Gopherspace. Gopher Archies enabled searching for specific stored or archived files. Similar to FTP Archies, Gopher found and listed all locations of a desired file. However, instead of just giving computer addresses for these sites, Gopher provided menu selections for retrieval. Another tool, Veronica, allowed searches for specific strings within Gopher menus.

Despite the rapid acceptance and widespread use of Gopher, invention of the World Wide Web and subsequent browser technologies incorporating text, graphics, audio, and video contributed to Gopher falling into disuse. By the late 1990s, few active Gopher sites remained on the Internet. Popular browsers such as **Netscape**'s Navigator and **Microsoft**'s Internet Explorer added features that enabled Gopher servers to be accessed without needing additional software. However, Gopher clients are unable to access Web sites.

FURTHER READING

Gopher Development Team. "The Internet Gopher: An Information Sheet." *Electronic Networking*, Vol. 2, No. 1, 1992, pp. 69–71.

Moschovitis, Christos J. P., Hilary Poole, Tami Schuyler, and Theresa Senft. *History of the Internet: A Chronology, 1843 to the Present.* Santa Barbara, Calif.: ABC-CLIO, 1999.

—Roger McHaney

Gordon Bell Prizes for Parallelism

The Gordon Bell Prizes for Parallelism were created in 1987 to recognize advances in high-performance scientific computing through parallelism. The prizes are awarded at the annual IEEE/ACM Supercomputing Conference and administered by a special group of the program committee. Bell, the first director for computing at the National Science Foundation, wanted to recognize and chronicle the progress of application programmers of newly created and evolving parallel computers.

In the early 1980s, the U.S. Defense Department's Advanced Research Projects Agency (**ARPA**) funded the Strategic Computing Initiative (SCI), aimed at achieving a teraops (1 million million operations per second) of computing power by the mid-1990s. Similarly, in 1994, the Department of Energy created the Advanced Strategic Computing Initiative (ASCI) to develop a computation facility aimed at a petaops (1 billion million operations per second).

Three prizes are awarded for actual application programs: total performance measured in operations per second; degree of parallelism (the number of independent operations being carried out simultaneously); and cost-effectiveness (cost per operation per second).

The 1988 prize was awarded to three researchers at the Sandia National Laboratory using an Ncube multicomputer with 1024 computing nodes for running three different applications. This demonstrated that parallelism was feasible. Furthermore, even greater performance could be obtained with larger computing nodes. The same year, a researcher at the National Center for Atmospheric Research (NCAR) introduced a global weather model program that used all four processors of the Cray XMP vector **supercomputer**.

With the exception of a prize awarded using the **Thinking Machines Corporation**'s CM2, a single-instruction-stream, multiple-parallel-data-stream computer, all winners have used clusters of computers connected by high-speed switching networks. In 1999, over 1 teraflops was reached on each of three machines with 5000 to 10,000 processors at Los Alamos National Laboratory (**SGI**), Lawrence Livermore National Laboratory (**IBM**), and Sandia National Laboratory (**Intel**).

FURTHER READING

Kumar, Vipin, et al. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Redwood City, Calif.: Benjamin/Cummings, 1994.

Hwang, Kai, and Zhiwhei Xu. *Scalable Parallel Computing: Technology, Architecture, Programming.* Boston: WCB/McGraw-Hill, 1998.

—Gordon Bell

"GO TO Statement Considered Harmful"

by Edsger W. Dijkstra

In 1968 **Edsger W. Dijkstra** (1930–), a professor at the Technological University of Eindhoven in

the Netherlands, wrote a letter to the editor of *Communications of the ACM* that would become a classic statement of programming philosophy. His letter, published under the title "GO TO Statement Considered Harmful," forcefully made the case for the abolition of the GO TO statement from all high-level languages. The GO TO statement is used when the programmer wants the code to continue executing at another line of code somewhere else in the program.

According to Dijkstra, since the GO TO statement is so flexible and allows any kind of branch in the code, it is an "invitation to make a mess of one's program." Sometimes programs written with many GO TOs and an extraordinarily tangled structure are referred to as *spaghetti code*. Dijkstra noted: "For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of GO TO statements in the programs they produce."

In a sense, Dijkstra's main argument was a psychological observation about human cognitive abilities. In his view, brains are geared toward the analysis and recognition of static relationships and not of dynamic processes. In other words, the programmer can better understand his creations when he or she does not have to mentally follow the evolution of the execution path. A more declarative style of programming would therefore make programming less error-prone.

Out of this and similar discussions about entangled code emerged the structured programming movement of the early 1970s. Languages such as **Fortran** and **ALGOL**, which were very popular at the time, made heavy use of the GO TO statement. Structured programs should avoid it; only three kinds of high-level structures should be present: sequencing of instructions (meaning that elementary instructions can be written one after the other), case selection (a condition picks up one of two or more alternatives), and iteration (an action is performed repetitively as long as a condition is true, e.g., the WHILE statement in some programming languages). Many programming languages designed after the 1970s eliminated GO TO from the instruction set, so that nowadays the programmer is forced to write structured code.

FURTHER READING

Dijkstra, Edsger W. "GO TO Statement Considered Harmful." *Communications of the ACM*, Vol. 11, No. 3, Mar. 1968, pp. 147–148.

Dijkstra, Edsger W., and Edward W. Dijkstra. *A Discipline of Programming*. Upper Saddle River, N.J.: Prentice Hall, 1976.

—Raúl Rojas

Graphical User Interface

A graphical user interface (GUI) is a method of displaying text and graphics on a computer screen. GUIs are considered to be more *user friendly* than a text-based **interface** because picture icons displayed on the screen help users develop a mental model of the operation of the computer. This style of computer interaction is called *direct manipulation* because it replaces complex command language syntax with direct manipulation of visual objects.

The first GUI was developed in the 1970s by **Alan Kay** (1940–) and his team at **Xerox's Palo Alto Research Center**. Kay's model for interface design is based on the work of Jerome Bruner (1915–), who suggested that there are three ways in which human beings translate experience into a model of the world: enactive or learning through action, iconic or visual learning, and symbolic or linguistic representation. Kay combined these learning mentalities together into a model called "Doing with Images Makes Symbols." The slogan refers to the idea that people should start learning how to use a computer with concrete images that are represented on the computer screen and then be carried into the more abstract level of programming.

However, the **Macintosh** and **Windows** implementation of the Desktop GUI were not meant to be easily programmable machines, as in Kay's original design. In contrast, the goal of Macintosh and Windows is to make **personal computers** easy to operate without learning complicated commands. In 1982, **Microsoft** began developing Windows, but it was not until Windows 3.0 and Windows 95 that GUI technology fully replaced the command-line interfaces found on Intel-based machines.