Estructura de un diseño generativo

PID_00267126

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: Antoni Isarch Borja

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

Índice

1. Pensar tu proyecto de diseño generativo 5

2. La ontología de un diseño generativo 8
2.1. Fijo frente a variable 8
2.2. Determinista y secuencial 9
2.3. Secuencialidad frente a aleatoriedad 11

3. Datos externos 13
3.1. Bases de datos 13
3.2. Procesamiento de datos y grados de generalización 14
3.3. Interactividad y personalización 15
3.4. Datos en vivo. Animaciones 16

4. Hibridación 18

Bibliografía 19

1. Pensar tu proyecto de diseño generativo

Tal y como hemos visto en el módulo «En qué consiste el diseño generativo», hay muchas formas de desarrollar un diseño generativo. Ni siquiera estamos obligados a utilizar tecnologías digitales para llevarlo a cabo. Disponemos de diversos paradigmas que hemos apuntado en el citado módulo: ser un grafista computacional como Maeda, utilizar algoritmos de *deep learning* como las *Generative Adversarial Networks* o bien algoritmos genéticos en los que jugamos a ser la naturaleza y decidir qué diseño será el que finalmente sobreviva.

Es importante que entendamos el cambio conceptual que implica movernos al diseño generativo. Al hacerlo, pasamos de ver el diseño gráfico como un proceso manual, de artesanía, a un proceso intelectual, de conceptualizar un problema, desarrollar un modelo, ejecutarlo, valorarlo, revisar nuestras apreciaciones, modificar el modelo y volverlo a probar.

Si nos lanzamos directamente al editor de código y empezamos a meter instrucciones «a ver qué pasa», los resultados serán probablemente decepcionantes e inútiles en un proyecto serio. Antes de programar es necesario planificar muy bien a dónde queremos llegar y qué procedimientos vamos a utilizar para llegar a esos objetivos. Parafraseando a Humpty Dumpty de *Alicia a través del espejo*: solo cuando no te importe a dónde quieres llegar te dará lo mismo el camino que elijas.

Esta planificación ha de organizarse desde tres fases diferenciadas:

- En una primera fase analizaremos el problema que queremos resolver, ya sea un objetivo propio o el encargo de un cliente. Describiremos los diferentes condicionantes asociados al proyecto y cómo interactúan entre sí, estableciendo qué interacciones ayudan a resolver el problema, cuáles lo dificultan y cuáles parecen irrelevantes para nuestros objetivos. Una vez entendido el contexto, procederemos a buscar una solución factible al problema planteado a partir de la información que hemos recopilado.

- En una segunda fase definiremos una estructura formal que caracterice la solución generativa que estamos buscando, especificando los diferentes objetos gráficos que van a formar parte de la solución y las reglas que los conectan entre sí al estilo del ejemplo que vimos en el apartado 1.2. del módulo «En qué consiste el diseño generativo».

- En una tercera fase testearemos el diseño gráfico que hemos ideado en el apartado anterior, viendo si consigue los efectos esperados o no. Si vemos que todavía quedan cosas pendientes que no funcionan, volveremos a la segunda o a la primera fase, dependiendo de si la falta de adecuación es

una cuestión de la estructura de nuestro diseño generativo (y entonces volveríamos a la fase 2) o si no hemos tenido en cuenta condicionantes relevantes (y entonces tendríamos que volver a la fase 1). Cuando sintamos que la solución realmente es eficiente y resuelve el problema que queríamos solucionar, daremos los últimos ajustes a nuestro modelo y procederemos a su implementación gráfica final. Este tipo de proceso recibe el nombre de iteración.

Tal y como argumentan autores como Nigel Cross, en una acción diseñada, problema y solución están íntimamente conectados. Es lo que en la literatura de teoría del diseño se describe como un *wicked problem*: un problema que en realidad no está bien definido, y en el que, inevitablemente, en el proceso de ofrecer una solución el problema se reinterpreta y se reescribe, creándose así un interesante bucle de interconexiones entre problema y solución.

Dicho de otra forma, cuando diseñamos no hemos de pensar que hay una solución única que es la correcta. En el proceso de buscar la solución, examinaremos el problema desde otras perspectivas, y es fácil que nos replanteemos el problema original y le demos la vuelta. Al replantear el problema, tendremos que volver a pensar la solución. Y el proceso puede irse repitiendo…

Una ingeniera se enfrenta normalmente a problemas bien definidos del tipo «construye un mecanismo de seguridad para un ascensor que sea capaz de resistir tantas toneladas de peso y haga que el ascensor baje a una velocidad máxima de tantos metros por minuto». Gracias a eso puede ofrecer una solución factible que se adapta científicamente a todos los constreñimientos. En cambio, un problema en diseño gráfico del tipo «Diseña un cartel que haga que la gente tenga interés en comprar mi libro» es mucho más abierto: tendremos que saber de qué va el libro, a qué públicos va dirigido, dónde se piensa distribuir los carteles, etc. Cada posible solución gráfica que propongamos tendrá más efecto en un entorno que otro, y llamará más la atención a un colectivo que a otro, por lo que según avanza el proceso de diseño la pregunta se irá transformando. Si decidimos hacer un diseño generativo, nuestro cartel probablemente atraiga más a gente joven, curiosa, que busca cosas nuevas, de manera que la pregunta original se transformará y, en lugar de preguntarnos en general sobre cómo generar interés en «la gente», tendremos posibles nuevos problemas como «diseña un cartel generativo que llame la atención a la gente joven que busca novedades pero al mismo tiempo no desanime a personas de gustos más clásicos». Pero también puedo generar otro tipo de situación como «diseña un cartel generativo que resulte tan llamativo y chocante que genere un gran interés entre el público más radical aunque provoque rechazo en personas de gustos clásicos».

En este módulo nos centraremos en la segunda fase de este proceso: cómo pensar la estructura formal de un proyecto generativo, previo a su implementación en un lenguaje o entorno de programación específicos. Las otras dos fases quedan fuera de los objetivos de este módulo.

De hecho, la primera fase se centra en los mecanismos creativos del profesional del grafismo, y sería complejo intentar desarrollar una teoría sistemática sobre cómo analizar el problema y generar una solución. Cada diseñadora y diseñador tienen su propio sistema: los hay más analíticos y los hay más intuitivos. La forma en que se procesa un problema de diseño dependerá también de la veteranía del profesional: una persona novata puede dedicar semanas a reflexionar sobre la mejor tipografía para un logo mientras que a un profesional se le aparece la solución en pocos segundos.

La tercera fase tiene diferentes ejemplificaciones en función del tipo de encargo que nos hagan y el modelo de diseño que tengamos en mente. Un grafista clásico con el encargo de solucionar cuestiones de comunicación gráfica con un cliente concreto podrá despachar la fase de evaluación con unos pocos encuentros con el cliente. Una diseñadora interesada en el *design thinking*, la co-creación o el diseño inclusivo tendrá que asegurar mecanismos de *feedback* directos con el público mucho más reglamentados y sistemáticos.

2. La ontología de un diseño generativo

Una vez tenemos una posible solución, ya sea bocetada, en un diagrama de flujo, en seudocódigo, o en nuestra forma favorita de representar la solución, tenemos que establecer de forma más exhaustiva qué elementos formarán parte de nuestra solución gráfica y de qué manera estarán organizados. En la jerga de las ciencias de la computación este proceso se conoce como ontología.

Originalmente, la ontología era una disciplina filosófica eminentemente abstracta, que se ocupaba de establecer qué es el ser. La parte más oscura y esotérica de la metafísica. Afortunadamente para nosotros, el término ha ido evolucionando y ahora significa básicamente establecer qué objetos forman parte de un sistema, qué propiedades pueden tener y cómo se interrelacionan unos objetos con otros. En el ejemplo de algoritmo generativo analógico para construir un logo de la UOC vimos un ejemplo muy simple de ontología.

La ontología en un sentido informático es una ciencia compleja, que intenta inventariar las propiedades de diferentes objetos y sus relaciones para tareas como enseñar a una inteligencia artificial. Hay ontologías muy completas que, por ejemplo, permiten organizar las informaciones que uno encuentra en la Wikipedia en formato clasificación, de manera que uno puede hacer búsquedas del tipo: encuéntrame todos los científicos nacidos en el siglo XX en Suiza un viernes.

A primera vista puede sonar a pura mística: ¿una ontología de toda la Wikipedia? Pero en realidad son sistemas que pueden consultarse de forma sencilla cuando uno aprende a programar. Quizás un día, si vais avanzando en el desarrollo de proyectos de diseño generativo, os planteéis conocer cómo acceder a ese tipo de ontologías para desarrollar proyectos realmente ambiciosos. Pero ahora no os preocupéis: la idea que vamos a trabajar aquí de ontología es totalmente interna, incluirá solo los objetos que formen parte directa de vuestro proyecto gráfico y serán muy fáciles de desarrollar.

2.1. Fijo frente a variable

Salvo que nuestro plan sea generar azar absoluto en nuestros gráficos –algo que finalmente resultará aburrido– la ontología de nuestro proyecto deberá incluir elementos fijos para hacer de contraparte a los elementos variables.

Ahí está precisamente el poder de los gráficos generativos: poder observar la aleatoriedad dentro de una cierta estructura que le da sentido. O, visto desde el otro lado, disponer de una estructura parcialmente estable pero que se adapta y transforma en función de cómo cambia el entorno.

Así, la primera tarea será especificar qué elementos van a darse de manera fija y cuáles serán variables. Esos elementos invariantes son muy importantes, pues son los que determinan una estructura básica de relaciones que darán sentido a los elementos variables.

Imaginad un cliente que os pide un logo para su empresa, GeneTex. El cliente tiene una mente abierta en general, pero hay algo en lo que no está dispuesto a ceder: la estructura básica del logo ha de ser un círculo violeta con el nombre de la empresa dentro. El tamaño y la tipografía del nombre podrán variar, pero dentro del círculo ha de estar solamente el nombre de la empresa y el color de ese círculo ha de ser violeta.

Así, la ontología de nuestro diseño gráfico sería más o menos así:

- Contenedor:
  - Forma: círculo (invariable)
  - Color: violeta (invariable)
  - Tamaño: variable

- Contenido:
  - Texto: GeneTex (invariable)
  - Tipografía: Time News Roman, Verdana, Baskerville, Futura, etc. (variable)
  - Tamaño letra: de 10 a 24 puntos (variable)

2.2. Determinista y secuencial

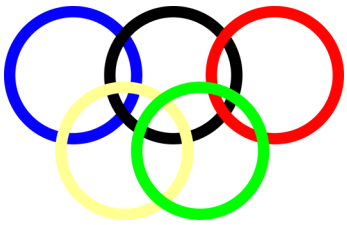
El diseño generativo, en el fondo, es como un arte en el que hay que calibrar un equilibrio entre los elementos que ya vienen prefijados por la grafista y los que queremos que sean variables.

Si dejamos todo en manos del azar, el resultado final acabará siendo indistinguible de ruido puro y duro. Si todos los pasos están determinados de antemano, el resultado acabará siendo equivalente al diseño convencional, con lo que perdemos la capa generativa.

No hay ningún problema en desarrollar un diseño computacional plenamente determinista, en el que hemos decidido de antemano de qué manera se van a comportar los elementos y somos capaces de visualizar *a priori* el aspecto que tendría nuestro objeto gráfico final. Imaginemos que nos encargan un rediseño de los aros olímpicos, el logo de las Olimpiadas. Queremos hacer un diseño muy preciso, en el que las distancias de los círculos sean perfectamente equidistantes, de manera que en lugar de dibujar el logo a mano lo hacemos en un lenguaje de programación gráfica en el que iteramos un círculo de un tamaño fijo, al que hacemos tomar cinco posiciones específicas, calculadas. Cada círculo tendrá un color específico, asignado por los códigos Pantone.

Este diseño sería generativo en el sentido de que ha sido generado a través de instrucciones en lugar de haberlo trazado a mano, pero sería un diseño fijo: cada vez que ejecutáramos el algoritmo el resultado sería exactamente igual. Si imprimiéramos doscientas copias, todas ellas serían idénticas, de manera que perdería algunas de las características que hacen que el diseño generativo se diferencie del clásico.

Figura 1. Los anillos olímpicos versión determinista. El programa siempre generaría la misma imagen



Pero insisto, no hay ningún problema en hacerlo así. Cada problema tiene diversas soluciones y si la grafista cree que la mejor solución es un proceso determinista, adelante con él. De hecho, John Maeda tiene diseños profesionales resueltos con programación en los que el resultado final es totalmente determinista: cada objeto gráfico que forma el cartel está predefinido de forma exacta por una serie de fórmulas matemáticas. Al contrario de nuestro insulso ejemplo de los anillos olímpicos, estos proyectos de Maeda son relevantes porque explota la capacidad de la programación para llevar a cabo un diseño que a mano resultaría imposible, como llenar la pantalla con la palabra *helvética* en diferentes posiciones, muy cerca las unas de las otras, comportándose como megapíxeles que dibujan una A de la tipografía Helvética.

Cuando generamos grafismos como los de Maeda, estamos aprovechando la secuencialidad de los lenguajes de programación. Si queremos llenar un DIN A3 de pequeños círculos hasta generar una malla compacta de fondo para un cartel, si lo hacemos con un programa de edición gráfica comercial nos tendremos que pasar un rato bien aburrido cortando y pegando círculos. En cambio, con un lenguaje de programación como Processing lo solucionaremos en dos líneas. Crearemos un bucle que hará que el programa dibuje, por ejemplo, un

círculo de 2 mm de diámetro cada mm, hasta llegar al final de la hoja. En segundos tendremos solucionado un problema francamente tedioso que habría consumido un buen rato de nuestro precioso tiempo.

Gracias a la secuencialidad también podemos crear variaciones progresivas de un diseño base. Así, podemos transformar el concepto de producción seriada y, en vez de generar varias copias del mismo diseño, ir generando diferentes variaciones, de manera que no haya dos diseños iguales en la serie. Por ejemplo, podemos ir variando de forma progresiva el color de un elemento o su posición a lo largo de todas las generaciones de un cartel.

2.3. Secuencialidad frente a aleatoriedad

Un diseño secuencial sigue siendo determinista y cada vez que ejecutemos nuestro miniprograma que llena de círculos una hoja saldrá exactamente la misma imagen. Pero no hay ninguna necesidad de quedarse aquí. Como explicamos en el módulo «En qué consiste el diseño generativo», hay formas matemáticas de generar procesos seudoaleatorios en un ordenador, de manera que podemos incluir azar en la secuencialidad y generar diversos resultados gráficos según qué números aleatorios surjan.

Volvamos a nuestro ejemplo de rediseñar los anillos olímpicos. La ontología original que definiría el proyecto sería la siguiente:

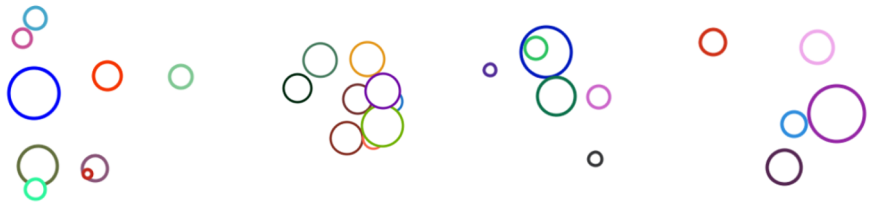
- Número de círculos 5
- Tamaño de cada círculo: radio de 50 píxeles
- Colores y coordenadas de cada círculo: (100, 100) azul; (140, 100) negro; (180, 100) rojo; (120,130) amarillo; (150,130) verde(al ser todos círculos con el mismo tamaño, con indicar la posición del centro de este es suficiente para ubicarlo en un plano)

Pero esta vez el Comité Olímpico nos deja más libertad y decidimos jugar con la aleatoriedad. Así, parametrizamos nuestro logo e incluimos en nuestra ontología los siguientes parámetros.

- Número de círculos: entre 5 y 10
- Tamaño de cada círculo: variará entre 10 y 70 píxeles
- Colores: completamente aleatorios
- Posición: un número aleatorio tanto para X como para Y entre 100 y 300

Con esta ontología tan sencilla nuestro diseño ha dejado ya de ser determinista y tiene ante sí millones de diferentes combinaciones. Cada vez que ejecutemos nuestro programa, de unas pocas líneas tendremos una imagen diferente. Que al Comité Olímpico le gusten nuestras propuestas es otro asunto…

Figura 2. Cuatro ejemplos de la cantidad de objetos gráficos diferentes que este simple ejercicio de añadir aleatoriedad a los anillos olímpicos genera



3. Datos externos

Hasta aquí hemos visto una descripción del diseño generativo donde trabajamos con objetos virtuales generados por el propio lenguaje de programación, pero nada nos impide añadir información externa. Básicamente, podemos dividir el tipo de datos externos que incluiremos en nuestro diseño generativo en tres grandes bloques: bases de datos, datos en directo e interacción con los usuarios.

En el primer caso, disponemos de una serie de datos organizados en una base de datos que nuestro programa leerá y parametrizará siguiendo una serie de informaciones.

En el segundo caso, un sensor como una cámara o un termómetro digital recibirá datos directamente y los reenviará a nuestro algoritmo, o bien el sensor reenviará los datos a un servidor online y nuestro programa los irá leyendo directamente en tiempo real, según los sensores los van ubicando en el servidor.

La última opción refiere a sistemas interactivos en los que el resultado final variará en función de las acciones del usuario y vendría a ser como un subconjunto muy especial e importante de los datos en directo.

3.1. Bases de datos

Además de los elementos que nuestro programa pueda generar de manera automática, podemos incluir bases de datos de diferentes tipos de objetos digitales que pueden ser incluidos de forma paramétrica en nuestro grafismo generativo.

Veamos un ejemplo sencillo. Supongamos que tenemos que hacer el cartel para un festival en nuestra ciudad donde tocarán diez grupos musicales diferentes, y tenemos tres fotografías en buena resolución de cada grupo. En lugar de hacer un único cartel con una foto en pequeño de cada grupo, o escoger el grupo más famoso y poner una foto suya, preferimos hacer un diseño generativo y combinar las fotografías con cierta aleatoriedad. Así, definimos por ejemplo un sistema de cartel donde habrá una serie de elementos fijos que son el nombre del festival, el lugar donde tendrá lugar, las fechas, etc. Esto formará una estructura fija sobre la que acoplar elementos variables y aleatorios. En concreto, pondremos dos fotografías de dos grupos, con el nombre de cada grupo abajo. Quiénes serán esos dos grupos se escogerá al azar, así como qué

fotografía de cada grupo se usará. El único criterio que añadimos a la selección es que los dos grupos sean diferentes. Esto nos permite crear más de setecientos posibles carteles diferentes.

Sin embargo, es fácil que queramos aprovechar las posibilidades que un lenguaje de programación nos ofrece y jugar más con esas fotos. Entramos así en la idea de procesar los datos. Para ello, utilizamos la información que nos ofrece la base de datos y la utilizamos para remodelar nuestro sistema gráfico.

Imaginemos que en la base de datos que incluye la fotografía y el nombre del grupo hay además una indicación del estilo que ese grupo musical practica. Podemos entonces decidir un criterio extra en nuestra ontología: que los dos grupos que comparten cartel sean de estilos muy diferentes, para así crear una contraposición conceptual. Evidentemente, un lenguaje de programación no sabe nada sobre estilos musicales (¡ni siquiera el increíble Processing!), de manera que tendríamos que incluir unas líneas de programación para establecer qué estilos son similares y cuáles no.

3.2. Procesamiento de datos y grados de generalización

Cuántas líneas deberíamos incluir en nuestro programa para decidir si dos estilos musicales son similares o no y cómo debería desarrollarse el tema es una cuestión importante. De fondo, hay un problema básico del diseño generativo: el grado de generalización que vamos a dar a nuestro algoritmo. ¿Queremos hacer un algoritmo que resuelva un encargo concreto? ¿Buscamos más bien un sistema que permita resolver toda una familia de problemas similares? ¿O tenemos en mente un programa mágico que solucione de manera genérica toda una tipología?

No es una decisión fácil, y dependerá de todo tipo de factores. Nuestra personalidad es un elemento relevante: ¿Somos personas sistemáticas que buscamos encuadrar un problema en una estructura tipo? ¿O preferimos solucionar cada problema al vuelo? Otro factor clave es lo común que es el problema: ¿nos encontraremos de forma repetida con este problema? ¿O es una situación que no volverá a ocurrir en años?

Volvamos al ejemplo del festival. Si esta es la primera vez que hacemos el cartel para el festival y no queda claro si nos volverán a contratar para hacerlo –o si el festival se va a volver hacer– lo más fácil es no complicarse la vida e incluir a mano cuatro reglas que nos indique que el Death Metal y el Grindcore son muy similares, pero que difieren mucho del Reggae. Total, son diez grupos, diez estilos como mucho…

Imaginemos ahora que hay un compromiso claro con la organización de poder dedicarnos varios años a trabajar con ellos, que además de este festival harán otros similares, que su intención es cada vez incluir más grupos en sus festivales y que les ha encantado la idea de poner dos fotos de dos grupos muy

diferentes. En ese caso, seguramente es una buena opción trabajar un poco más en el programa, buscar información sobre estilos musicales y dejar un algoritmo robusto que pueda decidir de manera automática qué estilos musicales son similares y cuáles no. Así, la próxima vez que tengáis que desarrollar un cartel para este cliente la decisión ya estará automatizada.

Finalmente, imaginemos ahora una profesional del grafismo famosa por sus carteles generativos. Recibe gran cantidad de encargos y no quiere rechazar ninguno. Es fácil que acabe creando su propia aplicación generadora de carteles donde el sistema leerá de forma automática texto, fechas, imágenes, etc. y genere carteles en un estilo de diseño evolutivo en el que la diseñadora lo único que tenga que hacer es hacer clic en las opciones que le parecen más prometedoras hasta tener el cartel perfecto para ese cliente que tiene tanta prisa y necesita el cartel para ayer.

Por otro lado, la decisión de generalizar puede no estar asociada al encargo propiamente dicho, sino a nuestro deseo de aprender. Volviendo al ejemplo del cartel de festival de música, aunque no tenemos nada claro que los organizadores del festival nos acaben contratando, podemos decidir igualmente hacer un programa muy generalista, capaz de producir muchos carteles diferentes de festivales y conciertos porque queremos aprender a programar mejor y la mejor manera de hacerlo es frente a un encargo concreto.

3.3. Interactividad y personalización

Hasta ahora hemos estado hablando de sistemas generativos totalmente autónomos. Creamos el algoritmo y una vez lo ejecutamos el sistema procesa los datos de manera unilateral y nos da un resultado final. Pero un diseño generativo no tiene por qué quedarse aquí.

> Una de las razones por las que el diseño generativo es tan importante es porque facilita la revisión de los diseños en tiempo real por parte del grafista y/o del público al que va dirigido.

Podemos distinguir, así, dos tipos de interactividad: interactividad en el proceso de creación e interactividad en el proceso de ejecución.

La interactividad en el proceso de creación es la que hemos comentado el apartado 2.5. del módulo «En qué consiste el diseño generativo» al hablar de diseño evolutivo. El proceso de creación del cartel no es autónomo, sino que el algoritmo va generando diferentes opciones, que el grafista o el público acepta o rechaza y poco a poco se va generando un objeto gráfico. Es decir, el cartel, portada de libro, etc. han sido creados de forma interactiva.

Claramente, esa es una versión muy limitada de interactividad, asociada a objetos estáticos que ya no pueden cambiar, como carteles o portadas de libros en papel. Pero nada impide imaginar objetos gráficos interactivos, que cambian a lo largo del tiempo en función de las decisiones que toman los usuarios que interactúan con ellos. Formalmente, sin embargo, los dos procesos son muy similares: cuando generamos un *website* cuyo aspecto gráfico sea personalizable por cada usuario o creamos una aplicación interactiva por la que un surfero empedernido puede customizar su tabla a su gusto antes de encargarla, tendremos que desarrollar una ontología en la que especifiquemos qué elementos son fijos y cuáles son variables, qué opciones ofrecemos a la usuaria que sean razonables y no causen problemas posteriores, y a través de qué canales esa usuaria podrá hacer llegar sus decisiones al algoritmo.

De hecho, esta es otra de las grandes revoluciones que nos trae el diseño generativo, y que solo ahora empezamos a vislumbrar: la separación entre objeto estático frente a objeto interactivo no tiene ningún sentido en un entorno digital. Si las portadas de nuestros *ebooks* son siempre iguales no obedece a ningún criterio técnico o estético, es la pura costumbre de asociar las propiedades de un objeto físico a un objeto virtual para facilitar nuestra interacción con él. En estos tiempos de impresión a demanda o de la fabricación de objetos vía *crowdsourcing*, no hay ninguna razón por la que todas las copias de un libro físico tengan que tener la misma portada, o unos zapatos solo se puedan encontrar en negro y marrón. Las razones son finalmente de proteger la capacidad de producir barato y a gran escala por parte de grandes multinacionales. Si el diseño generativo te llama la atención, ahora es un buen momento para meterse en él, pues queda mucho terreno por explorar y disponemos de herramientas cada vez más accesibles y que nos brindan muchas posibilidades.

3.4. Datos en vivo. Animaciones

Si nuestro proyecto incluye datos en vivo, podemos plantearlo de dos formas básicas: por un lado podemos simplemente considerar el tiempo como una dimensión extra y generar un objeto gráfico concreto cada X tiempo. Imaginemos que un equipo de geólogos nos contrata para ayudarlos a establecer la frecuencia de terremotos en diferentes países. La idea es recibir datos de diferentes sismógrafos mundiales y generar cada hora un documento imprimible que consistirá básicamente en un mapamundi donde un código de colores indicará la frecuencia relativa de terremotos en cada país. Si rojo es gran actividad sísmica y violeta nula actividad sísmica, cada hora tendremos una serie de informaciones distribuidas de forma determinista en un DINA4 como día, hora, etc. y un mapamundi en el que a través de un sistema de procesamiento de datos cada país tomará una gama del rojo si en aquel momento hay gran actividad sísmica y se acercará a tonos azulados si es una zona tranquila.

Con esta forma de trabajar los datos, en el fondo lo que estamos haciendo es reducir este nuevo caso a la situación anterior. En realidad, lo que hacemos es producir un gráfico generativo a partir de una base de datos. La única diferencia es que la base de datos cambia cada hora.

Es una decisión posible y en este caso adecuada para los intereses de los científicos que nos han contratado. Pero teniendo datos que están cambiando continuamente es una pena no capturar ese cambio de manera continua. Necesitamos así una animación.

Entramos así plenamente en un entorno digital en el que nuestra ontología incluye un objeto que es continuamente variable en función del tiempo. Podemos usar esos datos para mostrarlos directamente, como en el caso de una cámara, transducir esos datos a otra magnitud, como en el ejemplo anterior de convertir la actividad sísmica en una gama de colores, o podemos procesar esos datos en tiempo real cruzándolos con otras informaciones en vivo o de bases de datos, como cuando un navegador nos indica qué ruta para volver a casa en coche es la menos transitada a partir de la información que recibe en tiempo real sobre la movilidad en las carreteras cercanas.

Otra posibilidad es establecer un tipo de transducción menos cognitiva y más artística, como utilizar los datos de temperatura que recibimos de diferentes países para crear imágenes expresionistas abstractas al estilo de un Motherwell o un Rothko. O, como la bailarina cyborg Moon Ribas, que convierte en movimientos de danza los datos de sismógrafos que recibe a través de un sensor en sus tobillos.

De nuevo, la organización de nuestro sistema será la misma: tenemos una serie de elementos fijos y otros variables. En algunos casos la variación se generará a través de un proceso aleatorio. En otros se leerán datos de una base de datos; pero una parte central del objeto gráfico interactivo serán datos en tiempo real, ya sea internos o externos, que se organizarán en algún tipo de animación. Entendiendo aquí «animación» en un sentido amplio, que simplemente recoge la idea de elementos visuales que se van transformando a lo largo del tiempo y esas transformaciones pueden contemplarse en tiempo real.

4. Hibridación

Hemos hablado aquí de diferentes objetos gráficos producidos de manera generativa, siguiendo un principio de clasificación, buscando tipos puros. Pero nada impide jugar promiscuamente con esos tipos y generar todo tipo de objetos y aplicaciones híbridas.

Así, en lugar de representar fielmente los datos de la temperatura exterior en una animación podemos utilizar las variaciones en temperatura, como números aleatorios que cambien el aspecto de un banner cada hora. O utilizar cien fotos de una cámara digital en internet de un intervalo de 24 horas para hacer un cartel clásico que reflexiona sobre el paso del tiempo. Nada nos impide combinar información en bases datos con los movimientos de un usuario delante de una videocámara para inventar un nuevo tipo de videojuego o que un e-cartel colgado en internet muestre un paisaje nocturno o diurno en función de la zona horaria en la que se encuentra la persona que lo visita.

De la misma forma, en esta época de impresión bajo demanda y producción digital no hay razones de peso por la que los objetos gráficos físicos no puedan presentar diferentes aspectos. Nada impide tener 125 carteles diferentes de una exposición, o que cada vez que se imprime una novela la portada tenga un juego de figuras geométricas de estilo mediano diferente. Esos mismos principios los podemos trasladar al diseño de moda, de producto, de interfaces o de servicios.

Y las razones no han de ser puramente estéticas o de deseo de innovar. Hay también razones éticas y sociales para la pluralidad en el diseño gráfico. Imagina la interfaz de un sistema de e-comercio en el que en lugar de tener una solo sistema de interacción tenemos en cuenta la diversidad sensorial y neuronal, y así tenemos un sistema de interacción pensado especialmente para las personas ciegas o con problemas a la hora de distinguir colores, un cartel pensado especialmente para personas con síndrome de Asperger o con dificultades cognitivas. Los tiempos de «una sola talla va bien para todos» han pasado a la historia. Gracias al diseño generativo disponemos ahora de herramientas individualizadas para pulverizar distinciones y crear todo tipo de sistemas híbridos.

Bibliografía

Cross, Nigel (2001). «Designerly ways of knowing: design discipline versus design science». *Design Issues* (vol. 3, núm. 17 pág. 49-55). <https://oro.open.ac.uk/3281/1/Designerly-_DisciplinevScience.pdf>

Engelhard, Yuri (2002). *The language of graphics Institute for Logic, Language and Computation*. Universidad de Ámsterdam.

Fry, Ben (2007). *Visualizing Data*. O'Reilly.

Kress, Gunther; Van Leeuwen, Theo (1996). *Reading Images. The Grammar of Visual Design*. Routledge.

Stuikys, Vytautas; Damaševičius, Robertas (2008). «Design of ontology-based generative components using enriched feature diagrams and meta-programming». *Information Technology and Control*(vol. 37, núm. 4, pág. 301-310). <https://www.semanticscholar.org/paper/DESIGN-OF-ONTOLOGY-BASED-GENERATIVE-COMPONENTS-AND-Stuikys/5cbb085827a71f516e2e27b090adeef38b0a586b>

Wilkinson, Leland (1999). *The Grammar of Graphics*. Springer.