Alan Kay, le roi de l'informatique avec trois couronnes

Philippe Estival, Stefano A. Cerri

ALAN KAY
ROI DE L'INFORMATIQUE AVEC TROIS COURONNES

Philippe Estival, Stephano Cerri

LIRMM
UMR 5506 - CC 477
161 rue Ada
34095 Montpellier Cedex 5 France

1. RÉSUMÉ

Alan Kay est un informaticien américain. Il a rejoint les laboratoires PARC de Xerox en 1970, où il a travaillé sur le langage Smalltalk et sur la conceptualisation de l'ordinateur personnel moderne. Alan Kay est un des pères de la programmation orientée objet. Il est le concepteur du Dynabook, l'un des premiers prototypes d'ordinateurs portables, et a participé à l'élaboration des interfaces utilisateurs graphiques modernes. Il rejoint au début des années 1980 la firme Atari où il sera directeur scientifique, puis en 1984 la société Apple. À l'heure actuelle il travaille pour Hewlett Packard et il est président de Viewpoint Research Institute. Il a reçu en 2004 le prix Turing de l'ACM pour ses travaux sur la programmation orientée objet, ainsi que le prix Kyoto.

2. INTRODUCTION

Un ami m'a recommandé un jour de ne pas trop chercher ce que je voulais exercer comme informatique lorsque j'aurais fini mes études, mais plutôt de rechercher ce que serait l'Informatique dans 10 ans, et alors ce que je voudrais faire dans cet avenir là. Ce conseil est resté ma devise pendant mon cursus. Avec le recul, c'est une recommandation assez proche de la maxime de Pavese [1] reprise par Alan Kay : "le meilleur moyen de prédire le futur est de l'inventer".

Cet article retrace dans ses grandes lignes les travaux et les réalisations d'Alan Kay, personnage méconnu du grand public et pourtant figure de proue de l'informatique moderne, depuis le milieu des années 60.

3. BIOGRAPHIE

Alan Kay n'est pas, contrairement à Bill Atkinson, un programmeur. Lui aussi (comme tant d'autres chez Apple) est originaire du Palo Alto Research Center. Il est à l'origine des concepts de menus déroulants et de fenêtres. Son rôle chez Apple : penser. C'est en grande partie grâce à son travail de réflexion qu'Apple a pu mettre au point un système aussi abouti que celui du Macintosh.

Né en 1940 à Springfield, Massachusetts, Alan Curtis Kay étudie la biologie moléculaire et les mathématiques à l'université du Colorado et étudie ensuite l'informatique à l'Université de l'Utah. C'est là qu'il obtint ses diplômes, Maîtrise puis Doctorat en ingénierie électrique, et conçut le premier ordinateur personnel à programmation orienté objet doté d'une interface graphique. le Dr. Kay fut membre de l'ARPA (Advanced Research Project Agency). Il participa également à l'élaboration de l'ARPAnet, qui devint l'Internet dans les année 70.

Il rejoint le PARC de Xerox en 1970 et en devint l'une des élites. Leurs groupes de travail seront à l'origine du Smalltalk, l'Ethernet, l'impression Laser et le réseau client-serveur.

En 1979, c'est lui qui invite Steve Jobs à visiter le PARC, sans se douter que, cinq ans plus tard, ils inventeraient l'ordinateur individuel ensemble avec le Macintosh. En 1980, il travaille chez Atari, mais sa passion des ordinateurs personnels le ramène chez Apple en 1984.

L'interface graphique, il connaît parfaitement : après avoir conçu l'interface du Flex en 1967, puis Smalltalk (environnement utilisant la souris et les fenêtres multiples), il participe à la création de l'ALTO, qui est pour beaucoup le premier ordinateur moderne. Kay développa d'ailleurs sur cette machine un logiciel

1. Cesare Pavese est plus connu comme romancier et auteur noir mais il s'est voulu d'abord poète et philosophe. "Pour connaître le monde, il faut le construire".

de dessin capable de reconnaître les caractères dessinés à la souris. C'est également sur l'ALTO que des principes tels que le déplacement d'icônes ou de textes ont fait leur apparition.

Il considère l'ordinateur comme un média, et s'intéresse donc à l'objet et son interface. "Quand on demande à un enfant de faire un rond, il tourne sur lui-même. Si l'on pose la même question à un adolescent, il s'appliquera à dessiner un rond parfait avec un compas. Quant à un adulte, il donnera simplement la formule du cercle. On retrouve ces trois états d'appréhension du monde avec le Macintosh. La souris est le contact physique, les programmes correspondent à l'intelligence, et les icônes au symbolisme" [7].

Depuis, Alan Kay est devenu vice président du département de recherche et développement de Walt Disney. Il s'est tourné vers l'enseignement et les conférences. Il reste une grande figure de l'informatique et ses apparitions dans les salons informatiques sont très appréciées.

Kay est membre de l'Académie Américaine des Arts et des Sciences, de l'Académie Nationale d'Ingenierie et de la Société Royale des Arts. Ancien professionnel de guitare jazz, auteur de pièces de théâtres et amateur d'orgue classique.

Il a épousé Bonnie MacBird. Elle est l'auteur du script du film *Tron*, qui est une allégorie de l'œuvre de Kay : le héros, nommé Alan, se bat contre le Programme Maître de Contrôle (orienté ligne de commande).

Photo 1. Le héros en train de jouer au Dynabook.

4. LES DÉBUTS DE LA PROGRAMMATION ORIENTÉE OBJET

La programmation Orientée Objet (OO ou POO) se manifesta principalement pour deux raisons [Kay92] : la première était de trouver un meilleur agencement des modules dans les systèmes complexes et la possibilité de masquer les détails. La seconde de trouver un moyen plus flexible de réaliser les assignations. Comme dans bien des cas, la solution fut d'abord adoptée localement, avant de devenir un motif régulier, puis ce motif se transforma en une nouvelle façon de penser avant de devenir une religion inflexible.

L'idée lui apparu pour la première fois en 1961 alors qu'il était programmeur pour l'Air Force, sur le Burroughs 220 [2] dans le style du transport des fichiers (rouleaux de bandes magnétiques) depuis un poste de commandes vers un autre. Il n'existait pas de système d'exploitation standard ou de format de fichiers, aussi quelqu'un avait eu la finesse de prendre chaque fichier et de le diviser en trois parties : la troisième partie contenait tous les enregistrements des données de tailles et de formats arbitraires. La seconde contenait les procédures du B220 qui savaient comment accéder à un enregistrement et un champ pour copier et mettre à jour la troisième partie. Et la première partie était un tableau de pointeurs relatifs des points d'entrées des procédures de la seconde partie. Kay jugea l'idée très bonne et par la suite elle fut reprise dans de nombreux systèmes avant que le débarquement de COBOL ne l'envoie aux oubliettes.

Quand ATC remplaça le B220 par le B5000, Kay étudia le nouveau système en détail et commença alors à réfléchir aux méthodes de traductions et d'évaluations des langages de haut niveau. Parmis les mécanismes du B5000 – segmentation du stockage, protection, interface procédurale d'accès aux modules entre autres – se trouvaient les germes de l'OO.

Après l'Air Force, Kay poursuivit ses études et programma surtout des systèmes de fouilles de données pour le centre national de recherches atmosphériques. En 1965, Gordon Moore publia ses prédictions sur la croissance exponentielles de la densité et des coûts des circuits intégrés sur silicône. Kay entra à l'université de l'Utah en 1966. Dave Evans, qui dirige alors le département informatique, lui remet la

2. le remplaçant du Datatron, lui-même successeur de l'UNIVAC. Son CPU était composé de 1800 tubes à vides. Il utilisait une représentation décimale plutôt que binaire pour faire un compromis entre calcul scientifique et économique. Environ 55 exemplaires furent construit leur prix allant de 640.000$ à 1.209.000$. Le B220 fut programmé en assembleur STAR1 puis en BLEAP et enfin en ALGOL.

thèse de Ivan Sutherland (qui succédait à J.C.R. Licklider à la direction de l'ARPA, à 26 ans), intitulée "Sketchpad : A Man-machine Graphical Communications System" (Sketchpad : une interface de communication Homme-Machine). Le logiciel permettait pour la toute première fois à une personne de créer interactivement une image sur un écran d'ordinateur. Selon Sun Microsystems, dont Sutherland est actuellement vice-président, Sketchpad fut un pionnier des concepts de calculs graphiques en incluant les structures de mémoires permettant de stocker les objets, étirer des lignes, la capacité de zoomer sur l'affichage et la capacité de faire des lignes, des coins, et des joints parfaits. C'était le premier GUI (interface utilisateur graphique) bien avant que le terme soit inventé [12].

Dans Sketchpad, les structures de données sont envisagées sous un nouveau jour : les choses sont décrites par des "schémas maîtres" qui produisent des "schémas d'instances". Les contrôles sont gérés par "contraintes", associées aux représentations graphiques, et sont appliquées aux maîtres pour créer les formes et mettre les parties des formes en relations.

Dans le même temps on lui confie une documentation difficilement déchiffrable (du norvégien traduit littéralement) d'une thèse sur un langage dérivé de l'ALGOL, appelé SIMULA [3]. La traduction emploie les termes activités et processus dans un anglais impropre.

Kay réalise que le système de fichier du B220, le B5000, Sketchpad et SIMULA emploient tous le même concept dans des buts différents. Quelques jours auparavant, Bob Barton, concepteur du B5000 et professeur à l'Université de l'Utah a dit : « le principe de base de la conception récursive est faire en sorte que les parties aient la même puissance que le tout ». Kay imagine le tout comme étant l'ordinateur et les parties, des divisions qui s'appelleraient les structures de données et les procédures. Encore quelques années et il trouvera comment mettre en œuvre ses illuminations et concevoir les mécanismes nécessaires à leurs exécutions.

5. FLEX

Entre 1967 et 1969, Alan Kay et Ed Cheadle travaillent sur la construction du FLEX (l'acronyme de Flexible Extensible). La sortie graphique est prévue au cœur de l'interpréteur matériel.

« Un autre principe fort était de ne pas marquer de séparation entre matériel (hardware) et logiciel (software). Les deux cultures étaient totalement mélangées. Tout le monde au PARC savait faire les deux, c'est-à-dire écrire un système d'exploitation ou un compilateur, et fabriquer un ordinateur si c'était nécessaire. »

« Dans les années 60, quand il était encore très compliqué de fabriquer un ordinateur, toutes les universités américaines construisaient le leur. Il existait plusieurs milliers de langages de programmation différents. [...] Aujourd'hui, alors qu'il est plus facile que jamais de fabriquer un ordinateur, aucune université américaine ne le fait, pas plus qu'elle ne cherche à développer son propre OS ou son propre compilateur. Que vous alliez au HP Labs, au Sony Labs, à l'université de Standford ou au MIT, vous voyez des gamins qui travaillent sur les ordinateurs portables d'aujourd'hui, avec les logiciels d'hier. Je ne vois pas comment ils pourraient inventer le futur en travaillant avec les outils d'hier... [...] » [6].

Alan et Ed veulent que programmer cette machine soit à la portée des non-spécialistes dans un langage tel que le BASIC. Les candidats à l'étude seront JOSS, SIMULA, EULER et bien d'autres ; mais dans tout ces langages, Kay aura le sentiment que « l'utilisateur est poussé dans un coin d'où il ne peut voir son monde que par un trou de serrure ». Il fera cas à parts d'Interactive Lisp et de TRAC ("Text Reckoning and Compiling", très inspiré de Lisp). Leur défaut ? Des programme écrits dans l'un ou l'autre « ressemblent à des lettre du Roi Burniburiach aux sumériens en Babylonien cunéiforme ».

Tandis que le projet se précise, les contraintes font surfaces : comment l'utilisateur va interagir avec le système ? Ed Cheadle a déjà prévu des affichages en télétype sur ses premiers ordinateurs, mais un système comme Sketchpad dépasse largement les 16k mots de 16 bits alloué par leur budget et la sémantique du langage FLEX est devenue tortueuse...

3. Il faut toutefois reconnaître que SIMULA I (1962-65) et Simula 67 (1967) sont les deux premiers langages orientés objets. Simula 67 introduisit la plupart des concepts clés de l'OO : objets, classes et sous-classes (l'héritage), procédures virtuelles et liaisons dynamique. Conçu à l'origine dans l'esprit d'ALGOL 60 et utilisé pour la simulation d'évènements discrets, il fut réecrit pour devenir un langage de programmation générique. Simula fut crée au Centre de Calcul Norvégien à Oslo par Ole-Johan Dahl et Kristen Nygaard.

La visite de Doug Engelbart (un “prophète de dimension biblique”) aura un impact important sur le développement du FLEX, et Kay lui attribuera volontiers la paternité de son ordinateur. Sa conception de l’ARPA est d’être la couveuse d’un “Système oNLine (NLS) qui servira à augmenter l’intellect humain grâce à un véhicule de navigation interactif par vecteur de pensée dans un espace de concepts”. Son propre système gère l’hypertexte, les graphiques, les panneaux multiples, la navigation, le travail collaboratif [Kay92].

Mais le NLS est encore hors de portée des capacités du FLEX et la hiérarchie de navigation est contraignante (il faut retourner à la racine pour aller ailleurs). Kay conserve donc cette notion propre à Sketchpad de fenêtre générale qui regarde dans un monde virtuel, et récupère l’algorithme de clipping de Ed Cheadle, très semblable à celui que Sutherland et ses étudiants développent en 68 à Harvard dans un projet de casque de réalité virtuelle 3D.

Le FLEX poursuit son évolution : les références aux objets deviennent une généralisation des descripteurs du B5000. Au lieu d’avoir plusieurs formats pour référencer les nombres, les tableaux et les procédures, un descripteur FLEX contient deux pointeurs : l’un vers le “maître” de l’objet, le second vers son instance (il réalisera plus tard qu’un pointeur vers le maître dans l’instance fera gagner de l’espace).

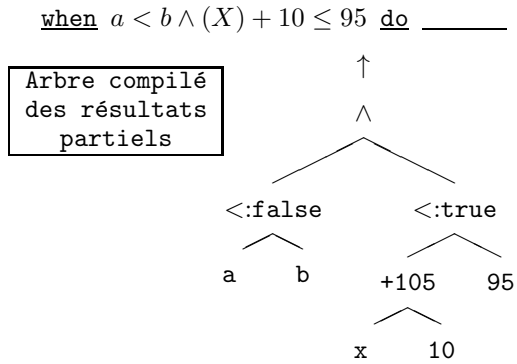

Figure 1. L’instruction **when** de FLEX

Parmi les autres caractéristiques intéressantes de FLEX se trouvait un principe d’interruption logicielle par événements appelée **when** [Kay69]. Son expression booléenne était compilée en un arbre de “tri par tournoi” (tournament sort) dont tous les résultats intermédiaires possibles étaient placés en cache. Les variables pertinentes étaient traitées dans tous les arbres de tri dans tous les *when* de telles sortes que les changements ne nécessitent de ezcalculer que les parties nécessaires des booléens. Cette technique très efficace était semblable à celle utilisée aujourd’hui dans les feuilles de calculs.

Pendant l’été 1968, Alan découvre le premier écran plat au plasma et estime selon la loi de Moore qu’il faudra attendre le début des années 80 avant de pouvoir intégrer cette technologie dans un ordinateur. L’invention de la tablette graphique (GRAIL, successeur de JOSS), par Tom Ellis, remet en partie en cause les plans de l’interface et la première maquette à l’étude est construite par les deux étudiants. Le FLEX cède la place au KiddiKomp.

Photo 2. Le modèle du Dynabook

6. LA FONDATION DU PARC

En juillet 1970, Jack Goldman, chef scientifique de Xerox établit un centre de recherche à Palo Alto, Californie. Bob Taylor est engagé pour y fonder un laboratoire d’informatique. Janvier 1971, le staff de Berkeley Computer Corporation les rejoint, suivi par la bande de Doug Englebart. Le projet ARPA basé à l’Université Washington intéresse la Défense. Les crédits sont débloqués en conséquence.

Pour remonter aux sources des idées fondamentales de l’informatique des années 60, idées qui occuperont majoritairement l’esprit des chercheurs du PARC, il convient de lire le travail précurseur fait en la matière par Licklider, directeur de l’ARPA en 1963. Voir notamment [8] qui discute de ce que l’ordinateur apporte à l’humanité sur des points stratégiques.

7. 1972

L'Ethernet n'est pas encore au point (Robert Metcalfe le concevra en 1973). Le processeur Intel 8008, cadencé à 108KHertz, vient de faire ses débuts. Le premier PC est proposé à 830€. La disquette 8 pouces à été inventée l'année précédente. Le langage de programmation Pascal est terminé et le langage C est crée. Le VLSI atteint 500 composants. ARPANET ne compte que quatre nœuds, et la souris n'existe pas encore. HP lance la première calculatrice de poche (la HP-35).

8. DES PC POUR LES ENFANTS

"Should the computer program the kid or should the kid program the computer ?"
– S. Papert

En août 1972, Alan Kay rédige une note au PARC à l'attention de la communauté scientifique. Il exprime ses idées sur l'émergence de l'ordinateur personnel portable et prévient d'emblée le lecteur que ce n'est pas de la science-fiction.

Tandis qu'à cette époque la plupart considèrent l'ordinateur comme un outil pour les ingénieurs ou les économistes, Kay pense que les ordinateurs doivent être rendus utilisables par tous, surtout par les enfants et dans un contexte créatif. Depuis 1968, il a examiné son utilité au service de l'éducation [Kay72b] : indiscutablement, il peut améliorer le processus d'apprentissage. L'exemple du simulateur de vol illustre bien ceci : il introduit de manière intuitive et pratique des notions de physique dans l'esprit de l'enfant. En cela, Kay se rapproche des idées de Piaget, plus que de celle de Skinner : l'enfant est l'acteur de son apprentissage et non l'objet.

En 1972, les chercheurs en intelligence artificielle se penchent déjà sur l'étude des modèles d'acquisition du savoir des enfants, dans la suite des travaux de Newell et Simon (les échecs), Papert [4] et Minsky [5], Moore et Andersen, et plus précisément des théories énoncées par Piaget [6], Hunt, Burner et Kagan [7]. Ces deux derniers ont démontrés que les enfants ont des capacités de discrimination et de généralisation visuelle plus grande que celles qui étaient supposées. Moore examina les questions relatives à l'attention de l'enfant : il s'avère qu'elle est plus soutenue si l'enfant est placé dans un rôle d'acteur, de juge ou de joueur, plutôt que d'auditeur patient. Ses travaux se fondent sur l'analyse des capacités de différenciation, d'abstraction et d'activité d'intégration mentale en fonction de l'environnement. Encore une fois, l'observation d'un enfant face à un jeu vidéo le démontre : c'est un environnement qui réagit immédiatement à l'activité et dans lequel on observe que l'enfant peut atteindre un haut niveau de concentration. Cependant, cet exemple ne répond pas au critère d'environnement productif, qui selon la terminologie de Moore, doit aider à la construction d'un modèle d'identité propre.

Parmis les idées de Piaget, Kay retient deux notions fondamentales du point de vue de l'informaticien [Kay77a].

1. La connaissance, surtout chez les jeunes enfants est acquise par une série de modèles opérationnels, délivrés "ad hoc" et qui ne nécessitent pas d'être logiquement consistants les uns avec autres. Ils sont davantage des algorithmes et des stratégies plus que des axiomes, des prédicats et des théorèmes. Ce n'est que bien plus tard dans l'apprentissage que le logique est acquise et encore, par des stratégies extralogiques.

2. Le développement procède par étape, chacune dépendante de et construite sur la précédente. La langage n'est pas maître de la pensée. Piaget et d'autres fournissent une large preuve qu'une grande part de la pensée n'est ni verbale ni iconifiée.

Ceci amène les conclusions suivantes :
□ Le "modèle opérationnel" est un algorithme, une procédure pour accomplir un but (ex : le calcul, dans notre contexte).
□ L'algorithme est informel est pas toujours consistant.
□ Le point de vue de l'enfant est global, et s'intéresse

4. Seymour Papert eut beaucoup d'influence sur les travaux de Kay. C'est lui qui conçut le premier l'ordinateur à usage créatif pour enfants et c'est au MIT qu'il élabora le langage LOGO. Pour la petite histoire, il fait partie des quatres qui perdirent un pari (et 250£) contre David Levy, à savoir qu'aucune machine ne battrait ce dernier aux échecs entre 1968 et 1978.

5. Marvin Minsky (1927-), cofondateur du Laboratoire d'IA du MIT. Distinction Turing 1969, Prix du Japon 1990, Médaille Benjamin Franklin 2001.

6. Jean Piaget (1896-1980) est un psychologue et pédagogue genévois (biologiste de formation) qui va développer dès 1925, les théories dites constructivistes. Ses travaux portent sur la construction des connaissances au cours du développement biologique de l'Homme. Ses théories transposent les modèles du développement biologique à la construction de la connaissance.

7. Pour une synthèse sur les différents modèles de l'apprentissage voir : www.chevaleyre.com/cc/Apprentissage/apprent5.htm

plus à la structure qu'à l'implication de "vérités".

De plus, l'ordinateur assiste à la formation de capacités de raisonnements : stratégie et tactique, planification, observation et chaînes de causalités, ajustement, correction et raffinement. Autant de capacité que l'enfant a rarement l'opportunité de mettre à l'épreuve dans un environnement patient, amusant et sûr.

Les enfants adoreront : la nature interactive du dialogue, le fait qu'ils soient aux commandes, le sentiment qu'ils font quelque chose de sérieux au lieu de jouer ou de travailler sur des problèmes donnés, l'aspect pictural et auditif des résultats. Tout ceci contribue à un sentiment d'accomplissement de leur propre expérience.

8.1. L'étude technique

Les contraintes de fabrication du Dynabook gravitèrent autour de la taille, du poids, du coût et des capacités.

Sans entrer dans les détails des calculs économiques – moins cher qu'une télévision sera la référence de Kay – on peut déjà se demander quel est le coût et la pérennité de l'ensemble des ouvrages scolaires nécessaires à la scolarité. Le Dynabook, lui, se perçoit tout naturellement comme un mélange entre un livre, un piano, un outil, jouet et moyen d'expression, avec toutes leurs vertues éducatives. Grosso Modo, le coût des manuels scolaires de l'école primaire au lycée peut être estimée entre 100 et 200€ par élèves et par an. Un Dynabook doit pouvoir servir au moins 3 ans. Donc s'il respecte cette contrainte de fiabilité, le coût est amorti. La rentabilité pour les éditeurs serait mirobolante et assurer l'infrastructure d'un réseau bibliothèque, un jeu d'enfant.

Mais voir le Dynabook comme un remplaçant du livre scolaire serait une erreur : l'objectif est de faire de l'ordinateur « l'objet d'une réflexion » pour l'enfant, notamment en l'incitant à le programmer. Aussi mettre le langage Smalltalk à la portée des plus jeunes sera le cheval de bataille d'Alan pendant le reste de sa carrière.

Le premier prototype du système d'exploitation du Dynabook s'appelle Bilbo. Les chercheurs du PARC sont bien conscients que les performances prévues pour le Dynabook dépassent encore la puissance disponible dans un portable, aussi ce prototype a la taille d'une machine à laver. Il exécute 6MIPs, dispose de 128K de mémoire vive et il est muni d'une tablette graphique à stylographe comme GRAIL. La première préoccupation de ces concepteurs est de mettre au point un affichage dynamique de caractères à hautes résolutions.

L'affichage est de 500.000 pixels sur un écran de 8.5" x 11". Pour comparer avec des standards qui nous sont bien connus, cela correspond à une résolution de 915 x 546 pixels sur un écran de 14"!

L'utilisateur pourra choisir sa police de caractères lors de la lecture de n'importe quel document, les menus déroulants et les fenêtres multiples voient le jour, tout ceci grâce à Smalltalk. Le langage de programmation permet de manipuler et de construire toutes sortes de programmes. Les premiers disponibles sont des outils de dessin et de peinture, d'animation et de musique. Le prototype sera soumis aux tests de 200 utilisateurs, essentiellement des étudiants et des écoliers. Parmis eux certains programmeront leur propre outil de dessins, un tangram, une CAO de circuit électronique, des jeux et jusqu'à la modification du logiciel d'animation SHAZAM. Ils ont entre 12 et 15 ans.

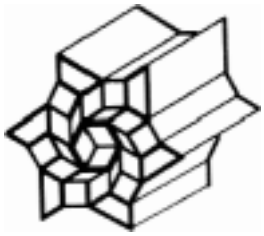

Figure 2. Un tangram réalisé à partir du programme écrit en Smalltalk par une fille de 14 ans.

8.2. Un rêve, le dynabook ?

Toutes les technologie contenues dans les portables d'aujourd'hui correspondent à la vision de Kay, mais il admit que le Dynabook resterait un rêve [8].

Les raisons sont multiples. En premier lieu, il faudrait former les enseignants à l'emploi des nouvelles tecnhologies. Les former en quelques semaines ne suffit pas pour développer une nouvelle façon de penser. Un autre argument qui fut avancé était qu'il est difficile d'apprendre aux enfants à raisonner et que cette nouvelle façon de penser était encore plus difficile. Remarquons que Kay semble adhérer à l'idée de l'effet Hawthorne, qui veut que nous soyons aussi optimistes que possible quant aux capacités mentales de l'enfant si nous voulons qu'il progresse. Tout comme on tombe

8. L'esprit est resté dans la gamme des ordinateurs portables d'Apple : les dimensions "livresques", le poids, les offres proposées aux institutions éducatives et la dénomination (*book).

plusieurs fois en apprenant le vélo, une culture de la difficulté et des échecs dans l'apprentissage serait de rigueur. C'est de ses erreurs qu'on apprend.

L'ordinateur comme moyen éducatif se résume encore au modèle télévisuel. Le contenu et les valeurs initiales d'un nouveau media sont toujours pris des anciens media. Une approche "Montessorienne" de L'Internet pourrait en faire un très bon moyen éducatif, si nous avions la possibilité d'adapter son contenu, à l'âge de l'internaute par exemple. Nous pourrions ainsi créer des vecteurs éducatifs déguisés en jeux où l'enfant serait acteur de son éducation [4].

Il est intéressant de constater la façon dont Kay s'est démarqué de ses confrères chercheurs : certains consacraient leurs efforts à la construction de calculateurs plus puissants et plus rapides, d'autres à la réduction des coûts des processeurs, lui, à faire cette petite machine pour les enfants. Les gens voyait le P.C. de la même façon que l'automobile (et IBM comme le train), mais au PARC la métaphore s'arrêta sitôt que Seymourt Paper eut mis ses programmes éducatifs à l'épreuve [3].

9. UN PETIT MOT SUR SMALLTALK

« The most important thing in the programming language is the name. A language will not succeed without a good name. I have recently invented a very good name and now I am looking for a suitable language. »

– Donald E. Knuth, 1967

9.1. La construction du langage

Kay mit en perspective son étude approfondie du LISP avec ses connaissances en biologie – la cellule membraneuse en tant qu'une interface uniforme avec un rôle dans l'évolution – pour inventer une structure de messages inter-objets.

Voici à quoi elle ressemble :

Global :	*L'environnement des valeurs paramètres*
Sender :	*L'expéditeur du message*
Receiver :	*Le receveur du message*
Reply-style :	*wait, fork, ... ?*
Status :	*progression du message*
Reply :	*résultat éventuel*
Operation Selector :	*relatif au receveur*
# Of parameters :	P_1.. P_n

Smalltalk-71 sera implémenté pour la première fois sur le miniCOM (la suite du KiddiKomp qui dispose en plus d'un périphérique de pointage et d'un système de stockage secondaire).

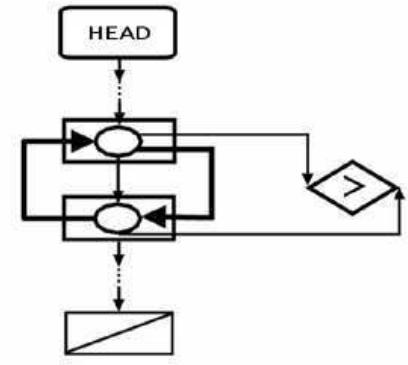

Figure 3. Représentation iconique du tri-bulles [Kay72b]

Mettre au point une programmation par icônes fût l'enjeu d'une partie difficile. On appliqua la méthode classique pour résoudre ce genre de problèmes complexes en le confiant aux thésards. La solution aboutira en 1975.

Quinze exemplaires du Bilbo étaient construits, grâce aux 230.000$ alloués par le budget, quand Kay édicta ses principes pour Smalltalk.

1. Tout est *Objet*.
2. Les objets communiquent par l'envoi et la réception de messages (eux mêmes des objets).
3. Les objets ont leur propre mémoire (toujours en termes d'objets).
4. Chaque objet est *instance* d'une classe (qui doit être un objet).
5. La classe détient le *comportement* partagé par ses instances (sous la forme d'objets dans un programme).
6. Pour l'évaluation du programme, le contrôle est passé au premier objet, le reste est traité par messages.

Les progrès dans Smalltalk-72 touchèrent au polymorphisme, à la protection et aux privilèges. Le potentiel du langage arrive à ce stade qu'il peut être envisagé dans l'avenir comme un système d'exploitation de réseaux.

9.2. Les applications de Smalltalk-72 à Smalltalk-76

Les premières applications furent une mise au point de la superposition des fenêtres. Le résultat conserva les conventions de GRAIL à savoir les coins préhentibles pour le déplacement, le redimensionnage, le clônage et la fermeture. Les classes suivantes implémentés sur un Dynabook provisoire furent la version OO de la tortue LOGO.

Vinrent ensuite :
- Un éditeur WYSIWIG déployé sur le NLS : *mini-Mouse* fut un temps l'éditeur de code alternatif.
- *Findit*, une interface de recherche par l'exemple, qui utilisait l'analogie des classes vers leurs instances pour former des requêtes de recherche.
- Un synthétiseur à douze canaux, relié à deux claviers d'orgues et un pédalier.
- TWANG, un système de capture et un éditeur de musique.
- SHAZAM, un système d'animation (1974).
- PYGMALION fut l'aboutissement d'une thèse sur la programmation par icônes (1975). C'était le plus gros programme jamais écrit en Smalltalk-72.

Smalltalk-76 fut implémenté en 7 mois (essentiellement par Dan Ingalls). Le système est composé de 50 classes (environ 180 pages de codes sources), inclut les fonctionnalités du système d'exploitation, fichiers, impressions et services Ethernet, l'interface utilisateur, les éditeurs, les programmes de graphismes et de dessins plus un navigateur parmis les méthodes statiques dans la hiérarchie des héritages et des contextes dynamiques pour le débogage à l'exécution. ST-82 est la version officielle qui fut livrée au public en 1982.

Rappelons que Smalltalk est un langage : interprété/compilé, à typage dynamique, à héritage simple, avec méta-niveau, réflexif, à sélection simple.

Et voici comment afficher dix fois "Hello" dans un programme Smalltalk (plus exactement en Squeak, une implémentation de Smalltalk) :

```
10 timesRepeat: [Transcript show: 'Hello '.].
```

10. APPRENDRE AUX ENFANTS LA P.O.O.

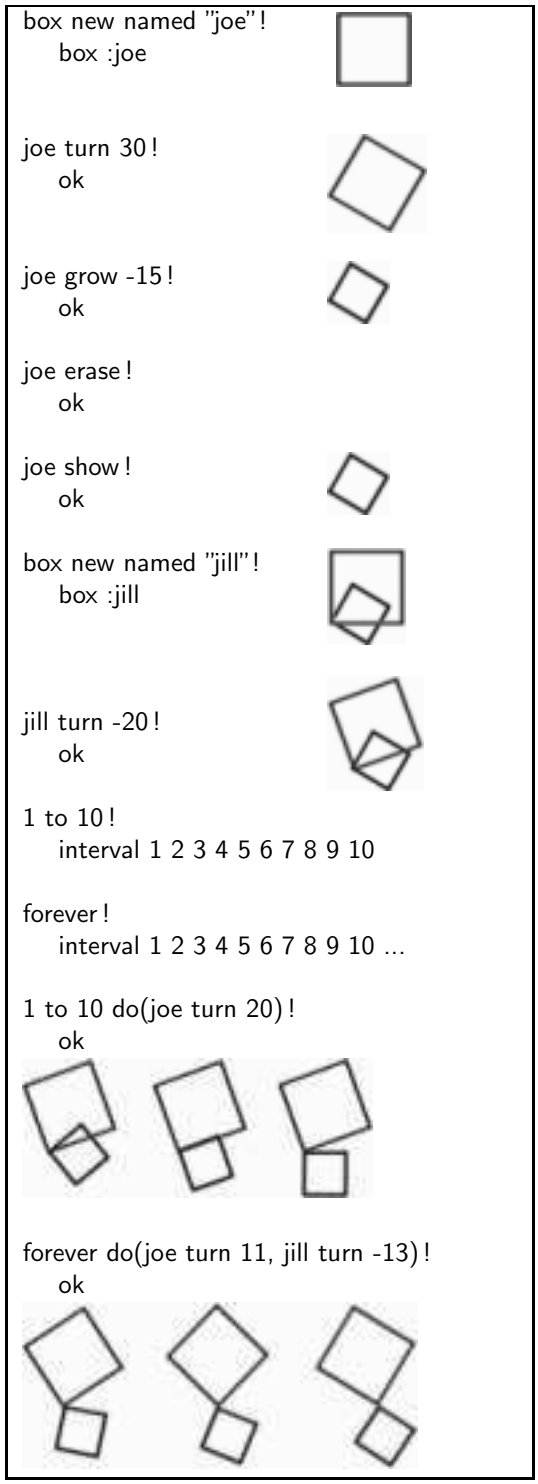

Figure 4. Une séquence de "Joe Book"

Adèle Goldberg fit une première tentative avec une adaptation en objet de la tortue LOGO. Les petits arrivaient à faire dessiner la tortue, mais le résultat demeurant superficiel, Kay chercha un moyen de mettre à leur disposition la possibilité de créer eux-mêmes les outils interactifs.

L'approche suivante de Goldberg pour apprendre Smalltalk comme un langage orienté objet fut "Joe Book" (Figure 4). On créait plusieurs instances de la classe boîte et on leur envoyait des messages pour finir par une animation simple à plusieurs processus. Les enfants devaient ensuite deviner ce que pouvait être une boîte. Pour finir on leur montrait ceci :

```
   to box | x y size tilt
(□ draw     • (@place x y turn tilt. square size.)
□undraw     • (@white. SELF draw. @black)
□turn       • (SELF undraw. 'tilt <- tilt + :. SELF draw)
□grow       • (SELF undraw. 'size <- tilt + :. SELF draw)
ISNEW       • (SELF undraw. 'size <- tilt + :. SELF draw)
```

Comme on s'y attendait, une myriade de projets naîtront de ces petites boîtes.

10.1. Vers un meilleur apprentissage

« [...] Une connaissance est ou obscure ou claire *; une connaissance claire est à son tour ou confuse ou* distincte *; une connaissance distincte est ou inadéquate ou* adéquate*, et encore ou symbolique ou* intuitive *; si elle est en même temps adéquate et intuitive, elle est la plus parfaite possible. ».*

C'est par ces mots que Leibniz débute son essai intitulé « De Cognitione, Veritate et Ideis ». Son principe monadologique – faisant de l'individu une image du monde dans son intégralité – fut source d'inspiration à la création de Smalltalk.

Lire et écrire ne suffit pas. Il y a aussi une *littérature* pour rendre des *idées*. Le langage est utilisé pour les lire et les écrire mais à partir d'un certain point l'organisation des idées commence à dominer une portion des facultés linguistiques. Avoir de bonnes idées aide beaucoup à l'acquisition d'idées encore meilleures [9].

Partant de ça, Goldberg et Kay décident d'enseigner les études et les analyses, les pré-projets, la conception des programmes avant leur fabrication. Adèle formule une autre idée brillante : elle constate qu'ils ont besoin d'un intermédiaire entre les vagues définitions des problèmes, et le codage/débogage. Elle nomma ces formes intermédiaires les *patrons de conceptions (design templates)*. Ce qui pourrait nous donner l'impression que l'informatique pour les enfants est l'enfance de l'informatique. On peut d'ailleurs prolonger le parallèle avec le phénomène du "hacker" adolescent... [9]

L'idée de Goldberg était bonne, mais l'euphorie du PARC retombe et le doute s'installe au regard de six années d'expériences. Les chercheurs ont cru en leur invention, impressionnés par les applications que quelques enfants étaient parvenus à réaliser. Mais dans son ensemble la portée pédagogique du projet est remise en cause. Le problème n'est pas de faire *faire* des choses aux enfants. Ils adorent *faire*, quand bien même ils ne sont pas sûr de ce qu'ils sont en train de faire. Kay fit le rapprochement avec les débuts de l'acquisition du langage : faire répéter les mots est nécessaire même lorsque leur sens n'a pas d'importance. La difficulté est de déterminer quelles sont les idées à mettre en avant, dans quelle mesure elle doivent pénétrer l'esprit de l'enfant, et à quel stade de son développement. Et il conclura "la musique n'est pas dans le piano" [1].

« The reason, therefore, that many of us want children to understand computing deeply and fluently is that like literature, mathematics, science, music and art, it carries special ways of thinking about situations that in contrast with other knowledge and other ways of thinking critically boost our abilty to understand the world. [10] *»*

Ce qu'il maintiendra, malgré l'argument (un peu court certes) que programmer n'a pas n'a pas d'influence discernable sur les capacités générales de réflexions ou de compréhensions du savoir humain.

Mais avant d'en arriver à cette conclusion, il dégagera les corollaires suivants : la connaissance est à son stade le moins intéressant quand elle est apprise pour la première fois. Les représentations – que ce soit des inscriptions, des allusions ou des contrôles physiques – entravent le cheminement (et deviennent parfois un but). Elles doivent être laborieusement et péniblement interprétées. De là partent deux notions. Premièrement, la *fluence*, qui fait partie du processus de construction de structures mentales pour faire disparaître l'interprétation des représentations. Les lettres et les mots d'une phrase sont apprises comme des significations au lieu d'inscriptions, la raquette de

9. La contre-culture pirate a souvent détourné l'étudiant de considérer sa discipline comme une science. Les aventures de la machine Enigma ont renvoyé cette image d'une technologie bricolée à la hâte pour les besoins de la guerre...

10. Ambigü, même dans son contexte ([Kay92]). Kay joue sur les mots et met l'accent sur la nécessité de développer une pensée critique.

tennis ou le clavier devient une extension du corps, etc.. Deuxièmement, prendre la connaissance comme une *métaphore* servant à éclairer d'autre aires. Mais en l'absence de fluence, le risque est que les connaissances précédentes prévalent, et que les métaphores soient alors floues et induisent en erreur.

Ceci mit à un terme temporaire à ces projets éducatifs. Par la suite, chez Apple, Kay s'orienta vers une approche plus pragmatique. Au passage il travaillera sur le problème de la justification des mots dans un paragraphe – problème difficile déjà abordé par Knuth et ses étudiants – pour en faire une unité de tests des systèmes orientés objets.

11. CROQUET

La dernière invention du Dr. Kay s'intitule Croquet. Le projet est "open source". Il est disponible (ainsi que ces sources) à l'adresse suivante :

`http://www.opencroquet.org/`

Il s'agit d'un système d'information bâti sur un ensemble de programmes intégrés dans une architecture en réseau. Le contexte collaboratif, le partage des ressources et l'aspect massivement multi-utilisateurs **et** pair-à-pair sont les fondements de cette architecture. Le système dispose d'une visualisation 3D (entièrement écrite en Squeak et où chaque objet rendu graphiquement a accès aux librairies OpenGL) et l'ensemble de la plateforme compose un vaste système distribué. Contrastant avec les architecture traditionnelles où seules les données sont répliquées, Croquet incorpore un principe de réplication exacte *d'objets et d'activités*. Le miroir fait d'ailleurs partie des primitives d'affichage, voulant signifier au passage que l'introspection est au cœur du langage.

Un environnement de développement est entièrement intégré à ce système d'exploitation – car s'en est un – toujours dans un esprit collaboratif ;

La 3D offre aux utilisateurs un espace de téléprésence – dans un monde virtuel – accentuant l'aspect social du système.

Le système est interopérable sur les plateformes Windows, Mac et Linux grâce à une machine virtuelle, mais il fonctionne également sur son propre matériel.

A y regarder de plus près, Croquet est le premier système d'exploitation orienté objet, plutôt que "orienté programme" [5].

Le langage de script de Croquet permet de contrôler l'intégralité des objets de l'environnement. En ces termes, objet prend le sens qu'il a dans le langage Squeak (objet/messages/activités). Implicitement ce langage est collaboratif car les actions scriptées déclenchent des réponse synchrone sur le réseau pair-à-pair.

Encore une fois, tout est objet. Ici ils sont tous manipulables simultanément par tous les participants avec les privilèges appropriés. Une preuve d'utilisation remarquable : non seulement un dessin réalisé en quelques secondes avec le programme *paintbrush* de Croquet évolue sous les yeux des participants, mais il peut en plus être exporté en un instant en trois dimensions. La transparence 2D/3D est tout à fait remarquable, et il y a fort à parier que Croquet impose cette « vue de l'esprit » d'un OS qui tire parti intelligemment de la 3D.

La couche de synchronisation temps réel est une technologie baptisée Tea-Time. C'est elle qui gère le versionnage des objets répliqués, la communication d'objet à objet et la synchronisation monde/objet entre les clients.

Croquet se veut un système d'exploitation de l'ère post-navigateur d'Internet. Ses principaux initiateurs sont : Alan Kay, Julian Lombardi, Mark McCahill, Andreas Raab, David P. Reed, et David Smith.

Les portraits des six accueillent l'utilisateur lorsque il entre dans son environnement personnel avec comme avatar Alice au pays des merveilles ou le Lapin Blanc !

12. LE PERSONNAGE

12.1. trois fois couronné

En 2004, il reçoit le prix Kyoto, pour ses travaux conduits au PARC dans les années 60-70, et sa recherche permanente pour faire de l'ordinateur un meilleur outil éducatif.

Ce prix accompagné de 50 millions de yens par catégorie (environ 400 000 euros), d'une médaille en or et d'un diplôme, compte avec le prix Nobel parmi les principales distinctions du genre au monde et a été créé en 1984 par Kazuo Inamori, le fondateur du groupe technologique japonais Kyocera. Il est décerné dans les catégories Art & Philosophie, Haute Technologie et Recherche Fondamentale.

Le représentant de la fondation Inamori expliqua que le prix était souvent attribué pour le travail accompli, mais surtout à ceux qui laisseront un héritage à la communauté. Il a été ainsi reconnu que les

travaux de Kay ont eu une influence positive sur la société.

Kay fut crédité pour avoir imaginé “un ordinateur qui soutient les efforts intellectuels des individus”, mais aussi pour avoir combiné l'Alto et les capacités du réseau Ethernet pour créer le premier LAN (Local Aera Network).

En février de la même année, Kay avec ses collègues du PARC Butler Lampson, Robert Taylor et Charles Thacker partagèrent le prix Charles Stark Draper pour leur développement qui ont aboutit au PC en réseau.

En octobre, il reçut la prestigieuse distinction de l'ACM (Association of Computing Machinery), le prix Turing 2003, pour avoir dirigé l'équipe qui inventa Smalltalk.

12.2. Citations

Alan Kay est aussi connu pour ses citations. Que le lecteur me pardonne si la traduction ne retranscris pas fidèlement le propos.
« Le meilleur moyen de prédire le futur est de l'inventer. »
« Un nouveau point de vue vaut bien 80 points de QI. »
« Faites-le marcher, faites-le correct, rapide et bon marché. »
« Java et C++ vous rappellent combien les idées nouvelles sont comme les anciennes. Java est la chose la plus embarrassante qui ait touchée l'informatique depuis MS-DOS. »
« J'ai inventé le terme Orienté-Objet et je peux vous assurer que je n'avais pas le C++ en tête. »
A propos du langage Perl : « ...C'est une culture pop. Comme un disque commercial qui serait un tube pour adolescents sans goût particulier pour la musique. Je pense qu'une grande part du succès de certains langages de programmation provient d'un besoin expéditif de boucher le trou. Perl est un autre exemple du remplissage d'un bref besoin à court terme, qui finira par devenir un vrai problème dans le long terme. »
A propos de la liaison tardive[11] : « Si vous utilisez des langages à liaison statique comme le font la plupart des gens, plutôt que des langages à liaison dynamique, alors vous serez coincé par des choses que vous avez déjà faites. »

Pour contrebalancer ses points de vues (voire carrément pour un démontage en règles des citations jugées douteuses) et pour d'autres citations de son cru, voir [11].

12.3. Le futur selon Alan

```
10 When's that?
20 RSN.
30 When's that?
40 Real soon now.
50 [GOTO 10]
```

Parmi les interviews de Kay dans des journaux ou sur des groupes de discussion [4, 9, 10] j'ai relevé que ses interlocuteurs essaient bien souvent connaître son avis sur l'informatique de demain.

« Il faut créer un nouveau champ de recherche, comparable à ce qui se faisait dans les années 60, centré sur les personnes. Le premier mot dans l'expression “PC” est “personne”. Il nous faut revenir à une vision humaine des choses, en donnant naissance à de nouveaux “amplificateurs humains” qu'ont été les pionniers de l'informatique il y a quelques décennies ».

Dans ses articles récents, Kay rappelle régulièrement que “la révolution informatique” n'a pas encore eu lieu. Ce point de vue recoupe celui de Doug Englebart, selon qui l'ordinateur va servir à augmenter l'intellect humain collectif.

En 1993, les travaux de Kay ont été remis d'actualités avec le développement de Playground, un environnement de programmation pour enfants des classes élémentaires, mené par le groupe Vivarium d'Apple.

Plus récemment, dans l'ouvrage de Mark Guzdial [2], Kay déclare sa confiance dans les systèmes multi-agents :

« Les capacités des ordinateurs à construire des modèles devraient permettre la construction de processus semblables à ceux de la pensée et aider les concepteurs à fabriquer des “agents” flexibles. Ces agents auront leur propre buts, conféreront sur des stratégies (en demandant des questions aux utilisateurs mais aussi en répondant à leur requêtes) et, en raisonnant, de fabriquer des objectifs de façon autonome. »

11. Quand les appels polymorphes – ou tout autre opération de liaison – ne sont pas résolues avant exécution. Similaire si ce n'est identique à la liaison dynamique (un appel de méthode sur un objet qui est sélectionné en concordance avec le type dynamique de l'objet. Fonctions virtuelles en C++, appels normaux de méthodes dans la plupart des autres langages).

13. CONCLUSION

L'oeuvre du Professeur Kay continue et poursuit une critique des technologies existantes afin de proposer des scénarios d'inspirations pour des technologies nouvelles.

Les conditions ont beaucoup changées depuis ces débuts et les orientations pédagogiques des premiers cours d'informatique doivent continuellement penser les bases fondamentales, concrètes, pratiques et ludiques sans rien négliger (comme estimer d'office que chacun possède un ordinateur à la maison). Le hardware préexiste, très largement répandu. Une importante séparation apparaît entre logiciel et matériel et des standards se sont imposés. L'informatique en tant que technique a mûri et n'est plus seulement un ensemble de recettes et de savoir-faire acquis par l'expérience, mais un savoir digne de ce nom et théorisable, qu'il convient d'accepter en tant qu'enseignement dans les écoles, pas seulement pour attester d'une reconnaissace en tant que science – ce qui ne contribue en rien à favoriser le projet éducatif, mais pour mener une réflexion qui porte davantage sur la situation de l'enseignement secondaire et moins sur celle de l'enseignement supérieur. Des questions du même ordre que l'utilité et l'usage de la calculatrice au bac, ou les résultats obtenus par un enseignement fondé sur des valeurs de coopérations – tel que nous y incite internet – à contrario d'un enseignement basé sur le critère unique de la compétition.

L'approche, encore éminemment constructiviste et expérimentale, doit parvenir à élargir le spectre quant aux usages de la programmation. Si on examine l'enseignement supérieur public, on est d'accord pour dire que les épreuves algorithmiques gardent toute leur importance. Bien entendu, offrant une représentation mieux organisée de l'information, la méthode orientée objet facilite le traitement, la manipulation et la structure de l'information, ce qui a grande échelle, continue de jouer un rôle architectonique. Mais il apparaît que l'on a insisté bien lourdement sur les aspects dits théoriques de la méthode, lors de cours magistraux qui n'ont plus vraiment lieu d'être. Il ne s'agit pas de les supprimer mais 1/ de les rendre intéressants et 2/ de réflechir aux méthodes utilisées pour former des informaticiens comme on formerait des pilotes et non des scientifiques purs ou des techniciens spécialisés.

La discussion sur l'enseignement de l'informatique dans les écoles (et sur internet) reste un débat ouvert, à commencer par la question de la mise à disposition d'un ordinateur pour chacun. Certains ont optés pour le financement d'un ordinateur portable par élève, d'autres ont choisi des tablettes. La majorité des établissements sont équipés d'Environnements Numériques de Travail (ENT) pour faciliter la gestion et la centralisation des cours. Puisqu'il s'agit de décisions régionales et publiques, elles font l'objet de programmes politiques qui devraient être éxaminés avec soins par des commités d'expert reconnus, pour éviter les dépenses inutiles et les apories du système.

L'autre grand chantier est la création d'outils informatiques à vocation éducative et créative. Si je connais des instituteurs qui sont d'ores et déjà prêts à rediscuter les méthodes traditionelles, les institutions, elles, tardent à réaliser la transition. Cet apprentissage est resté très en deçà de qu'il aurait du être, malgré que tous les moyens, tant humains que techniques soient à notre disposition. Les établissements sont équipés mais les cours de Pascal, en options, sont marginalisés. Et pourtant, bon nombre d'élèves font d'excellents programmes sur leur calculatrice (construites en séries, vendues 70€). Alors à quoi servent-elles ?

Bibliographie d'Alan Kay

[Kay68] Kay, A., Flex : a flexible extensible language. M.S. thesis University of Utah, mai 1968.

[Kay69] The reactive engine. PhD thesis, Université de l'Utah, septembre 1969.

[Kay70] Ramblings towards a KiddiKomp, *Stanford AI Project Lab Notebook*, novembre 1970.

[Kay71] Display transducers, *Pendery Papers for Parc Planning Purpose*, Xerox PARC, juin 1971.

[Kay71a] Draft design for miniCOM, *PARC Lab Book Xerox*, août 1971.

[Kay71b] Computer Structures–Past Present and Future, Panel paper, *in Proceedings of the FJCC* Vol. 39, novembre 1971.

[Kay72] MiniCOM proposal, *PARC Lab Book Xerox*, mai 1972.

[Kay72a] Learning research group 3 year plan. Xerox PARC juillet 1972.

[Kay72b] A personnal computer for children of all ages, *Proceedings of ACM National Conference*, Boston, août 1972.

[Kay72c] A dynamic medium for creative thought *Proceedings of the National Council of English Conference*, Minneapolis, novembre 1972.

[Kay72d] Smalltalk Blue Book, Fall 1972.

[Kay76] Kay, Alan & Goldberg, Adele. SmallTalk-72 Instruction Manual. Xerox PARC, 1976. Technical Report SSL-76-6, mai 1976.

[Kay77] Microelectronics and the personal computer, Scientific American, (pp. 125-136) septembre 1977.

[Kay77a] & Goldberg Adele., Personal dynamic media. IEEE Computer. Vol 10, (pp.31-42), mars 1977. Réimprimé dans *A History of Personnal Workstations*, Academic Press, 1988.

[Kay79] Programming your own computer, Science Year 1979, World Book Encyclopedia, 1979.

[Kay84] Computer software, Scientific American, septembre 1984.

[Kay90] User interface : a personal view, *in TheArt of Human-Computer Interface Design*, ed. Brenda Laurel, Addison-Wesley Publishing Co.,1990, (pp. 191-207) ISBN 0 20151797 3.

[Kay91] Computers, networks, and learning, Scientific American, Vol. 265, Nb. 3, (pp. 138-l48), septembre 1991.

[Kay91a] Computers, Networks and Education. In : Scientific American, Sept. 1991, p. 148

[Kay92] The Early History of Smalltalk. HOPL-II/4/93/MA, USA

[Kay95] Powerful Ideas Need Love Too ! *Written remarks to a Joint Hearing of the Science Committee and the Economic and Educational and Opportunites Committee*, Oct. 15, 1995

[Kay96] Revealing the elephant : The use and misuse of computers in education. Sequence. Vol. 31 No. 4 (Jul./Aug.) p. 22-28, 1996.

[Kay97] Kay A., et Goldberg A. « Personal Dynamic Media », The Newmedia Reader 1997 chap26. Publication originale dans Computer 10(3) :31-41.

[Kay97a] The Computer Revolution Hasn't Happened Yet. *Keynote address at the ACM SIGPLAN Conference on Object Oriented Programming Systems, Languages, and Applications.*

Références

[1] The Book & The Computer, une interview d'Alan Kay, `http://www.honco.net/os/kay.html`.

[2] Guzdial, M., Rose, K., Squeak : Open Personal Computing and Multimedia. Prentice Hall, 2001.

[3] Discussion entre Alan Kay et Danny Hillis, Wired Magazine numéro 2, janvier 1994.

[4] Educom Review, Interview d'Alan Kay, volume 34 numéro 2 1999.

[5] Human-Computer Interaction Seminar 2003, archives vidéos de l'Université de Stanford.

[6] Intervention d'Alan Kay au symposium de Sony CSL, octobre 2004. `http://www.internetactu.net/?p=5635`

[7] L'ordinateur Individuel, n°108, novembre 88.

[8] Licklider J.C.R., Man Computer Symbiosis, IRE(IEEE) *Transactions on Human Factors in Electronics*, volume HFE-1, pages 4-11, mars 1960.

[9] Papert, S., Teaching children thinking, Massachusets Institute of Technology,

[10] Portland Pattern Repository's Wiki, `http://c2.com/cgi/wiki?AlanKay`.

[11] The programming language weblog `http://lambda-the-ultimate.org/node/view/531`.

[12] Shklyar D., L'histoire du rendu 3D, `http://3DFV.com`.

[13] Alan Kay on messaging, `squeak@cs.uiuc.edu`.