Werkzeuge zum Gestalten interaktiver PC-Programme für den Unterricht

Diss. ETH Nr. 12078

Werkzeuge zum Gestalten interaktiver PC-Programme für den Unterricht

Abhandlung
zur Erlangung des Titels
Doktor der Technischen Wissenschaften
der
Eidgenössischen Technischen Hochschule Zürich

vorgelegt von
Hans Ulrich Gerber
Dipl. El.-Ing. ETH
geboren am 17. Mai 1953
von Langnau BE

Angenommen auf Antrag von
Prof. Dr. Walter Schaufelberger, Referent
PD Dr. Christian Hafner, Korreferent

1997

Zusammenfassung

Arbeitsplatzrechner können als nützliche Werkzeuge im Unterricht oder beim Selbststudium dienen. Besonders gut eignen sie sich als Vorführ- und Experimentiergeräte. Simulationsprogramme bilden Vorgänge aus Natur und Technik nach und veranschaulichen Phänomene, die uns sonst verborgen blieben. Bei Experimentierprogrammen kann der Anwender ins Geschehen eingreifen, Zusammenhänge zwischen Ursachen und Wirkungen entdecken.

Wer entwickelt die Programme? Ausbildner und Fachpersonen des jeweiligen Gebietes wären dazu prädestiniert, sie kennen ihren Unterrichtsstoff und den Abnehmerkreis. Fachexperten sind aber meist nicht zugleich Informatiker oder Computerbastler. Viele kennen zwar eine höhere Programmiersprache, aber haben keine Zeit, alle Strömungen auf dem Kleincomputermarkt zu verfolgen, Programmierwerkzeuge auszuprobieren und Spreu von Weizen zu trennen. Einige von ihnen haben über Jahre hinweg Programmteile entwickelt, die sich bewährt haben, aber auf neuen Maschinen und Betriebssystemen nicht mehr laufen.

Unsere Fachexperten und Anwendungsprogrammierer brauchen Werkzeuge, um die Kluft zwischen ihren erprobten Programmen und modernen Betriebssystemen samt grafischen Benützungsoberflächen zu überwinden. Gleich wie Benützer von Computerprogrammen das Recht auf eine verständliche, an ihre Bedürfnisse angepasste Bedienoberfläche haben, sollen Anwendungsprogrammierer bei ihren Programmierwerkzeugen mit einer leicht erfassbaren Schnittstelle rechnen dürfen. Programmierer sind dann am produktivsten, wenn sie über massgeschneiderte Werkzeuge verfügen, die auf ihren Problemkreis und ihre Kenntnisse zugeschnitten sind. Käufliche Programmbibliotheken können aber oft kein einfaches Benützungsmodell vermitteln, weil sie einem möglichst grossen Abnehmerkreis mit verschiedensten Ansprüchen dienen müssen.

Der Schreibende hat in der vorliegenden Arbeit die Rolle des Werkzeugmachers gespielt. Er hat Software-Instrumente gefertigt, die sich nahtlos in die Vorstellungswelt unserer Anwendungsprogrammierer einfügen, zugleich den Wert ihrer Programme erhalten helfen und sie von allzu flüchtigen Strömungen des Marktes abschirmen. Es handelt sich um

- eine einfache Programmierschnittstelle zu grafischen Benützungsoberflächen, die sich von klassischen, prozeduralen Programmiersprachen wie FORTRAN oder Modula-2 aus leicht benützen lässt. Damit können Anwendungsprogrammierer innerhalb ihrer vertrauten Umgebungen Programme gestalten, die sich an der Oberfläche nicht von kommerziellen Programmen unterscheiden. Unsere Programmierer können weiterhin diejenigen Programmiersprachen einsetzen, in denen sie sich am gewandtesten ausdrücken.
- Unterstützung zum Anpassen älterer FORTRAN-Programme ("legacy applications") an grafische Benützungsoberflächen.
- Hilfsmittel zum Erzeugen elektronischer Bücher, die Programmteile und Erläuterungen vereinen. Die neusten Entwicklungen erlauben uns, die gleichen Dokumente und Programme auf verschiedensten Rechnern anzubieten. Mit einem Anwendungsbeispiel aus dem Gebiet des Genetischen Programmierens zeigen wir, dass sich die Programmiersprache Java für Neuentwicklungen interaktiver Unterrichtsprogramme besonders gut eignet.

Abstract

Personal computers are useful tools for teaching and learning. They serve well as demonstration aids and experimentation tools. Simulation programs imitate natural and technical processes. They visualize phenomena that otherwise would remain hidden from our senses. In experimentation programs or virtual laboratories, the user interacts with the simulated processes. By exploring, he may discover relationships between cause and effect.

Who should develop these programs? Teachers and experts of the different fields are the prime candidates; they know their subjects and understand their target audience. Unfortunately, those who are experts in their fields are generally not computer scientists or software geeks at the same time. Many of them know some classic high-level programming language, but they do not have the spare time to keep track of the latest developments on the personal computer market, to test programming tools and to separate the useful from the gimmicks. Some of them have developed program modules over the years that have proven useful and reliable, but which no longer run on new machines or operating systems.

Our experts and application programmers need tools to help them bridge the gap between their proven programs and modern operating environments with graphical user interfaces. Just as users of computer programs may rightfully expect understandable interfaces, application programmers have a right for clean and simple interfaces to their tools as well. Programmers are most productive if they can use tools that are tailored to their problems and their knowledge. Commercial toolkits often do not exhibit a simple programming model, since they have to serve a wide range of users with different needs.

In this report, the author plays the role of software toolsmith. He has crafted software instruments which seamlessly fit into the world of our application programmers. The tools help protect the value of their programs by shielding them from too many rapid changes of the market. They are

- a simple program library for commercial graphical user interfaces. Application programmers are not locked into an unwieldy toolkit from a single vendor, instead they can continue to use their familiar classic procedural languages, be it C, FORTRAN or Modula-2. Programs built with this tool look and behave like commercial applications.
- a software library to adapt existing FORTRAN programs ("legacy applications") to a graphical user interface.
- tools to create electronic books, combining program and documentation parts. The latest developments make it possible to offer such compound documents across politicial and technical boundaries. An application example from the field of Genetic Programming shows the potential of the programming language Java to produce interactive simulation programs that can easily be published for a world-wide audience.