Art de la programmation et programmation esthétique

Annick Bureaud

Art Press, n°283, octobre 2002

Un programme n'est ni de la matière ni de l'énergie, c'est juste un type d'information. Stephan Diehl[1].

Un programme est un matériau avec une esthétique inhérente. Boris Müller[2].

En 2001, le festival *Transmediale* de Berlin crée une catégorie *Software art* (art du programme), la même année, le Macros-Center de Moscou lance une compétition d'art du programme qui trouve son aboutissement dans le festival *read_me* au printemps 2002. Juillet 2002, le séminaire *Aesthetic Computing* (programmation esthétique) réunit, au Centre de Dagsthul, informaticiens, designers en informatique et artistes, à l'initiative des premiers. Eté 2002, l'ICC ouvre l'exposition *art.bit collection*[3].

Florian Cramer et Ulrike Gabriel définissent l'art du programme pour Transmediale comme "un art dont le matériau est le programme lui-même"[4]. Olga Goriunova et Alexei Shulgin écrivent pour read_me qu'il s'agit de "programme créé dans un but différent des buts pragmatiques habituels"[5] et le jury déclare "considérer comme art du programme un art dont le matériau sont les instructions algorithmiques et/ou qui met l'accent sur les concepts culturels des logiciels"[6]. Les participants de Dagstuhl définissent la programmation esthétique comme "la théorie, la pratique et l'application de l'esthétique à la programmation, l'esthétique étant la théorie et la philosophie de l'art".

Au sein de l'art numérique, l'art de la programmation a pris un nouvel essor et constitue un axe clé de la création et de la discussion théorique qui n'ont jamais été aussi vivantes. Un de ses mérites, et non des moindres, est de sortir l'analyse d'une typologie par média pour la recentrer sur le matériau de l'art, la création et son processus, et l'esthétique.
La variété du vocabulaire témoigne de l'intensité et de la richesse des débats et des enjeux : art du code, art du programme (*software art*)[7], programme artistique (*artistic software*), art algorithmique, *code-based media art* (art des médias reposant sur le code, introduit par Anne Nigten), art de la programmation, art numérique.
À ce stade de notre réflexion, nous utilisons "art de la programmation" comme terme générique désignant un ensemble de pratiques au sein de l'art numérique, tout en étant consciente de ses limites et du biais qu'il peut introduire. L'art de la programmation ne serait-il pas lui-même une catégorie plutôt qu'un terme réellement général ?

Le terme "art du code", s'il est séduisant, n'est pas un outil opératoire en ce qu'il n'apporte pas de réelle discrimination. En effet, toute pensée, toute expression se formule dans un code[8]. Par ailleurs, l'univers numérique est constitué de couches superposées, voire tissées, de code(s). Il y a toujours un programme sous le programme, à chaque strate, ce qui est le signifié devient le signifiant et *vice-versa*[9]. Opérer une stratification permet de distinguer entre des objets dont le *matériau* est identique : un certain type d'information, pour reprendre l'expression de S. Diehl. Sommairement, nous en distinguons trois.
La première strate est le niveau perceptible et sensoriel[10]. On y retrouve les œuvres relevant de l'esthétique du desktop et de l'interface graphique qui jouent et se jouent du design informatique et montrent que l'interface *relève aussi* de la programmation[11].

La seconde est celle que nous qualifions de niveau du modèle. L'œuvre est le processus de création en action[12]. Les œuvres génératives, de vie artificielle, algorithmiques (comme celles de Manfred Mohr[13]) et la plupart de celles présentées dans les manifestations de *software art* appartiennent à ce niveau.
Tout comme pour le niveau précédent, deux tendances, bien souvent antagonistes, s'en dégagent : celle d'une approche esthétique formelle[14] et celle d'une critique culturelle et sociale[15].
La troisième strate est celle du méta-modèle ou de l'écriture du langage d'écriture. Linux et le mouvement de l'open source et du libre en sont les meilleurs exemples. Dans ce territoire métisse, à la croisée de l'informatique et de l'art, l'écriture de logiciels libres est pour Florian Cramer, une des plus hautes formes de la littérature numérique[16].

L'art de la programmation dispose d'une problématique établie et d'une filiation artistique. La programmation esthétique (*Aesthetic Computing)*, en revanche, a tenté pour la première fois à Dagstuhl de définir ses contours et ses enjeux. Elle est issue, pour faire bref, d'informaticiens qui ne considèrent pas qu'ils "font de l'art" parce qu'ils écrivent des programmes, mais qu'il y a bien des éléments esthétiques et culturels fondamentaux dans leurs pratiques à caractère utilitaire —puisqu'il s'agit de "donner une forme à des concepts"— avec une volonté (un désir ?) d'élaborer des esthétiques alternatives ... et néanmoins fonctionnelles (ou que les propositions seront d'autant plus fonctionnelles qu'elles sont ... esthétiques).
La programmation esthétique s'étage également sur plusieurs niveaux, chacun avec des problèmes différents à résoudre, des structures de modélisation formelle, à la visualisation de logiciels, au design de l'interaction et des interfaces. Différents axes de recherche et de questionnement articulent ses problématiques :
- Comment sortir d'un système de notation et de codification abstrait (lié aux mathématiques) ? Peut-on envisager un système iconique ? ou associé à des objets tangibles ? ;

- La nécessité d'inventer de nouvelles représentations et métaphores ;
- Une "programmation de l'ouverture", considérée ici essentiellement comme la possibilité de "personnalisation" par l'utilisateur. Celle-ci est largement une illusion et reste dans une logique consumériste : le fait d'avoir une voiture rouge avec des sièges bleus ou bleue avec des sièges rouges ne change rien au fait qu'elle a le même moteur (en langage informatique, les mêmes fonctionnalités). Une ouverture réelle devrait permettre la programmation des interfaces, voire des fonctionnalités, donc la modification du programme par l'utilisateur. Ceci ouvre à des problèmes ardus dont la communication inter utilisateurs et avec le concepteur n'est pas l'un des moindres. À notre connaissance, pas plus en art qu'en informatique, ceci n'est offert, à l'exception, précisément, de l'élaboration des logiciels libres ;
- La nécessité de s'interroger sur les bases de la doxa de la fonctionnalité et de l'utilisabilité[17] ;
- L'approfondissement du design et de la conception de l'interaction humains/machines (HCI) dans une approche systémique qui met en jeu les qualités sensorielles, perceptuelles, sociales et culturelles de la pratique des technologies de l'information.

L'utilité —ou la fonctionnalité— d'un programme est souvent posée comme distinction entre ce qui relève de l'art de ce qui relève de l'informatique. À la lumière des œuvres et des discussions en cours, cette opposition montre toute sa faiblesse. En visualisation de logiciels, est esthétique, "un programme qui remplit les minima des fonctionnalités ayant été définies comme devant être accomplies". Cette définition pourrait tout aussi bien s'appliquer à un objet artistique, c'est ce que nous appelons la "cohérence" de l'œuvre, entre le fond et la forme, l'intention et sa réalisation.
Une dichotomie plus intéressante est celle établie par Jay Bolter entre transparence et réflectivité[18]. La transparence tend à faire disparaître la programmation, à la rendre invisible, tandis

que la réflectivité met en avant un regard réflexif sur la technologie. Une analyse, sur cette base, des logiciels "utilitaires" et des programmes artistiques reste à faire.

---

[1] Dans l'introduction de *Software Visualization*, Springer, LNCS 2269, Berlin, Heidelberg, 2002.
[2] Lors du séminaire *Aesthetic Computing,* Centre International de conférence et de recherche en informatique, Château Dagstuhl, juillet 2002.
[3] www.art-bit.jp
[4] "Software Art" dans le catalogue-livre *Do It Yourself Media*, Transmediale 01, Berlin, 2001.
[5] "Artistic Software for Dummies and, by the way, Thoughts About the New World Order, site de read_me, www.macros-center.ru/read_me
[6] Email à la liste Spectre annonçant les prix, le 22 mai 2002.
[7] Pour notre part, et en français, nous établissons une différence entre *logiciel* et *programme* bien que ce soient des artefacts de même nature. Nous utilisons le mot "programme" de manière générique et le mot "logiciel" pour désigner des programmes commerciaux, vendus avec les équipements ou séparément et quel que soit le niveau auquel ils agissent (du logiciel système au traitement de texte ou graphique, etc.).
[8] À propos du code, voir le texte de Vilèm Flusser, "Le monde codifié", conférence prononcée le 3 mai 1973 à l'Institut de l'Environnement.
[9] Cf. "Corps virtuels et signifiants clignotants", Katherine Hayles, in *Connexions : art, réseaux, media*, A. Bureaud & N. Magan (dir.), Paris, Ensba, 2002.
[10] Ce qui correspondrait en partie à ce que Jean-Pierre Balpe appelle le niveau de "surface".
[11] Voir notre article "Small is beautiful" dans *Art Press* et les écrits de Lev Manovich (www.manovich.net).
[12] En fait, l'œuvre est le processus de création dans sa formalisation et dans sa mise en œuvre, et la relation perceptuelle entre les deux. Voir, *Contextes de l'art numérique,* de Jean-Pierre Balpe, Paris, Ed. Hermès, 2000 et "The Aesthetics of Generative Code" de G. Cox, A. McLean et A. Ward, www.generative.net.
[13] www.emohr.com
[14] Par exemple : www.generative.net
[15] Par exemple : le groupe Mongrel et le programme *Linker*, www.linker.org.uk/Linker/
[16] "Des logiciels libres comme texte collectif", in *Connexions*, op. cité.
[17] En anglais *usability*, vocable parfaitement monstrueux, et premier des commandements de la programmation !
[18] Ce que Maurice Benayoun appelle la "rugosité".