En qué consiste el diseño generativo

PID_00267125

David Casacuberta

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

David Casacuberta

Como profesor de Filosofía de la ciencia en la Universidad Autónoma de Barcelona (UAB), su línea de investigación actual son los impactos sociales y cognitivos de las TIC, tema sobre el que ha publicado varios libros y artículos.

Actualmente es miembro del Grupo de Trabajo de Ética, Seguridad y Regulación de Bioinformática Barcelona e investigador del Grupo de Estudios Humanísticos en Ciencia y Tecnología (GEHUCT). También es codirector del máster de Diseño y dirección de proyectos para Internet de Elisava y participa como profesor en varios posgrados de gestión cultural, teoría del arte contemporáneo y diseño de tecnologías digitales.

Ha recibido el premio Eusebi Colomer de la Fundación Epson al mejor ensayo sobre los aspectos sociales, antropológicos, filosóficos o éticos relacionados con la nueva sociedad tecnológica con su libro Creación colectiva. También ha ganado el premio Ingenio 400, organizado por el Ministerio de Cultura y la Sociedad Estatal de Conmemoraciones Culturales, al mejor proyecto de net.art con su obra X-Reloaded (en colaboración con Marco Bellonzi).

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Quelic Berga (2019)

Primera edición: septiembre 2019
Autoría: David Casacuberta

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

1. Qué es el diseño generativo

El diseño gráfico inició una transformación profunda con la aparición del ordenador personal en los años 80 –especialmente el Macintosh, con programas específicos para el dibujo o la edición de texto, su interfaz gráfica y la inclusión de tipografías. Mientras los profesionales del diseño trabajaban duro para reconvertir su creatividad y sus habilidades en el nuevo entorno digital, otros profesionales, con sólidos fundamentos en ingeniería y programación, imaginaban otra forma de entender el arte y el diseño, consistente en crear procesos automatizados de diseño basados en matemáticas. Nacía así el diseño generativo.

¿Cómo podemos definir el diseño generativo? La Wikipedia conecta el diseño generativo con herramientas asociadas a la arquitectura y desde un modelo concreto de cómo automatizar los diseños, de manera que no resulta demasiado útil. Una búsqueda por internet os conducirá por caminos similares en los que cada definición se organiza desde una disciplina concreta y a partir de una técnica específica. Mi favorita es la de Lars Hesellgren, que afirmó que «El diseño generativo no busca crear un edificio. Quiere diseñar el sistema que construye un edificio». Aunque una vez más está basado en la arquitectura, creo que apunta a la distinción más relevante para entender las particularidades del diseño generativo. En síntesis, nos está diciendo que el objetivo final del diseñador es otro: ya no queremos hacer un cartel, una tostadora o un hospital: queremos crear un sistema que sirva para generar diferentes carteles, tostadoras u hospitales para poder decidir cuál nos parece más apropiado o, simplemente, ofrecerlos todos y que el cliente escoja. Los diferentes sistemas que creemos que serán los diferentes modelos de diseño generativo con los que queremos trabajar.

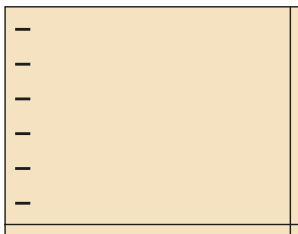
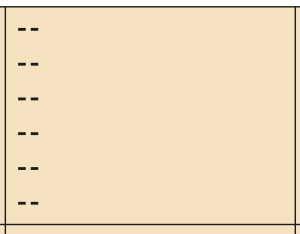
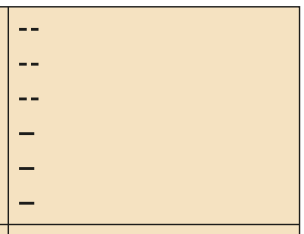
1.1. Diseño generativo analógico

Es importante no confundir diseño mediante ordenador con diseño generativo. La inmensa mayoría del diseño gráfico que se hace en la actualidad se hace con ordenador. Incluso la diseñadora más habilidosa y analógica del mundo probablemente acabe introduciendo la computadora en algún momento del proceso, aunque solo sea para escanear el dibujo que ha realizado y pasarlo a la imprenta. El ordenador facilita ciertamente el proceso de automatizar sistemas de diseño, pero no es obligatorio.

Considerad por un momento uno de los libros más antiguos de la historia: el *I Ching*. Este libro oracular consta de 64 hexagramas, todas las combinaciones posibles de seis elementos de una línea entera o partida. Cada una de estas combinaciones tiene un nombre: Así, el primer hexagrama, formado por seis líneas continuas se llama «El poder», el segundo, con seis líneas partidas, re-

presenta la «Recepción», etc. Cada hexagrama va acompañado por un texto que ofrece consejo sobre cómo actuar. No lo imaginéis como un horóscopo que predice que «te encontrarás por sorpresa con una vieja amistad que hacía años que no veías». Son más bien propuestas genéricas, de múltiple interpretación como «El movimiento del Cielo es poderoso. Del mismo modo, el noble se va fortaleciendo sin descanso».

Figura 1. Algunos hexagramas del I Ching

		
El poder	Recepción	Paz

Para recibir consejo del *I Ching* el lector primero formula en su mente la pregunta a la que quiere tener respuesta. Por ejemplo se cuestiona: «¿Cómo debo afrontar los ejercicios prácticos de este módulo de diseño generativo?» Tira tres monedas seis veces y apunta las caras y cruces que aparezcan. En función de cuántas caras y cruces surjan, el lector apunta una línea entera o partida. Ello le generará un hexagrama que será la respuesta a su pregunta. De una forma muy sencilla hemos creado así un sistema de lectura no convencional. En lugar de leer el libro de principio a fin, hemos creado un sistema que nos selecciona un hexagrama al azar. De hecho, el sistema de adivinación es algo más complejo de lo explicado aquí, pero como esto no es un texto de filosofía oriental, lo podemos dejar así.

De la misma forma, no hay nada más fácil que inventarse sistemas analógicos para diseñar generativamente. Podemos parametrizar aleatoriamente el diseño de un logo, haciendo que el color de fondo venga decidido por el lanzamiento de un dado, el tipo de tipografía por otro, o asociarlo a otros eventos azarosos, como el tiempo que hace hoy, echar una rápida ojeada a un reloj digital y utilizar el número de segundos que marca, etc. Unos párrafos más abajo tenéis un ejemplo de diseño generativo analógico y en la bibliografía tenéis algunas referencias curiosas para explorar el diseño generativo analógico.

De todas formas, tenemos que ver estos ejemplos como lo que son, curiosidades, pues sacaremos mucho más jugo a nuestro sistema de crear diseños gráficos si los implementamos en un software. Sin embargo, nos ha parecido importante insistir en este tema, pues apunta a una cuestión central sobre qué es el diseño generativo. Como decíamos en la definición anterior, no se trata de hacer un edificio, sino de crear un sistema que se encargue de hacer edificios. En un contexto de diseño gráfico, queremos un sistema generador de carteles que nos genere todo tipo de carteles. Para hacerlo, en lugar de crear nosotros el cartel con un software al uso como Photoshop o InDesign (o con papel y

rotuladores) lo que hacemos es crear una serie de instrucciones que definan el proceso por el que se va a crear el cartel. En el lenguaje de las ciencias de la computación a esa colección de instrucciones la llamamos «algoritmo».

1.2. ¿Qué es un algoritmo?

Un algoritmo no ha de ser algo misterioso y matemáticamente inescrutable.

> Un algoritmo son simplemente una serie de instrucciones que le damos a un agente (humano o digital) para que lleve a cabo una tarea.

Un software para jugar al ajedrez contra un ordenador está generado a partir de un algoritmo, pero una receta en un libro de recetas es también un algoritmo.

> El único requisito realmente vital a la hora de crear un algoritmo es que el agente sea capaz de interpretar cada instrucción del algoritmo de forma unívoca, sin crear confusión.

Así, si en una receta ponemos «poner dos litros de agua a hervir y añadir una pizca de sal», el chef que siga la receta sabrá qué hacer. En cambio, si decimos: «pon huevos, leche, harina, yogurt y ralladura de limón en un bol y lo mezclas hasta que quede la masa del pastel, la metes en el horno y lo sacas cuando esté hecho», esta receta es básicamente inútil, pues no especifica la cantidad de huevos, leche y harina que hay que poner, o el tiempo que ha de estar en el horno, por lo que no podremos ejecutar las instrucciones con seguridad.

Los humanos toleramos la ambigüedad mucho mejor que los ordenadores, por lo que un algoritmo que especifique un sistema de diseño generativo tendrá que ser mucho más preciso que una receta de cocina. Esta será una de las habilidades clave que aprenderéis a lo largo de este curso, especificar una serie de instrucciones, objetos y parámetros para que vuestro ordenador diseñe por vosotros.

Independientemente de cómo se ejecuta un sistema de diseño generativo, si es un algoritmo analógico que interpreta un humano o lenguaje de programación para que lo ejecute un ordenador, podemos caracterizarlo formalmente de la siguiente manera:

Un sistema de diseño gráfico generativo está formado por los siguientes componentes:

1) Un listado de los diferentes objetos gráficos que pueden aparecer en nuestro diseño.

2) Un listado de las diferentes propiedades que esos objetos pueden tener.

A 1 y 2 los llamaremos «ontología» pues explicitan el tipo de objetos que aparecerán en nuestro diseño y cuáles serán sus propiedades.

3) Una serie de reglas que indican, en función de lo que ya ha aparecido en el diseño gráfico, qué nuevo(s) elemento(s) hay que incluir, y qué propiedades han de mostrar. Estas reglas incluirán también instrucciones de cuándo el diseño se considera acabado y se debe detener la ejecución del proceso.

4) Un agente que ejecute las reglas y cree el diseño generativo.

5) Unos objetos (físicos o virtuales) que el agente use para seguir las reglas y desarrollar el diseño.

Veamos un ejemplo sencillo para diseñar el logo generativo de la UOC.

Ontología:

- El nombre «UOC», que puede aparecer:
  - En seis tipografías diferentes: Times New Roman, Helvética, Futura, Optima, Courier o Baskerville.
  - Dos tamaños diferenciados: pequeño o grande.

- Un contenedor que puede ser:
  - Círculo, dos círculos concéntricos, triángulo isósceles, triángulo equilátero, triángulo rectángulo, cuadrado, rectángulo, pentágono, hexágono, heptágono, octógono.
  - Ese contenedor puede ser: rojo, azul, verde, amarillo, naranja o blanco.
  - Si es blanco, entonces la línea que lo define puede ser roja, azul, verde, amarilla, naranja o violeta.

Reglas:

R.1 Lanzamiento de un dado para el nombre:
1 = Times New Roman
2 = Helvética
3 = Futura
4 = Optima
5 = Courier
6 = Baskerville

R.2 Lanzamiento de una moneda para tamaño del nombre:
Cara = Pequeño
Cruz = Grande

R.3 Lanzamiento de dos dados para el contenedor:
2 = Círculo
3 = Dos círculos concéntricos
4 = Triángulo isósceles
etc.

R.4 Lanzamiento de un dado para color contenedor:
1 = rojo
2 = azul
3 = verde
4 = amarillo
5 = naranja
6 = blanco

R.5 Si el lanzamiento de dados en R.4 = 6 (Color blanco) ir a R.7.

Si no:

R.6 Detener el algoritmo.

R.7 Lanzamiento de un dado para decidir el color de la línea del contenedor:
1 = rojo
2 = azul
3 = verde
4 = amarillo
5 = naranja
6 = violeta

R.8 Detener el algoritmo.

Ejecutor: Humano

Objetos para el ejecutor:

- Dos dados de seis caras
- Una moneda de un euro
- InDesign

Vamos a ver cómo funcionaría el sistema:

Lanzo un dado y saco un 3. Por lo tanto, el nombre «UOC» irá en Futura. Seguidamente lanzo una moneda al aire. Sale una cara, luego el tamaño del nombre «UOC» será grande.

Ahora lanzo dos dados y sale un 2. Así, el contenedor será un círculo.

Lanzo un dado y sale un 6. Ello quiere decir que el círculo será blanco con lo que tengo que aplicar la regla R.7 Tiro una vez más el dado y obtengo un 3 con lo que el color de la línea será verde.

Así pues, una vez ejecutado el algoritmo obtendría este logo:

Figura 2. Logo UOC

Sí, estamos totalmente de acuerdo. No voy a ganar ningún premio nacional de diseño con este logo, pero nos ha servido para ejemplificar qué es un diseño generativo y comprobar que no es nada misterioso ni hace falta tener un doctorado en inteligencia artificial para desarrollar diseños generativos.

2. Historia y evolución del diseño generativo

En esta apartado vamos a describir de forma breve cómo ha ido evolucionando y transformándose el diseño generativo. Ello nos servirá también para ofrecer un principio de clasificación de los diferentes tipos de diseño generativo que uno puede encontrar actualmente y las características de cada uno. No se trata de llevar a cabo un ejercicio académico historicista, ni tampoco de ofrecer una taxonomía sistemática. Es simplemente una primera visión de las diferentes formas en las que el diseño generativo se puede expresar.

2.1. Enter Maeda

Aunque hemos hablado en la introducción del ordenador personal como el punto de inflexión de la transformación del diseño, lo cierto es que el diseño generativo es de hecho anterior al ordenador personal. Ingenieros y programadoras que empezaron a experimentar con pantallas y ordenadores en los años sesenta ya vislumbraron las posibilidades que los entornos digitales ofrecían al arte y el diseño, y, siendo de formación matemática, sus proyectos y propuestas se basaban sobre todo en parametrizar formas geométricas conectadas, creando poderosos visuales de polígonos engarzados, estructuras fractales, etc. Estas prácticas son precedentes interesantes que marcan el camino a seguir a los futuros diseñadores.

Sin embargo, hay una figura clave, un diseñador que marca un antes y un después del diseño generativo. Se trata de John Maeda. John Maeda es un diseñador con formación en ciencias de la computación y que en los años noventa se dio cuenta de una serie de problemas en el desarrollo de la profesión de diseñador gráfico. Maeda observó, por un lado, la inexistencia de un lenguaje común entre diseñadores y desarrolladores de software. Las conversaciones típicas en aquella época entre un diseñador y una programadora colaborando en el desarrollo de un *website* eran del estilo de: «Quiero una estructura de contenidos similar a esta página de periódico, con la foto aquí, un encabezado allá, etc.» y la programadora respondía: «Eso es técnicamente imposible.» Debido a esta incapacidad de comunicarse, los diseñadores estaban limitados por lo que el software les dejaba hacer. El cartel de una película hecho con Photoshop o la *homepage* de un *website* desarrollado con Dreamweaver no era exactamente como el grafista esperaba que fuera, sino como la programadora había concebido el software. A este fenómeno, Maeda lo bautizó «La autocracia del PostScript.» Es decir, el acto de diseñar quedaba limitado por las características que los desarrolladores de software daban a sus programas.

Para Maeda, la única forma de evitar esta «autocracia del PostScript» era que la diseñadora o diseñador se construyeran sus propias herramientas. Es decir, que aprendieran a programar. Pero debido a la falta de lenguaje común entre

desarrolladores de software y diseñadores gráficos, los lenguajes de programación existentes en los noventa convertían los procesos de creación gráfica en algo muy farragoso y complejo, asociado a difíciles fórmulas matemáticas. Para evitarlo, Maeda desarrolló su propio lenguaje de programación, al que bautizó como Design by Numbers.

2.2. De Design by Numbers a Processing

Design by Numbers era un lenguaje pensado para enseñar conceptos de programación a diseñadores, y así facilitar la conversación con desarrolladores de software ofreciendo un lenguaje común.

No estaba realmente concebido para poder desarrollar creaciones funcionales que un profesional de diseño pudiera ofrecer a un cliente. Pero Design by Numbers ofrecía una novedad desconocida hasta entonces en el mundo de los lenguajes de programación: en Design by Numbers, para trazar una línea, un círculo o un polígono y darles color no hacía falta desarrollar complejas funciones algorítmicas. Todas esas acciones estaban recogidas en instrucciones primitivas. Así, una instrucción como «circle (56, 46, 55)» dibujaba un círculo con centro en las coordenadas x=56,y=46 con un radio de 55 píxeles.

Dicho de otra forma, Design by Numbers era el primer lenguaje de programación en el que las instrucciones primitivas eran conceptos que los diseñadores conocían a la perfección y por tanto, era la manera perfecta de introducir a un profesional de diseño gráfico en el mundo de la programación, y así poder plantearse aprender a programar sus propias herramientas.

Dos estudiantes de Maeda, Ben Fry y Casey Reas, fascinados por Design by Numbers, decidieron dar un paso más adelante y desarrollar un lenguaje de programación que fuera realmente funcional y que permitiera a un profesional del diseño gráfico crear sus propios diseños en dos dimensiones, así como todo tipo de proyectos de diseño de interacción.

Nació así Processing, un lenguaje de programación mucho más fácil de aprender y aplicar en el diseño gráfico y que permite a una diseñadora crear sus propias herramientas y no depender de la autocracia del PostScript, no estar limitada por lo que un software comercial le deja o no hacer.

Originalmente, Processing era muy básico; resultaba complejo incluso crear PDF profesionales del diseño que uno acababa de hacer, pero al ser un sistema de código abierto, libre y gratuito, mucha más gente se unió a colaborar con el proyecto, creando nuevos algoritmos, funciones y rutinas, con lo que ahora en Processing podemos generar carteles profesionales, maquetar libros, desarro-

llar complejas instalaciones interactivas, procesar datos de sensores y cámaras para crear videojuegos, etc. Si el Processing original se basaba en el lenguaje de programación Java, ahora también hay versiones para Javascript que facilitan el desarrollo de aplicaciones interactivas en la web, o la versión en Python, otro lenguaje de programación que muchos desarrolladores de software utilizan.

Processing será también el lenguaje de programación que utilizaremos en este curso, ya que es, con diferencia, el código mejor estructurado para hacer diseño generativo, y donde resulta más fácil hacerlo.

2.3. Diseño computacional

Maeda llamó a su estilo de diseñar «diseño computacional», para distinguirlo del diseño clásico y del «*Design Thinking*». En el *Design Thinking* quien diseña usa la empatía para establecer una conexión con el cliente y trabajar juntos para conseguir un diseño que realmente solucione los problemas que se le plantean al cliente. En el diseño clásico, el profesional del diseño es el experto que analiza el problema y ofrece la solución que le parece más adecuada, considerando el problema a analizar y el contexto concreto que lo define. Ese profesional del diseño probablemente trabaje con ordenador la mayoría de las fases del proceso.

Pero el diseño computacional no es simplemente diseño hecho con ordenador. Es diseño que aprovecha las capacidades del ordenador para generar infinitas variaciones de un tema y para construir herramientas específicas que permitan tratar los problemas de una forma individualizada y seguidamente automatizar esas soluciones.

> En el diseño computacional, el grafista deja de ser un creador de objetos visuales (carteles, portadas, *homepages*, etc.) para ser el creador de instrucciones que generen esos objetos gráficos de forma automática.

La analogía con la música nos puede ser útil. En música clásica distinguimos entre el intérprete y el compositor. El compositor toma una partitura y llena el pentagrama de símbolos (compases, claves, notas musicales, etc.). El intérprete toma esa partitura y la ejecuta, haciendo posible al público poder escuchar la composición del autor. Así, Johann Sebastian Bach compone en su mente una sonata solo para violín. La transcribe a papel pautado y así queda convertida en instrucciones. Un tiempo después –hasta siglos– una intérprete toma esa partitura y la ejecuta, haciendo así audible la música de Bach para nosotros una vez más.

El diseño computacional funciona de la misma manera. El grafista ya no genera él mismo un cartel, clicando botones y arrastrando objetos en el Photoshop. Escribe un programa en el que se describe un algoritmo para el tipo de cartel que queremos que se genere. Esa «partitura gráfica» la entregamos al ordenador, que hace de «intérprete» y hace visible el cartel para nosotros.

2.4. Un golpe de dados

La metáfora musical, aunque sin duda es útil, puede confundirnos y limitar excesivamente las posibilidades del diseño generativo. Cuando el violinista interpreta la sonata de Bach, su ejecución está muy bien delimitada. Cuando interpreta para el público, no se puede inventar notas, no puede repetir un pasaje que le guste especialmente a no ser que la partitura lo indique expresamente, no puede acelerar una parte que considere especialmente emocionante y, en general, ha de respetar todas las indicaciones que Bach dejó. Si pensáramos que el diseño generativo es exactamente equivalente a componer una sinfonía, el diseño generativo no sería muy diferente del diseño clásico: hay una solución concreta al problema que el cliente plantea. La única diferencia es que en lugar de que la solución la cree el grafista directamente la lleva a cabo el algoritmo.

Afortunadamente, el diseño generativo se parece más al jazz: el ejecutor de la pieza puede improvisar, inventarse notas, acelerar, frenar, repetir… Hay un esquema a seguir, pero hay también libertad creativa.

Los ordenadores no tienen libertad creativa *strictu sensu*, claro está, pero podemos conseguir algo similar añadiendo azar en los algoritmos que especifican cómo generar nuestra propuesta gráfica. En el ejemplo descrito en el apartado 1.2. de este módulo conseguíamos darle al algoritmo cierta libertad creativa al hacer que la elección de la tipografía, la forma geométrica del contenedor y su color dependieran del lanzamiento de diversos dados.

La mayoría de lenguajes de programación incluyen instrucciones para generar esos números aleatorios y disponerlos en la franja que queramos.

Generación de números aleatorios

Un ordenador sigue sus instrucciones de manera determinista, así que en realidad no podemos contar con él para generar auténtico azar, pero existen fórmulas matemáticas complejas que nos permiten crear números seudoaleatorios: números que siguen un proceso de generación tan complejo que para los efectos prácticos del diseño generativo son tan buenos como si fueran generados por un proceso realmente aleatorio, como el lanzamiento de dados.

Imaginemos que tenemos 100 fotografías de gatos que vamos a usar en un cartel, y queremos que sea el azar el que decida qué foto vamos a seleccionar. En Processing tenemos instrucciones muy sencillas para que el ordenador asocie un número del 1 al 100 a cada foto de gatos, generar un número aleatorio del 1 al 100 (como si lanzáramos un dado de cien caras) y poner la foto seleccionada

al azar. El proceso podría continuar, y podríamos decidir aleatoriamente en qué coordenadas ha de estar la foto del gato, y otro número aleatorio decidiría el tamaño final de la foto, etc.

2.5. Diseño evolutivo

Recordemos cómo funciona la selección natural. Hace millones de años, en África, los parientes cercanos de las actuales jirafas tenían un cuello corto. De entre los millones de antepasados de las jirafas que había entonces, algunos tenían el cuello más largo, con lo que podían comer hojas de ramas de árboles donde sus congéneres no llegaban. Esta habilidad extra facilitó su supervivencia y ayudó a que tuvieran más hijos. Sus hijos se parecían a ellos, y buena parte de ellos tenían también el cuello más largo. Algunos de esos hijos tuvieron hijos que tenían el cuello un poco más largo aún, con lo que se podían alimentar aún mejor, y así tenían más descendencia. A cada generación, el rasgo de tener el cuello más largo se habría ido seleccionando al ser útil en el contexto concreto de la sabana, y así, millones de años después, tenemos nuestras jirafas de cuellos larguísimos.

Los algoritmos genéticos son una forma alternativa de programar en la que el desarrollador del software imita este proceso de evolución por selección y crea diferentes programas al azar, cientos de ellos, para que procesen un problema. Al principio los programas lo harán realmente mal, y la solución que ofrezcan estará a años luz de la respuesta correcta. Sin embargo, al trabajar con cientos de ellos, siempre habrá algún programa que lo haga algo mejor que los demás. Esos programas quedan seleccionados y el resto eliminados. Se cruzan entonces los programas sobrevivientes entre sí, imitando el proceso de reproducción sexual y así se crea una nueva generación de programas. De esos programas, habrá unos pocos que lo seguirán haciendo fatal, pero se acercarán un poco más a la solución correcta. Esos serán los programas que sobrevivirán y se cruzarán. Al cabo de unas horas –afortunadamente aquí no hay que esperar millones de años como en biología– tendremos programas que solucionarán de manera óptima el problema que estábamos analizando.

El diseño evolutivo utiliza esta técnica para generar sistemas de creación gráfica. Imaginemos que queremos un fondo multicolor de puntos aleatorios para un cartel que estamos diseñando. Queremos que ese fondo muestre cierta estructura –no queremos aleatoriedad pura y dura– pero tampoco queremos algo rígido: nos apetecen zonas de colores marcadas, pero con transiciones sorprendentes. En lugar de hacerlo a mano, desarrollamos un pequeño programa de diseño evolutivo. Ese programa generará cien imágenes de píxeles multicolores aleatorias y nos las presentará en pantalla. Al principio, todas estarán muy lejos de la imagen que tenemos en mente, pero seguro que algunas se parecerán más que otras. Después de examinarlas, seleccionamos las cuatro que nos han parecido más interesantes y el programa nos creará una nueva generación de imágenes, resultado de combinar las cuatro imágenes que habíamos seleccionado de forma aleatoria. De esta iteración surgen cien nuevas

imágenes. Volvemos a seleccionar las cuatro más prometedoras y generamos un nuevo listado de cien imágenes. En unas cuantas iteraciones tendremos una imagen con exactamente la carga de estructura y aleatoriedad que buscábamos ¡y sin tener que hacer ni un solo dibujo!

También podemos tomar una posición intermedia, por ejemplo crear veinticinco imágenes de salida básica nosotros y entonces utilizar diseño evolutivo que combine al azar esas creaciones nuestras hasta conseguir algo nuevo, algo que recuerda nuestros diseños originales, pero que ha sido transformado y evolucionado por el algoritmo y nuestras selecciones.

Debemos la idea de utilizar los algoritmos genéticos para hacer diseño y arte evolutivo sobre todo a William Lathan y Karl Sims, que desarrollaron impactantes proyectos visuales interactivos que ayudaron a extender esta forma de entender el diseño generativo en la profesión y también en el arte contemporáneo. En la bibliografía encontraréis referencias a obras suyas.

2.6. Inteligencia artificial

Si observamos las diferentes fases del diseño generativo que hemos estado viendo hasta ahora, veremos que la tendencia general es desarrollar sistemas cada vez más autónomos en los que el individuo que diseña tiene cada vez menos capacidad de decisión. En el diseño computacional, el ordenador es el que lleva a cabo el diseño, pero este diseño ha sido pensado en su completitud por el autor, que ha decidido las propiedades de cada uno de los objetos gráficos que forman su propuesta. A través del azar dejamos que el programa improvise, y esta improvisación se convierte en el mecanismo central de creación en el diseño evolutivo.

Siguiendo esta tendencia, el siguiente paso lógico es dejar las decisiones en manos del sistema, quedando así el autor como un mero constructor de un algoritmo generativo. Este es el paso en el que nos encontramos ahora, en que diseñadores y artistas crean programas de inteligencia artificial que se ocupan de forma básicamente autónoma de generar las obras.

Un ejemplo clásico es el programa Aaron de Harold Cohen. Aaron es un sistema artificial que controla un plotter modificado que ha desarrollado su propio estilo de pintura y que es capaz de pintar del natural, desarrollando bodegones, retratos, etc. A diferencia de los casos anteriores, Cohen –el autor humano– no interviene directamente en el proceso creativo. En su momento desarrolló unas reglas de cómo el sistema aprendería a dibujar a partir de la información ofrecida por una cámara, y seguidamente el programa Aaron se encarga de plasmar esa imagen en un lienzo. Y Aaron no es simplemente una curiosidad de la ingeniería. Sus cuadros están suficientemente bien considerados como para ser expuestos en museos y galerías de arte y están bien cotizados entre los coleccionistas.

Actualmente hay una serie de proyectos de arte y diseño generativo fascinantes utilizando redes neuronales. En estos proyectos se le ofrece al programa una serie de ejemplos que se quieren desarrollar, por ejemplo, rostros de celebridades humanas, o cuadros famosos de la historia del arte, y el sistema, a través de un complejo proceso de aprendizaje en el que se localizan patrones relevantes en las imágenes, intenta generar sus propios ejemplos de cuadros o rostros famosos.

Red neuronal

Entendemos por red neuronal una forma de producir algoritmos inspirada en la manera en que trabajan las neuronas. Básicamente, se codifica una serie de inputs que queremos procesar en una primera capa de neuronas que representan esas entradas de forma numérica. Esa primera capa de neuronas está conectada a una segunda capa que procesa esas entradas. Cada neurona de la primera capa está conectada con las neuronas de la segunda capa, imitando las sinapsis que conectan las neuronas entre sí. Cada conexión entre neuronas tiene asociado un «peso», es decir, un número que amplifica o disminuye la señal que genera una neurona. Después puede haber más capas o no de neuronas intermedias hasta que llegamos a una capa final que ofrece el *output*, la respuesta asociada al *input*.

Las redes neuronales se «entrenan» a través de una serie de algoritmos que comprueban la distancia que hay entre la respuesta correcta asociada a un *input* y la que la red genera. Esos algoritmos van modificando los pesos de forma progresiva hasta que la red es capaz de generalizar el problema lo suficiente como para que haya una buena correspondencia entre el *input* y el *output*.

Por ejemplo, para enseñar a una red neuronal a reconocer letras manuscritas le damos como *input* gifs de letras escaneadas y de *output*, el código ascii de esa letra. Después de una serie de procesos de entrenamiento tendremos una red con unos pesos de conexión optimizados que le permiten reconocer diversas «a» manuscritas como una letra «a» y así ser capaz de transcribir un texto escrito a mano en un documento de texto.

El tipo de algoritmo más utilizado en estos procesos recibe el nombre de GAN, siglas de *Generative Adversarial Networks*. La idea es trabajar con dos redes neuronales: una de ellas recibe el *input* de una imagen e intenta determinar si pertenece al conjunto de imágenes de entrenamiento –diferentes caras de famosos, por ejemplo. La otra red neuronal genera imágenes intentando pasar el filtro de la primera. El proceso de creación y evaluación se va iterando, mejorando el funcionamiento de las dos redes neuronales, hasta que el sistema está convencido de que ha creado algo lo suficientemente bueno como para pasar por un ejemplo de las imágenes seleccionadas.

De hecho, es un poco como el diseño evolutivo que explicamos en el apartado anterior, pero en el que el humano ya no es necesario pues la fase de evaluación la lleva a cabo otra red neuronal.

En esta última iteración del diseño generativo, la persona encargada del proyecto solo tiene una función: decidir qué tipo de imagen quiere generar: ¿Se trata de generar rostros humanos creíbles para un proyecto de creación de bots en redes sociales? ¿O quizás queremos un sistema que cree retratos al estilo de Cezanne? La única tarea creativa aquí es hacer la selección del tipo de imágenes que nos gustaría ver replicada. El artista/diseñador no necesita ni siquiera crear el algoritmo, ya que existen algoritmos en el dominio público plena-

mente funcionales y que trabajan de forma muy genérica, de manera que solo son necesarias nociones de programación para saber dónde poner las imágenes, de qué forma han de ir organizadas y qué parámetros hay que tocar para maximizar el resultado final.

3. ¿Qué nos aporta el diseño generativo?

Superficialmente, podríamos decir que hacer proyectos de diseño generativo nos hace parecer modernos. Es una ventaja competitiva que tendremos ante otros profesionales más convencionales. Podemos hablar de inteligencia artificial, diseño evolutivo y otros términos que suenan cool, y nos dará una aureola de creador innovador que está a años luz de la competencia.

Afortunadamente, hay otras ventajas mucho más claras y funcionales en el uso del diseño generativo, que aportan verdadero valor a nuestros diseños, más allá de lo modernos que nos encuentren.

Por un lado la posibilidad de parametrizar un diseño nos permite llevar a cabo una serie de acciones que en el diseño convencional resultan muy complejas. La más clara e impactante es la capacidad de crear obras únicas. El diseño es un producto derivado de la revolución industrial, y piensa en sus creaciones como objetos repetidos, hechos en serie, con las facilidades y los problemas que ello comporta, tal y como analizó Walter Benjamin en su famoso texto *El arte en la época de la reproductibilidad técnica*. Si el diseñador convencional está atado al modo de reproducción industrial y a plantearse crear series de carteles, portadas de discos y libros, etc., un grafista generativo, apoyándose en las técnicas de creación y distribución digitales, puede crear mil carteles o mil portadas de libros y que todas sean diferentes entre sí. En lugar de ir a un cliente y darle a escoger entre tres opciones de logos puede presentarse con un sistema iterativo y jugar con parámetros y crear trescientos logos diferentes para ese cliente en una hora, hasta que los dos estén satisfechos. Y nada obliga a que ese logo se quede fijo. Le podemos vender directamente un sistema generador de logos y que su logo se transforme cada semana en un objeto gráfico diferente.

El diseño generativo también puede ayudarnos en momentos de bloqueo creativo. Cuando ninguna idea nos funciona, en lugar de buscar directamente la solución, podemos desarrollar un sistema generativo –por ejemplo, un diseño evolutivo– e ir mirando diferentes configuraciones hasta que una nos permita salir del bloqueo y tener una propuesta realmente interesante. Las mentes humanas se configuran desde preferencias y desagrados, muchos de ellos resultado de nuestra inclusión en una cultura concreta. Un algoritmo, al ser un mecanismo mucho más básico de creación, está paradójicamente libre de esos condicionantes culturales, y puede generar un diseño que nunca se nos habría ocurrido directamente.

Otra razón central es cómo el diseño generativo nos permite crear objetos gráficos que requirieran semanas de esfuerzo si se hicieran a mano o que directamente resultaran imposibles. Si John Maeda triunfa como diseñador en los

noventa es en buena parte porque muestra cómo con programación pueden hacerse cosas que resultan imposibles de hacer a mano o con el software comercial de edición gráfica. Intentar replicar el cartel Absolut Maeda donde este diseñador crea la silueta de la conocida marca de vodka con miles de líneas curvas es una empresa imposible sin programación.

Como hemos visto al hablar del diseño evolutivo, otro factor disruptivo del diseño generativo es su capacidad de recibir *feedback* en tiempo real y modificar el diseño a partir de las decisiones que un humano tome en directo. Cuando hablábamos de diseño evolutivo en el apartado 2.5., hablábamos más del diseñador que iba seleccionando la mejor imagen y así el algoritmo iba creando nuevas secuencias. Pero nada impide que el público tome la posición del diseñador y sea cada persona individualmente la que escoja cómo ha de ser su logo, su tarjeta de visita o el cartel que anuncia su exposición. Algunos de los proyectos de Karl Sims van en esa línea.

Pero la razón más importante por la que el diseño generativo importa es la cuestión del control. En una época en la que las herramientas del profesional del diseño las desarrollan ingenieros informáticos, es muy importante que entendamos cómo funcionan esas herramientas, cuáles son sus limitaciones y podamos crear nuestras propias herramientas para solucionar aquellas cuestiones para las que las herramientas de software comerciales son claramente insuficientes. En una época en la que uno puede predecir determinadas tendencias a partir de los nuevos filtros que aparecen para Photoshop, disponer de una herramienta propia, adaptada perfectamente a nuestros intereses, gustos y posibilidades, es la mejor forma de garantizar nuestra originalidad y valor como profesionales del diseño.

4. ¿Quién es el autor del diseño generativo?

Se podría replicar al apartado anterior que todo eso es muy bonito, pero que finalmente al utilizar este tipo de herramientas matamos al autor: es el programa el que crea el grafismo, no la persona. Mi respuesta ante esa objeción es que se está partiendo de una visión caduca de autor, que no tiene ningún sentido en el mundo digital y que en realidad el diseño generativo revaloriza el concepto de autor. Cuando la fuerza de un cartel viene dada en buena parte por los efectos y transformaciones que conseguimos crear en una fotografía a través de unos filtros de software, ¿quién es más autor? ¿Quién aplica los cuatro filtros comerciales que esta temporada están de moda? ¿O la persona que entiende cómo funcionan los filtros y ha creado sus propios filtros? Está claro que es la segunda persona, la persona que tiene control del diseño generativo.

> En realidad, todo diseño digital es un diseño generativo: aplicamos algoritmos para transformar nuestros objetos gráficos. La única diferencia es que el grafista que sabe programar tiene el control de sus propias herramientas.

Y por otro lado, nuestra concepción de autor está en continua transformación. El *Design Thinking* y la co-creación son dos ejemplos de tendencias muy fuertes en el diseño actual en las que el profesional del diseño interacciona continuamente con el público al que va dirigido su proyecto, con lo que la creación queda distribuida entre el grafista y el público participante en esas sesiones de co-creación. Si originalmente el diseño de un libro era trabajo básicamente de una persona, ahora tenemos todo tipo de profesionales que interactúan: diseñador, maquetador, ilustrador, impresor, etc. y no tenemos ningún problema en asignar a cada profesional su parcela creativa y su influencia en el diseño final. Cuando escuchamos a Glenn Gould interpretando a Bach a nadie se le ocurre decir que todo el mérito es de Gould y que Bach no pinta nada. No tenemos ningún problema en reconocer que hay dos procesos creativos en paralelo y que, finalmente, las variaciones Goldberg se las debemos a Johann Sebastian Bach y él es el creador de esas increíbles piezas para piano.

Nuestra apreciación del oficio del diseño ha de ir así deslizándose de la idea de creador que genera un objeto gráfico con sus manos a un creador que idea un sistema que generará objetos digitales únicos, parametrizados y adaptados al público que lo recibe y el contexto en el que se encuentra. Y en este nuevo contexto de apreciación tendrán cada vez más peso aquellas personas que, en lugar de utilizar sistemas prediseñados, son capaces de desarrollar sus propias herramientas para así no poner trabas a su originalidad creativa.

Bibliografía

General

Ceccato, C.; Hesselgren, L.; Pauly, M., Pottmann, H.; Wallner, J. (eds.). (2016). *Advances in Architectural Geometry 2010*. Birkhäuser.

Generative Design Blog <https://generativedesign.wordpress.com/>

Pérez, Eduardo (2018). «Diseño generativo: Inteligencia artificial y diseño». *Revista Código*. <https://revistacodigo.com/diseno/diseno-generativo/>

Roncoroni Osio, Umberto (2017). *Manual de Diseño Generativo*. Fondo editorial Universidad de Lima.

Soddu, Celestino. *Generative Design Futuring Past* <http://www.generativeart.com/ga2015_WEB/FuturingPast_Soddu.pdf>

Diseño generativo analógico

Munari, Bruno (2015). *Dibujar un árbol*. Anti.

Pacheco, Joshua. *Paths: An exploration of analogue algorithms*. <https://interface.fh-potsdam.de/gestalten-in-code/projects/analogue-paths/>

On Analogue Generative Design <https://www.youtube.com/watch?v=xVJF4ufv8Ng>

Yi Jing (2012). *El Libro de los Cambios*. Editorial Atalanta.

Diseño computacional

Flake, Gary William (1998). *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. MIT Press.

Maeda, John (2000). *Maeda @ Media*. Thames & Hudson.

Maeda, John (2001). *Design by Numbers*. MIT Press.

Maeda, John (2010). *Las Leyes de la Simplicidad*. Gedisa.

Processing <www.processing.org>

Reas, Casey; Fry, Ben (2014). *Processing: A Programming Handbook for Visual Designers and Artists*. <https://processing.org/handbook/>

Website oficial de Design by Numbers <https://dbn.media.mit.edu/>

Diseño evolutivo

Banzhaf, Wolfgang; Nordin, Peter; Keller, Robert; Francone, Frank (1998). *Genetic Programming - An Introduction*. San Francisco, CA: Morgan Kaufmann.

Climent, María (2018). «El Divino Diseño generativo. La máquina piensa ya como la naturaleza». Innovadores. *El Mundo*. <https://www.elmundo.es/economia/2018/01/08/5a5340be268e3e8a648b4610.html>

Latham, William. Homepage <https://www.doc.gold.ac.uk/~mas01whl/>

Martín, Santiago. *Diseño generativo y naturaleza*. Gijón: TEDX. <https://www.youtube.com/watch?v=C7UYPQJbP7o>

Núñez, Laura (2019). «¿Qué son los algoritmos genéticos?» *El País*. <https://elpais.com/elpais/2019/01/31/ciencia/1548933080_909466.html>

Sims, Karl. Homepage <https://www.karlsims.com/>

Diseño e inteligencia artificial

Cohen Harold. Homepage <http://www.aaronshome.com/aaron/index.html>

Knight, Will (2018). «Las caras falsas imaginadas por una nueva IA son cada vez más reales». *MIT Technology Review*. <https://www.technologyreview.es/s/10818/las-caras-falsas-imaginadas-por-una-nueva-ia-son-cada-vez-mas-reales>

Elgammal, Ahmed y otros (2017). *CAN: Creative Adversarial Networks Generating «Art» by Learning About Styles and Deviating from Style Norms*. <https://arxiv.org/pdf/1706.07068.pdf>

Colectivo Estampa. *El mal alumne*. <https://tallerestampa.com/estampa/el-mal-alumne/>

Greene, Tristan (2018). «Someone paid $432K for art generated by an open-source neural network». *The Next Web*. <http://thenextweb.com/artificial-intelligence/2018/10/25/someone-paid-432k-for-art-generated-by-an-open-source-neural-network/amp/>