PALOQUE-BERGÈS, Camille [2009]

***Poétique des codes sur le réseau informatique*.**

Paris: Éditions des archives contemporaines, 193 pages

AB, octobre 2009

Exploration des façons selon lesquelles les «écritures qui mettent en scène l'art de la programmation» [17] explicitent et articulent la question du code, *Poétique des codes* de Camille Paloque-Bergès repose son développement sur les critères très efficaces à travers lesquels Florian Cramer décrit les qualités propres à l'écriture du code:

– son *exécutabilité*: le code est écriture de la performance, de la performativité (le code, en fonction, agit ce qu'il énonce, agit en l'énonçant);

– sa *textualité*: le code est écriture textuelle, où reconnaître des propriétés esthétiques à part entière (ses qualités esthétiques et intentionnelles).

Suivant l'insistance avec laquelle les œuvres textuelles «à code informatique» travaillent l'une ou l'autre de ces deux qualités, Camille Paloque-Bergès distingue deux moments en fonction desquels très schématiquement saisir la performance du code dans la qualification poétique des œuvres textuelles à code numérique:

– *poiesis* pratique du code, tendant vers l'implémentation (la mise en œuvre) du code, expérimental;

– médiatisation rhétorique du code, tendant vers la représentation (la mise en scène) du code, délié de ses fonctions pragmatiques et performatives.

Le code mis en œuvre

L'une des originalités de la recherche de Camille Paloque-Bergès est d'articuler la question de la génération programmée de textes (poétiques, littéraires) avec celle, plus pointue encore, des pratiques expérimentales du code par diverses communautés de hackers-progammeurs: pour l'une comme pour les autres, l'exécution du code (l'opération du code exécutable) constitue le cœur de leur projet créatif. Cette qualification de l'écriture du code par son exécutabilité (par le fait que le code soit donc constitué d'instructions adressées à un processeur qui les exécute) suggère à Camille Paloque-Bergès un rapprochement avec la notion austinienne de «performativité».

Performatif

1. le performatif comme acte illocutoire

2. le texte du code est performatif au sens où ce qu'il dit, la machine le fait

Pratiques expérimentales du code: génération programmée et hack

Dans le champ de la génération programmée, le code est la formulation logique du processus génératif. Se pose alors, pour la génération poétique, un paradoxe, celui de l'innovation poétique: si le code est générateur d'une classe de textes reconnaissables (comme poèmes, …), ses productions ne valent esthétiquement que pour autant qu'elles échappent à la prédictibilité de leur programmation. Avec les pratiques de génération programmée, l'attention esthétique se porte ainsi sur le code, sur le texte-producteur où s'expérimentent des méthodes

innovantes d'exploration du langage, plutôt que sur le/s texte/s produit/s. Il s'agit dès lors d'investiguer les qualités esthétiques (intentionnelles) de l'écriture du programme, du code lui-même, script/source d'un processus implémenté, dont les réalisations (les textes générés) ne constituent somme toute qu'un sous-produit, un produit dérivé (*a by-product*).

À la question, non-triviale, de savoir comment intéresser leurs lecteurs à l'œuvre de la génération programmée, plusieurs créateurs répondent que:

– les actions d'un morceau de code dépassent de beaucoup l'acte de traduction des intentions d'un auteur: il y a quelque chose comme une intentionnalité machinique (Cox, McLean, Ward, d'une part; Bootz, d'autre part). «C'est quand l'accident se produit que la machine "s'exprime"», que «le code devient expressif» [124]. Les comportements aberrants (interférences, accidents, pannes, virus, spams, bugs, …) du point de vue d'une lecture humaine, jouent sur cette équivoque entre code et bruit.

– si le quantitatif est du côté du/des texte/s produit/s (l'excès combinatoire), le qualitatif réside dans le code, dans l'écriture du programme: le code joue donc un rôle de premier plan (Jim Carpenter, *Electronic Text Composition*; Hartman, *Travesty*), quoiqu'il ne soit généralement, pour ces œuvres-là, pas mis en avant.

De ce point de vue (qui consiste donc à considérer le code dans son fonctionnement producteur), les pratiques du code telles que diverses communautés de hackers-programmeurs les expérimentent, constituent un champ exploratoire idéal pour une poétique du code: il s'agit là de «comprendre comment des modèles esthétiques peuvent émerger de […] pratiques expérimentales du code» [53], de considérer comment, plutôt que de décoder un code – ce qui constitue une approche littérale du code comme mécanique utilitaire de production –, en «faire émerger une forme dont on n'est pas certain qu'elle existe» [70] certes, mais dont on décèle les indices, les marques d'une configuration «poétique»: «le code [y] opère une reconnaissance de lui-même comme écriture par une série d'acte réflexifs» [58], de «jeux de langage»: «poétisation du code» [63]. Exécutable par la machine, ce code poétisé perd sa transparence, son évidence fonctionnelle à la lecture humaine dont le processus est supsendu, en arrêt: fonctionnel mais obfusqué (*obfuscated*), opacifié, bruité.

Nous tenons là une première réponse à la question posée: l'illisification, l'obfuscation, constituent des procédés esthétiques à part entière, à travers lesquels le code «prend forme», poétique. Ou: à travers lesquels le code fait résistance par sa matérialité-même (William Morris: «You can't have art without resistance in the material»). Ou encore: à travers lesquels le code met en évidence le côté palpable de ses signes (Roman Jakobson). Ou encore – les références sont ici nombreuses pour créditer ce geste paradoxal qui associe la poéticité aux «bruits» dans la langue, aux irruptions et interruptions, «herbes entre les pavés» (Michel de Certeau, Umberto Eco, …): à travers lesquels le code expose le conditionnement «physique» de son fonctionnement en lequel il finit par consister (Jerome McGann).

Les exemples convoqués à l'illustration de différents procédés d'obfuscation sont:

– pour la réflexivité: *The Hello World Collection*, «premiers gestes de réflexivité esthétique», dont la performance est alimentée «par la force illocutoire de ces codes qui deviennent textes» [60-61];

– pour l'auto-référencialité: la collection de programmes *Quine*, dont l'enjeu est d'«écrire un programme qui génère une copie exacte de son code, i.e. de lui-même»: «le résultat textuel est constitué par ses propres conditions de productions, le code lui-même» [61];

– pour l'obfuscation proprement dite: *The International Obfuscated C Code Contest* (IOCCC). Camille Paloque-Bergès remarque une forte tendance calligrammatique des programmes obfusqués;

– pour le codage créatif (*creative coding*): les langages ésotériques (*weird languages*), «poétisation[s] radicale[s] du code, […] entièrement tournés vers le processus – le résultat n'important plus» [71], et dont 4 propriétés sont remarquables: la qualité parodique, le caractère minimaliste, l'effet intrigant (puzzle), le (double-)jeu de langage.

Camille Paloque-Bergès consacre tout une analyse [75-85] au langage PERL pour lequel le développement d'un genre poétique à part entière est désormais reconnu sous l'expression «poésie PERL»: ce genre fonctionne essentiellement sur le principe du double-codage, le code programmé étant interprété par la machine autant qu'il est lu, mais sur un registre lexical différent, par un lecteur humain.

Le code mis en scène

Par distinction d'avec l'approche «performative» (au sens de «ce que [le code] dit, la machine le fait» [87], le met en œuvre) consistant, par double-codage, à *poétiser le code* (sa lecture à l'œil humain figuralisant son interprétation par la machine), *littérariser le code* c'est abstraire le code de sa fonction: le code écrit imite «sans prendre en compte la pragmatique du langage» [84]. Plutôt qu'à son *implémentation expérimentale* (*poesis*, pragmatique à travers laquelle le code, exécutable, est révélé comme forme), le travail sur le code tend alors davantage à sa *représentation* (imitation, *mimesis*, image de code): «les processus sont échangés pour la représentation» [123]. Camille Paloque-Bergès décrit cette appropriation des langages informatiques par une énonciation «littéraire» («Je littéraire» [118], encore rapportée comme une «écriture artiste du code» [121]), qui délie le code de ses fonctions pragmatiques et performatives [115], comme une remédiatisation rhétorique (la qualification en «rhétorique» connote ici négativement – faux-code [112], imitation, image [112, 115], phénomène [115] – autant qu'elle signe le caractère positivement figural de la rémédiatisation en question) de l'expérience du code: ce code qui émerge en surface de l'œuvre est une sorte de «faux-code, non pas un texte de programme montré pour ses capacités à agir, mais une image du texte-code, une projection du code» [112]. Le code fait ici figure métonymique: figure (*in absentia*) de surface qui «signe vers les profondeurs du code» [111], symbole de conventions culturelles [121], réflexion sur la nature digitale de l'écriture informatique [115], …

De cette littérarisation du code, Camille Paloque-Bergès en trouve l'exemplification

– dans les langages cypher/cipher, tels *l33t*, qui agissent «au niveau de l'unité graphique» [127] du code ainsi exposé, manifesté dans sa matérialité, qui «reproduisent l'idée du code comme langage d'expression de la machine, à la fois pourvoyeuse de solution et génératrice de bruit» [128];

– dans les textes des codeworkers (Giselle Beiguelman, Mary-Anne Breeze [*aka* mez], NN, Jodi, Alan Sondheim, Talan Memmott, …) qui «formulent et représentent les discours des perturbations et de la subversion par le code» [129]. Les *codeworks* imitent les langages de programmation *obfusqués*, procèdent par prélèvement au sein de jargons informatiques.

Giselle Beiguelman: «@+he !n+e®$ec+!0n 0f \/\/0®&$ & $ymb0|$ \/\/e beg!n +0 ®e7ef! ne 0u® b0unda®!e$ (At the intersection of words and symbols we begin to redefine our boundaries)» [128], déconstruit/reconstruit un langage dont l'orthographe textuelle hybride les signes linguistiques traditionnels et les marqueurs symboliques d'un code informatique. Ces marqueurs paraissent certes être des signes linguistiques dégradés (le "i" subit un renversement en "!", le "t" l'amputation de sa jambe qui ne laisse qu'un "+", le "w" sa désarticulation en "\ / \ /", etc.), ils figurent («symbols») cependant tout autant les morphèmes d'un code affleurant en surface du texte: le chiffre binaire "0", le marqueur de variable "$", le "@" d'une adresse email, les opérateurs de concaténation "+", de négation "!", etc.

Mary-Anne Breeze, mez, invente le langage *Mezangelle* qui «utilise les "langues" informatiques comme outil de déconstruction de la langue anglaise […]. Un ensemble de signes sont des marqueurs stylistiques critiques et des procédés rhétoriques, [par exemple utilisés] pour démultiplier les processus de signification» [132]. Ce langage produit de l'illisible (*obfuscation*), introduit un déficit communicationnel en lequel précisément reconnaître son geste «poétique». mez «tisse son écriture d'un commentaire permanent et auto-réflexif» [145] qui «illustrate the x.pansion of software potentialities of co:d]iscours[e on an environment entirely reliant on it». mez «compose un langage polyphonique où chacune des unités textuelles est prise dans une tension dialectique: cohérence et perturbation, idiosyncrasie et méta-commentaire, altérité et égotisme réflexif».

Alan Sondheim «s'attache à faire émerger une conscience du code dans le texte, même si le code reste caché» [141-142]. Ses textes sont une «hybridation de code et de données» [143].

Ces différentes pratiques «réinjectent du *bruit* dans l'environnement informationnel». Mimant la performativité d'un code programmé, elles la minent tout autant: le code «perd sa fonction d'instruction au profit de la fonction expressive. Les signes du pseudo-code deviennent un signifiant du code de programmation qui devient lui-même un signifié» [141]. Ces pratiques recèlent toutes une forte *composante auto-référentielle*, ce en quoi consiste aussi leur caractère «poétique». Elles se pensent encore dans le cadre d'une théorie de l'*émergence*: code émergeant des profondeurs en surface du texte, formes émergeant de la combinatoire des «symboles» et des «mots» (Beiguelman).

Conclusion

L'analyse permet ainsi de détailler les éléments d'une poétique généralisée telle que la mise en œuvre, expérimentale, comme la mise en scène, littéraire, du code permettent de la caractériser:

– la déconstruction de l'opposition bruit/information, pour une promotion de l'illisification comme geste esthétique (obfuscation, etc.);

– l'auto-référentialité;

– le double-codage;

– l'émergence du code comme texte.

Pour Camille Paloque-Bergès, les deux postures se distinguent cependant par ce point essentiel que l'une exerce, du code, sa «performativité» – c'est d'ailleurs par sa mise en œuvre, par son exécution, qu'émergent les formes de son aperception esthétique («la question de la valeur qualitative et esthétique de la textualité du code n'est pas séparable de son économie, i.e. d'une perception quantitative et pragmatique de ses fonctionnements» [49]) – tandis que l'autre ne fait que la mimer dans un geste largement rhétorique. Cette distinction vaut comme une dénonciation de l'utopie des *Codeworkers* que Camille Paloque-Bergès reprend de John Cayley lorsqu'il défend que «ce serait tomber dans un discours de l'illusion, qui se trompe lui-même, que de ne travailler qu'à la surface du texte et de représenter les langages de programmation sans activer leur degré de performativité et d'instabilité autrement que dans une performance rhétorique» [159]. Il vaut cependant la peine de ne pas déprécier *a priori* l'efficacité de cette performance rhétorique ni de l'opposer à la performativité du code machiniquement exécutable. Le code «en représentation» n'est pas texte inerte, pour lequel on peinerait à définir quel type de lecture pourrait alors s'y appliquer [159]. La distinction, qui tend à opposer processus et représentation, est sans doute à réévaluer: il convient de reconnaître à la représentation du code son efficacité performative, non plus adressée à la machine (ce que le programme dit, la machine l'exécute), mais transposée à l'instance lectorielle: ce que le texte-code dit, les lecteurs l'activent. Un fragment cité d'un *codewurk* de mez illustre exemplairement ce moment «rhétorique» où

nous nous voyons intimé notre comportement de lecture à l'écran au moment même où nous nous en exécutons: «$N scroll the pages thru // $N n0.s N linkz that breakE the KodE / $N thenne merge the stringz / $N u shell B phree» [147]. Charles Hartman le pressent quand il définit l'instance lectorielle comme «the arena in which the poem acts itself out» [184, n.xxxv]. L'écriture du texte-code, mieux que d'inactiver sa performativité, en réoriente l'adresse illocutoire de la machine à l'instance lectorielle qui devient la *scène* (*arena*) où *œuvre* le code.

Camille Paloque-Bergès conclut son essai en revenant sur la délicate question de l'intentionnalité machinique déjà évoquée à propos de la génération programmée de textes, suivant laquelle l'attention portée au code générateur relèverait d'une intention propre au programme. 3 réponses «littéraires» lui sont rendues:

1/ François Rastier («Écritures démiurgiques», *Visio*, v.2, nº4 (2002)) dénonce ces mythes techno-scientifiques, relents du romantisme tardif, qui instituent une «sorte de démiurge avec prothèse numérique, cyborg appareillé, qui puise dans sa soumission à l'instrument sa puissance créatrice». Il analyse les difficultés des écritures numériques (multiplication, abondance) qui ruinent l'autorité et la présence (laissant ainsi place au jeu et à l'interaction), qui réduisent l'œuvre à une combinatoire (risquant ainsi sa clôture et amputant son texte de son contexte et de sa situation). Cette première «réponse», à bien en juger, n'en est pas une: il s'agit pour Rastier de faire place à une reconception (praxéologique, déliée de toute ontologie) du statut de l'art, où l'art «consiste en ses formes et dans les systèmes de signes», qui cèle l'aberration des conceptions «démiurgiques» des créations programmées qu'il s'agit ici de considérer:

– l'art numérique, par son retrait instrumental (*potentia in machinam*), incline au ludique

– l'art numérique, par son ouverture à l'interaction, n'offre guère qu'un principe de plaisir où le seul sujet subsistant (interacteur) se résorbe (immersion) en se voyant agir.

2/ Mario Costa (*Le sublime technologique*) interprète les environnements électroniques en termes de paysages hyperboliques. La pratique de l'excès textuel (telle qu'elle est déjà problématisée par Rastier) appelle à une «pose conceptuelle» – sur le mode de l'aperception esthétique du sublime, ici technologique. Camille Paloque-Bergès le remarque très justement, cette deuxième réponse contourne le problème posé – celui de l'intention machinique lisible dans le texte-code – en considérant que l'enjeu des écritures numériques est ailleurs et consiste par essence dans l'épreuve de «l'excès textuel» – «le littéraire est ce qui reste quand l'information a disparu dans l'excès d'elle-même» – à la démesure duquel la seule attitude adaptée est «un comportement de boulimie et d'indigestion» [164] de la part de son «consommateur».

3/ Pour Camille Paloque-Bergès, la spéculation des *codeworkers* à propos du code, dont l'efficience machinique est désamorcée par sa mise en scène représentationnelle en surface du texte, ne constitue pas non plus une «réponse» satisfaisante. L'entreprise poétique des *codeworkers* se contenterait d'appuyer la légitimation d'une pratique littéraire du code où seraient confondues «représentation et pratique des environnements du code» [167]. D'où sa proposition conclusive d'«une exploration des écritures du code [...] menée dans les régions de la programmation plutôt que dans celles où le drapeau littéraire est déjà planté» [168]. Malgré l'avantage qu'elle a d'ouvrir à l'étude poétique ce nouveau champ expérimental des pratiques de programmation, cette dernière proposition risque de paraître inadaptée en ce qu'elle écarte un peu lestement du champ de la création programmée les œuvres du code en représentation, dont la performativité est réactivée sur la scène de son aperception esthétique.