De l'informatique pédagogique et de sa place dans un musée de l'informatique et de la société numérique

Bastien Guerry

1. Introduction

Un musée de l'informatique et de la société numérique ne peut pas être un entrepôt d'ordinateurs éteints, ni se contenter d'exposer des visiteurs passifs à des données sociologiques. Les ordinateurs doivent être utilisés et les visiteurs doivent pouvoir interroger les éléments de sociologie du numérique, les mettre en perspective. Mais que faire des ordinateurs allumés ? Comment présenter des faits historiques et des données de manière vivante ? Quelles activités, quelles interactivités mettre en place autour des objets d'un tel musée ? Ces questions reviennent à celle, plus fondamentale, de la *pédagogie* à mettre en oeuvre.

Voici les trois thèses que nous développerons dans cet article : (1) il n'existe pas de « culture générale » de l'informatique mais de multiples cultures, qui ne communiquent les unes avec les autres que par certains aspects superficiels ; (2) le lien entre pédagogie et informatique est un lien fondamental, irréductible à une histoire de l'informatique scolaire ; (3) c'est en comprenant ce lien qu'il devient possible de lier les différentes cultures informatiques entre elles et de les rendre vivantes.

Notre objectif est de comprendre pourquoi les rapports si particuliers qu'entretiennent informatique et pédagogie doivent avoir une place à part dans ce projet de musée et d'esquisser quelques pistes pour définir cette place.

2. Une culture générale de l'informatique ?

Nous voyons d'ici la scène : un grand-père montre à sa petite fille une carte perforée, et tente de lui expliquer à quoi elle sert. S'il n'arrive pas à lui faire comprendre le contexte technique dans lequel fonctionnait cette carte, il lui racontera une anecdote sur la manière dont les cartes ont changé son métier, la redistribution des plaisirs et des peines dans le maniement des machines, d'abord avec ces cartes, puis avec les technologies qui les ont remplacées. Il concluera en se disant, peut-être à voix haute, que tout cela est très difficile à imaginer. Les deux

sentiront le fossé culturel qui sépare leurs représentations de l'informatique, et il y a peu de chances que l'enfant associe ces cartes à des objets informatiques qui lui sont familiers ou à des concepts informatiques toujours en cours. De manière générale, les objets d'un musée de l'informatique risquent de rester déconnectés les uns des autres et de ne « parler » qu'à un petit nombre de personnes. Que faire avec les objets de ce musée ?

Une idée serait de solliciter les associations de passionnés et de leur demander d'animer des ateliers autour des machines dont ils sont la « mémoire vive » : ils pourraient faire des démonstrations, initier des visiteurs, les inviter à mettre la main à la pâte. Mais peut-être faudra-t-il solliciter autant de clubs qu'il existe de machines, et ces communautés pourraient ne pas faire bon ménage entre elles[1]. Nous nous heurtons là à une première difficulté : l'histoire de l'informatique est en partie celle des nombreuses communautés qui se sont identifiées à telle ou telle machine, tel ou tel langage, et qui ne se reconnaissent pas forcément dans les pratiques des autres communautés. C'est une histoire morcelée.

Supposons qu'un muséologue parvienne à relever ce défi. Un deuxième problème nous attend : combien faut-il de machines en état de marche ? Autant qu'il y aura de visiteurs ? Devra-t-on réinventer un système d'inscription où chacun pourra réserver une heure d'interaction, comme au heures des premiers machines PDP-1 ? Surgit là une deuxième difficulté : nos représentations de l'informatique, même lorsqu'elles sont partagées par une communauté, sont construites dans l'interaction intime entre un utilisateur et sa machine. Faire revivre ces rapports subjectifs passés dans l'espace et la durée limités d'un musée est peut-être impossible.

Admettons encore cette difficulté levée : différents clubs animent des ateliers pour chaque technologie, et chaque visiteur y fait une expérience l'exposant à la micro-culture qui s'est formée autour d'elle. Reste un dernier obstacle à éviter pour conserver au lieu son équilibre : comment rendre un atelier d'initiation au Lisp aussi attractif qu'un atelier autour de la X-Box ? Comment intéresser les visiteurs à des machines qui ressemblent à de gros frigidaires clignotants quand ils passeront à côté de gadgets aux atours plus attrayants ? Comment faire que la pédagogie autour des ordinateurs ne se dissolve pas dans les seules interactions ludiques ?

La raison commune de ces difficultés est celle-ci : il n'existe pas *une* culture de l'informatique, mais un agrégat de cultures disparates, plus ou moins proches les unes des autres.

Nous voyons au moins trois obstacles à l'émergence d'une culture *générale* de l'informatique : la séparation des utilisateurs dans l'espace (ou le rapport subjectif qui s'établit entre un utilisateur et une machine), l'apparente séparation dans le temps entre les technologies (ou l'idée que chaque progrès rend obsolètes les technologies passées), et la difficulté d'engager les utilisateurs au-delà du plaisir immédiat que telle ou telle interface suscite chez chacun. À ces trois obstacles correspondent trois

[1] Sans même parler des querelles qui opposent aujourd'hui les tenants d'un langage comme Python contre les adorateurs de Ruby, on se souvient des querelles qui opposaient jadis les adeptes de l'Amiga contre ceux de l'Atari.

images « romantiques » de l'informatique : celle de l'autodidacte (l'ordinateur devenant l'emblème de celui qui apprend seul, souvent en marge des institutions) ; celle d'un champ technique en perpétuelle révolution (existe-il un domaine dans lequel ce mot soit plus galvaudé ?) et celle de la génération spontanée d'usages intelligents au contact de belles machines. La figure du « hacker »[2] synthétise ces perspectives imaginaires : c'est un jeune adolescent qui apprend seul et comme en jouant à maîtriser des machines aux capacités révolutionnaires.

Au lieu d'une culture générale de l'informatique, avons en commun ces images et quelques représentations superficielles de nos interactions avec les interfaces les plus répandues – nous savons à quoi ressemble un « bureau », un « fichier », etc. mais nous ne savons pas bien ce qu'est « l'information », un algorithme ou une base de données. Ce que nous percevons en commun de l'informatique, ce sont à peine quelques usages standardisés tout à fait limités, et la possibilité que certains, plus doués que d'autres, aillent au-delà.

Nous faisons l'hypothèse (à vérifier) que notre imaginaire informatique fonctionne comme un *obstacle épistémologique*[3] à l'acquisition d'une culture *générale* informatique, laquelle irait à l'encontre de ces trois images : elle partirait de l'observation que les gens *apprennent* à se servir des ordinateurs les uns avec les autres (leurs enseignants, leurs parents et leurs amis) ; que c'est bien parce que l'informatique s'est développée autour de certains concepts clefs que nous pouvons réutiliser ce que nous savons d'un ordinateur à un autre, d'un langage à un autre ; et que c'est en dépassant le plaisir immédiat des rapports superficiels à la machine que les usages se creusent et forment une culture.

3. Du rapport entre informatique et pédagogie

3.1. Les rapports de surface : quelques traces historiques

3.1.1. Alan Turing et les machines qui apprennent

Revenons à l'article fondateur d'Alan Turing de 1950 : *Computing Machinery and Intelligence*. Après avoir proposé un cadre expérimental dans lequel il y aurait du sens à se demander si une machine peut être qualifiée « d'intelligente »[4], Turing achève son article sur des considérations concernant la capacité de la machine à *apprendre* (cf. la section intitulée *Machine learning*.) Turing imagine un apprentissage de la machine par essais et erreurs avec un système de « récompenses

[2] Pour une histoire des premiers hacks, informatique et physiques, voir *Nightwork, A History of Hacks and Pranks at MIT*, T. F. Peterson, The MIT Press, 2003.

[3] Voir *La formation de l'esprit scientifique*, Gaston Bachelard, Paris, Librairie philosophique Vrin, 1999 (1ère édition : 1938).

[4] Ce cadre expérimental est appelé par Turing «jeu de l'imitation» et sera par la suite connu sous le nom de « Test de Turing ». Dans une conversation avec Marvin Minsky, alors que je lui expliquais avoir consacrée une année de travail à l'article de Turing, celui-ci me dit "… but, this paper was a joke !" Le problème de savoir si Minsky plaisantait est resté pour moi indécidable.

» et de « punitions ». La machine pourrait ainsi ajuster ses heuristiques et devenir de plus en plus apte à imiter les comportements que nous qualifions spontanément d'intelligents[5].

C'est la première fois que la notion centrale d'intelligence est aussi clairement liée aux machines à calculer : l'idée apparaît ici d'un mode d'intelligence commun aux hommes et aux machines, permettant aux unes d'apprendre et aux autres de se servir des machines pour autre chose que de simple calculs.

3.1.2. J. C. R. Licklider et l'ordinateur comme interface pédagogique

La deuxième « trace » de rapport entre informatique et pédagogie remonte à J. C. R. Licklider qui, dès le milieu des années 1950, s'amuse à écrire des programmes pédagogiques pour ses enfants. Nous n'avons pas conservé ces programmes mais d'après la description qu'en fait rapidement M. Mitchell Waldropp dans *The Dream Machine*[6], il pourrait s'agir de petits jeux de questions-réponses où l'ordinateur indique à l'élève s'il tombe juste ou s'il se trompe. Licklider va jusqu'à imaginer que la tonalité des réponses de l'ordinateur pourrait varier aléatoirement, afin de renforcer le plaisir et l'engagement de l'élève. Quelle que fut la nature de ces logiciels pédagogiques, c'est ici qu'apparaît l'usage de machines pour *apprendre* plutôt que pour calculer, pour *interagir* plutôt que pour donner des instructions. La finalité pédagogique donne aux problématiques d'interaction une importance nouvelle : ce n'est pas malgré eux et pour la machine que les utilisateurs interagiront avec elle, mais *pour eux* et *avec la machine*.

3.1.3. La pédagogie comme problématique ambiante aux sources de l'informatique

Intelligence et interaction : voilà les deux concepts phares, la polarité au sein de laquelle se développera l'informatique comme science et comme technique pendant les deux décennies d'après guerre.

D'un point de vue sociologique, c'est aussi la période où se dégagent au moins trois classes distinctes d'utilisateurs. D'abord la communauté des chercheurs, ceux qui explorent le champ des possibles informatiques, qui inventent les nouveaux

[5] Si le mode d'apprentissage décrit par Turing est influencé par les théories béhavioristes de l'époque, Turing ne dit rien sur la pertinence de ce mode d'apprentissage pour les êtres humains, et les conclusions de son article sont indépendantes de toute hypothèse à ce sujet.
[6] *The Dream Machine*, Mitchell Waldrop, 2001.

concepts, langages et interfaces. Ensuite celle des *hackers*, ces étudiants à qui l'on donne un accès libre aux machines et qui auront le droit de « jouer » avec. Enfin celle, plus large, des opérateurs techniques, responsables de la maintenance de ces calculatrices géantes[7]. En grossissant le trait, nous dirons que les opérateurs sont les premiers à devoir apprendre à se servir de l'informatique comme d'un *outil*, que les hackers, eux, apprennent à en *jouer* (comme on apprend à jouer d'un instrument), et que les chercheurs se consacrent à tout ce qui permet de faire interagir l'intelligence des machines et celle des hommes.

Il est difficile de se replonger dans l'effervescence intellectuelle de l'époque, mais nous imaginons une période où, dans un champ que les chercheurs à la fois découvrent et définissent, tout paraît à apprendre. Les mêmes personnes s'entraînaient à manipuler des ordinateurs, à leur enseigner de nouvelles techniques de calcul, à en apprendre de nouvelles façons de poser les problèmes. En lisant *The Dream Machine*, on est frappé de voir comment l'esprit de ces pionners est imprégné de problèmes pédagogiques : repensant leur relation à l'informatique chaque fois qu'il apprivoisent un nouvel outil, ils perçoivent peu à peu que ce nouveau champ scientifique et technique aura des conséquences sur nos façons d'enseigner et d'apprendre, sur notre rapport aux connaissances et à l'intelligence.

3.1.4. L'informatique pédagogique en amont de l'informatique générale

Une histoire de l'informatique scolaire (c'est-à-dire de l'informatique utilisée dans les écoles) nous dirait quels outils grand public ont passé les barrières de l'institution scolaire, et ce que celle-ci en a fait. Une histoire plus générale du rapport entre l'informatique et la pédagogie nous donnera au contraire des exemples d'innovations dont la visée fut d'emblée « pédagogique », et qui eurent ensuite des répercussions sur l'informatique en général. Voici trois brèves illustrations de telles innovations.

Le premier exemple est l'invention du LOGO par Seymour Papert à la fin des années 60. La finalité du LOGO est d'abord pédagogique : le langage s'inspire du Lisp mais se veut plus facile à lire, à écrire, et surtout à *voir*, grâce au retour visuel sur les instructions que l'utilisateur donne à la fameuse Tortue. S'il est difficile d'évaluer l'impact qu'aura eu le LOGO dans les écoles où il est entré, il est en revanche indéniable que l'idée centrale d'ajouter un retour visuel aux instructions aura un impact sur la manière d'envisager la programmation, et sur l'informatique en général[8].

Le deuxième exemple est la popularisation de la programmation orientée objet (POO), par Alan Kay. Ici encore, le Lisp est source d'inspiration, ainsi que le

[7] À ces trois catégories s'ajoutera plus tard celle du « grand public ». Il serait intéressant de voir comment les trois attitudes décrites précédemment se retrouvent dans les différentes attitudes du grand pulic à l'égard des ordinateurs.

[8] Il n'est qu'à voir aujourd'hui la résurgence de l'intérêt pour la programmation visuelle. Voir par exemple le site Web de Bret Victor sur le Learnable programming : celui-ci s'inspire largement de l'idée centrale du LOGO, réunissant dans un même espace d'interaction un programme et ses retours visuels.

Sketchpad[9], une interface permettant de manipuler des objets graphiques. Le sketchpad introduit pour ces objets la notion de « patrons » (ou *masters*), ancêtres des objets en POO. En retraçant les origines de SmallTalk[10], le premier environnement à implémenter et diffuser largement les concepts de la POO, Alan Kay indique qu'il cherchait à ajouter une couche d'abstraction plus intuitive, plus pédagogique, qui permette aux utilisateurs d'apprendre à se servir d'un ordinateur plus facilement, le tout dans une période où il pressent l'avènement de l'ordinateur personnel. Aujourd'hui, ce qui fut inventé dans une perspective d'apprentissage est devenu l'une des méthodes de programmation les plus couramment employée.

Le troisième exemple est la popularisation des ordinateurs ultra-portables par le projet *One Laptop Per Child* (OLPC), lancé en 2005. L'ancêtre de ces ordinateurs ultra-portables destinés aux enfants date de 1968, quand Alan Kay présente le *Dynabook*[11]. Mais ce qui déclencha réellement l'émergence du marché des ultra-portables fut le projet OLPC, un projet humanitaire et éducatif visant à fournir des machines à bas prix pour les pays en voie de développement. Avant que des pays en voie de développement ne s'intéressent à ces ordinateurs, aucun industriel n'aurait parié que leur format deviendrait populaire. Depuis 2005, celui-ci est quasiment devenu plus courant que celui de nos anciens ordinateurs portables. Ici encore, c'est un projet à vocation pédagogique qui vient bouleverser le paysage de l'informatique.

Ces trois innovations ont en commun d'être issue d'une même famille de chercheurs (les héritiers des pionniers de l'après-guerre, continuant d'explorer le rapport entre intelligence et interfaces) et d'une même motivation pédagogique initiale.

3.2. Le rapport profond : quelques concepts clefs

J'ai présenté quelques « traces » historiques du rapport entre informatique et pédagogique et j'ai donné des exemples où l'informatique pédagogique innove en amont de l'informatique générale. J'en viens maintenant à l'exposé du rapport conceptuel entre informatique et pédagogie, au lien plus profond qui unit ces deux champs de problèmes dès les début de l'informatique.

3.2.1. Seymour Papert et de l'ordinateur comme catalyseur de micro-cultures

Le penseur le plus influent du rapport entre informatique et pédagogie est Seymour Papert. Nous mettrons ici en regard deux de ses articles : dans le premier, *Computer Criticism vs. Technocentric Thinking* (1987) [12], Papert décrit et critique l'approche *technocentriste* des questions qui entourent le LOGO comme objet

---

[9] Voir l'article « Sketchpad » sur la Wikipédia anglaise : http://en.wikipedia.org/wiki/Sketchpad

[10] Voir *The Early History of Smalltalk*, Alan Kay, 1993. http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html

[11] Voir cette vidéo où Alan Kay présente le Dynabook:

[12] *Computer Criticism vs. Technocentric Thinking*, S. Papert, 1987, Educational Researcher (vol. 16, no. I) (Lien)

informatique utilisé dans les écoles ; dans le second, *Why School Reform Is Impossible* (1995)[13], il souligne que notre manière d'aborder la question de l'informatique pédagogique est implicitement tributaire d'un point de vue centré sur l'école *telle qu'elle existe* – point de vue que nous appelons « école-centré ». Dans les deux cas, son objectif est de montrer qu'un ordinateur n'est pas un outil pédagogique « neutre » que l'école pourrait assimiler sans que changent avec lui les pratiques des enseignants et des élèves : l'outil informatique modifie la *culture* du groupe qui l'utilise. Ce n'est donc pas l'efficacité de l'ordinateur en tant qu'auxiliaire pédagogique qu'il faut évaluer, mais la richesse pédagogique des micro-cultures qu'il fait émerger.

Le techno-centrisme est le présupposé qui transparaît immédiatement dans la question « Est-ce que le LOGO *marche* ? ». Cette question n'a de sens que si l'on imagine que le rôle de la machine est d'être un « agent actif » dans le processus d'instruction, et si l'on imagine l'apprentissage comme quelque chose qui « arrive » aux élèves, plutôt qu'un acte, une dynamique. Pour montrer comment cette manière de voir les choses ne rend pas justice à « l'expérience LOGO », Papert donne un exemple où le LOGO est utilisé *parmi d'autres techniques* pour construire des horloges. Au cours de leurs essais, les élèves utilisent le LOGO soit pour formuler une partie des problèmes qu'ils se posent, soit pour simuler entièrement une horloge. Dans les deux cas, le LOGO est vraiment un *langage*, dans toute la dimension culturelle qu'a d'ordinaire ce mot : c'est un outil pour penser, pour communiquer, pour construire et partager des problèmes – bref, un outil pour apprendre.

L'école-centrisme est un ensemble de présupposés sur l'école et les rôles de l'enseignant et de l'élève : à quoi doit ressembler une classe, un élève, un maître, leurs outils, leurs relations. Papert est pessimiste sur la capacité de l'informatique à jouer un rôle *direct* dans une réforme de l'école : celle-ci plie les nouveaux outils à ses contraintes, conservant ceux qui s'adaptent aux pratiques établies, rejetant les autres. Tant que l'informatique sera perçue comme un élément devant être *introduit* dans les écoles, les tentatives d'intégration sont vouées à l'échec car l'équilibre de l'école rendra impossible l'émergence d'une micro-culture pédagogique telle que celle décrite ci-dessus. Mais Papert est optimiste sur la capacité de la société à évoluer avec l'informatique, et de l'école à évoluer avec la société, accordant une place de moins en moins contrainte à l'ordinateur. L'école-centrisme ne sera alors plus un obstacle à l'usage de l'informatique en classe, car cet usage ce sera d'abord répandu hors de l'école : la part de l'informatique dans la culture ordinaire aura rendu l'outil plus apte à être utilisé de façon pédagogique au sein de l'école.

S'il est difficile d'imaginer l'informatique pédagogique comme outil favorisant l'émergence de micro-cultures, c'est que celles-ci doivent être contruites et conçues en résistant à la fois au techno-centrisme et à l'école-centrisme[14] :

---

[13] *Why School Reform Is Impossible*, S. Papert, 1995, The Journal of the Learning Sciences, 6(4), pp. 417-427. (Lien)

[14] *Computer Criticism vs. Technocentric Thinking*, S. Papert, 1987, Educational Researcher (vol. 16, no. I) (Lien)

> *[…] Si nous nous intéressons à l'élimination du technocentrisme dans la manière de penser le rôle des ordinateurs dans l'éducation, nous nous retrouverons peut-être à réexaminer nos présupposés sur l'éducation, faits bien longtemps avant l'avènement des ordinateurs. (On pourrait même soutenir l'idée que la principale contribution faite à l'éducation jusqu'à présent par les ordinateurs a été de nous forcer à repenser complètement des questions qui n'ont en elles-mêmes rien à voir avec les ordinateurs.)*

3.2.2. Alan Kay et l'analogie avec l'instrument de musique

Pour décrire le rôle pédagogique de l'ordinateur, Alan Kay recourt souvent à l'analogie avec un instrument de musique : un instrument ne contient pas la musique, pas plus qu'il n'instruit l'élève directement. Mais il est un « amplificateur » d'idées musicales : il donne accès à ces idées, permet de les tester, de les « jouer » – et d'en inventer d'autres. De la même façon, un ordinateur ne contient rien qui soit pédagogique en soi, mais il est un amplificateur d'idées fortes (voir le concept, récurrent chez Kay, de *powerful ideas*) : il propose un environnement dans lequel l'élève peut lire, écrire, et mettre oeuvre de nouvelles idées[15].

Il ne viendrait à l'esprit de personne de se demander si un élève doit apprendre la théorie de la musique *ou* le maniement d'un instrument : les deux paraissent essentiels au fait de pouvoir accéder à la musique en général. L'analogie faite par Kay permet aussi de dépasser une vision de l'informatique à l'école qui serait tournée soit vers l'apprentissage d'une nouvelle discipline soit vers l'apprentissage du maniement des ordinateurs. Dans les deux cas, c'est manquer le fait que l'informatique pédagogique est un moyen de créer un environnement dans lequel des idées fortes (qu'elles soient liées aux mathématiques, à la littérature, voire à l'informatique elle-même) puisse être « jouées » et apprises par des élèves. Kay résume cela : l'interface informatique doit être une interface d'apprentissage.

Cette analogie avec la musique permet en outre à Kay de noter que le problème des efforts à faire pour apprendre avec l'informatique est un faux problème : lorsqu'un élément de culture semble assez important à ses aînés, un enfant fait l'effort de l'apprendre. Ainsi de la musique et du sport. Ainsi, peut-être, de l'informatique, le jour où le fait de comprendre ce qu'elle est et ce qu'elle peut sera devenu culturellement discriminant[16].

3.2.3. D. Engelbart et J. C. R. Licklider : Framework et « symbiose »

Micro-cultures et idées fortes : deux concepts indépendants à la fois de l'outil informatique et des contenus mêmes de l'apprentissage, mais qui sont au coeur de la relation informatique/pédagogie. Ce deuxième concept d'idées fortes apparaît pour la première fois en lien avec l'informatique dans les travaux de Douglas Engelbart[17]:

---

[15] L'environnement SmallTalk s'appuie sur la complémentarité des approches kinesthétiques, visuelles et symboliques : Alan Kay explique ces approches et leur combinaison dans cette vidéo.

[16] Voir *Powerful Ideas Need Love Too!*, Alan Kay, 1995. (Lien)

> *Par « augmentation de l'intellect humain », nous entendons la capacité croissante pour un homme d'approcher une situation problématique complexe, de gagner en compréhension de ce dont il a besoin, et de trouver des solutions à ses problèmes. […] Nous ne parlons pas d'astuces isolées qui aident dans une situation particulière. Nous faisons référence à un mode de vie intégré dans lequel les intuitions, les essais, tout ce qui relève de l'intangible et du « sentiment d'une situation » coexistent de manière efficace avec des concepts forts, une technologie et une notation fluides, des méthodes sophistiquées, et des aides électroniques très puissantes.*

Pour décrire l'environnement dans lequel un utilisateur peut expérimenter ces « idées fortes », Engelbart utilise la notion de *Framework*, voisine de celle de *symbiose* déjà mise en avant depuis longtemps par Licklider. Ces deux notions suggèrent que le rapport d'un utilisateur à un ordinateur va au-delà du rapport d'une personne à un *outil* : celui-ci est extérieur, contingent, dirigé vers une finalité qui n'est ni dans l'outil ni dans la personne qui s'en sert ; celui-là est un rapport réflexif, où l'homme et la machine sont forment un tout *intégré*, et où la finalité n'est pas nécessairement tournée vers l'extérieur.

La marteau est une extension de l'homme plantant un clou, mais la relation avec l'ordinateur instaure un *cadre*, un environnement où l'homme et la machine dialoguent pour accomplir des tâches dont certaines sont définies hors de l'environnement, d'autres à l'intérieur. Vu ainsi, l'environnement informatique crée un « double » de l'esprit, sachant exécuter certaines tâches mieux que moi, d'autres moins bien ; mais tant qu'une interface facilite nos interactions, ces différences forment un espace *pédagogique*. C'est toute la vision contenue dans l'idée de *symbiose* développée par Licklider, puis plus tard dans celle de *Framework* mise en oeuvre par Douglas Engelbart[18].

3.2.4. L'ubiquité de l'informatique comme science et comme technique : un espace pédagogique ?

Licklider formule son idée de *symbiose* à une époque où l'informatique n'est ni tout à fait une science, ni seulement une technique, et il la précise à l'aide de la notion de « modèles dynamiques ». Les sciences établissent des modèles : la valeur de ces modèles est dans leur force prédictive et l'universalité de leur description. Mais un programme informatique est plus que cela, c'est un *modèle en action*. Interagir avec un ordinateur, c'est créer une relation dynamique entre un modèle en action et les représentations d'un cerveau. Comment mieux cerner ce qui fait l'essence de l'acte d'apprendre ? Dans *Role Models, Mentors, and Imprimers and Thinking*[19], Marvin Minsky va jusqu'à proposer d'imaginer qu'un élève fonctionne

---

[17] *Augmenting Human Intellect: A Conceptual Framework*, Douglas Engelbart, October 1962. (Lien)

[18] La démonstration du premier *framework* proposé par Engelbart est connue comme la « mère de toutes les démonstrations ». Voir la série de vidéos ici et l'article de la Wikipédia anglophone.

[19] OLPC Memo 3, *Role Models, Mentors, and Imprimers and Thinking*, Marvin Minksy, 2008. http://web.media.mit.edu/~minsky/OLPC-3.html

comme un ordinateur, et qu'apprendre avec un ordinateur, c'est progressivement résoudre les problèmes qui se posent au couple ordinateur-élève, dans sa dynamique. Et l'on note au passage le renversement de la perspective de Turing : ce n'est plus l'élève qui sert de modèle pour penser l'apprentissage des ordinateurs, mais l'ordinateur qui sert de modèle (au moins méthodologique) pour penser l'apprentissage par l'élève.

Au coeur de la relation entre informatique et pédagogie, nous avons donc ces trois concepts : idées fortes, micro-cultures, environnement. Ce que nous appelons la relation « profonde » entre l'informatique et la pédagogie se résume ainsi : l'interaction homme-machine crée un *environnement dynamique* dans lequel l'homme peut découvrir et manipuler des *idées fortes* (à un premier niveau subjectif de réflexivité), et le partage de cet environnement crée une *micro-culture* (à un deuxième niveau collectif de réflexivité) dans laquelle les uns et les autres *apprennent*.

4. Un musée pédagogique de l'informatique ?

Est-ce que l'exposé de ce rapport « profond » entre informatique et pédagogie peut nous indiquer quellle pédagogie mettre en oeuvre au sein d'un musée de l'informatique et de la société numérique ? Quelles activités imaginer qui, tout en donnant accès à des cultures numériques spécifiques, permettent d'en abstraire les traits communs et de les faire communiquer entre elles ?

4.1. Babel : inventer son langage de programmation

Nous prendrons ici l'exemple d'une activité imaginaire que nous baptisons *Babel*, et dont le principe est de permettre à des visiteurs d'inventer un langage informatique. Un premier but serait d'être exposé par degrés à quelques unes des contraintes qui donnent aux langages l'allure qu'on leur connaît ; un second but, plus élaboré, pourrait être de transformer le visiteur (sur place ou en ligne) en « apprenti informaticien-linguiste », capable de trouver des rapports entre le langage qu'il invente et ceux qui existent.

Imaginons pour cela une interface simple : dans la moitié droite de l'écran, l'ordinateur attend que l'utilisateur dessine un objet ; dans la moitié gauche, l'ordinateur attend que l'utilisateur associe un nom à cet objet. Par exemple : l'utilisateur dessine un rond et l'appelle *rond*. Ce rond est celui affiché à l'écran, ce n'est pas un rond en général. En plus de dessiner des objets, l'utilisateur pourrait ensuite *agir* sur eux, et enregistrer le nom des actions. Par exemple : l'action de choisir le rond dans la pile d'objets disponibles et de le faire apparaître au centre de l'écran serait nommé *afficher rond*. Au sein de ces deux registres (d'objets et d'actions), l'utilisateur construirait les éléments primitifs d'un langage par accumulation de noms et de verbes, lesquels seraient tous très spécifiques au

départ[20]. Puis il serait invité à écrire un programme mettant en oeuvre les éléments de l'univers qu'il aura défini.

En coulisse, la machine contiendrait un moteur de suggestions pour guider l'utilisateur dans une démarche d'abstraction. Dans *J'affiche le rond* est-ce « J' » et « le » sont indispensables ? Est-ce que *afficher rond* et *rond afficher* sont équivalents ? Outre la définition progressive de contraintes grammaticales, le logiciel aiderait à abstraire des classes d'objets, d'actions, et de conditions-actions. Par exemple : la classe *rectangle*, la classe *déplacer*, et la classe *si un objet de classe X est à gauche de la ligne centrale, le cloner à droite par symétrie axiale*. Si possible, en même temps que son langage gagnerait en abstraction, l'utilisateur disposerait d'un espace où il pourrait accéder visuellement aux objets, actions et structures conditionnelles, comme c'est le cas dans un logiciel comme /Turtle Arts/[21].

Babel rappellera les activités Scratch[22] et Etoys[23] à ceux qui connaissent ces environnements inspirés du LOGO et de la Tortue. Sauf qu'il ne s'agirait pas d'apprendre à programmer à partir de blocs *prédéfinis*, mais de comprendre la démarche d'abstraction à partir d'éléments visuels et symboliques *définis* par l'utilisateur. Comme motivation, celui-ci pourrait avoir à relever un défi du genre « faire se déplacer une voiture de gauche à droite de l'écran, puis de droite à gauche quand elle heurte le bord droit » ; essayer d'écrire un programme avec le langage inventé par son voisin, etc. D'autres extensions sont imaginables, comme de voir les affinités du langage inventé avec d'autres langages existants.

4.2. S'inspirer de la démarche des design patterns

En informatique, un « patron de conception » (*design patterns*) est une solution générique utilisée pour la conception de logiciels. Ce n'est pas une solution propre à un langage mais à une classe de problèmes : ce sont les éléments primitifs d'une *architecture* modulaire des logiciels[24].

Notre logiciel imaginaire *Babel* permet de comprendre comment passer d'objets et d'actions spécifiques à un *programme générique*, en laissant le visiteur libre du langage dans lequel il décrira l'univers des objets et des actions possibles. D'une façon similaire, et en s'inspirant de la démarche des patrons de conception, on pourrait permettre aux visiteurs de tenter des rapprochements entre différents logiciels sous l'angle des problèmes qu'ils résolvent et des solutions qu'ils implémentent. A partir d'une description d'abord informelle de leur comportement, le

---

[20] … tellement spécifiques que nous sommes encore à la limite de ce qui s'appelle réellement « langage » en informatique.

[21] Voir l'activité *Turtle Arts* de l'environnement pédagogique libre Sugar: http://wiki.sugarlabs.org/go/Activities/Turtle_Art

[22] http://scratch.mit.edu/

[23] http://www.squeakland.org/

[24] Pour une présentation rapide des idées de Christopher Alexander, voir "Une expérience d'urbanisme démocratique", Éditions du seuil, 1976. La théorie des *design patterns* est plus concrètement détaillée dans *A pattern language*, Oxford University Press, 1977.

visiteur serait amené à formaliser et abstraire sa description progressivement, jusqu'à atteindre un niveau où problèmes et solutions sont communs à plusieurs logiciels.

Ce travail pourrait se faire autour de logiciels devenus des « classiques » de l'histoire de l'informatique. Pour ne prendre que deux exemples, un éditeur de texte et un jeu du type *Arkanoid*[25]. Quel est le domaine des problèmes propres à un éditeur de texte ? Quelles sont les solutions apportées par différents éditeurs de texte à tous ces problèmes ? Quels modes d'interaction sont mis en place par les uns et les autres ? Même questions pour le jeu *Arkanoid*. Se concentrant ainsi sur des classiques, le visiteur apprendra à les « lire » autrement, à aiguiser un regard critique sur des grands problèmes, communs à de nombreux logiciels.

4.3. Une critique de l'informatique ?

Les exemples ci-dessus paraîtront certainement trop exigeants ou trop difficiles à mettre en place dans l'espace et le temps limités d'une visite de musée, mais ils illustrent la manière dont le lien profond entre informatique et pédagogie peut inspirer des activités dont le but est de *créer des liens* entre les aspects épars de nos cultures informatiques. Du lien entre les langages informatiques, entre les logiciels, entre les interfaces. Ils relèvent de ce que nous appelons, après Papert, une *critique informatique*.

La « critique informatique » (*computer criticism*) est l'expression que Papert oppose au technocentrisme[26]. Il définit cette activité en la comparant à la critique littéraire : une critique informatique n'est pas un jugement sur la qualité ou l'efficacité d'un logiciel, pas plus qu'une critique littéraire n'est un jugement sur la qualité ou l'efficacité d'une oeuvre. C'est une exploration active, une tentative pour comprendre les problèmes que l'oeuvre pose, et le champ esthétique dans lequel elle tente de les résoudre. C'est une recherche qui plonge dans la relation entre l'oeuvre et son public, plutôt qu'un description extérieure d'une oeuvre qui exiterait objectivement.

Selon la suggestion de Papert, il en irait de même pour la critique informatique : il faudrait plonger à l'intérieur non seulement des logiciels mais surtout des environnements, des micro-mondes qu'ils créent avec l'utilisateur, de la micro-culture qu'ils induisent dans un groupe, pour explorer les idées qui en émergent, définir le domaine des problèmes et des solutions. On est tentés de donner une dimension kantienne à cette « critique » : son rôle est de déterminer les limites des interactions rationnelles à l'oeuvre entre un homme et une interface informatique, la redéfinition du champ des possibles que cette interface opère.

---

[25] Voir la page de la Wikipédia anglophone sur ce jeu et son histoire. http://en.wikipedia.org/wiki/Arkanoid

[26] *Computer Criticism vs. Technocentric Thinking*, S. Papert, 1987, Educational Researcher (vol. 16, no. I) (Lien)

5. Conclusions

Nous rêvons d'un musée de l'informatique et de la société numérique qui soit le lieu où résister aux trois « images » mentionnées au début de cet article. Premièrement parce que c'est un espace privilégié pour prendre conscience des rapports *collectifs* que nous entretenons avec tel ou tel objet numérique ; ensuite parce que ce lieu mettrait l'accent sur la *continuité historique et conceptuelle* qui relie ces objets les uns aux autres ; enfin parce que les muséologues ont l'habitude de devoir rendre attratifs et intéressants des objets qui ne le sont pas d'ordinaires. En déconstruisant l'image du « hacker » solitaire, le musée poserait les fondements d'une culture générale de l'informatique.

Nous croyons aussi que c'est au musée de construire cette culture, non à l'École. Sans haut-lieu d'une culture informatique générale, l'École pourra bien sûr intéresser les élèves à une vision riche et englobante de l'informatique, mais nous croyons que c'est au musée de définir cette vision, comme c'est le musée qui fait les Beaux Arts avant que l'École ne les enseignent.

Dans *Du Contrat Social*[27], Rousseau définit l'opinion publique et les moeurs comme la clef de voûte de tout l'édifice législatif – clef qu'il est impossible à décréter, mais sur laquelle tout repose. Il en va de même pour la culture générale de l'informatique : elle est impossible à décréter, mais c'est sur elle que repose notre rapport quotidien aux objets numériques. Nous souhaitons un musée qui ne soit pas le simple reflet de ce rapport spontanément décousu et contingent, mais qui aide à comprendre ce rapport en profondeur, et qui mette en place des outils pédagogiques pour l'expérimenter.

Nous avons exploré quelques aspects des débuts de l'informatique dans le but d'y repérer l'omniprésence, factuelle voire conceptuelle, des questions pédagogiques, et nous espérons que ces problèmatiques aideront à concevoir les outils dont ce musée disposera pour poser les fondements d'une culture générale de l'informatique. À l'heure où les musées intègrent de plus en plus le numérique dans leurs pratiques, l'invention d'une pédagogie numérique autour de l'informatique serait une aventure passionnante.

[27] *Du Contrat Social*, Chapitre 2 division 12. J.-J. Rousseau, 1762. (Lien sur Wikisource.)