Le rôle de l'esthétique dans les compréhensions du code source

Pierre Depaz
sous la direction d'Alexandre Gefen (Paris-3)
et Nick Montfort (MIT)

ED120 - THALIM

Juin 2021

1 Introduction

Les discusssions autour du l'impact de la computation sur l'esthétique, la créativité et la littérature se sont largement développées lors des dernières décennies, avec l'avènement des machines de Turing, puis de leur miniaturization et de leur personalization (i.e. des *personal computers*). En effet, l'implication des algorithmes fait apparaître de nouvelles sortes de textes, souvent regroupés au sein du champ de la littérature numérique. Ce projet de recherche se penche sur un autre aspect des ces "textes", ceux constitués par le code source, condition nécéssaire à l'existence des programmes.

En tant que texte dont le but même est de disparaître (transformé en changements de courant électrique), le code source est un object hybride, autant description qu'action, aux interlocuteurs multiples. Il peut être communication entre humain et humain, entre humain et machine, ou entre ma-

chine et machine. Cependant, en tant que création humaine, il est possible de s'interroger sur les modalités de ses manifestations, notamment en termes d'expressivité. Si l'expressivité fonctionnelle d'un code source implémentant un algorithme est indéniable, l'expressivité artistique d'un texte de code source demeure élusive. Quelle est donc la place de l'esthétique, d'une manifestation formelle sensuellement plaisante, dans l'écriture ou la lecture du code source? Le code source, basé sur un système syntaxique similaire au language naturel, se révèle être un système de communication aussi bien d'humain à humain que d'humain à machine. Il semble néanmoins être principalement destiné les humains, et secondairement aux les machines[1]. Une fois que la condition principale d'existence du code source (sa validité d'exécution) est satisfaite, une condition secondaire semble être non pas sa beauté mais sa compréhensibilité.

Se pose donc la question d'un texte dont les manifestations formelles sont vouées à disparaître, et dont la lecture n'est qu'un processus collatéral de son exécution. Donald Knuth, dans son oeuvre *The Art of Computer Programming*[2], commence par établir que la programmation (l'écriture et la lecture du code source) est bel et bien un art. Cette déclaration, dans les premières pages du premier volume, n'est pourtant pas réitérée ni élaborée dans les autres volumes du monographe. Si écrire du code peut être un art, certains codes sources peuvent donc exhiber des propriétés esthétiques, mais ces dernières sont rarement explicitées par la littérature sur le sujet.

La problématique principale de ce projet de recherche est donc celle du *rôle de l'esthétique dans les compréhensions du code sources*. Il s'agit de mettre à jour et d'interroger les standards esthétiques appliqués au code, d'identifier si ces standards sont similaires à d'autres activités créatives humaines, notamment celle de la littérature, et de mettre à jour la dimension cognitive du phénomène esthétique.

Une approache rapide de l'état de l'art sur ce sujet révèle deux tendances séparées: d'une part, la littérature en informatique et ingénierie prend en compte l'évidence de l'existence d'une esthétique du code source,

du point de vue la productivité de ceux et celles qui écrivent du code, et d'un point de vue cognitif de la compréhension des bases de code[3, 4, 5, 6, 7, 8]. De l'autre, les recherches en sciences humaines traitent de l'interaction entre esthétique et code, tout en restant majoritairement attachées à une conception abstraite et désincarnée du "code", élaborant une esthétique du digital sans pour autant rentrer dans les détails des codes sources eux-mêmes (e.g. il ne semble pas pertinent qu'ils soient écrits en Perl, ou Python, ou Ruby, etc.)[9, 10, 11, 12]. Il existe cependant un certain nombres d'ouvrages de sciences humaines approchant la question matérielle du code de manière directe, que ce soit au niveau culturel[13], politique[14] ou sociologique[15]. C'est au sein de cette approche sur les manifestations du code source que cette recherche s'inscrit.

2 Méthodologie

L'approche principale de ce projet est donc empirique. Il s'agit avant tout d'examiner les codes sources eux-mêmes, ainsi que les discours de ceux et celles qui les écrivent et les lisent.

Cette examination se fait également à travers un cadre théorique partant de la philosophie esthétique et de la littérature afin de définir mon utilisation du terme *esthétique*. D'un point de vue littéraire, je m'appuie sur les travaux de Gérard Genette et sa distinction entre fiction et diction[16], considérant l'esthétique du code comme sa diction, tandis que la poétique serait sa fiction. Entre littérature, philosophie et sciences cognitives se trouvent les oeuvres de Paul Ricoeur[17] et de George Lakoff[18], reliant composition verbale et évocation d'images mentales. Enfin, cette manifestation en surface est reprise par le philosophe de l'art Nelson Goodman dans son analyse des languages de l'art[19], et notamment dans son exploration entre systèmes syntactiques et communication, ainsi que son approche scientifique des phénomènes artistiques.

C'est donc aussi un angle analytique et cognitive du phénomène artistique que je prends, et cela implique la question de la compréhension. Pour celà, je m'appuie sur les travaux de Kintsch et van Dijk et leurs études des stratégies de compréhension du discours[20], l'approche métaphorique considérée comme une stratégie de compréhension. Le code source étant un texte, il produit donc un discours.

Enfin, il est nécéssaire de clarifier deux types de compréhension de l'action du code source: son aspect prescriptif (ce que le code *doit* faire) et son aspect effectif (ce que le code *fait*).

L'approche empirique de ce projet a tout d'abord fait émerger une variété d'individus écrivant et lisant du code, que j'ai regroupé en quatre catégories[21]. Les *développeurs* sont les individus qui écrivent du code dans un contexte économique, à but fonctionel et soutenable à long-terme; ils travaillent sur des bases de code extensives et souvent en collaboration. Les *artistes* sont les individus écrivants du code dans un cadre expressif et artistique, tels que des *code poems*; leur code est destiné principalement à être lu par un public, et seulement de manière secondaire à être exécuté. Les *hackers* sont les individus qui écrivent du code en tant que solution technique unique, idiosyncratique et hautement contextuelle; leur code est destiné à être exécuté par une machine particulière, et de manière secondaire par des humains. Enfin, les *académiques* sont les individus qui écrivent du code afin d'illustrer des concepts de computation, représentant des abstractions plutôt que des usages concrets.

Ces quatres groupes ont des limites poreuses, et un code source peut, souvent, appartenir à plusieurs de ces catégories en même temps (une référence artistique dans un contexte commercial, un hack présenté comme oeuvre d'art, ou une approche particulière au sein d'une base de code commerciale, etc.).

La constitution du corpus de cette recherche s'est faite principalement par la consultation de ressources en ligne. En effet, la plupart des bases de code existe sur des sites d'aggrégations (GitHub, BitBucket, etc.) ou

sous formes d'extraits (*snippets*) présentés et commentés dans des forums, sur des sites personnels, ou des sites de question/réponse (e.g. StackOverflow, Quora). Cette constitution du corpus inclue non seulement ces sources primaires (le code en soi), mais également des sources secondaires (le commentaire par un auteur, ou par un public, justifiant de l'aspect esthétique d'un code présenté). De cette manière, je considère la valeur esthétique d'un code source comme étroitement lié au jugement de groupe émis à son encontre.

Ma métholodologie consiste en une approche interprétative du corpus constitué, dans un double-mouvement. D'une part, les références et discours des individus écrivants du code sont pris en compte afin de constituter un ensemble de standards esthétiques: ces standards sont des manifestations concrètes du code permettant une facilitation de la compréhension du *propos* du code (propos prescriptif, ou effectif). D'autre part, les standards exhibés sont appliqués à d'autres bases de code afin de vérfier leur applicabilité. Ã travers la multiplicité de ces standards, il s'agit donc d'identifier les stratégies métaphoriques de compréhensions des lecteurs de code.

3 Avancement

Jusqu'à présent, les corpus analysés ont été ceux des développeurs, des artistes et en partie celui des hackers. Cette dernière catégorie étant particulièrement vaste et polymorphe, il faut encore la définir d'avantage, et peut-être d'identifier des sous-groupes qui pourraient particulièrement être mis en avant (e.g. les participations au IOCCC, les *one-liners* et les *demoscenes*). Le corpus académique, principalement tiré des manuels d'informatique n'a été qu'esquissé.

Les analyses effectuées jusqu'à présent montrent plusieurs aspects. Premièrement, il n'y a pas "un" code source, abstrait et désincarné, mais

bien une multiplicité de codes sources, existant au sein de pratiques et de groups différents. Ces codes sources possèdent tous une possibilité de manifestation esthétique, manifestations soumise au désir de communiquer *quelque chose*. Ce quelque chose, et les modalités esthétiques qui l'accompagne, varient avec les groupes.

Pour les développeurs, il s'agit de communiquer l'intention du programme à travers son implémentation à travers une métaphore non seulement linguistiques (tentant de faciliter le processus de lecture par des choix de structures, de syntaxe et de vocabulaire) mais également architecturale (notamment à travers les travaux de Christopher Alexander sur les *patterns*, et sa récupération par la communauté de programmation orientée-objet).

Pour les artistes, il s'agit de développer une certaine vision du et un certain commentaire sur le monde à travers divers niveaux de métaphores linguistiques (rhétorique procédurelle[22], double-codage[14], double-signification[15]). Le code est donc ici une double sorte de langue: langue machinique et langue humaine dont les tensions syntactiques peuvent créer une richesse sémantique.

Pour les hackers, il s'agit d'exprimer une certaine connaissance de la machine, du language de programmation, ou d'un algorithme—il ne s'agit pas de communiquer cette connaissance de la machine à autrui, mais de communiquer le fait que l'on a cette connaissance. La stratégie métaphorique ici est celle du matériel, avec des connections entre *hackers* et *tinkerers*, et une expérimentation concrète par le faire. C'est en "brisant" le code qu'il permet d'atteindre des niveaux de virtuosité inégalés, ou c'est en "rasant" un ou deux caractères d'un programme qu'on démontre cette virtuosité.

L'hypothèse qui reste à explorer pour la communauté des académiques est celle du code en tant que mathématiques, et donc d'un rapprochement entre l'élégance mathématique et l'élégance d'un programme.

Enfin, j'ai aussi examiné la question de la beauté d'un code du point de vue de la machine: un code peut-il être beau pour un ordinateur? Au vu de la représentation de la sémantique dans les languages de programmation

(esssentiellement un arbre syntactique différé, ”décorant” le premier arbre syntactique lors de l'interprétation d'un programme), on peut peut-être dire qu'un code présente des manifestations esthétiques lorsqu'il minimise la taille de l'arbre généré et optimise la localisation des données stockées en mémoire vive. Additionellement, les languages de programmation peuvent eux-mêmes supporter des manifestations esthétiques sous certains contextes. Une comparaison des langaues C++, Go, Ruby et Java a été entreprise, basée sur les standards évoqués plus haut.

4 Prochaines étapes

Ce projet a donc jusqu'à présent montré que les différents standards esthétiques des différentes communautés repose notamment sur des représentations métaphoriques du code, représentations du code comme language, comme architecture, et comme matériau. Le dernier groupe de notre corpus semble traiter le code comme une sorte de mathématique et il faut donc vérifier si les standards esthétiques présents en mathématique[23] sont similaires. Cette étude a été effectuée sur un corpus de code court, et doit être vérifiée sur un corpus plus extensifs (i.e. UNIX v6 et LaTeX).

Par ailleurs, il s'agit d'approfondir les différentes manières d'écrire et de lire du code en tant que hacker, afin de vérfier si une telle catégorie fait sens, ou si sa complexité est justement l'illustration du fait que les groupes identifiés se superposent et qu'un même individu, ou un même code-source peut être examiné à travers le prisme de différentes métaphores, et en résulter des jugements et expériences esthétiques différentes.

Enfin, en guise de conclusion, je compte connecter l'esthétique de ces codes sources avec leur poétique—les structures, concepts et idées qu'ils peuvent permettre d'exprimer, notamment en termes de fiction, afin de reconnecter avec leur action (exécution) dans le monde, et avec l'extension de la littérature numérique.

En termes de calendrier, je compte donc conclure ces recherches lors du semestre d'été et d'automne 2021, avant de finir la rédaction de la thèse durant les semestres de printemps et d'été 2022, afin de soutenir en automne 2022.

References

[1] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs - 2nd Edition*. Justin Kelly, 1979.

[2] Donald E. Knuth. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., USA, 1997.

[3] Andy Oram and Greg Wilson, editors. *Beautiful Code: Leading Programmers Explain How They Think*. O'Reilly Media, Beijing ; Sebastapol, Calif, 1st edition edition, July 2007.

[4] A. Cox and M. Fisher. Programming Style: Influences, Factors, and Elements. In *2009 Second International Conferences on Advances in Computer-Human Interactions*, pages 82–89, February 2009.

[5] Richard P. Gabriel and Ron Goldman. *Mob Software: The Erotic Life of Code*. 2001.

[6] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education, August 2008.

[7] Francoise Detienne. *Software Design – Cognitive Aspect*. Springer Science & Business Media, December 2012.

[8] Gerald M. Weinberg. *The Psychology of Computer Programming*. Dorset House Pub., 1998.

[9] Florian Cramer. *Exe.cut(up)able statements: Poetische Kalküle und Phantasmen des selbstausführenden Texts*. Wilhelm Fink, December 2019.

[10] N. Katherine Hayles. *My Mother Was a Computer: Digital Subjects and Literary Texts*. University of Chicago Press, March 2010.

[11] Adrian Mackenzie. *Cutting Code: Software and Sociality*. Peter Lang, 2006.

[12] Pierre Lévy. *De la programmation considérée comme un des beaux-arts*. Textes à l'appui. Anthropologie des sciences et des techniques. Éd. la Découverte, Paris, 1992.

[13] Nick Montfort, Patsy Baudoin, John Bell, Ian Bogost, and Jeremy Douglass. *10 PRINT CHR$(205.5+RND(1)); : GOTO 10*. The MIT Press, illustrated edition edition, August 2014.

[14] Geoff Cox and Christopher Alex McLean. *Speaking Code: Coding as Aesthetic and Political Expression*. MIT Press, 2013.

[15] Camille Paloque-Bergès. *Poétique des codes sur le réseau informatique*. Archives contemporaines, 2009.

[16] Gérard Genette. *Fiction & Diction*. Cornell University Press, 1993. Google-Books-ID: OdgPWk1Sq6YC.

[17] Paul Ricoeur. *The Rule of Metaphor: The Creation of Meaning in Language*. Psychology Press, 2003.

[18] George Lakoff. *Metaphors We Live By*. University of Chicago Press, 1980.

[19] Nelson Goodman. *Languages of Art*. Hackett Publishing Company, Inc., Indianapolis, Ind., 2nd edition edition, June 1976.

[20] Walter Kintsch and Teun A. van Dijk. Toward a model of text comprehension and production. *Psychological Review*, 85(5):363–394, 1978.

[21] Brian Hayes. *Cultures of Code*. February 2017.

[22] I. Bogost. *The Rhetoric of Video Games*. 2007.

[23] Gian-Carlo Rota. The Phenomenology of Mathematical Beauty. *Synthese*, 111(2):171–182, 1997. Publisher: Springer.