Code = Design

Par Kévin Donnot, publié dans *Graphisme en France 2012*

Après plus d'une quinzaine d'années de design graphique conçu sur ordinateur, la technique informatique reste mystérieuse pour la majorité des graphistes et encore peu de praticiens osent mettre les mains dans le cambouis. En leur temps, William Morris et le mouvement britannique des Arts and Crafts défendaient une création graphique intimement liée à la production artisanale et une maîtrise des outils du début à la fin de la chaîne, en réaction à l'industrialisation de la fin du xixe siècle. Morris était à la fois imprimeur, calligraphe, graveur de poinçons et responsable de la composition typographique, c'est-à-dire graphiste.

Comme Morris, on peut constater aujourd'hui une uniformisation de la production graphique. Par ailleurs, la grande majorité des designers utilisent les mêmes outils, créés par la même société (Adobe). L'homogénéisation des outils et celle de la production ne sont-elles pas liées ? Edward Tufte démontre dans The Cognitive Style of PowerPoint[1] que la conception de PowerPoint conduit non seulement à une uniformisation graphique, mais également, dans certains cas, à des décisions aberrantes, prises à l'issue de raisonnements faussés par le logiciel.

Pourquoi, comme William Morris, ne prendrions-nous pas nos outils en main ? Pourquoi ne pas sortir du sentier balisé par Adobe ? John Maeda[2] fut l'un des premiers à revendiquer de nouvelles formes visuelles basées sur le développement de ses propres logiciels. Il fut étudiant de Paul Rand et de Muriel Cooper, cofondatrice du Media Lab du MIT et pionnière de l'expérimentation visuelle numérique. D'autres ont suivi cette voie, comme les typographes de LettError[3], dessinateurs de caractères génératifs, ou le groupe bruxellois Open Source Publishing[4] qui travaille exclusivement avec des logiciels libres.

---

[1] Tufte, Edward. The Cognitive Style of PowerPoint, Cheshire, Graphics Press, 2006.

[2] John Maeda est l'auteur de plusieurs livres majeurs sur le design interactif : Maeda@media (2000), Design by Numbers (2001) et Creative Code (2004). Il préside aujourd'hui la Rhode Island School of Design.

[3] LettError est une fonderie digitale créée par les typographes néerlandais Erik Van Blokland et Just Van Rossum. www.letterror.com

[4] Open Source Publishing est un collectif de graphistes basé à Bruxelles. Il est lié à Constant, une association d'artistes travaillant sur la culture et l'éthique du web. http://www.ospublish.constantvzw.org

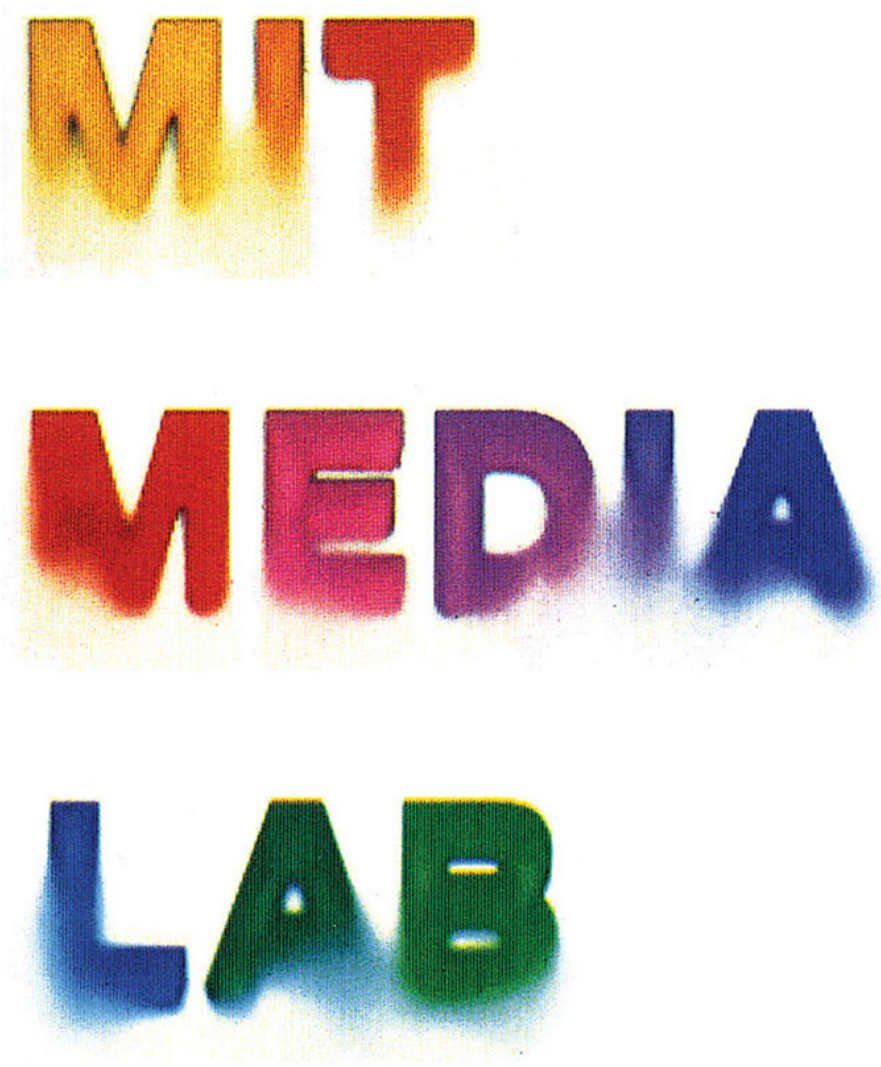

Figure 1: Muriel Cooper, MIT Media Lab, 1990. Déformation typographique rendue avec le système « Soft type » développé au Media Lab par Muriel Cooper.

La maison-prison des logiciels graphiques

Adobe Systems Incorporated est la société qui édite les cinq applications standard utilisées universellement par l'industrie graphique pour l'édition d'images et de textes, numériques et imprimés : InDesign, Illustrator, Photoshop, Flash et Dreamweaver. Ces programmes sont exemplaires et la majorité des designers s'en satisfait très bien, mais ils sont standard et, comme tout outil, ils ont leur empreinte propre. Si l'outil est standard, ce qui est produit a tendance à se standardiser.

Pour le designer tentant de se singulariser dans le brouhaha graphique ambiant, ces solutions logicielles peuvent être inadaptées. En effet, sous le prétexte de faciliter l'accès au plus grand nombre, la marge de manoeuvre laissée à l'utilisateur est réduite. Erik Van Blokland de LettError parle ainsi de « l'illusion de l'exhaustivité [...] soit l'idée que tout peut être fait en utilisant un menu déroulant et une barre d'outils[5] ». David Reinfurt, graphiste cofondateur de Dexter Sinister, affirme que « les fonctions présentes, les paradigmes logiciels et les scénarios d'utilisation sont planifiés pour chaque projet de programme, afin de s'assurer de la plus large utilisation possible. En résulte un outil moyen, qui passe outre les hauts, les bas, les erreurs et les bizarreries[6]». Toutes les spécificités qui pouvaient ainsi apparaître durant le développement de ces programmes sont supprimées pour répondre à l'usage médian, conventionnel.

Ces outils sont paramétrés par défaut en vue d'une plus grande facilité d'accès. Par exemple, un automatisme permet de saisir immédiatement du texte dans un nouveau document InDesign. Ce texte sera alors composé automatiquement avec un caractère acceptable (Arial), un corps acceptable (12 points), un interlignage acceptable (120 % du corps) et une couleur acceptable (noir). Tout cela est merveilleux, mais n'est-ce pas un peu réducteur, graphiquement parlant ? Ces réglages par défaut n'influencent-ils pas nos choix, quand on oublie de les modifier et ainsi de prendre une décision ? Comme le dit Loretta Staples, graphiste spécialisée dans les interfaces utilisateur, ces programmes créent « un espace [...] où la facilité d'utilisation prend le pas sur notre autorité d'auteur[7] ».

Un outil est conventionnellement perçu comme un objet servant l'expression du créateur et devant interférer le moins possible avec l'idée abstraite qu'il s'agit de matérialiser. Pierre-Damien Huyghe affirme ainsi que « l'ingéniosité (d'un outil) [...] consiste à obtenir que le travail matériel de construction effectué

[5] « The illusion of completeness [...] The idea that anything can be achieved using dropdown menu and toolbox sidebar. » Crow, David. « Magic box : craft and the computer », Eye Magazine no 70, hiver 2008, p. 25.

[6] « Function sets, software paradigms, and user scenarios are mapped out for each software project to ensure the widest possible usability, resulting in an averaged tool which skips the highs, lows, errors, and quirks. » Reinfurt, David. « Making do and getting by », in : Kyes, Zak ; Owens, Mark. Forms of Inquiry : The Architecture of Critical Graphic Design, Architectural Association Publications, Londres, 2007, p. 132.

[7] « The new computer-generated environment [...] is a space [...] where user-friendliness overrides the authority of the author. » Staples, Loretta. « What happens when the edges dissolve ? », Eye Magazine no 18, automne 1995.

pas à pas ne vienne pas faire de bruit dans la présence ultime de l'oeuvre[8] ». Pour lui, un outil est ingénieux, de qualité, s'il n'influence pas ce qu'il produit, c'est-à-dire l'oeuvre. Or, comme tout pinceau laisse une trace spécifique, tout logiciel façonne les décisions de son opérateur par la conception même de son interface et de sa logique interne. Cette affirmation a été étayée par la thèse d'Amod Damle, professeur de Computing and New Media à l'université du Wisconsin : « Les processus impliqués dans une activité créative comme le design peuvent être influencés de manière fondamentale par les spécificités de l'outil mis à disposition[9]. » Pourquoi ne pas assumer cette influence et choisir un outil en fonction de son empreinte ? Ne faudrait-il pas s'interroger sur l'outil qu'il serait juste d'employer avant de se tourner machinalement vers son logiciel habituel ? La vraie question serait : quel conditionnement[10] choisit-on pour mener à bien tel projet ? Il existe des alternatives aux logiciels graphiques standard. Ces autres programmes ne sont pas meilleurs en termes de rendement ou de facilité d'accès, mais proposent souvent une approche différente du WYSIWYG[11].

Logiciel libre et hacking

Les alternatives aux outils commerciaux sont principalement des logiciels libres[12]. L'idée d'un tel programme fut lancée par Richard Stallman alors qu'il travaillait sur le système d'exploitation GNU[13] au MIT en 1983. Stallman avait pour but de « ramener l'esprit de coopération qui avait prévalu autrefois dans la communauté hacker, quand la question de la propriété intellectuelle du code n'existait pas et que tous les codes sources s'échangeaient librement[14]. » GNU donnera plus tard naissance au système Linux permettant à quiconque d'exploiter un ordinateur librement et gratuitement. En 1985, Stallman crée également la Free Software Foundation pour assurer une structure légale et financière à son projet. Il y définit les quatre libertés fondamentales que doit

[8] Huygue, Pierre-Damien. Modernes sans modernité. Éditions Lignes, Fécamp, 2010, p.80.

[9] « The problem-solving processes involved in a creative activity like design can be influenced in fundamental ways by the features of the tool provided .» Damle, Amod. Influence of design tools on design problem solving. Thèse de philosophie, Colombus : Département Industrial and Systems Engineering, université d'État de l'Ohio, 2008 . Résumé. Damle a formé aléatoirement deux groupes de quinze designers expérimentés. Chaque groupe devait dessiner une lampe en sélectionnant et en combinant deux éléments de chacune des deux autres lampes présentées comme références. Afin de créer ce dessin, il a été demandé aux participants d'assembler plusieurs segments de droite de tailles et d'orientations différentes sur un ordinateur. Pour le premier groupe, les lignes étaient d'une seule couleur, pour le second, elles étaient multicolores. Damle a pu observer que le second groupe apportait plus d'attention au détail de chaque élément plutôt qu'à la forme globale de la lampe.

[10] Il faut ici entendre le mot conditionnement à la fois au sens psychologique, comme un schéma de pensée préétabli par autrui, et au sens marchand, désignant le packaging, l'emballage du produit.

[11] Acronyme de What You See Is What You Get.

[12] À ne pas confondre avec la notion d'open source.

[13] Acronyme récursif de GNU's Not Unix (GNU n'est pas Unix), Unix étant un système d'exploitation propriétaire développé en 1969.

[14] Richard Stallman, 1984.

garantir un logiciel libre.

- liberté d'exécuter le programme pour tous les usages ;
- liberté d'étudier le fonctionnement du programme et de l'adapter à ses besoins – ceci impliquant un code source ouvert ;
- liberté de redistribuer des copies, donc d'aider son voisin ;
- liberté d'améliorer le programme et de publier ses améliorations, pour en faire profiter toute la communauté, ceci impliquant également un code source ouvert[15].

Cette définition donne une éthique au logiciel, dès lors considéré comme un outil de libération détaché de toute logique commerciale[16]. Processing est un bon exemple d'application libre potentiellement employée pour le design graphique. Logiciel généraliste, il est avant tout destiné aux artistes réalisant des pièces interactives ou génératives via un langage de programmation dédié. Ce langage possède une syntaxe simple et le programme est facile à mettre en oeuvre. Il est fréquemment utilisé pour visualiser des données (graphisme d'information). Ce logiciel, malgré de grosses lacunes dans la gestion de la typographie, propose un tout nouvel espace d'expérimentation visuelle, où le design n'est plus WYSIWYG mais piloté par du code. Cette approche différente implique des processus de création différents et donc des propositions graphiques différentes. Cependant, la définition de Stallman ne relève que de considérations éthiques. Prenons l'exemple de Scribus, un logiciel libre de mise en page. Ce n'est qu'une pâle copie d'InDesign version free software : où est alors l'intérêt graphique ? Moins efficace que son concurrent, ce programme est incompatible avec les formats de fichiers en vigueur et défaillant dans la production de fichiers PDF[17]… Certes, on a toute liberté de l'améliorer puisqu'il est sous licence GNU, mais à quoi bon réinventer la roue[18] ?

La vigilance est de mise afin que l'éthique logicielle ne prenne jamais le pas sur la production visuelle et que le statut d'auteur soit préservé de tout diktat idéologique. Les grandes firmes commerciales ont d'ailleurs contribué à l'établissement de normes standard ouvertes comme le PDF[19] ou l'Opentype[20]. Graphiquement parlant, le libre ne présente pas d'autre intérêt que sa source ouverte qui autorise le façonnement personnalisé du programme par la modification de son code source . L'utilisateur peut ainsi intégrer les fonctions de son choix, mais aussi et surtout comprendre comment fonctionne son outil. Il évite ainsi de se voir « réduit à la situation d'usager ou de consommateur[21] » et con-

[15]Définition d'un logiciel libre, GNU Project/Free Software Foundation. www.gnu.org/philosophy/free-sw.fr.html

[16]Libre ne signifie pas nécessairement gratuit. « Think of ‹ free » as in ‹ free speech ›, not as in ‹ free beer › » : pensez ‹ libre › comme dans ‹ liberté d'expression ›, pas comme dans ‹ bière gratuite ›. Richard Stallman sur www.gnu.org

[17]Notamment des problèmes avec les plaques en tons directs.

[18]Si ce n'est pour apprendre.

[19]PDF est un format de document dédié à l'impression développé par Adobe.

[20]Opentype est un format de police universel développé par Microsoft et Adobe.

[21]Huygue, Pierre-Damien. Modernes sans modernité, op. cit., p. 111.

damné à la passivité technique. Il s'agit de passer du statut de consommateur de logiciel à celui de créateur. Cette attitude libertaire et autonome est relayée par la culture hacker, également liée à l'univers DIY[22]. Tel un tourneur qui façonne son outil pour tourner sa pièce comme il l'entend, un graphiste-hacker pourrait créer ses programmes à sa main, pour répondre à ses exigences propres, lesquelles participent de son statut d'auteur.

« The craft of programming[23] »

Même si tous les graphistes travaillent avec un ordinateur, il est paradoxal de constater que quelques-uns seulement ont un rapport créatif à la technique informatique. La machine est très souvent considérée comme une black box mystérieuse, hermétique, compliquée, quand ceux ou celles qui en saisissent la logique ne sont pas taxés de techniciens ou d'exécutants… La programmation n'est pas un impératif technique pour les travaux d'édition, mais la conception dépend néanmoins de l'informatique, ce qui pourrait induire une réflexion sur les outils numériques. « Quiconque est impliqué dans la production culturelle, avec, pour, ou autour d'un ordinateur devrait savoir comment lire, écrire et penser les programmes informatiques[24]. » Par ailleurs, malgré l'engouement actuel pour le livre d'art et l'édition bien pensée[25], il est absurde d'envisager un avenir sans commande sur média numérique. Comme il est inimaginable de travailler sur des formes papier sans connaître les processus d'impression, comment penser pouvoir produire des formes numériques sans en maîtriser les rouages ? Comment concevoir sérieusement un site web si l'on n'est pas familier, d'une part avec le média lui-même, et d'autre part avec la réalisation technique, c'est-à-dire la programmation ? Il est nécessaire que « les designers et les développeurs ne soient placés en équipe que si chacun a une connaissance du champ d'expertise de l'autre[26]. » Il ne s'agit pas de former des développeurs professionnels, mais des designers suffisamment autonomes pour développer leurs prototypes, ayant aussi la possibilité de faire entrer la programmation informatique dans leur méthode de création. On peut alors envisager le design logiciel non plus comme

[22] Acronyme de Do It Yourself.

[23] « La fabrication de l'acte de programmation » Crow, David. « Magic box : craft and the computer », art. cit., p. 24.

[24] « Anyone involved in cultural production on, in or around a computer should know how to read, write and think about programs. » Davidson, Drew. Beyond Fun : Serious Games and Media, p. 81

[25] On peut observer depuis quelques années une multiplication des salons indépendants d'édition d'art (salons Light et Offprint à Paris, Motto Fair à Berlin…)

[26] « Only practitioners who combine procedural literacy with a conceptual and historical grounding in art and design can bridge this gap and enable true collaboration. » Seuls les praticiens possédant à la fois des techniques de programmation et un bagage historique et conceptuel en art et en design peuvent franchir ce fossé et engager une véritable collaboration (avec les développeurs). Mateas, Michael. « Procedural literacy : educating the new media practitioner ». On the Horizon, no 1, 2005.

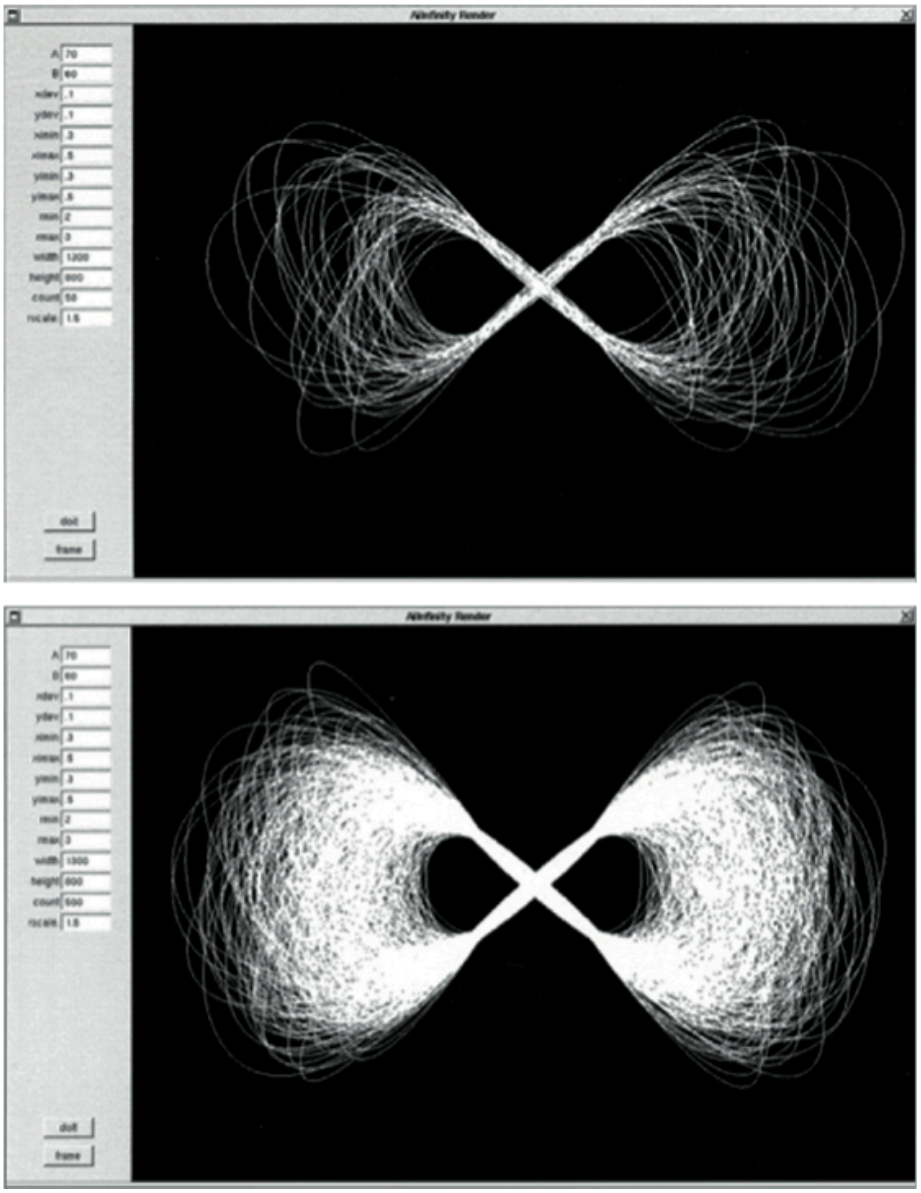

Figure 2: John Maeda, Infinity, 1993. Image générative dessinée avec un programme informatique développé par John Maeda. Différents paramètres (à gauche sur les captures d'écran) influent le tracé mathématique de la ligne. La superposition de plusieurs centaines de courbes, chacune modulée par une variable aléatoire, génère l'image finale. « This was the first image I made as a means to understand the computer less as a tool, and more as a new material for drawing. », « Ce fut la première image que je fis me permettant de considérer l'ordinateur moins comme un outil et plus comme un nouveau matériau pour dessiner. »

une technique au sens réducteur du terme, mais comme partie intégrante du processus de design graphique. « La puissance créative, c'est écrire le code du filtre, c'est décider comment il marche, ce n'est pas l'utiliser[27]. »

Créer son outil, c'est faire des choix qui détermineront le résultat final, comme n'importe quel choix de conception. David Crow évoque The craft of programming, craft étant entendu au sens des Arts and Crafts, avec une relation à l'artisanat, à la main. Le code est une matière à modeler comme peut l'être un pain de terre glaise ou un bloc de texte. Programmer, c'est articuler des structures logiques en les appliquant à des données, ce qui génère un résultat. Le travail du code ne regarde pas tant les données de base, ni le résultat, mais le traitement de ces données par l'enchaînement des instructions. La finalité, ce sont les données représentées, le contenu mis en forme, sans qu'aucun autre artefact extérieur n'intervienne. Dans le cas d'un logiciel comme Processing, le code textuel est interprété par la machine pour générer une matrice de pixels, elle-même perçue comme une image. L'équivalence du texte à l'image est directe, mécanique. Le rapport image/texte, à la base même du design graphique, n'est plus ici directement maîtrisé et visualisé (comme avec un logiciel WYSIWYG ou des morceaux de papier), mais dissocié dans le temps, asynchrone. Ce décalage entre conception et visualisation du résultat entraîne nécessairement une perte de maîtrise : il n'y a plus de retour visuel immédiat sur ce que l'on dessine. C'est dommageable si l'on poursuit un objectif formel précis, mais cela génère également souvent des surprises graphiques nées du contenu même. Ces surprises sont autant d'ouvertures formelles potentielles, émanant directement des données de base. De nouvelles méthodes de création pourraient ainsi émerger des techniques de programmation : Github[28], par exemple, est un site web destiné aux programmeurs qui permet à la fois un partage du code et une archive, version après version. La spécificité de cet outil est de pouvoir forker un projet, c'est-à-dire se l'approprier et en proposer une modification ou un autre développement. Les programmes sont ainsi enrichis de nombreuses variations incluant de nouvelles fonctions ou de nouvelles applications. Le code est partagé par tout le monde et tout le monde peut faire évoluer n'importe quel projet dans n'importe quelle direction.

Si l'on reconnaît la possibilité de créer des objets de design graphique en programmant, ce type d'outil propose une toute nouvelle méthode de conception : un design mutualisé, partagé. Il s'agirait alors à la fois d'utiliser des fragments développés par d'autres et de mettre ses créations à disposition de la communauté[29]. Même s'il est admis depuis une quinzaine d'années que les designers graphiques devraient être à l'aise avec leur outil, l'ordinateur, la réalité est tout

[27] « Creative power comes from writing the code of the filter, deciding how it works, not from using it. » Texte de présentation de la typographie Robotfont éditée par LettError. www.letterror.com/content/nypels/robotfont.html

[28] http://www.github.com

[29] C'est dans cet esprit de coopération qu'ont été créées les licences Creative Commons, qui protègent le créateur de l'oeuvre originale en même temps qu'elles en autorisent l'adaptation. http://www.creativecommons.org

autre et peu de graphistes se frottent au développement. Mais depuis 2007, la dimension sociale d'Internet est de plus en plus marquée et de véritables réseaux de designers-développeurs se concentrent autour d'initiatives comme Github, Processing ou Arduino, installant de nouvelles structures de création – autant d'incitations à repenser les relations entre design, outils de création et programmation.

Le texte « Code = design » est basé sur le mémoire de fin d'études Outils numériques et design graphique de Kévin Donnot, suivi par Catherine de Smet et Isabelle Jégo et soutenu en mars 2011 à l'École européenne supérieure d'art de Bretagne – site de Rennes.