The Computer Revolution Hasn't Happened Yet

Grand Challenge: Make It Happen In The Best Possible Way

by Alan Kay

I think most of the Grand Challenges in Computing lie ahead. For example: three Grand Challenges would be to do the next phases and complete the intertwined destinies of:

- Internet
- Personal Computing
- Teaching real Computer Literacy to all.

A simple way to think of these Grand Challenges is that they are a call for a second wave of the ARPA/ONR/ PARC processes--Licklider's Dream--that existed in the 60s and early 70s. A related and more fundamental area that is ready for a major advance is:

- "Dynamic Math" For an Entire System

Internet

On the one hand, the Internet works extremely well because of its quasibiological architecture and simple messaging structure, mostly for isolated noncoordinated messaging. But one of the destinies of the Internet and the personal computers on it is to provide "Engelbart's Dream" of many groups in close collaboration, much of it real-time and immersive. Since the number of groups that can be made from N people is the order of 2^N, no solution that requires special servers will scale. A really good solution would be peer-peer and distributed, in which each group member's personal computer can furnish its needed part of the simulation, rendering, replication, fault tolerant, and resynchronization processes which could do their job with DSL rates and above.

Personal Computing

If we really mean "personal computing" as "real persons doing computing," then the commercialization of the personal computer ideas of the 70s has really missed the boat. Beyond the simple ideas of using the computer as a metamedium to simulate old media and create media that can only exist on a computer lies a more profound idea: that real computer literacy is learning how to shape and understand computer stuff itself. This is a kind of "architectural process math" that has many new powerful properties on the one hand, yet can be formulated in a way that children can learn it (as they learn how to read and write). Since we want to share many of our ideas, this destiny of personal computing intersects very strongly with the Internet and "Engelbart's Dream" of augmenting group intelligence.

Teaching Real Computer Literacy to All

The completion of the Internet and Personal Computing has to go beyond invention and systems building to helping an entire generation of people to learn the powerful ideas and techniques. Though there are many routes for this Grand Challenge, the one that needs to be most supported is to be able to teach these ideas to all via the Internet and Personal Computing. There are several Grand Challenges here. Let me single out the old unrealized dream of a flexibly competent computer tutor for subject matter that is best taught by helping the learner construct their own knowledge.

• "Sustained Relationship" Programming

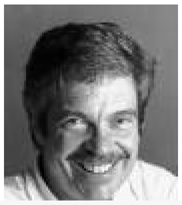
Most traditional "sequential procedural style simple state space programming," even within dynamic objects, is too fragile and verbose to scale well into the new computing environments. Instead, sustainable "dynamic relationships" must be added to dynamic objects in order to allow complex combinations of them to be more simply stated and controlled. This kind of programming has to be extended to the scripting worlds of children and nonexpert adults.

• Adaptable User Interfaces and Knowledge-That-Teaches

Most interfaces today present the very same "face" to a vast variety of users, from children to adults, from novices to experts. Most pieces of knowledge on the Internet have exactly the same form regardless of who is trying to understand it. The future demands that adaptable learning be the center of all knowledge presentations, including that of the user interface itself. The user interfaces have to be able to create a fairly accurate model of each user and use that model to modify the presentation of its knowledge structures, including finding other humans on the net that might be able to help. This area constitutes new frontiers in adaptable systems, models of humans and human knowledge, etc.

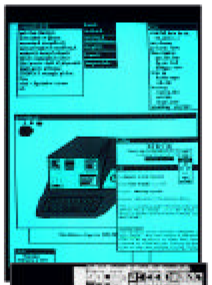
Dynamic Math for An Entire System

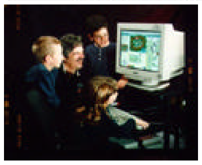
WIndows XP is "only an operating system" but has more than 60 million lines of code. Here is an unfair comparison: Squeak is a late-bound dynamic OOP personal computing system in which its OS, UI, development environments, and its many applications (including email, DTP, presentation, network server, sound and movie lab, etc.) require only about 220,000 lines of code in total. This is because there is some "dynamic math" as part of Squeak's metasystem and system that allows some of the important relationships to be expressed more directly, simply and powerfully than in most systems. There is so much more that could be done that what remains constitutes an exciting edge of the art research project on its own. For example, the Squeak team is pretty sure that the current 220,000 lines of code could be reduced to no more than 40,000, and there is a good argument that the current system should take no more than 10,000. Such a re-rendering of an entire system would be a tremendous accomplishment in higher level programming and would greatly increase a programmer's leverage on new problems.

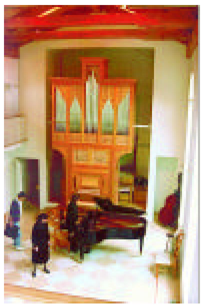


The FLEX Machine

The Dynabook







Music Room

Alan Kay

Alan Kay is one of the earliest pioneers of personal computing, and his research continues today.

In 1966 he made a major contribution to the invention of "object-oriented programming" — the main kind of programming today. He also contributed to the invention of 3D graphics.

In 1967-9 he and Ed Cheadle invented the FLEX Machine, a very early modern desktop machine they called a "personal computer". It had a display, a pointing and drawing tablet, a multiple window graphical user interface, and the first object oriented operating system.

In 1968, after a visit to Seymour Papert's early LOGO work with children, he designed "a personal computer for children of all ages" — the Dynabook — in the form of a very portable notebook, with a flat-screen, stylus, wireless network, and local storage. During this time he also contributed to the development of the ARPAnet.

At Xerox PARC in the 70s he invented Smalltalk, which was the first complete dynamic object oriented language, development, and operating system. It is still the leading such system today, especially in the free open-source version called Squeak.

At PARC he was one of the instigators for the first bitmap displays (that all computers use today), and the main inventor of the now ubiquitous overlapping windows, icons, point-click-and-drag user interface.

He was head of one of several groups at PARC that together created much of modern computing, including: the overlapping windows GUI, WYSIWYG word processing & desktop publishing, object-oriented OS, music synthesis, painting and animation, laser printing, ethernet, client-server (and peer-peer) networking, and parts of the Internet.

Most of his contributions from 1968 onwards have been the result of trying to invent and test better learning environments, mainly for children.

He has been a Xerox Fellow, Chief Scientist of Atari, Apple Fellow, Disney Fellow, and is now President of Viewpoints Research Institute.

Formal Education: BA in Mathematics and Molecular Biology with minor concentrations in English and Anthropology from the University of Colorado, 1966. MS and PhD in Computer Science (both with distinction) from the University of Utah, 1968 and 1969.

He started his career as a professional jazz guitarist. Much of his show business experience combined music and theatrical production. Today he is an avid amateur classical pipe organist.

Honors include: J-D Warnier Prix d'Informatique, ACM Software Systems Award, 2001 Computers & Communication Foundation Prize, 2002 McLuhan Lecture, etc.

He has been elected a Fellow of the American Academy of Arts and Sciences, the National Academy of Engineering, and the Royal Society of Arts.