ANIAV REVISTA DE INVESTIGACIÓN EN ARTES VISUALES

| Año 2024 | marzo | Nº. 14
https://doi.org/10.4995/aniav.2024.20454

Funciones estéticas y pedagógicas del software: una reflexión teórico-práctica con estudiantes de máster

Aesthetic and Pedagogical Functions of Software: A Theoretical-Practical Reflection with Master Students

Pérez-Campos, Marta
UPV/EHU Universidad del País Vasco
mperez272@ikasle.ehu.eus

Recibido: 05-10-2023
Aceptado: 20-02-2024

Citar como: Pérez-Campos, Marta (2024). Funciones estéticas y pedagógicas del software: una reflexión teórico-práctica con estudiantes de máster. *ANIAV - Revista de Investigación en Artes Visuales*, n. 14, p. 81-94, marzo. 2024. ISSN 2530-9986. Doi: https://doi.org/10.4995/aniav.2024.20454

PALABRAS CLAVE
Software; practice-based research; *esolangs*; programación; prototipado; estética; pedagogía.

RESUMEN
Este artículo analiza el taller *Aesthetic and Pedagogical Implications of Software,* cuyo objetivo principal era reflexionar sobre el hecho de mostrar el *software* detrás de aquellos proyectos que están diseñados utilizando algoritmos y, a partir de allí, comenzar a indagar sobre qué valor estético-pedagógico puede tener ese desvelamiento. Dividido en teoría y práctica, en la parte teórica fueron presentadas prácticas contemporáneas en las que el código es utilizado de una manera estética. Por otro lado, se propone un análisis de la vertiente pedagógica que posee esta estetización del *software*. En la parte práctica del taller, los participantes eran invitados a crear un prototipo en relación con la parte teórica y su propia práctica. La evaluación del taller se realizó mediante encuestas, las cuales también se emplearon a la hora de extraer las conclusiones principales. En ellas se muestra cómo pese a que en su práctica no habían considerado el *software* como un ente albergador de un valor estético, la asistencia al taller supuso una apertura hacia esa posibilidad, hacia una nueva percepción del *software* y a su vínculo con la pedagogía. Otro hecho para tener en cuenta fue la presencia de lo analógico y lo performativo en algunos de sus prototipos.

Este taller es el segundo de los realizados dentro de la tesis doctoral *Más allá de la poesía computacional: una exploración de las implicaciones estéticas y pedagógicas del software* en la que se busca reflexionar sobre la relevancia de una aproximación estética al *software* y qué consecuencias puede tener en cuanto a conocer el funcionamiento interno de la

máquina. El taller tuvo lugar en dos sesiones de 90 minutos en la Kunstuniversität Linz (Austria) y fue parte de los cursos con créditos de libre elección que son ofertados a los estudiantes del Máster *Interface Cultures*.

KEYWORDS
Software; practice-based research; esolangs; coding; prototyping; aesthetic; pedagogy.

ABSTRACT
This article analyzes the workshop Aesthetic and Pedagogical Implications of Software, whose main purpose is to reflect on the option of showing the software behind those projects that are designed using algorithms and, from there, to begin to inquire about what aesthetic-pedagogical value this unveiling may have. Divided into theory and practice, the theoretical part presents contemporary practices in which the code is used in an aesthetic way. On the other hand, an analysis of the pedagogical aspect of this aesthetization of software is proposed. In the practical part of the workshop, the participants were invited to create a prototype in relation to the theoretical part and their own practice. The workshop was evaluated by means of surveys, which were also used to draw the main conclusions. They show how, even though in their practice they had not considered software as an entity with an aesthetic value, attending the workshop meant an opening towards this possibility, towards a new perception of software and its link with pedagogy. Another fact to consider was the presence of the analogical and the performative in some of his prototypes.

This workshop is the second one of those carried out within the doctoral thesis *Beyond Code Poetry: An Exploration of the Aesthetic and Pedagogical Implications of Software* in which the aim is to reflect on the relevance of an aesthetic approach to software and what consequences it may have in terms of knowing the inner workings of the machine. It took place in two 90-minute sessions at the Kunstuniversität Linz (Austria) and was part of the elective courses offered to students of the Interface Cultures master's degree.

INTRODUCCIÓN

La realización de este taller se enmarca en la tesis doctoral *Más allá de la poesía computacional: una exploración de las implicaciones estéticas y pedagógicas del software,* en la que se propone una aproximación al *software* alejada de su funcionalidad, incidiendo en su valor estético y pedagógico. Este taller es una iteración de "(Des)cifradores de mensajes(…)" cuyo objetivo principal fue buscar la posibilidad de explicar el funcionamiento interno de un compilador a niños de 6 a 12 años, desde la experimentación estética (Pérez-Campos, 2022). La hipótesis defendida en este taller es que considerar hacer el código visible dentro de un proyecto artístico, puede ser valioso para el proyecto: los algoritmos que lo componen pueden tratarse como material no solamente destinado e interpretable por máquinas, sino también por personas. Los participantes se aproximaron a esta idea tanto desde lo analógico, con la *performance Mindless Routine* propuesta por Maria Konstantinova, como desde lo digital, conceptualizando lenguajes de programación esotéricos (*esolangs*) como fue el caso de *Your North Is My South!* De Ahmed Jamal y Salma Aly, entre otros. El código fue sacado de su invisibilidad dentro de la máquina para convertirse en el objeto central sobre el que se construían sus propuestas artísticas.

El valor artístico de estas propuestas reside en que plantean una reflexión sobre el propio medio que utilizan (*software* y/o lenguajes de programación) y sobre la propia percepción del mundo de los y las participantes. Si "(Des)cifradores de mensajes (…)" demostró que desde una perspectiva creativa y con el uso de analogías era posible hablar y educar sobre un componente tan abstracto como es el compilador de un ordenador; el taller que nos ocupa se centró en cómo desde la creación artística se puede abordar el *software* para crear proyectos artísticos que lo ubiquen en el centro y le permitan poseer en sí mismo valor estético.

El objetivo principal de este taller era enfrentar a estudiantes de máster con la idea de programar de una manera estética *(aesthetic programming*[1]*),* hacerles conscientes de esta posibilidad y de la presencia que ha tenido dentro del arte de los nuevos medios desde sus inicios. A través de este concepto, se genera un intercambio de maneras de hacer y saberes con los participantes, los cuales en su mayoría escriben código dentro de su práctica artística. Tener el Máster *Interface Cultures* (IC) como contexto para el taller fue ideal por la naturaleza de los estudios que en él se cursan. Estos estudios de máster se centran en proporcionar a los estudiantes una formación en el diseño de *software*, *hardware* y las interfaces necesarias para interactuar con el sistema, interacción basada en la web, VR e instalaciones interactivas, entre otras. Aparte de ello, también busca proporcionar a sus estudiantes formación como investigadores, ofreciéndoles conocimiento en teoría artístico-científica *(artistic/scientific theory)* y estrategias para publicar artículos de investigación. Tener este máster como contexto aseguró que los participantes contarían con ciertas nociones sobre *software*, estética y arte de los nuevos medios en general. Junto a la idea de programación estética, con el taller se buscaba propiciar una reflexión sobre el *software*, entendiendo este como un texto no creado únicamente para ser procesado por máquinas: ¿qué valor o importancia tiene mostrar lo que hay detrás? Formulando esta pregunta, de manera implícita, se buscaba reflexionar sobre la idea de 'aura' y sobre si consideraban que ocultar el *software* muestra sus proyectos como 'dispositivos mágicos'. Por último, se buscaba una reflexión desde el hacer artístico personal con la creación de un prototipo, el cual sería presentado y discutido con el resto de los participantes.

METODOLOGÍA

A nivel estructural, se decidió dividir el taller en dos sesiones de 90 minutos, así, se permitió que los participantes tuviesen más tiempo de desarrollar su prototipo y presentarlo durante la segunda sesión. El taller fue principalmente teórico, con explicaciones y discusiones entre los participantes, siendo imprescindible que todos participasen activamente. Por eso, se decidió tomar como contexto a estudiantes de un máster, considerando que ya contarían con su propio discurso artístico al que aplicar las ideas defendidas en el taller. Con la finalidad de evaluar si se habían logrado los objetivos propuestos inicialmente, se elaboró una encuesta que permitió, entre otras cosas, medir el interés y relevancia que este taller tenía en relación con su práctica artística. Esta manera de estructurar el taller, dividida en parte teórica y práctica con el uso de encuestas

[1] Término acuñado por Geoff Cox y Winnie Soon en su libro *Aesthetic Programming.*

como método de evaluación, se utilizó en el ya mencionado taller "(Des)cifradores de mensajes (…)" al considerarse que la alternancia de una parte práctica con una teórica era la fórmula que mejor permitiría analizar si se conseguía el objetivo principal.

En cuanto a los antecedentes y el marco teórico en el que se desarrolló este taller, una de las primeras cuestiones que se plantearon sobre la naturaleza del código fue la división entre *software* y *hardware*, abordada desde la perspectiva del filósofo e informático Timothy Colburn. Se defendió su idea del *software* como una abstracción concreta de la máquina (*software as a concrete abstraction*) (Colburn, 2000, p.199). En esta concepción del *software*, este está compuesto por texto, en tanto que es la manera en la que escribimos los algoritmos a ejecutar, pero a su vez es una máquina; una abstracción de las señales eléctricas que tienen lugar dentro del ordenador. Así, en este taller, el *software* fue entendido en dos variantes: como texto y como máquina. Esto se puede ejemplificar en la diferencia existente entre encontrar un algoritmo impreso en papel, donde se presenta como puro texto y deja de ser ejecutable, o dentro de un USB, donde sí sería ejecutable y podría considerarse como una máquina en potencia.

Pasando directamente al *software*, los lenguajes de programación (LP) y a su uso estético, se tomó como punto de partida al artista e investigador Daniel Temkin. Su web https://esoteric.codes ('esoteric.codes', s.f.) es un archivo de propuestas experimentales realizadas con código, tanto con la creación de nuevos LP como con preexistentes. Este compendio de proyectos computacionales, en los que el planteamiento de cuestiones en torno a la naturaleza, al uso y percepción que tenemos del *software*, ha sido una importante fuente de recursos para este taller, así como algunas de las entrevistas con sus creadores. Entre ellas, cuenta con una al creador de *esolangs* Keymaker, en la que defiende que los programas creados utilizando *esolangs* son en sí mismos *visual art* ('Interview with Keymaker', 2011).

Los *esolangs* basan su existencia en extender la definición de lenguaje de programación. Proponen acercarse a la programación desde la experimentación formal y alejándose de la funcionalidad y eficiencia tan buscada en los LP convencionales. Pese a que con su mayoría es posible generar algoritmos compilables, esta experimentación formal que buscan conduce a que, algunos de ellos, busquen ser meramente conceptuales. Ello no evita que puedan ser implementables posteriormente por otros artistas o programadores. Un ejemplo de *esolang* conceptual sería *A programming language is a formal language comprising a set of strings (…)* creado por el artista e investigador Daniel Temkin. Este *esolang* está basado en la definición de LP que recoge Wikipedia y la cual va siendo modificada a lo largo del tiempo('Programming language', 2024). Según su creador, cada vez que una persona escribe esa definición, está escribiendo el único programa posible existente en este *esolang* ('A programming language is a system of notation for writing computer programs. - Esolang', s.f.). Este acto puede entenderse como, aparte de conceptual, poético, presentando una visión expandida del *software*. Esta relación de la estética con el código la encontramos en las publicaciones (Soon y Cox, 2020) y (Montfort, 2016). En ellas, a modo de curso y de una manera bastante pedagógica, se presentan unos modos de programar alejados de la funcionalidad con la que normalmente es asociada, por ello también fueron considerados como referentes para la parte pedagógica. Ejemplo de ello es el segundo capítulo titulado "Variable

Geometry" en el que tratan el tema de la composición con formas geométricas y, tras explicar las funciones y estructuras para programar este tipo de geometría, proponen una discusión sobre qué es una cara (*what constitutes a face?)* y la abstracción que de ellas hacen los emojis. El código es empleado, entonces, como un vehículo de reflexión que sirve para educar sobre aspectos que no siempre son él mismo. En el caso de "(Des) cifradores de mensajes(…)" sirvió para aproximar a niños al funcionamiento de un compilador. En el caso que nos ocupa, el código ha servido para reflexionar sobre la propia idea de algoritmo y de lenguaje de programación.

En torno al valor poético de la palabra, un texto altamente influyente ha sido *The Uprising: On Poetry and Finance.* Escrito en 2011, cuando movimientos como el 11M parecían suponer un despertar en las conciencias y conducir a un cambio político, el filósofo autonomista italiano Franco 'Bifo' Berardi escribió que la poesía sería la herramienta que guiaría ese cambio en tanto que sería capaz de librarnos de los automatismos lingüísticos impuestos en la sociedad actual mediante la economía financiera y el desarrollo tecnológico (Berardi, 2012). De la misma manera que en dicha publicación Bifo entiende que el lenguaje ha de ser liberado del automatismo actual, el poder expresivo de los lenguajes de programación puede ser liberado si empezamos a leer código de una manera que va más allá de ser palabras con las que comunicarnos con un ordenador. Esta idea la desarrolló en su introducción para el libro *Speaking Code: Coding as Aesthetic and Political Expression* en la que afirma que cada vez más estamos intentando escapar de los automatismos que conlleva la escritura de código (Cox y McLean, 2013). Esta también será una de las ideas en torno a las que Mark C. Marino creó el grupo de trabajo llamado *Critical Code Studies,* que acabó cristalizando en un libro con ese mismo título (Marino, 2020). La dependencia existente entre el algoritmo y la máquina, dentro de la cual tienen lugar procesos que están fuera de nuestro control, es la que provoca la intervención de lo desconocido y su relación con la magia, con el no saber cómo se ha generado ese resultado. La investigadora Betti Marenko enunció el concepto *algorithm magic* para referirse a esa situación. Marenko entiende que tanto en la magia como en la tecnología se busca 'alterar el medio natural con medios artificiales (*change the natural environment by artificial means*)' (Marenko, 2019, p.213). Por tanto, explorar las capacidades estéticas de los lenguajes de programación y los algoritmos es necesario en tanto que la incertidumbre digital que generan fomenta la idea de conducirnos a una especie de universo mágico.

Todos estos referentes, tanto los que cuestionan nuestra concepción del *software* de una manera teórica como práctica, han sido la base sobre la que construir la parte teórica, la cual que fue completada con preguntas cuya finalidad era abrir el debate a los participantes y con la propuesta de creación de un prototipo.

DESARROLLO DEL TALLER

1. Primer día: ¿qué es el código?, *software art*, *software aesthetics* y propuesta de prototipado

Antes de comenzar, se consideró necesario saber cuál era la práctica artística y formación con la que contaban cada uno de los participantes del taller, por lo que se propuso una ronda introductoria de presentaciones.

Se realizó una división en dos partes principales. Por un lado, se reflexionó sobre la propia definición de código y su relación con las ideas de transcripción, traducción y transcodificación. Dentro de la propia idea de transcripción, del paso de la mente del programador a la línea de código, se comentó, entre otras cosas, la naturaleza textual del código y la pérdida de expresividad, de la entonación, que se produce con la escritura. En contraposición a ello, se presentaron dos performances en las que el código es leído, recitado. La primera de ellas fue la lectura del código del virus *ILoveYou* (*LoveLetter.exe)* que en 2001 realizó Bifo en el festival D-I-N-A de Bolonia (Italia). La segunda fue la lectura de la pieza Internet Directory que Daniel Temkin realizó en 2019, en la cual listaba en un libro de 37 000 páginas todos los dominios .com en orden alfabético ('Daniel Temkin | Internet Directory', s.f.) .

En este punto se abrió el primer debate, en el que se planteaba qué sentido y relevancia consideraban que tienen este tipo de performances. La mayoría coincidieron en que era una manera de descontextualizar el código y de experimentar con el hecho de que son textos que no han sido concebidos para ser leídos por personas, sino para ser procesados por máquinas. En el caso del virus *ILoveYou,* este pierde su poder y se convierte en poesía sonora *(sound poetry*); mientras que en el caso de *Internet Directory*, se llegó a la conclusión de que lo que se enfatiza es el tiempo, la imposibilidad de leer todos los dominios terminados en .com, y una reflexión sobre la cantidad de información que está *en la nube*, pero a su vez ocupando espacio dentro de servidores.

El siguiente punto fue la idea de traducción, la cual se apoyó en la idea de W. Benjamin que defiende que para conseguir una buena traducción es necesario encontrar la esencia del texto (Benjamin, Bullock, Jennings, Eiland y Smith, 1996), imprescindible a la hora de traducir el texto desde la idea, en la mente del programador, hasta conseguir la línea de código. Junto a ella, se presentó la concepción de que la traducción se basa en un diálogo constante entre el humano y la máquina y en cómo es necesario tomar en consideración los propios límites de la máquina.

De la transcodificación, el proceso que permite la traducción y almacenamiento de un archivo en diferentes formatos y también de lo analógico a lo digital, se comentó la capa cultural *(cultural layer*) (Manovich, 2002, p. 46) que según Lev Manovich, quien acuñó el término en 2001, posee cualquier objeto digital. Esta capa cultural se refiere a la razón por la que se genera un objeto digital en concreto y qué valor tiene dentro del marco cultural en el que es creado. En este punto se volvió a abrir el debate: '¿por qué decides emplear un tipo de *software* y no otro?' Coincidieron en que la elección se basaba en la facilidad de uso y disponibilidad. Esta pregunta condujo a la dicotomía entre *software* libre y *software* de pago y algunos de los participantes opinaron que parte del *software open source* tenía una curva de aprendizaje mayor y era menos intuitivo que su homólogo no libre o privativo. Por ello, aunque preferiblemente se decantan por el *open source*, la elección de su herramienta de trabajo depende de lo que el *software* en cuestión les pueda facilitar el trabajo cuando los plazos de presentación son ajustados. Una de sus quejas principales fue, como dentro de las instituciones académicas normalmente se promueve el uso de *software* no libre, como es en el caso de la Adobe Creative Suite, con la que normalmente se trabaja en todas las universidades artísticas o escuelas de diseño, pese a que todas esas aplicaciones cuentan con un equivalente de código abierto.

El siguiente tema con el que se buscaba reflexionar sobre la naturaleza del código fue la contraposición digital/analógico y *software/hardware*. Antes de hablar sobre definiciones, se les plantearon dos preguntas en apariencia obvias: ¿cómo influencian una a otra? y, ¿cómo de borrosa es la línea entre ellas? Algunos veían muy claro que el código era el *software* y la máquina el *hardware*, pero no era fácil encontrar la línea en la que uno pasa a ser otro, el momento en que las instrucciones pasan a realizar algo tan físico como apagar la máquina. Se llegó a la conclusión de que ese paso intermedio se realizaba dentro del compilador; esa pieza de *software* que traduce el código a cambios en el *hardware*. A partir de ahí, se planteó la idea de que el *software* es una abstracción concreta *(concrete abstraction)* y tras ello el origen que sobre los términos hardware y *software* recoge el diccionario *Online Etimology Dictionary*. A continuación, se planteó la pregunta: ¿el código es *software* o *hardware*? A lo que, obviamente, respondieron que era *software*; aunque las dudas comenzaron a aparecer cuando se habló de las tarjetas perforadas como ejemplo de código con las propiedades del *hardware*: tangibles y físicas, y las cuales son *hardware* porque aparecieron antes de que el término *software* se relacionase con la computación a partir de los años 60. Dentro del taller no se llegó a una respuesta definitiva para esta pregunta.

La segunda parte teórica se centró en el *software art* y la estética vinculada a él, relacionada con la idea de *algorithm magic*. Dentro de este apartado se habló de *live coding*, *codeworks* y *esolangs*. El *live coding* se presentó como práctica que enfatiza la presencia de la máquina, sus procesos y cómo nos hace partícipes del acto de programarla y la información de salida que se genera con ello. Los *codeworks* y la poesía computacional *(code poetry)* se presentaron en relación con la presencia de la máquina y los lenguajes creados para comunicarnos con ella, para finalmente hablar de los *esolangs*. Una vez presentadas estas manifestaciones, se plantearon dos preguntas: ¿hay otras manifestaciones que vengan a tu mente?, ¿aplicarías la percepción estética del *software* a alguno de tus proyectos? Pese a su familiaridad con el código, en sus proyectos no lo habían considerado un elemento visible y también compartieron que, al no definirse como programadores, tampoco creían que su código era interesante mostrarlo, porque opinaban que quizá no todo lo que hay en él sea correcto. Algunos de los participantes afirmaron que no buscaban en sus proyectos la belleza, sino que, a veces, les interesaba mostrar parte del funcionamiento, en lugar de lograr una estética muy depurada, pero no habían pensado en mostrar el código en sí. Coincidían en que la pérdida del aura no era la razón por la que preferían no mostrar el código, ya que entendían que esta se perdió con la foto, como enunció Walter Benjamin en su ensayo *La obra de arte en la época de su reproductibilidad técnica* (Benjamin et al., 1996), pero en su práctica artística entendían el código como la herramienta que está detrás y que no es necesario mostrar porque carecía de interés. En este punto, se comentó la idea de *aesthetic programming* y cómo la entendían sus creadores: Winnie Soon y Geoff Cox. Cercana a ideas como *creative coding* o *exploratory programming*, utiliza una definición de estética que se aleja de la idea burguesa de belleza y fealdad: es una estética relacionada con la política en tanto que no es ajena al contexto social en el que es creada: 'cultural production – which would now naturally include programming – must be seen in a social context' (Soon y Cox, 2020, p.15). Se habló de cómo este vínculo entre estética y contexto social se relaciona con la idea marxista de estética, la cual defiende que el arte, más allá de su relación o no con lo bello, debe estar envuelto en la transformación de la sociedad (Sánchez Vázquez, 2005).

Figura 1. *Imagen de Catn-IP y Pizza.*

La última parte consistió en proponerles la creación de un prototipo en el que se utilizara el *software* de una manera estética y, si fuese posible, que estuviese en línea con su discurso artístico.

2. Segundo día: presentación de prototipos y exploración de la función pedagógica del código

De los trece participantes, cinco optaron por la presentación de lenguajes de programación esotéricos puramente conceptuales, siendo la mayor parte de ellos visuales. Dos ejemplos de ello serían *Pizza* (Fallica, 2023) o *Catn-IP* (Šermukšnis, 2023) [Figura 1] dos lenguajes basados en utilizar ingredientes de una pizza e imágenes de gatos, respectivamente, como elementos de programación. Ambos proyectos cuentan con un componente humorístico que comparten con otros *esolangs* preexistentes.

Junto a estas interpretaciones de lo que podía ser un proyecto de *aesthetic programming*, otro de los participantes, Yuma Yanagisawa, propuso la creación de un *esolang* que, dada una imagen de arte abstracto como información de entrada, fuese capaz de producir el código necesario para ser replicada algorítmicamente. Este *esolang* sería capaz de generar tres algoritmos diferentes: 'Just like a poet would describe a scene in various ways, this project addresses the various possibilities to produce a creative coding sketch based on an abstract painting'[2] (Yanagisawa, comunicación personal, 24 marzo 2023). En este caso, es interesante cómo se plantea una humanización de la máquina, en tanto que sería capaz de generar distintas interpretaciones de una misma imagen.

Por su parte, el *esolang Your North is my South* planteó el problema que se genera con la división norte-sur y las tensiones que provoca en cuanto a la marginalización de ciertas regiones. Para plasmar esta problemática, se planteó un lenguaje que contaba con tres símbolos para representar el sistema binario y la inversión de dichos valores: '∧' representa el 1, '∨' representa el 0 y '°' invierte los valores. Por ejemplo '∧∨°∧∨' sería equivalente a 1001 en binario. Se destaca de este proyecto cómo los autores han trasladado una

[2] 'Al igual que un poeta describiría una escena de varias maneras, este proyecto aborda las distintas posibilidades para producir un boceto de codificación creativa basado en un cuadro abstracto'.

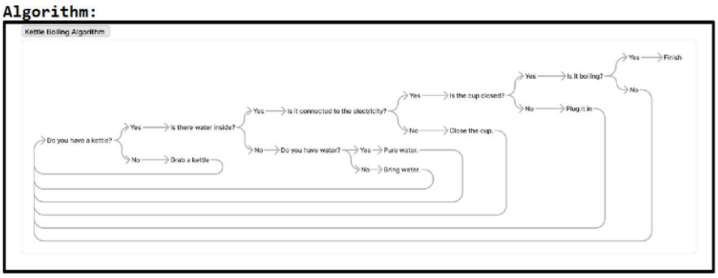

Figura 2. *Partitura de la performance Mindless Routine.*

problemática contemporánea a un *esolang*, que permite que los programas generados con él sean atractivos visualmente y, a la vez, posean una función crítica.

Fuera del ordenador hubo dos propuestas, una que utilizó la idea de programar aplicada al acto de tejer y con la que Maria Dirneder buscaba plantear la imposibilidad de utilizar un sistema binario puro en el mundo analógico, en sus palabras:

> 'I was very interested in coding nothing. Thinking of weaving in a binary system, black and white, I thought of how could it be possible in coding nothing with weaving. This led me to weave holes or loops, because in between there was neither black nor white will. Within the loop it was possible to look through, it opened up a new dimension. Thinking of Duchamp and the 3rd and 4rd dimension it was difficult imagine to transfer this experience into digital.'[3](Dirneder, comunicación personal, 24 marzo 2023)

La segunda propuesta puramente analógica fue *Mindless Routine* de Maria Konstantinova (Konstantinova, 2023) [Figura 2], quien planteó una performance en torno a hervir agua en una jarra eléctrica. Utilizó este algoritmo, porque es el que repetían constantemente en la escuela primaria durante las clases de programación y pese a que en su momento llegó a aborrecerlo, ahora lo considera útil para reflexionar sobre la correlación entre el algoritmo y la lógica humana: 'The performance is inviting the viewer to reflect on the correlation between algorithm and humans' logic'[4] (Konstantinova, 24 marzo 2023). Aparte de reflejar la relación entre la lógica humana y la de los algoritmos, saca del ordenador la estricta y repetitiva ejecución de un algoritmo creando una situación cómica.

Desde una posición crítica respecto a nuestra sociedad capitalista, Miguel Rangil propuso *Neutralising Machine* (Rangil, 2023) [Figura 3]; un *script* escrito en Python en el que el usuario es invitado a introducir un concepto y el algoritmo lo *neutraliza* reduciendo cada letra del concepto al número de su posición en el alfabeto. Este *script* junto con el

[3] 'Me interesaba mucho codificar la nada. Pensando en tejer en un sistema binario, blanco y negro, pensé en cómo sería posible codificar la nada con el tejido. Esto me llevó a tejer agujeros o bucles, porque en medio no había ni blanco ni negro. Dentro del bucle era posible mirar a través, se abría una nueva dimensión. Pensando en Duchamp y la 3ª y 4ª dimensión era difícil imaginar transferir esta experiencia a lo digital.' [3](Dirneder, comunicación personal, 24 marzo 2023)

[4] 'El espectáculo invita al espectador a reflexionar sobre la correlación entre el algoritmo y la lógica humana'. (Konstantinova, 24 marzo 2023)

de Kathrine Hardman (Hardman, 2023), en el que realizaba un autorretrato en el que fluctuase la imagen con funciones como *eatImages(ImageSet allImagesOnMyDrive, lambda tearUp()) o*

reinventSelf(FacePartsSet i) concluyendo con un llamamiento al constante cambio:

while(iStillLive):
showWorld(reinventSelf(everyFaceICouldHopeToHave);

Por su parte, Sofia Talanti creó un conjunto de *stories* de Instagram llamadas *Tales of a 3D Artist Learning Coding* (Talanti, 2023) [Figura 4] en las que mezclaba su práctica artística basada en el modelado 3D con la frustración que le provocaba tener que programar en sus proyectos. A modo de microhistorias divididas en capítulos, la artista crea textos en 3D los cuales eran presentados junto al código que los genera.

Tras la presentación de los proyectos, se comentó la función pedagógica del código. Alrededor de la pregunta '¿qué puede ser enseñado a través del *software*?' se presentó el trabajo de científicos e investigadores como Seymour Papert –creador del lenguaje de programación LOGO, con el que se buscaba enseñar a programar a los más pequeños (Papert, 1980)– o la psicóloga Edith Ackermann –centrada en investigar sobre la intersección entre juego, diseño, aprendizaje y tecnología (Ackermann, 2004). Junto a ellos, se destacó la labor del poeta y artista Nick Montfort, quien con su práctica y publicaciones experimenta con la poética dentro de los algoritmos. Dentro de los artistas que buscan reflexionar sobre qué aporta mostrar el código que compone sus proyectos

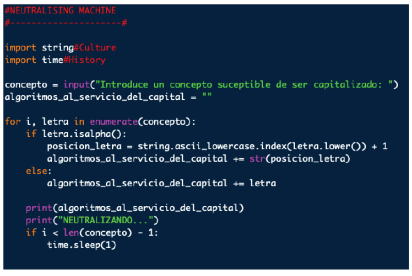

Figura 3. *Imagen de Neutralising Machine.*

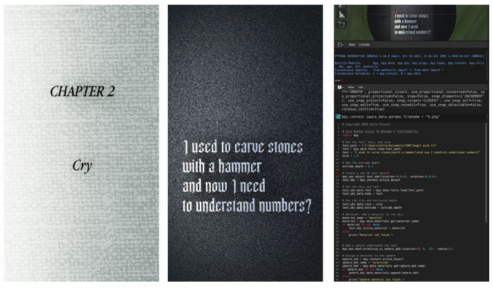

Figura 4. *Imagen de Tales of a 3D Artist Learning Coding.*

se destacó la propuesta de los *livecoders* o programadores en directo y cómo algunos de ellos encuentran su labor pedagógica en tanto que abogan porque los colegios adopten el *live coding* como metodología de enseñanza:

> 'We also think that schools will adopt live coding to teach programming and music at the same time because it is such a good way to learn both of the subjects! Teachers and students will love it' (Blackwell, Cocker, Cox, McLean y Magnusson, 2022, p.51)[5].

El taller se cerró preguntándoles a los participantes si consideraban que los prototipos que habían presentado poseían un componente pedagógico y se llegó a la conclusión de que, pese a no haber sido concebidos con una función pedagógica, algunos de ellos la presentaban, como en el caso de *Mindless Routine*, en el que la ejecución de un algoritmo era encarnada en una performance y se podía ver físicamente el funcionamiento de los bucles y comprobaciones que realiza el algoritmo.

ENCUESTAS

Una parte fundamental para evaluar el grado de implicación de los participantes en este taller consistió en la elaboración de una encuesta. De este modo, se buscaba medir en qué nivel habían sido alcanzados los objetivos propuestos y cuáles eran las impresiones de los 13 participantes. Estas encuestas contaban con doce apartados entre los que había preguntas de escoger una opción (2 preguntas), respuesta libre (6 preguntas) y escalas de Likert (4 preguntas).

En comparación con las encuestas desarrolladas en el taller previo, se subsanó la carencia con la que contaban las anteriores: no haber contemplado el nivel de conocimientos sobre el tema. En una escala del 1 al 5, de las 13 participantes, tres marcaron 1, cuatro marcaron 2 y 3; y los dos restantes marcaron 4; nadie consideró que tenía mucho conocimiento sobre el tema, quedando el 5 vacío. Así, se entendió que el conocimiento previo sobre el tema era medio. Sobre su formación, 4 habían estudiado Bellas Artes, 1 arquitectura, 1 escultura y 3D, 1 ilustración y 6 arte de los nuevos medios, por lo que la mitad de ellos había tenido contacto directo con la programación antes de comenzar a estudiar el Máster Interface Cultures. En el taller no se contó con ningún estudiante de Ingeniería u otra carrera puramente técnica.

En cuanto a la evaluación de la estructura del taller, la mayoría sugirieron una mayor duración, ya que hubiera permitido discusiones más largas y prototipos más elaborados. La parte teórica se les hizo muy extensa en comparación con la parte práctica, por ello, si este taller se realiza en el futuro se buscaría plantearlo en más sesiones y más separadas en el tiempo.

Una parte esencial del taller fue valorar cómo lo percibieron en relación con su práctica artística [Figura 5] y ver que la parte pedagógica fue la única que algunos de ellos consideraron 'Nada interesante' *(Not interesting at all),* aunque en una de las preguntas

[5] 'También creemos que las escuelas adoptarán el *live coding* para enseñar programación y música al mismo tiempo, ¡porque es una forma muy buena de aprender ambas materias! A profesores y alumnos les encantará' (Blackwell, Cocker, Cox, McLean y Magnusson, 2022, p. 51).

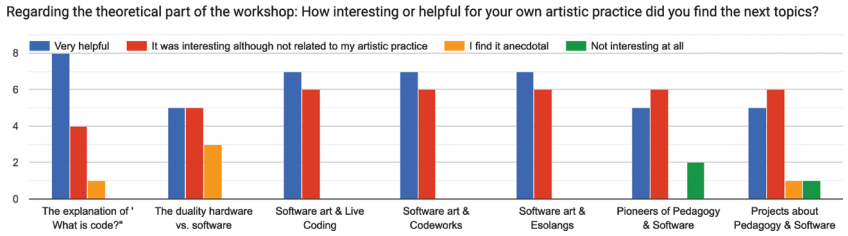

Figura 5. *Percepción del contenido del taller en relación con su práctica artística.*

finales, sobre cuál había sido la parte que les había resultado más interesante, algunos afirmaron que esta.

Conocer su definición de estética también era importante para ver si en algunos casos ya se habían planteado la idea de que la estética podía ir más allá de la belleza y tener una función transformadora, pero la mayoría se centraron en el aspecto visual y agradable. Uno de los participantes afirmó que la definición dada en el taller le era desconocida, pero que le había servido para modificar su concepción previa de este término.

CONCLUSIÓN

La apertura de posibilidades e interpretaciones que generó la idea de visibilizar el *software* y hacerlo de una manera estética fue uno de los resultados que se buscaba alcanzar con este taller y que se puede afirmar se consiguió tras la puesta en común de los prototipos diseñados por los participantes. Es notable como algunas propuestas incluso sacaron la programación fuera del ordenador para conducirla hasta el acto de tejer y a la *performance* como es el caso de las propuestas de Maria Dirneder y Maria Konstantinova, respectivamente.

Una de las conclusiones más importantes que se extrae en relación con la parte práctica es que la mayoría consideraron interesante crear una pieza de *software* estético, reflexionar sobre su valor pedagógico y también los debates que surgieron a lo largo del taller, aunque no consideraron que esta puesta en valor del *software* estuviese relacionada con su práctica artística directamente. El desarrollo de este taller permitió presentar el tema principal de mi tesis doctoral a estudiantes que trabajan con los nuevos medios y demostrar que la idea de trabajar con el *software* como materia visible y parte central de un proyecto puede resultar interesante pese a no ser una propuesta que se suela incluir en programas de grado o máster centrados en los nuevos medios. La fluidez en el desarrollo del taller fue posible gracias a la apertura de mente y colaboración de los participantes, pese a no ser un tema con el que estuviesen familiarizados.

En cuanto a los prototipos presentados, fue sorprendente la alta calidad de las propuestas. Esta se reflejó en la capacidad que prototipos como *Catn-IP* o Pizza tuvieron para proponer *esolangs* construidos utilizando imágenes y humor. Estas dos propuestas de *esolangs* visuales están muy conectados a la contemporaneidad: al mundo de internet en el primer caso y a una aproximación a los estereotipos gastronómicos de Italia en

el segundo. Aparte del humor, prototipos como *CollageTheftSelfPortrait y Neutralising Machine* emergieron como dos ejemplos de poesía computacional que denunciaban la falta de libertad para definir qué o cómo debe ser nuestra imagen y la capacidad devoradora del capitalismo, respectivamente. De este modo, es claramente observable la habilidad de los participantes para trasladar sus intereses al contenido del taller, demostrando así que los postulados del taller eran correctos y que una aproximación estética del *software* genera unas reflexiones y posibilidades nuevas.

Indirectamente, proyectos como *Catn-IP*, *Mindless Routine* o la propuesta textil de Maria Dirneder cumplen una función pedagógica respecto al *software*. En el primer caso, al emplear analogías visuales con imágenes de gatos tomadas de internet para hablar de estructuras inherentes a los lenguajes de programación (bucles, condicionales, imprimir, etc.) En el segundo, al mostrar el carácter repetitivo de los algoritmos si los aplicamos a acciones cotidianas y en el tercero por como mediante el acto de tejer propone una reflexión sobre la dificultad de implementar el sistema binario a un mundo físico lleno de matices.

Junto a los aspectos a mejorar de cara al futuro, cabe mencionar que la parte pedagógica necesitaría de un mayor desarrollo, debido a que en esta primera edición no tuvo el mismo peso que la parte estética. No obstante, por los resultados obtenidos, se propone la realización de otro taller en el futuro, de una mayor duración y en el que los contenidos puedan trabajarse más en profundidad ya que la carga teórica se consideró muy elevada para el tiempo en el que se contó para impartir el taller. Esta opinión también la compartieron la mayoría de los participantes.

AGRADECIMIENTOS

Agradecer su confianza al departamento Interface Culture de la Kunstuniversität Linz (Linz, Austria) y a los trece participantes del taller por su colaboración y compromiso con el buen desarrollo del taller.

FUENTES REFERENCIALES

Ackermann, E. K. (2004). A learning zone of one's own: Sharing representations and flow in collaborative learning environments. In *Constructing Knowledge and Transforming the World* (pp. 15–37). Amsterdam, Berlin, Oxford, Tokyo, Washington DC: IOS Press.

A programming language is a system of notation for writing computer programs. - Esolang. (n.d.). Recuperado el 14 febrero 2024 de: https://esolangs.org/wiki/A_programming_language_is_a_system_of_notation_for_writing_computer_programs.

Benjamin, W., Bullock, M. P., Jennings, M. W., Eiland, H. y Smith, G. (1996). *Selected writings*. Cambridge, Mass: Belknap Press.

Berardi, F. (2012). *The Uprising: On Poetry and Finance*. Los Angeles: Semiotext(e).

Blackwell, A. F., Cocker, E., Cox, G., McLean, A. y Magnusson, T. (2022). *Live coding: a user's manual*. Cambridge, Massachusetts: The MIT Press.

Colburn, T. R. (2000). *Philosophy and computer science*. Armonk, N.Y: M.E. Sharpe.

Cox, G. y McLean, A. (2013). *Speaking code: coding as aesthetic and political expression*. Cambridge, Mass: The MIT Press.

Daniel Temkin | Internet Directory. (n.d.). Recuperado el 4 octubre de 2023 de: https://www.danieltemkin.com/InternetDirectory

esoteric.codes. (n.d.). Recuperado el 13 de febrero de 2024 de: https://esoteric.codes/

Interview with Keymaker. (2011). Recuperado el 4 de octubre de 2023 de: https://esoteric.codes/blog/keymaker

Fallica, A. (2023) *Pizza.*

Hardman, K. (2023) *Collage Theft SelfPortrait.*

Konstantinova, M. (2023). *Mindless routine*.

Manovich, L. (2002). *The language of new media* (1st MIT Press pbk. ed). Cambridge, Mass: MIT Press.

Marenko, B. (2019). Algorithm Magic: Gilbert Simondon and Techno-Animism. In B. Marenko, *Believing in Bits* (pp. 213–228). Oxford University Press. https://doi.org/10.1093/oso/9780190949983.003.0013

Marino, M. C. (2020). *Critical code studies: initial methods*. Cambridge, Massachusetts: The MIT Press.

Montfort, N. (2016). *Exploratory programming for the arts and humanities*. Cambridge, Massachusetts: The MIT Press.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books.

Pérez-Campos, M. (2022). Taller (Des)cifradores de mensajes: una manera de presentar el funcionamiento interno de un compilador a niños de 6 a 12 años desde la experimentación artística. *Artnodes*, 0(30). doi: https://doi.org/10.7238/artnodes.v0i30.400553

Programminglanguage.(2024).In*Wikipedia*.Recuperadoel14defebrerode2024de:https://en.wikipedia.org/w/index.php?title=Programming_language&oldid=1206983426

Rangil, M. (2023). *Neutralising Machine.*

Sánchez Vázquez, A. (2005). *Las ideas estéticas de Marx* (Primera edición 2005, segunda reimpresión). México, D.F.: Siglo Veintiuno Editores.

Šermukšnis, D. (2023) *Catn-IP*.

Soon, W. y Cox, G. (2020). *Aesthetic Programming: A Handbook of Software Studies.* Open Humanities Press.

Talanti, S. (2023) *Tales of a 3D Artist Learning Coding.*