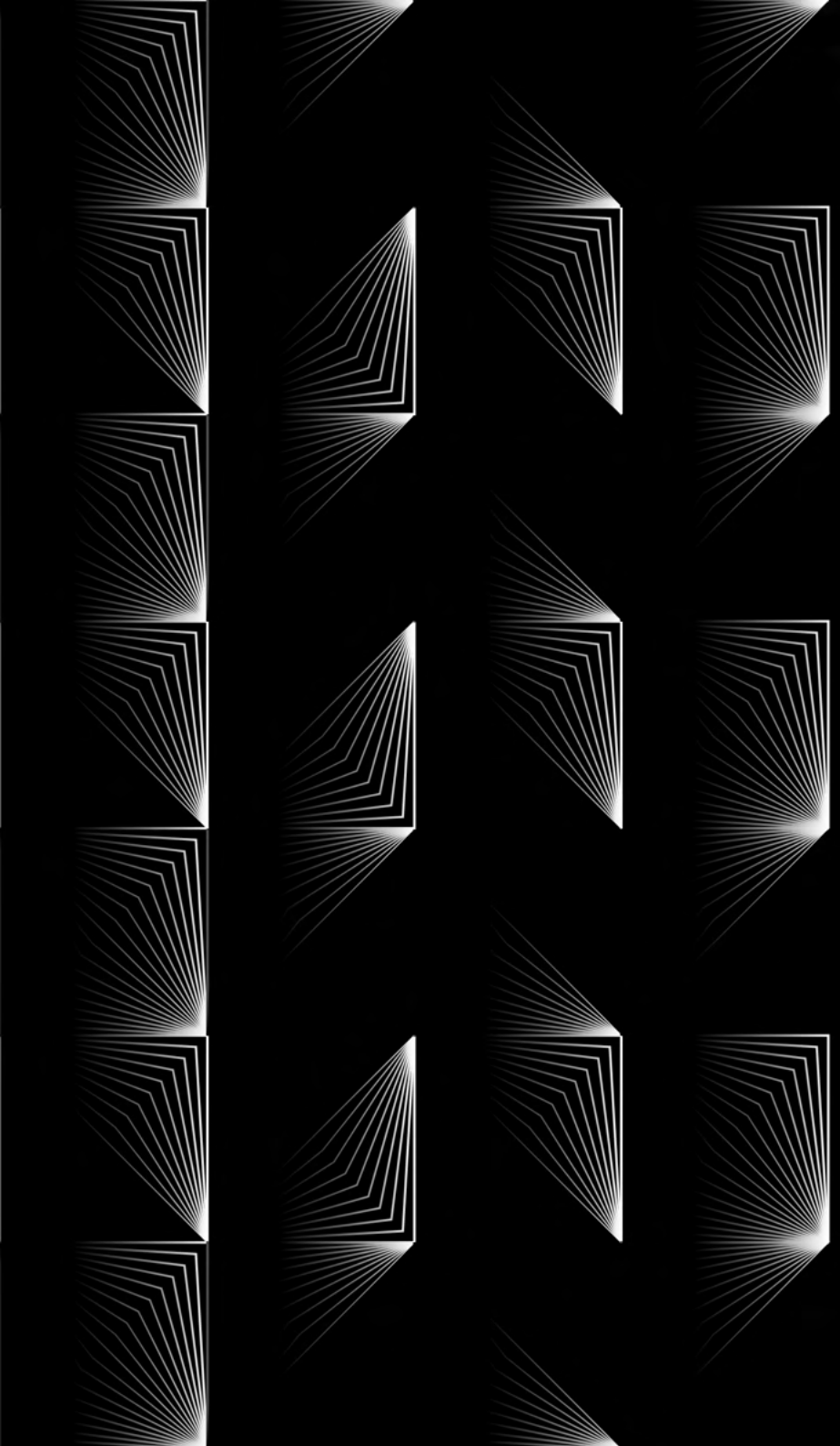
Figuras 31/32. Principles of two-dimentional design. Wucious Wong, 1972. Recuperado de https://www.amazon.com/Principles-Two-Dimensional-Design-Wucius-Wong/dp/0471289604

Figura 33. Untitled. Karel Martens, 2015. Recuperado de https://martens-martens.com/

Figura 34. Untitled. Karel Martens, 2014. Recuperado de https://martens-martens.com/

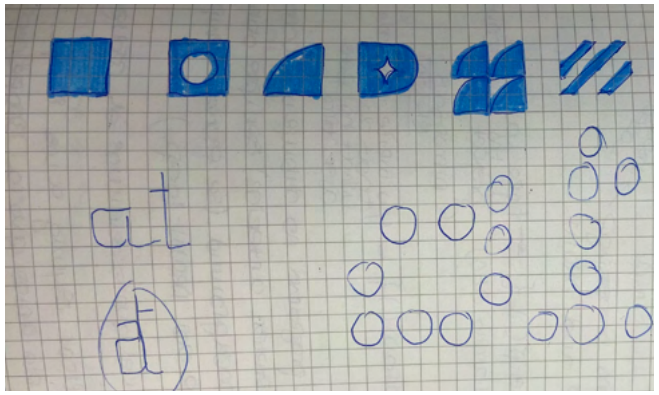
Figura 35/36. Twopoints.net. Center for Complexity, Rhode Island School of Design, 2020. Recuperado de https://new.twopoints.net/risd-cc/

Figura 37. Flexible Visual Systems. Martin Lorenz, 2021. Recuperado de https://flexiblevisualsystems.info/

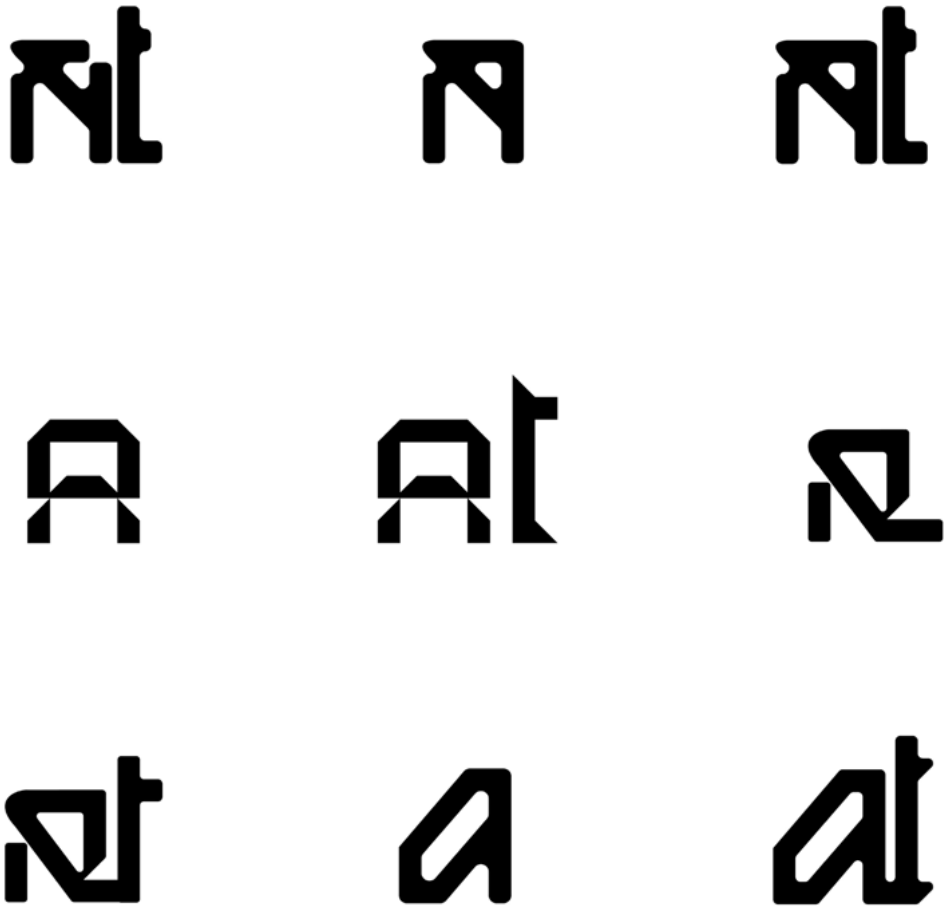


ANEXOS

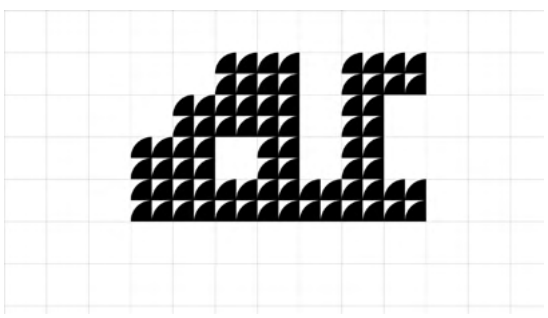
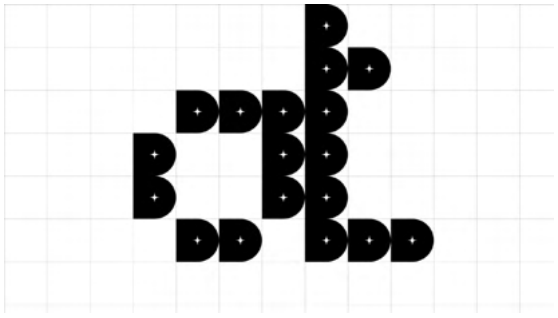
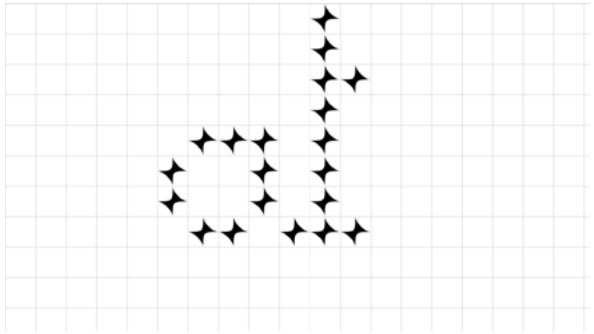
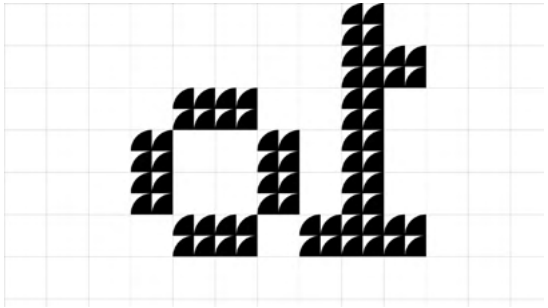
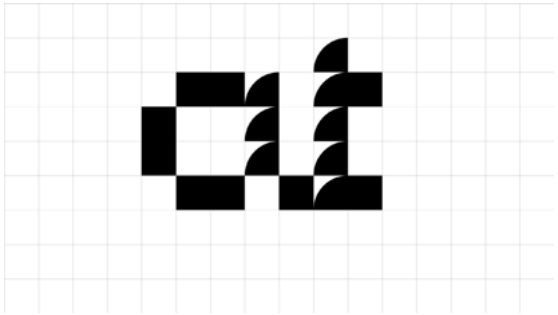
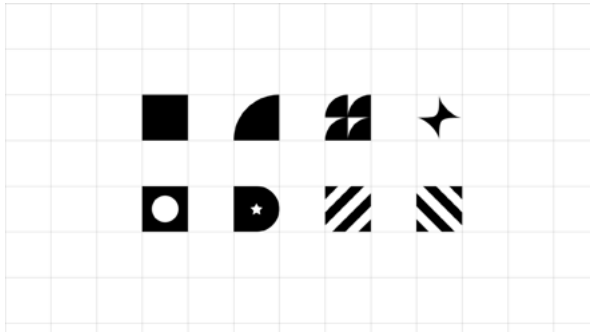
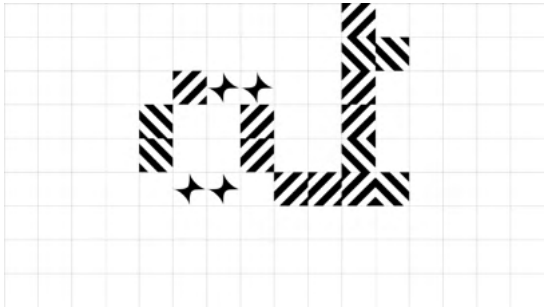
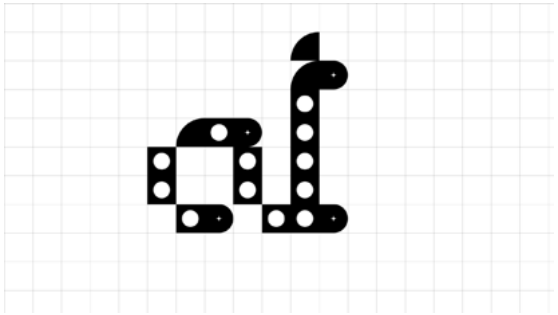
1. Bocetos de AT (primera idea de nombre del festival) y contrucción modular variada.



2. Diseño tipográfico modular del logo AT basado en los bocetos iniciales.



3. Diseño tipográfico modular en Processing basado en 8 plantillas de módulos diferentes diseñadas en adobe illustrator previamente.

4. Entrevistas completas

4.1 En conversación con Tim Rodenbröker. Codificación creativa.

Entrevista en inglés.

1. Is your academic background related with creative coding / generative design?

No, I don't have a background in this field, at least I didn't when I started. I studied communication design and then got involved with the subject out of pure interest. It wasn't until much later, when I already had a lot of experience in teaching, that I did another master's degree, which I finished a few weeks ago.

2. If not, how did you discover this new way of working?

I am a totally curious person who has always loved to tinker and make things. Code has me particularly excited that I can do an infinite number of things with a single tool. Over time, I realized that I would like to deal with this very intensively and then I just experimented endlessly. First it started with web development, then with creative coding. I still enjoy doing all of this to this day!

3. Has your experience with creative coding helped your skills as a graphic designer?

I published my master's thesis under the title "Creative Coding as a School of Thought." I think the name says it all! In today's world, everything moves infinitely fast, and we need a great deal of flexibility in our thinking in order to remain capable of acting. Learning to think is what it's all about, and it doesn't just help you become a good designer.

4. Can you solve design problems that you previously solved with adobe softwares, with creative coding?

Yes indeed! With coding you can do many things that are even not possible with Adobe software. Two examples: Design automation is a huge topic, as well

as custom design tools for specific project.

5. What are the Pros and cons of creative coding?

Difficult question. I'd say that the main supposed disadvantage of creative coding is the fact that it takes a lot of time and effort to get into it. It takes quite a long time to get to the point where you can use it effectively. Traditional design tools are much easier to learn and accordingly they are tempting for young people who are looking for their way into the world of design. At the same time, there are many prejudices and fears associated with programming. However, it's totally worth it to get into it, because there is so much to learn and discover.

6. Do you think in the future Communications design and technology will go hand by hand?

John Maeda once said that nothing has ever had o great influence on design as the development of the computer. The influence of technology on design probably cannot be overestimated.

7. Would you recommend any online/offline communities around this theme?

Sure: https://timrodenbroeker.de/

4. Entrevistas completas

4.2 En conversación con Martin Lorenz. Flexible visual systems.

Entrevista en inglés.

1. Is your academic background related with creative coding / generative design?

Not really. I had one teacher though that tried to explain us a bit of python in a day. Petr van Blokland. If you wanted to learn more, you had to do it in your spare time.

2. How did you discover this new way of design? Was it just an adjustment to these technological times? Or you actually needed some new tools that made your designing process easier?

I am not that rational. I am just constantly looking for things I am bad at and try to learn them. Learning is what I enjoy most in life. All the new tools coming up shape the design of today. Learning the new tools make you think with your hands. What I discovered was that we were heading towards a systemic approach. Not just in design. The shift of perspective is happening everywhere and in every discipline. We see and make everything in relation to something else. In order to solve one problem, you have to see how it is affected by and has an effect on other systems.

3. What are the advantages of designing flexible visual systems?

- Working with systems is more efficient because the application can be automated.
- Working with systems is more durable because the components can be optimized without having to overhaul the entire system.
- Working with systems enables teamwork, as it is based on objectively comprehensible rules.
- Thinking in systems makes empathy necessary for communication because design components have to be understood in context, as people will encounter them, which may be very different from

how you encounter them.
- Thinking in systems creates a sense for responsibility, as your work affects the system as well as everyone and everything involved.
- Thinking in systems leads to the opportunity of criticism of systems because you realize that the system may be the problem, not its components.
- Not working or thinking in systems corresponds neither to our today's communication networks nor to contemporary communication behavior.

4. Is modularity related with flexible visual systems?

Modularity is one way of building a flexible system, but there are many more. Most form-based systems are modular systems, because they are easy to understand and apply.

5. Can you desing flexible visual systems with out coding?

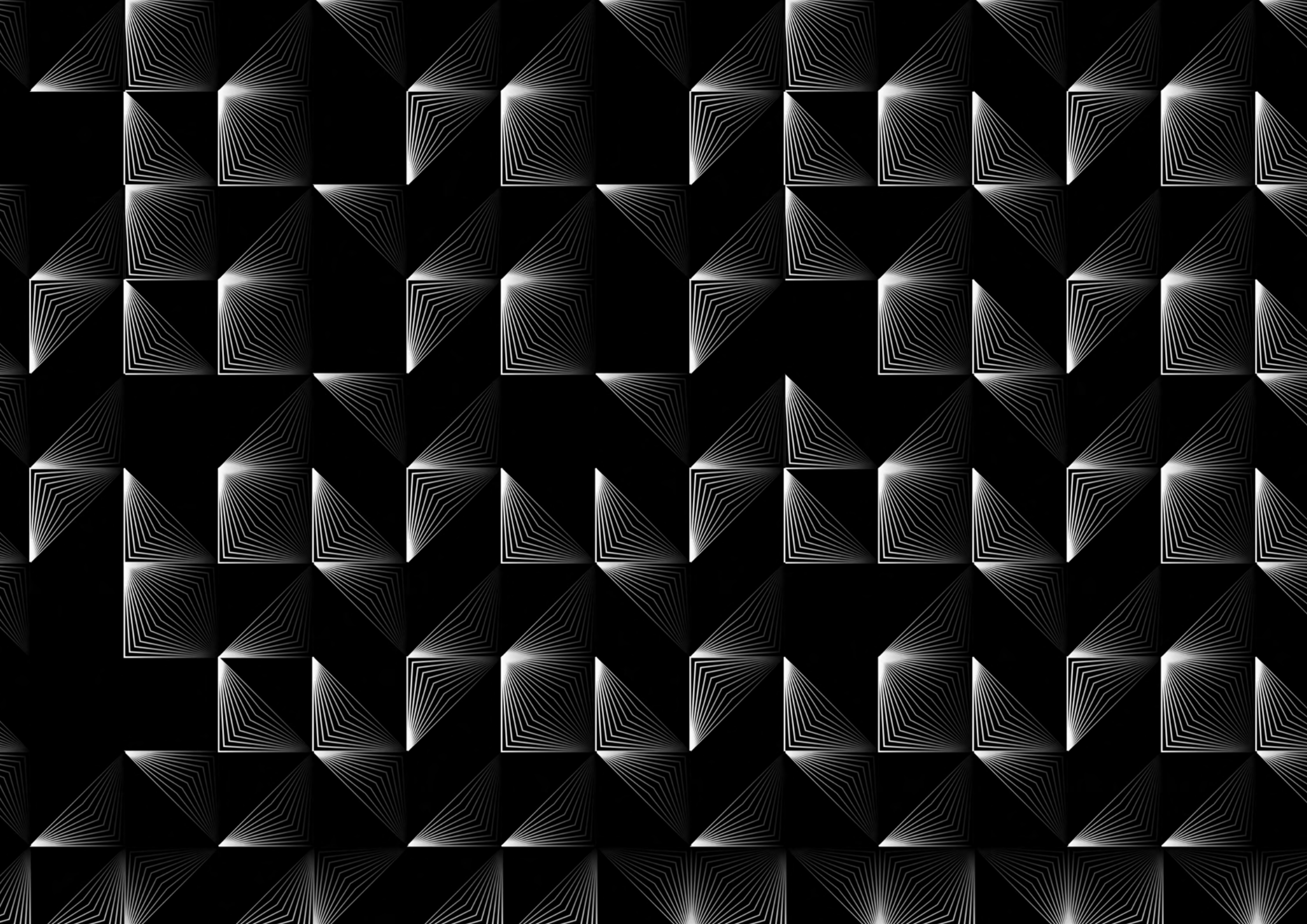
Of course! In my classes we work a lot with paper and photography.

6. Is creative coding fully integrated in your working flow?

I use it whenever I need it. There is stuff I can't do with code and I do not want to be limited to one tool. If you only have a hammer, everything will be a nail to you. I want to have different solutions for different problems.

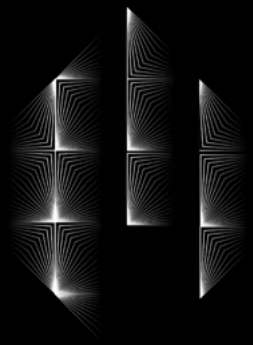
7. What are the advantages of designing the way you do?

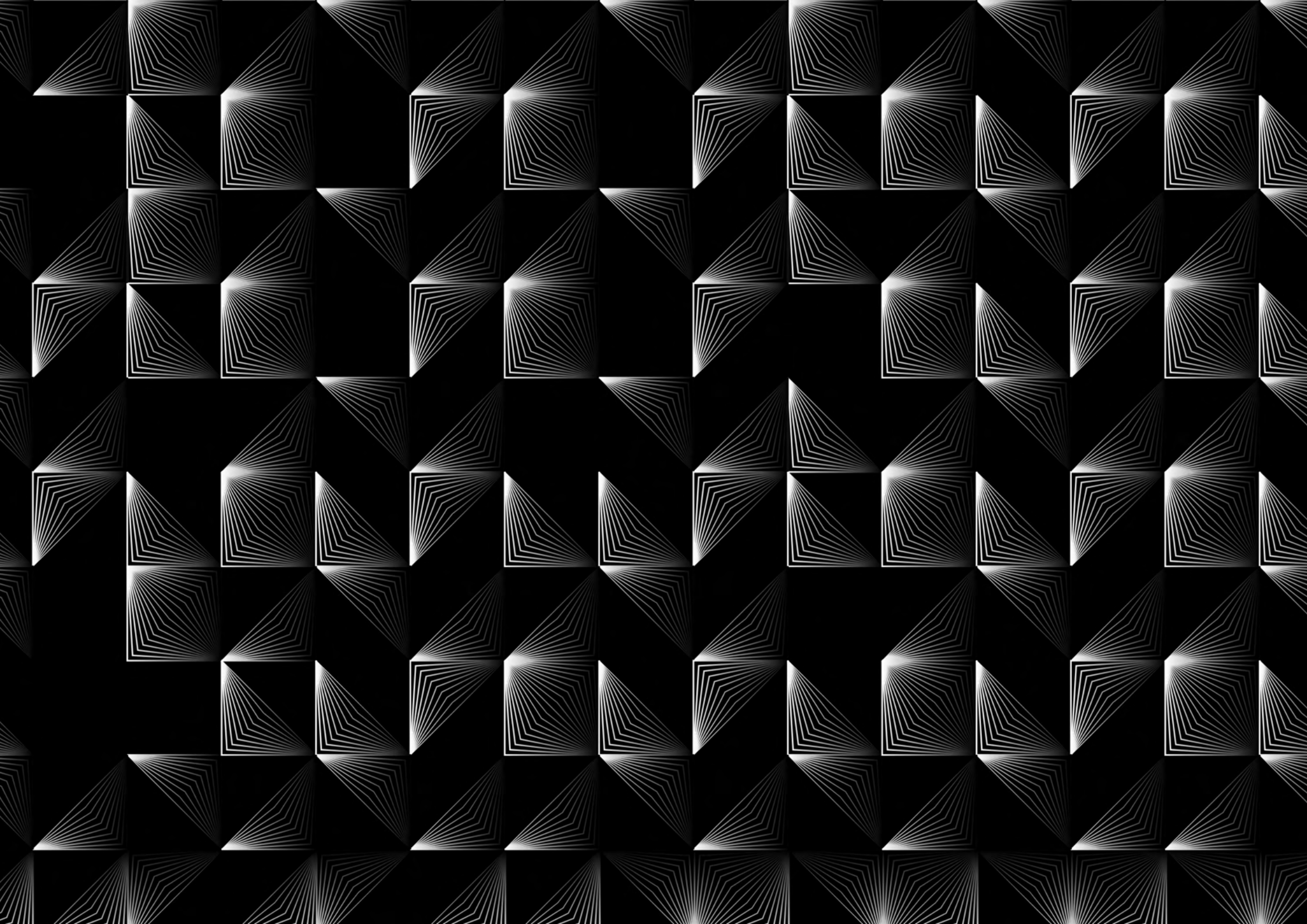
As I design systems, I think I already answered that question. :) ... but I like to think, I am not limited to one way of designing. I want to keep flexible. :)



VISUAL BEINGS

MANUAL DE IDENTIDAD





Manual de identidad visual.
VISUAL BEINGS.

Alumna
Paula Martín Gómez de la Serna

Tutor académico
Alfonso Ruiz Rallo

Trabajo de Fin de Grado
Grado en Diseño
Facultad de Bellas Artes
Universidad de La Laguna

Curso académico 2021-2022
Septiembre 2022

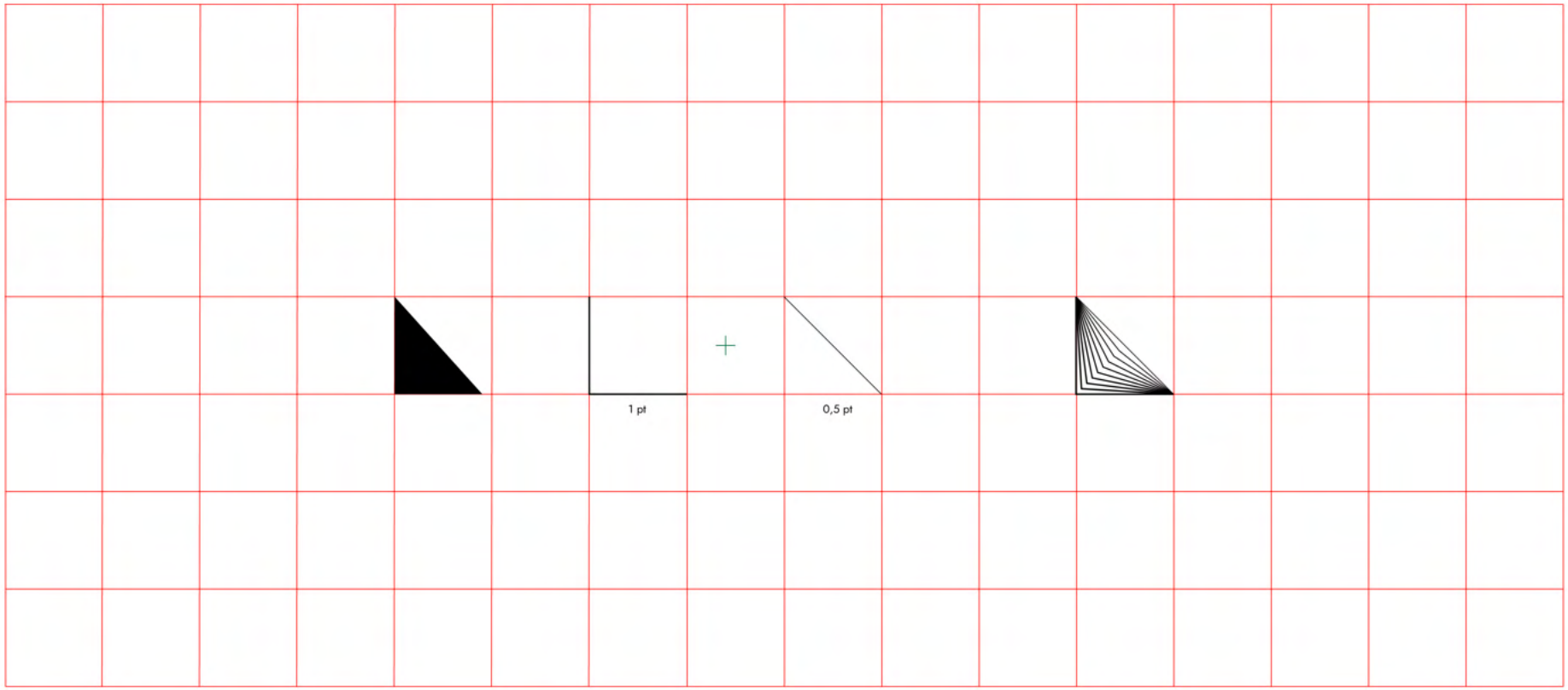
Este documento sienta las bases del correcto uso y aplicación de la marca visual VISUAL BEINGS.

Las directrices aquí establecidas, son de obligado cumplimiento siempre que se quiera hacer un uso correcto de la marca para garantizar su reconocimiento, legibilidad y coherencia gráfica.

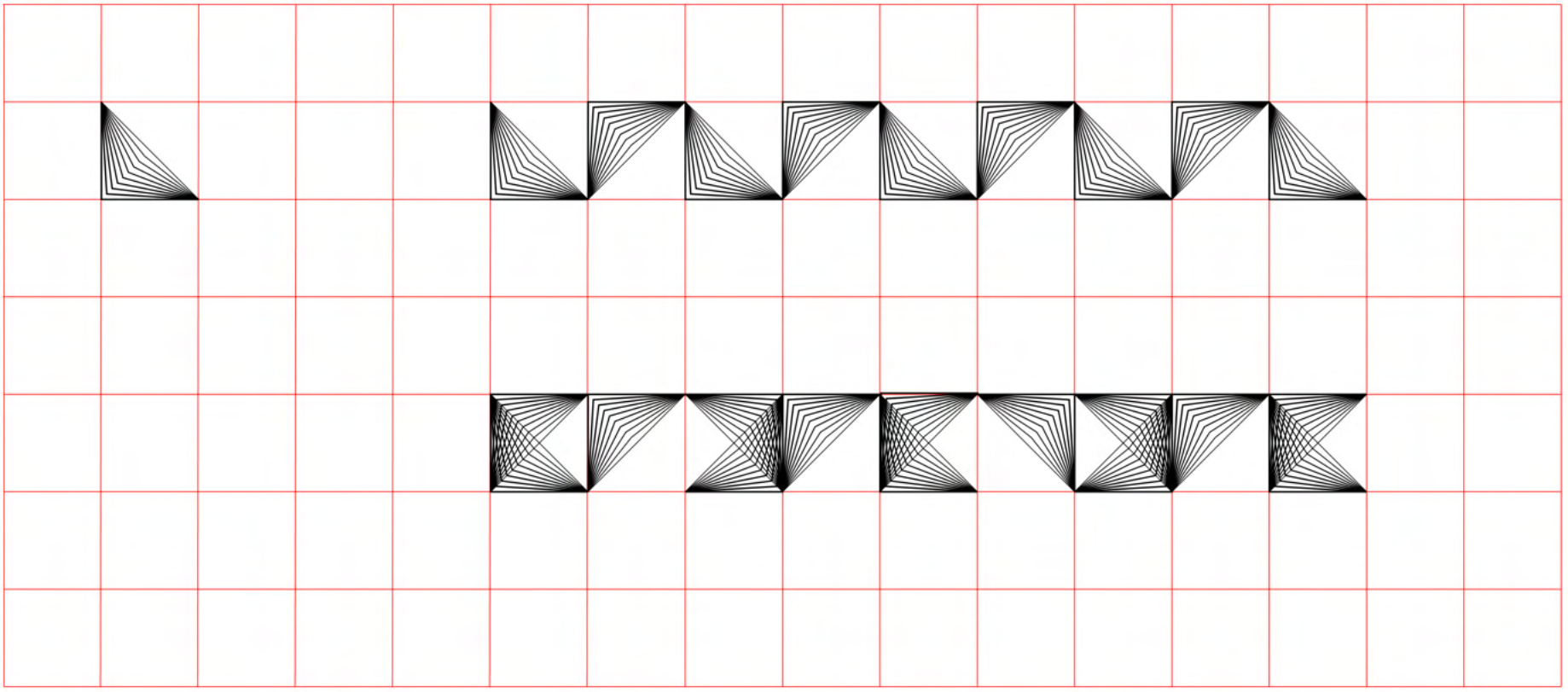
ÍNDICE

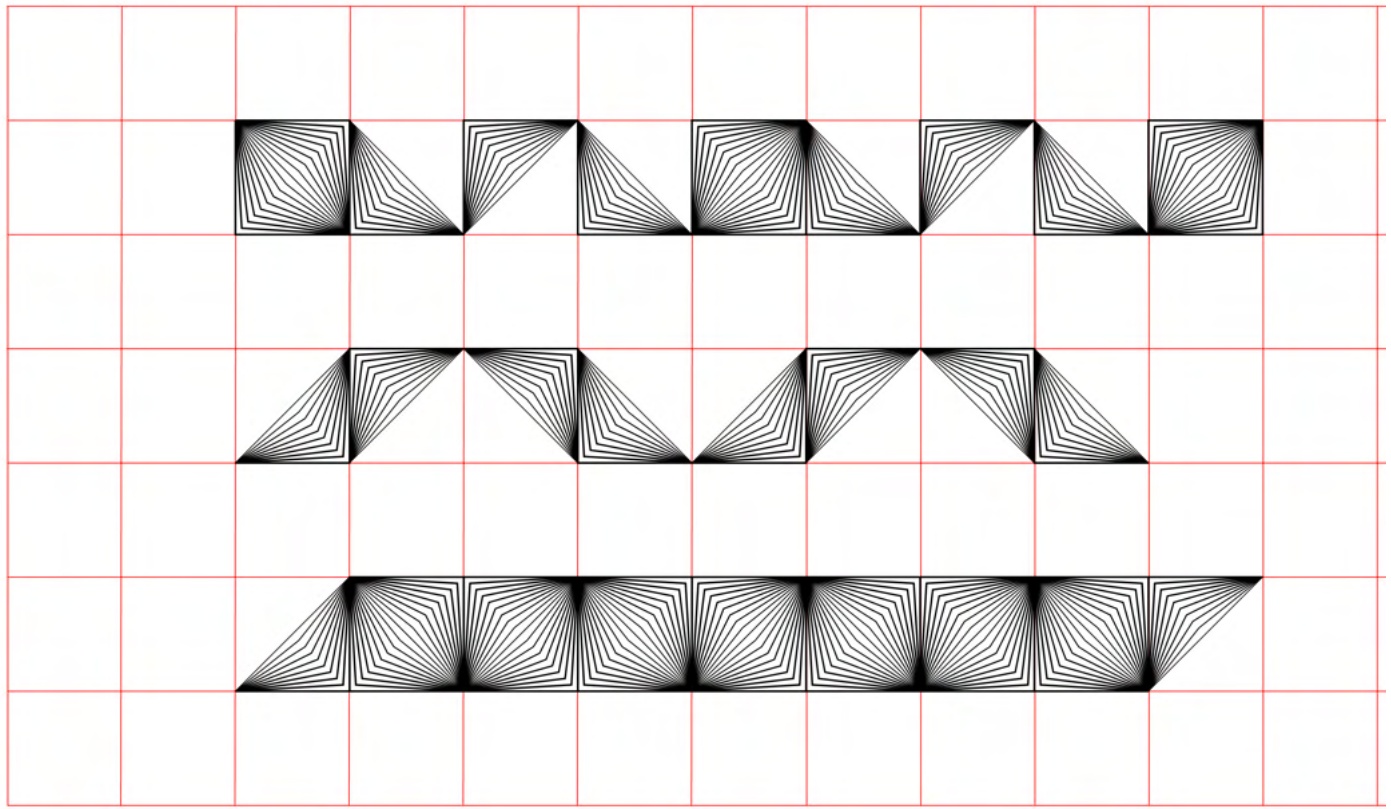
1. CONSTRUCCIÓN DEL MÓDULO Y ACTIVOS

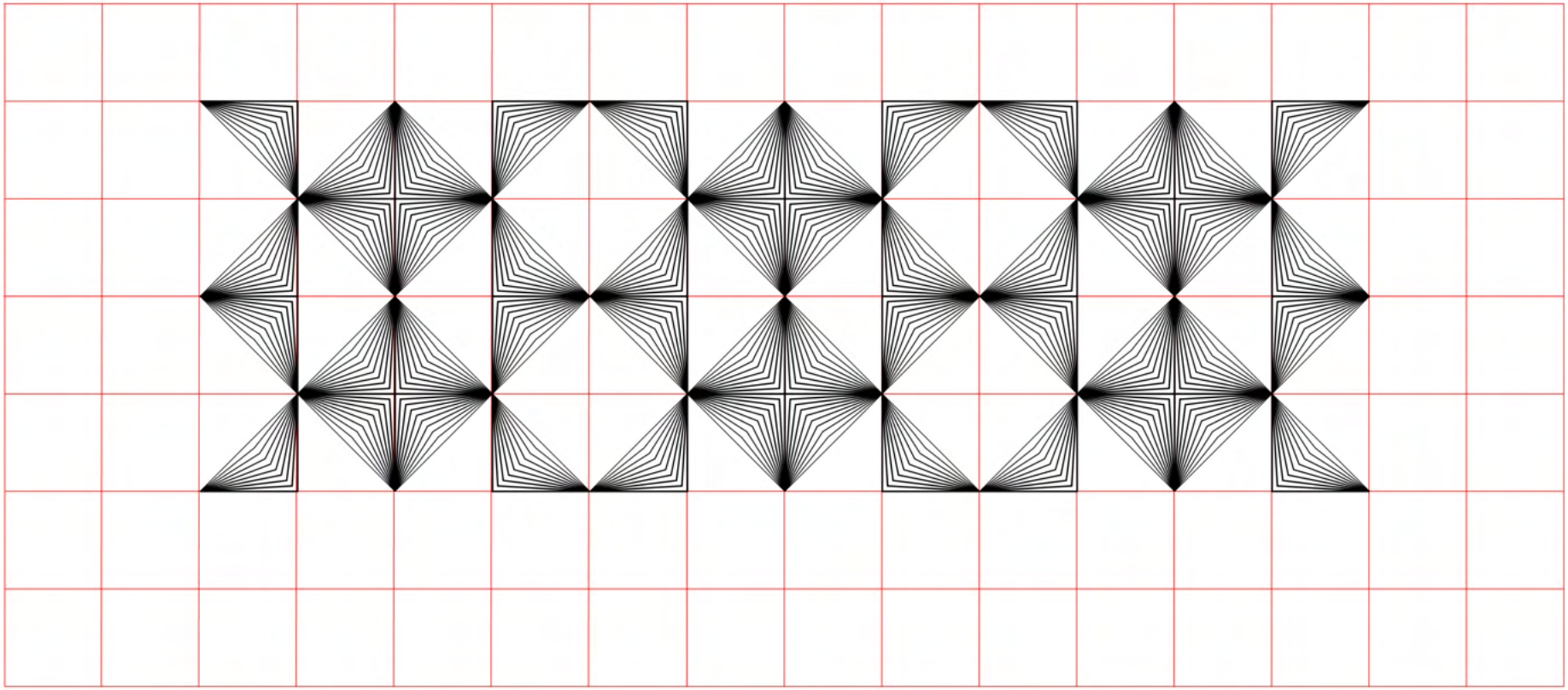
La idea de definir cómo se realizó la construcción del módulo y los activos, es que el cliente pueda entender el sistema de diseño y así poder generar infinitas variaciones de nuestro diseño de manera correcta.


1 pt
0,5 pt

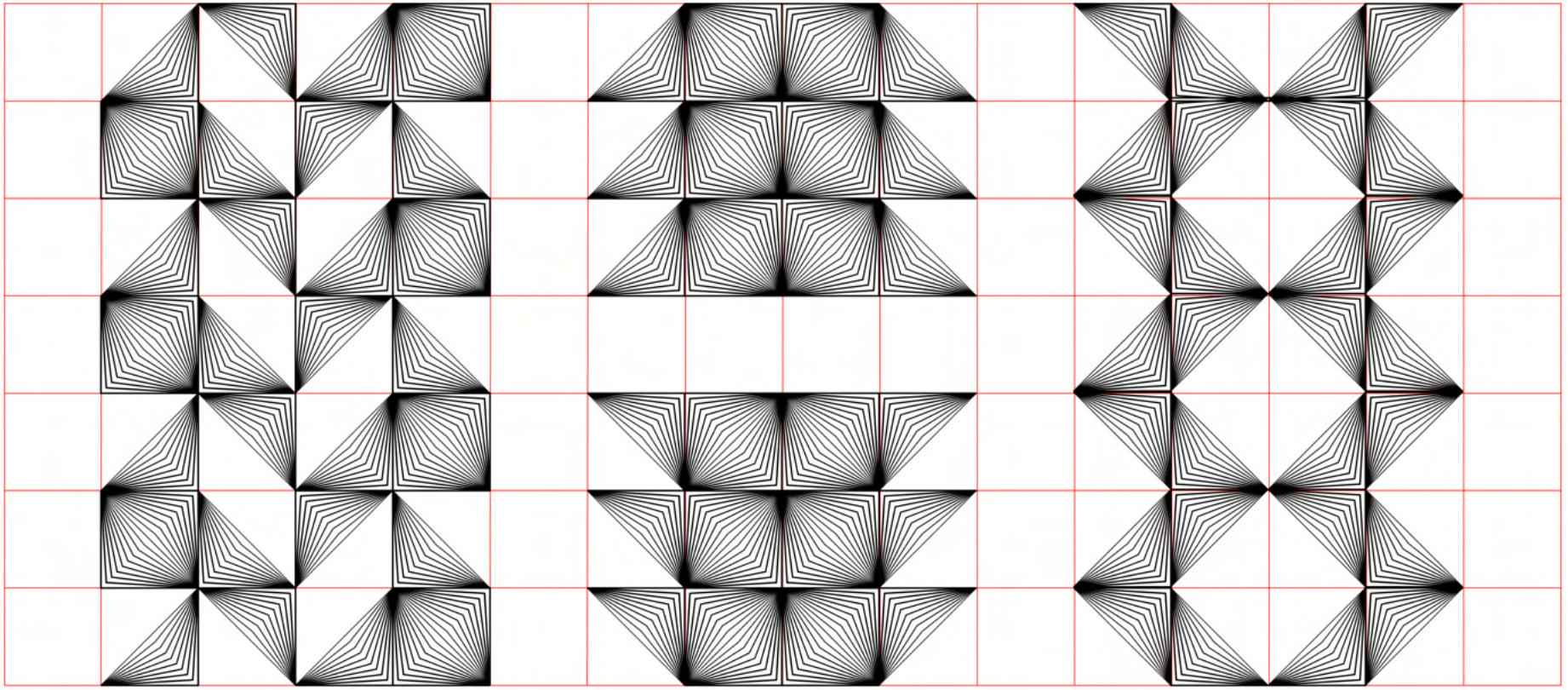
Activos horizontales simples



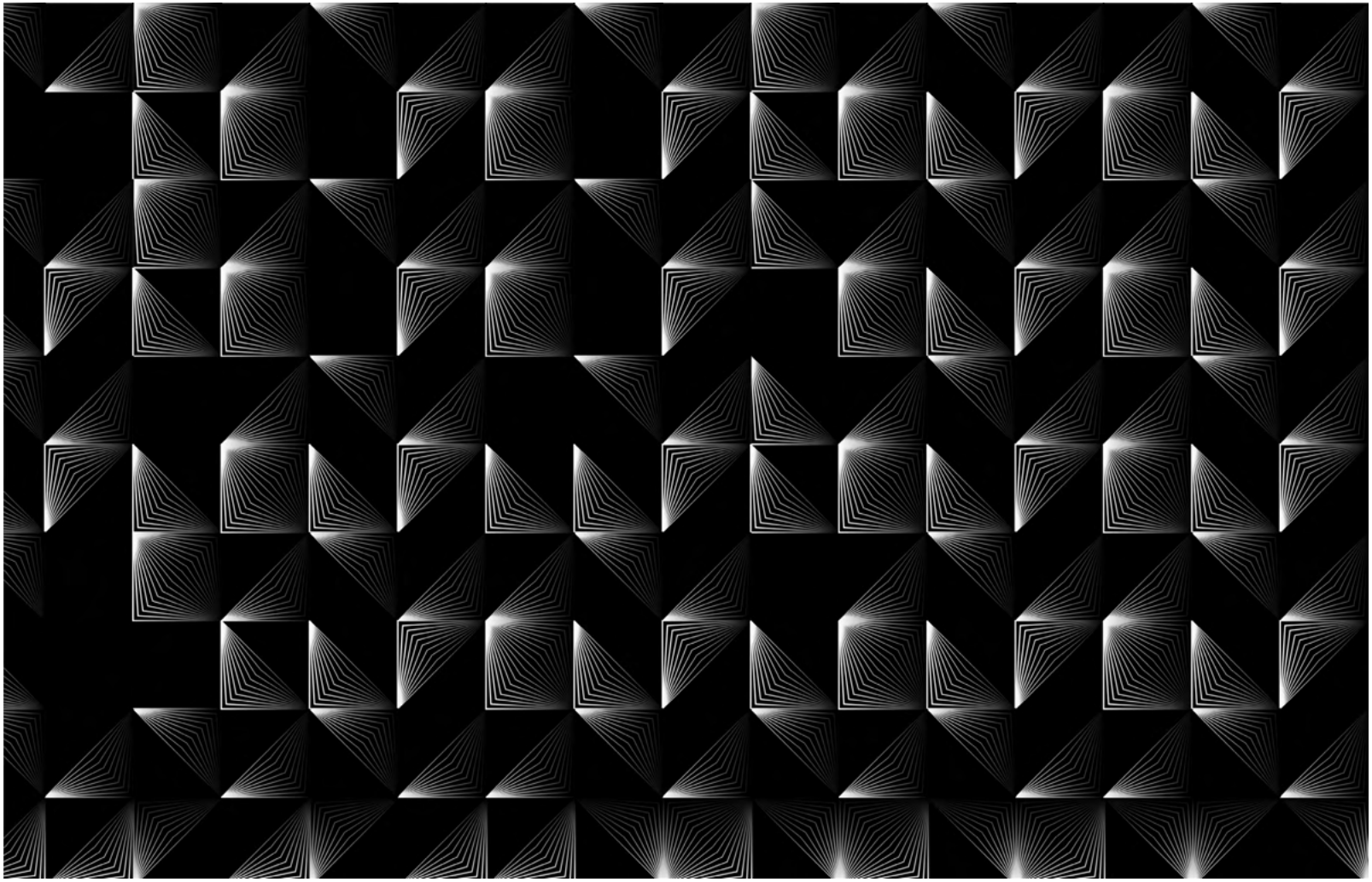


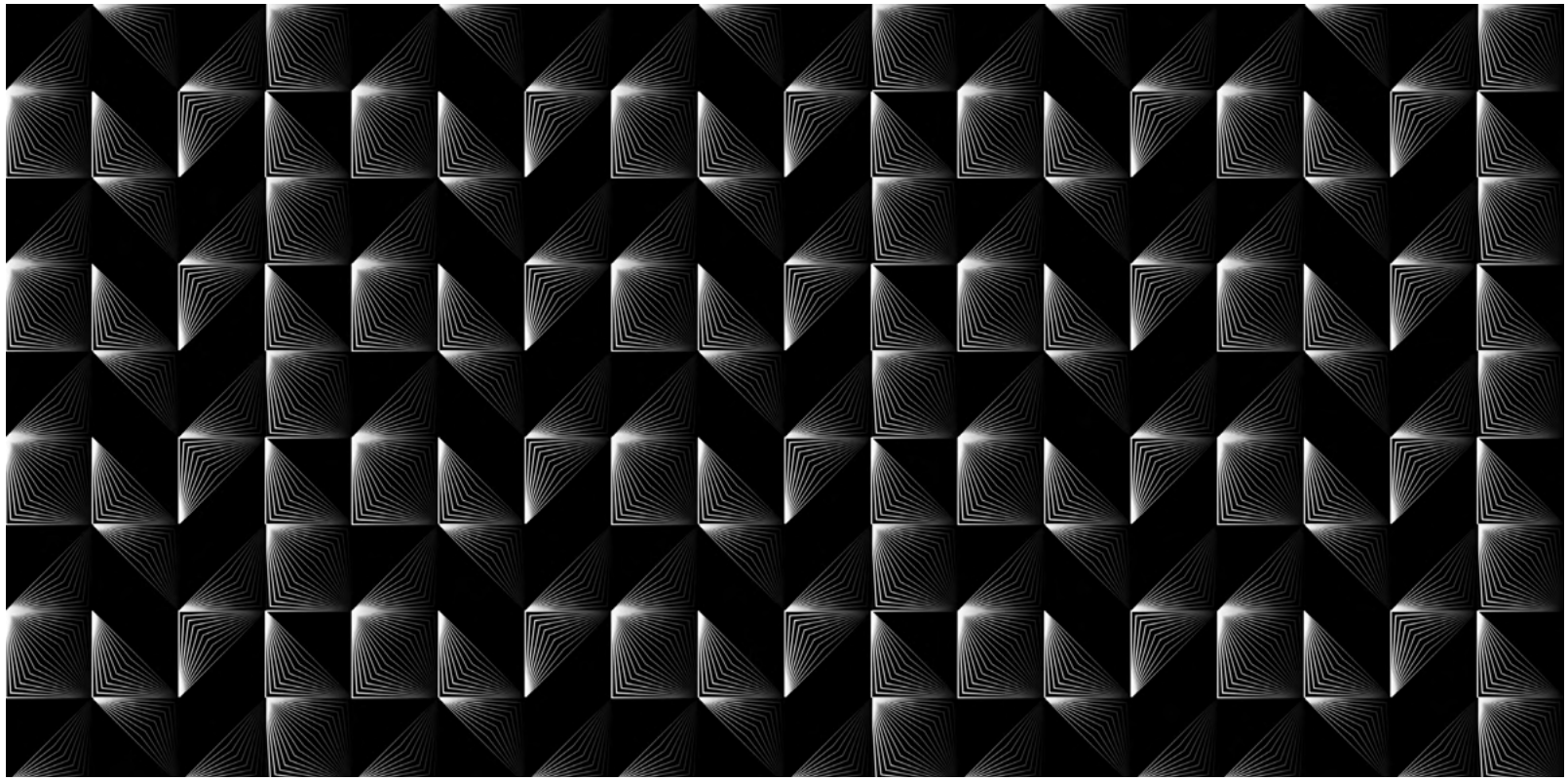


Activos verticales



Patrones

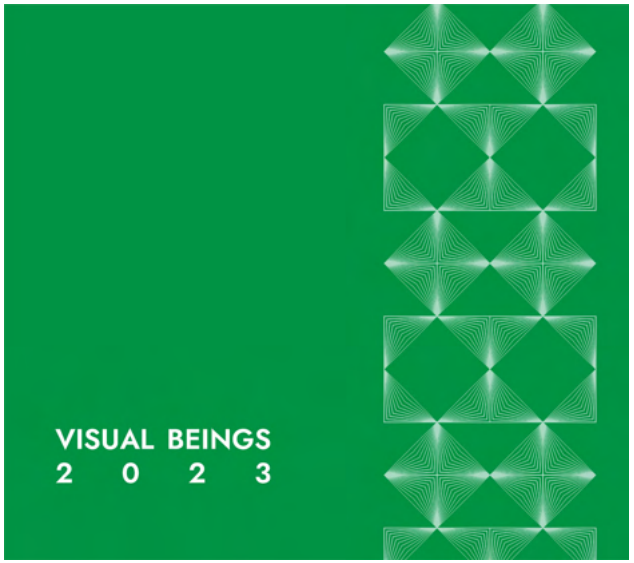


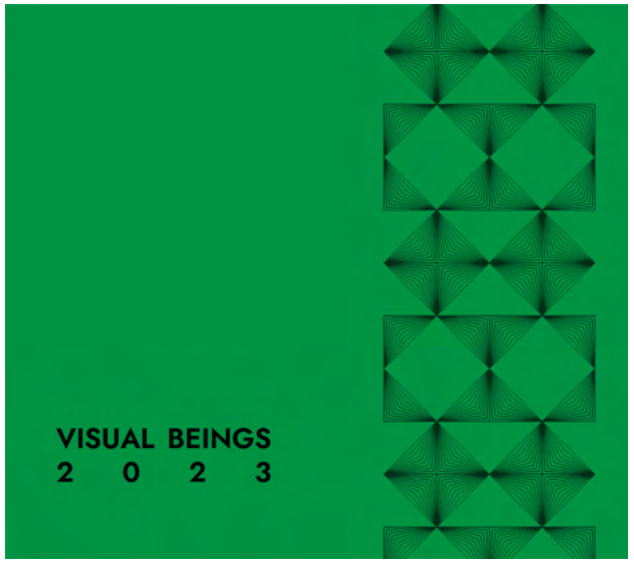


7. USO DEL COLOR

El color en nuestra identidad se utiliza exclusivamente como color de fondo para diseño de cartelería, banners etc, y nunca como color aplicable en nuestros módulos.

El color siempre llevará tipografías blancas encima, al igual que el trazo de los módulos también debe ser blanco (o con un degradado de blanco a negro) nunca negro.





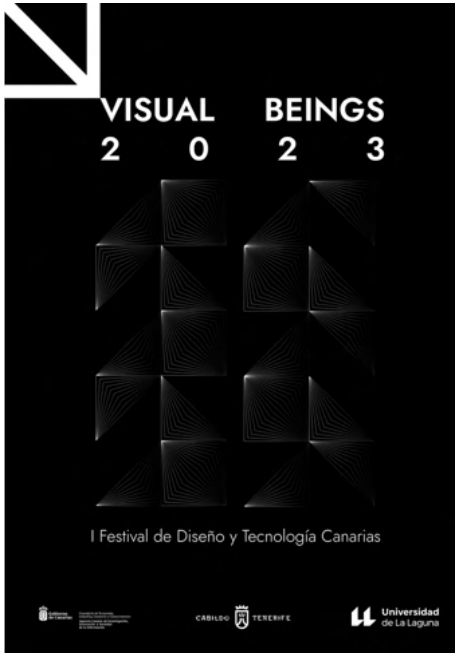
2. LIBERTAD DE USO

La ventaja del diseño modular flexible, es la libertad de uso que nos ofrece.

El diseño de esta identidad se creó con el fin de que el cliente tuviera total libertad de creación una vez fuera entregado el producto.

Nuestro activos están diseñados de tal manera que sean la idea central de cualquiera de nuestras creaciones, de tal manera que el uso del texto, queda relegado a un segundo plano pudiendo combinar de la manera que quiera el cliente el mismo (siempre y cuando sean las tipografías de identidad)

Se muestran a continuación algunos ejemplos.



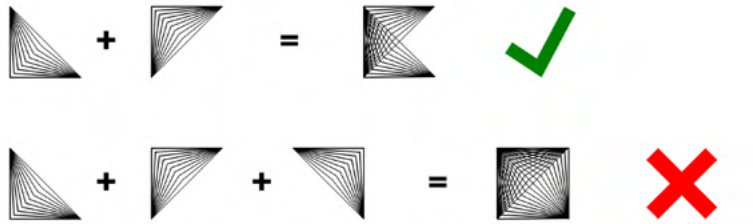
3. RESTRICCIONES

Si bien nuestra identidad nos permite infinitas posibilidades y formas de desarrollar nuestro diseño, debe haber algunas reglas claras que nos enseñen a diseñar con nuestro sistema.

Existen varias restricciones y recomendaciones de uso que se explicarán a continuación.

1. No debemos colocar ninguna tipografía sobre nuestros módulos

2. No debemos colocar más de dos módulos superpuestos.



3. No debemos reducir nuestro módulo más de 60 x 60 px.

Al tratarse de un módulo con un degradado lineal, al aumentar o reducir el módulo, los trazos seguirán siendo los mismos, por lo que la apariencia del módulo será distinta.

En aumento, los resultados son óptimos sin embargo cuando lo reducimos no.

4. TIPOGRAFÍAS DE IDENTIDAD

Jost SemiBold
Aa Bb Cc Dd Ee Ff
Gg Hh Ii Jj Kk Ll Mm
Nn Ññ Oo Pp Qq Vv
Ww Xx Yy Zz

Se usa fundamentalmente para el nombre del festival. VISUAL BEINGS.

Jost Medium
Aa Bb Cc Dd Ee Ff
Gg Hh Ii Jj Kk Ll Mm
Nn Ññ Oo Pp Qq Vv
Ww Xx Yy Zz

Se usa para el resto de información importante como localización o fecha.

Montserrat Light
Aa Bb Cc Dd Ee Ff
Gg Hh Ii Jj Kk Ll Mm
Nn Ññ Oo Pp Qq Vv
Ww Xx Yy Zz

Se usa para cualquier tipo de texto informativo.

5. PALETA CROMÁTICA

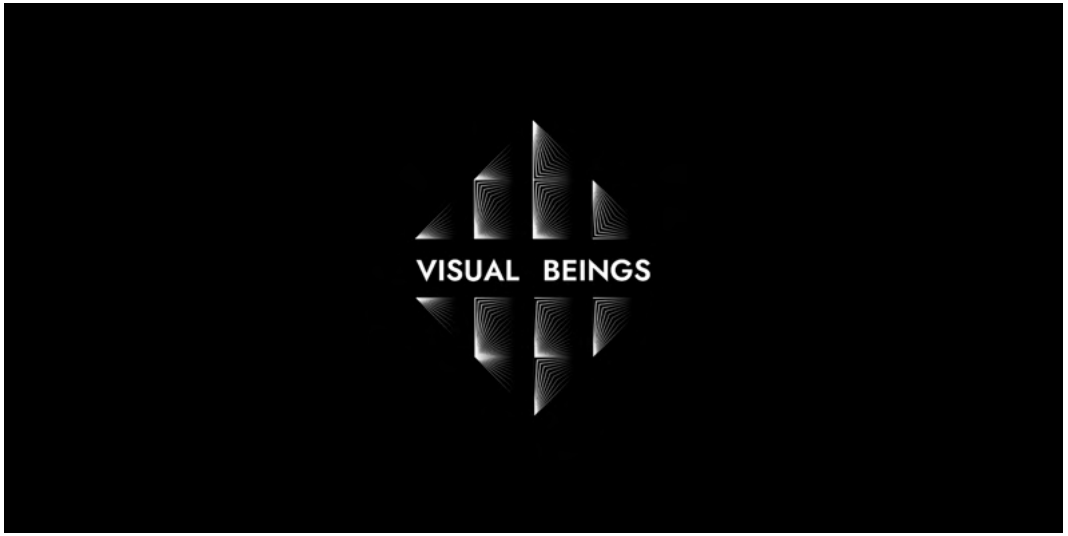
CMYK 0,0,0,100
RGB 0,0,0
HEX #000000

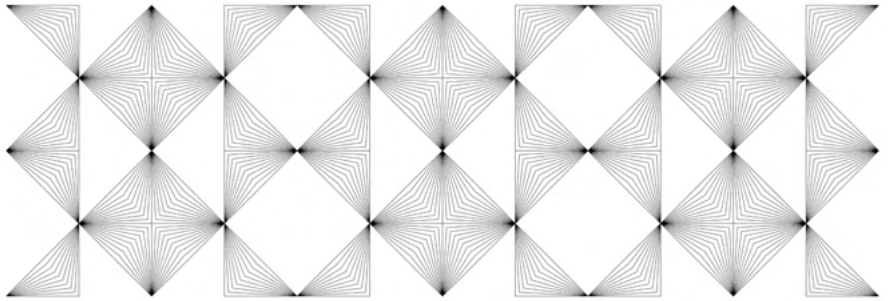
CMYK 0,0,0,0
RGB 255,255,255
HEX #ffffff

CMYK 84,14,93,2
RGB 0,168,69
HEX #009245

6. VERSIONES POSITIVAS Y NEGATIVAS DE LOS ACTIVOS

Nuestros diseños que incluyan texto, irán siempre con un fondo negro y tipografías en blanco. De ser lo contrario y no haber texto, si podemos usar la versión positiva de nuestros módulos, siempre que se usen como ilustraciones individuales.

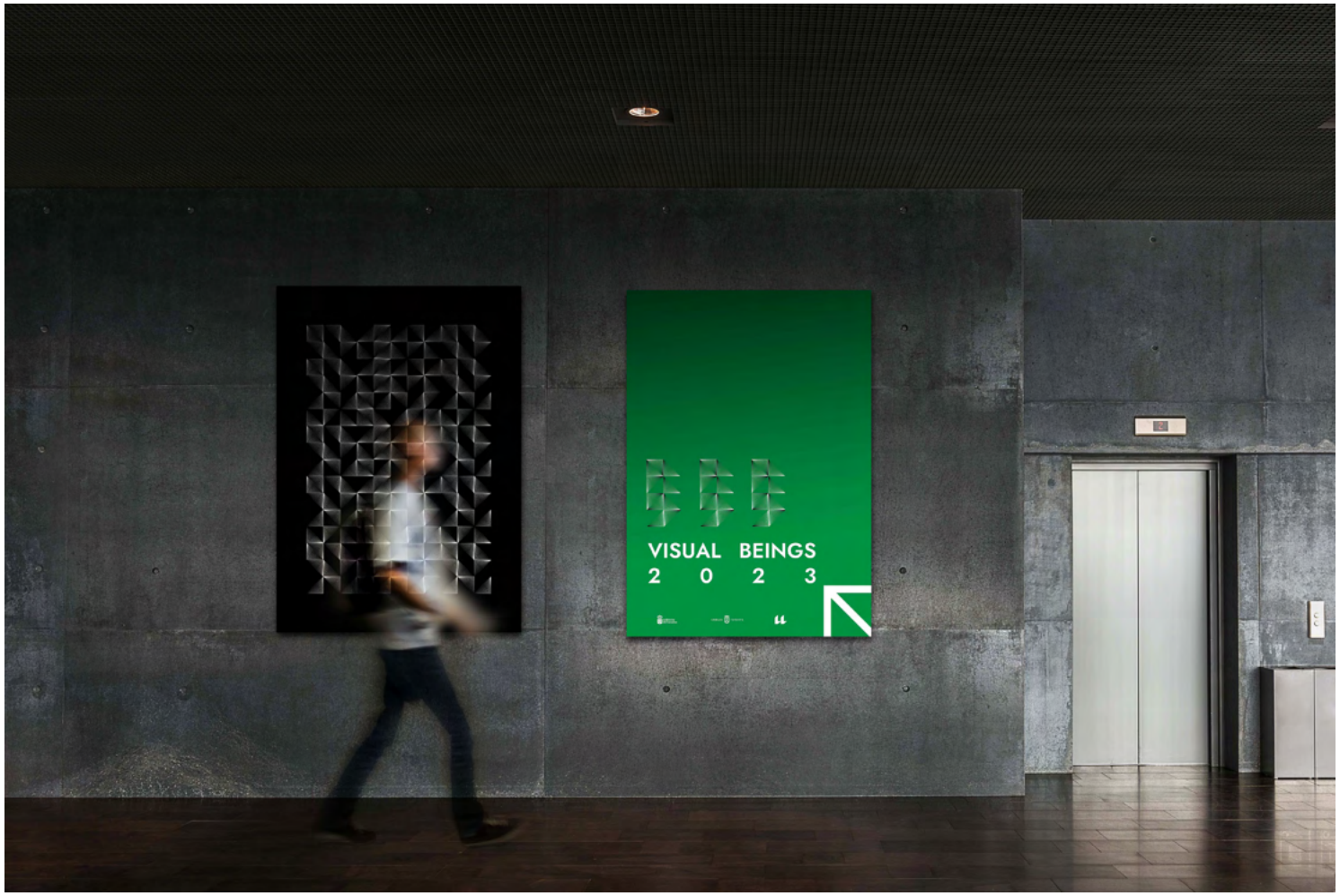




8. MATERIAL P.O.P Y MERCHANDISING

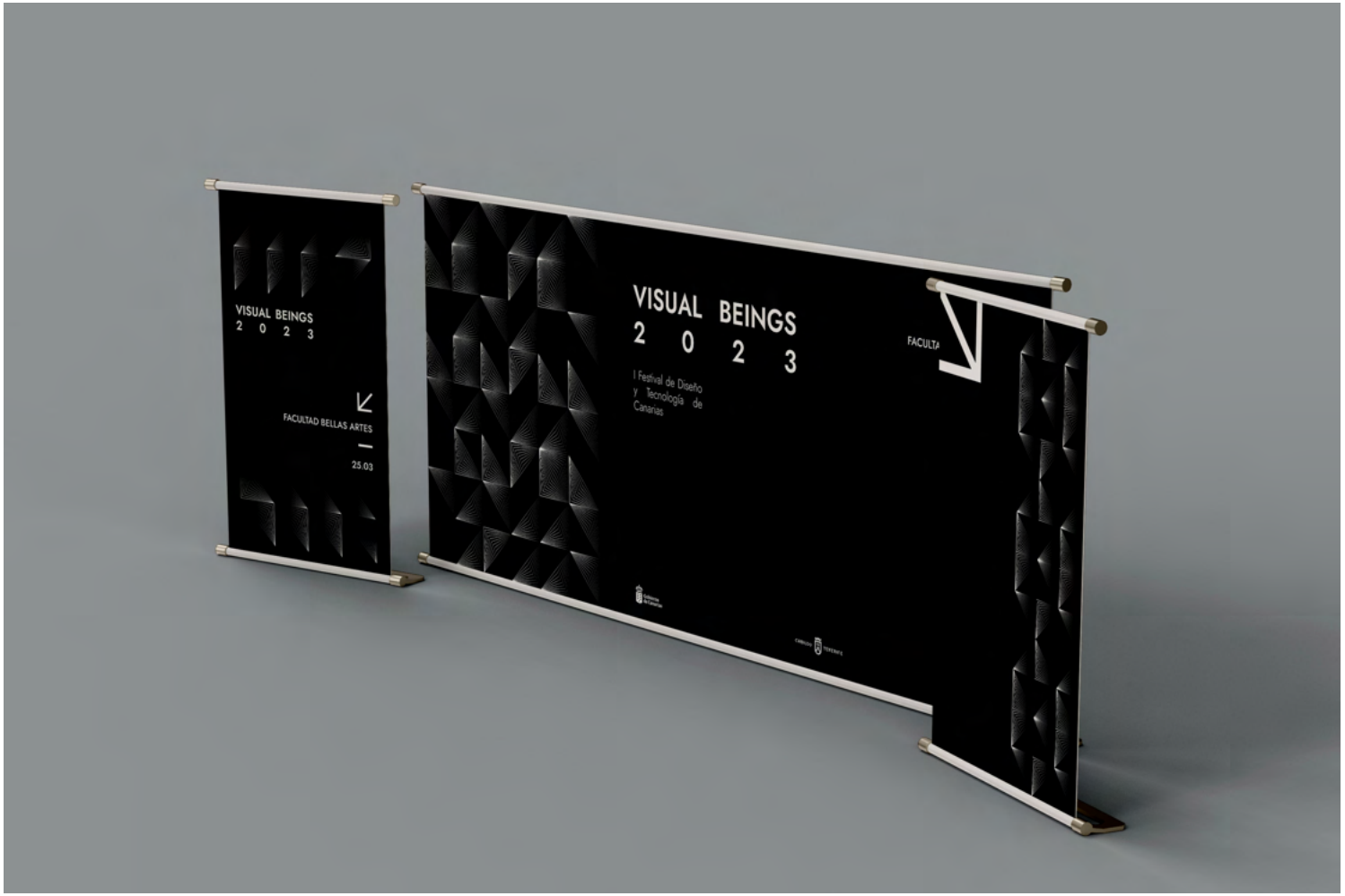
Se realizó una serie de cartelería con aplicación en póster, banners, mupi, banderolas y pegatinas.

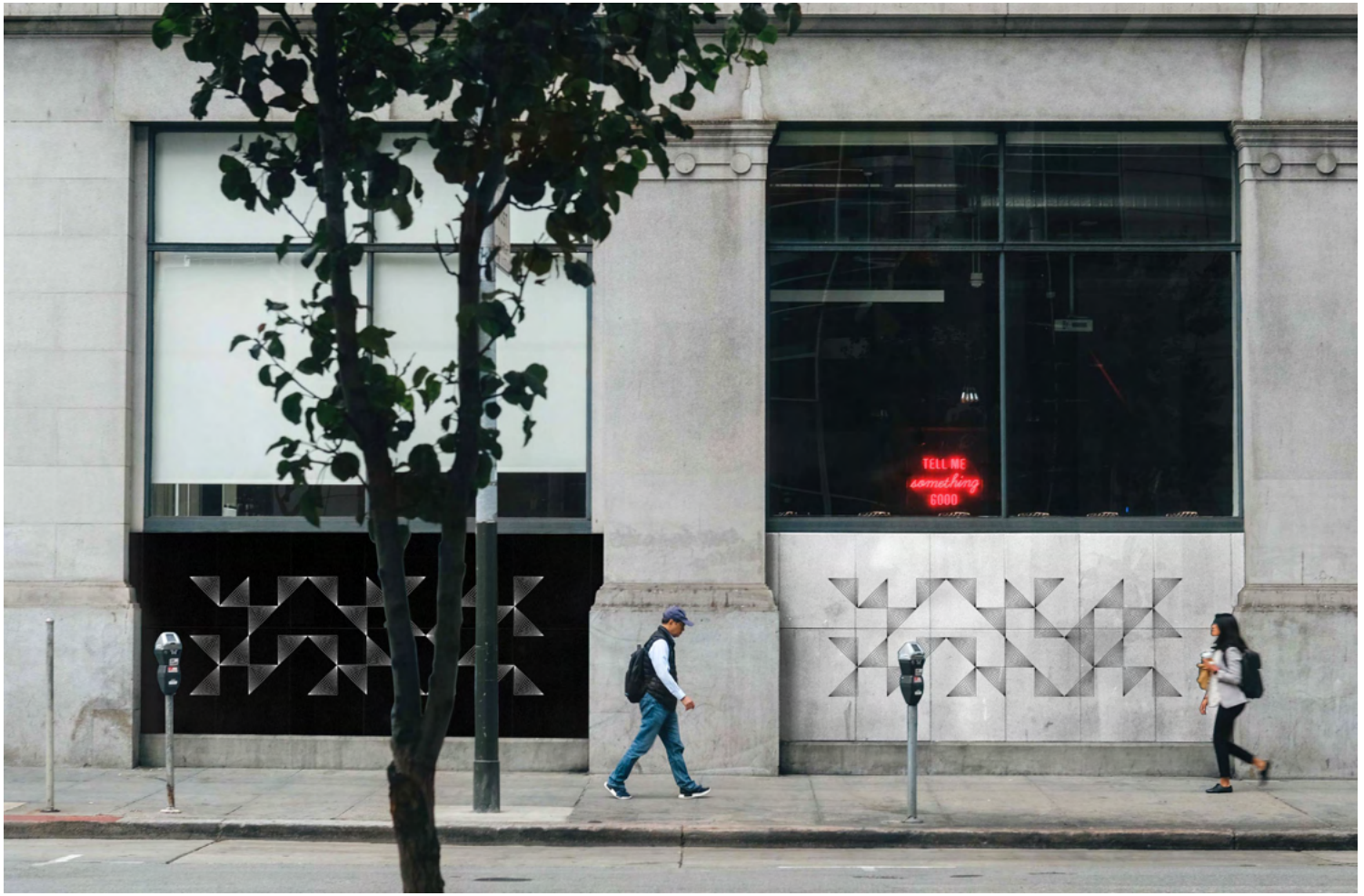
Además también se diseñan unas identificaciones para ponentes, entradas para el festival y un folleto informativo, además de unos totebags.

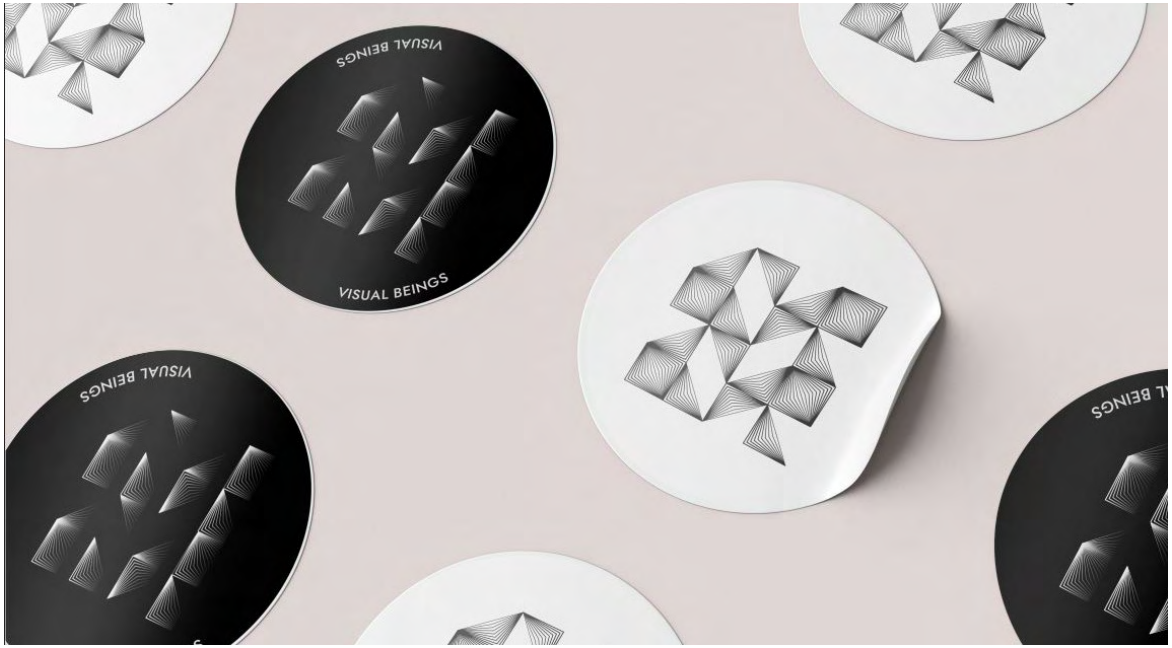

VISUAL BEINGS
2 0 2 3

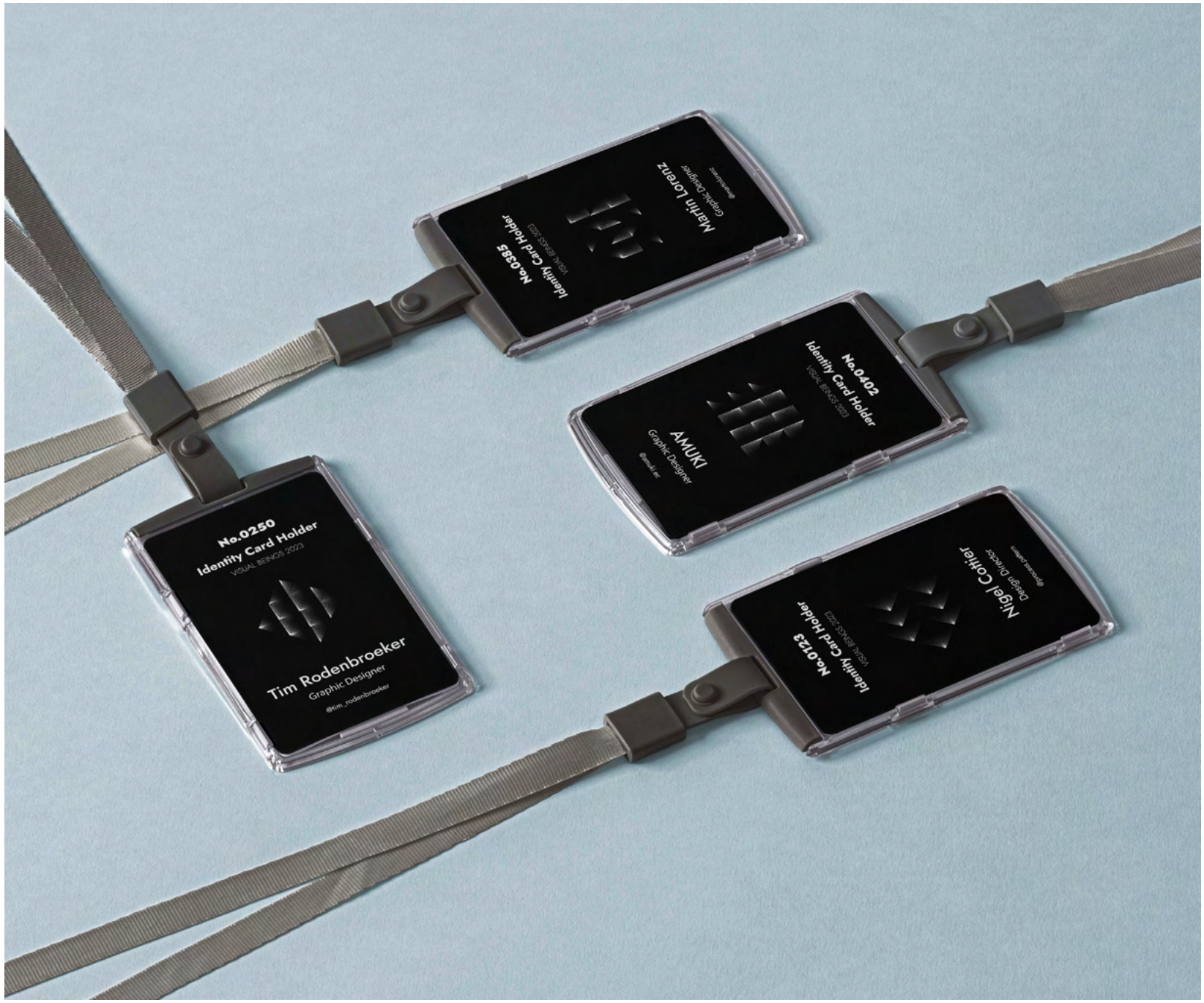

VISUAL BEINGS
2 0 2 3
I Festival de Diseño
y Tecnología de
Canarias


FACULTAD BELLAS ARTES
25.03
VISUAL BEINGS
2 0 2 3


VISUAL BEINGS
2 0 2 3
FACULTAD BELLAS ARTES
25.03
VISUAL BEINGS
2 0 2 3
I Festival de Diseño
y Tecnología de
Canarias
FACULTA


TELL ME
something
GOOD


VISUAL BEINGS
VISUAL BEINGS
VISUAL BEINGS


No.0250
Identity Card Holder
VISUAL BEINGS 2023
Tim Rodenbroeker
Graphic Designer
@tim_rodenbroeker
No.0402
Identity Card Holder
VISUAL BEINGS 2023
AMUKI
Graphic Designer
@amuki.ac
No.0385
Identity Card Holder
VISUAL BEINGS 2023
Martin Lorenz
Graphic Designer
@martinlorenz
No.0123
Identity Card Holder
VISUAL BEINGS 2023
Nigel Cortiier
Design Director
@nigelcortiier

20 €
ENTRADA GENERAL
I FESTIVAL DE DISEÑO Y TECNOLOGÍA
DE CANARIAS


VISUAL BEINGS
2 0 2 3
I Festival de Diseño y Tecnología Canarias
PROGRAMA
25.03
1. Inauguración del festival.
2. Presentación de los ponentes.
3. Mesa redonda.
26.03
1. Ponencias.
2. Taller código creativo con Tim Roderbroeker.
3. Taller de sistemas flexibles con Martin Lorenz.
27.03
1. Presentación de proyectos resultantes de los talleres.
2. Cierre del festival.
20 €
Entrada General
10 €
Entrada Estudiantes
BEINGS
2 3
20 €
Entrada General
10 €
Entrada Estudiantes


VISUAL
BEINGS

9. FORMATOS DIGITALES

El punto fuerte de esta identidad son justamente los formatos digitales.

Es en estos formatos que se demuestra la versatilidad de el diseño modular y además, se pueden ver todas las animaciones que han sido creadas con nuestra herramienta interactiva.

Esta herramienta es bastante fácil de adaptar a cualquier formato porque lo único que debemos hacer es escribir el formato que queremos al inicio del código, y el programa adaptará la cuadrícula a tal formato.

Para esta identidad se han realizado vídeos animados y Giff's, mostrando la versatilidad de los módulos y creando contenido que hará reconocible a la marca.

Se creó una página de Instagram (@visualbeingsfestival) donde se muestran estas películas, además de material informativo o simplemente ilustrativo.