Document Nbr: CSI426/kaypro-PD01.02
Date: Oct 8, 1999
Copy Nbr: ____

A JAVA IMPLEMENTATION OF A KAYPRO II MICROCOMPUTER SYSTEM

A PROJECT FOR COMPUTER SCIENCE 426 (CSI426)

METROPOLITON STATE COLLEGE, DENVER, COLORADO

- **Brandon Buist**
- **Joe Diethorn**
- **Jim Gilmer**
- **Shannon Steinmetz**
- **Hung To**

1. Abstract

1.1 The Class

CSI 426 was the second class in series of senior project classes. The series is a practical synthesis of all courses taken at Metro State. The course was presented in a "real life" context. In many ways, CSI 426 was an independent study. The classroom experience was comprised of brainstorming sessions, and discussion of independent issues.

1.2 Assignment

The task assigned was to produce an emulation of an older computer system. The class was given a list of computers to emulate. Among the list was the Kaypro II. The assignment was to completely emulate the computer system. The programming language was Java. The emulation was to be delivered over the Internet within a conventional browser. Most of the students had little or no experience in Java. They were asked to come up to speed in Java over the summer (it was a fall class). There was no instruction in Java. That was left up to the student.

The emulation was to be completed in a single semester. The group was to use good design practices. This included full documentation. Documentation deliverables were:

- User manual
- Requirements documentation
- Design documentation
- Test documentation

The project was to be implemented as designed. Periodic requirement and design reviews were held. Due dates were assigned for each document. The final deliverable was to be a working demonstration of the assigned computer.

1.3 Groups

The class was divided up into groups of five. Our group was diverse, in that it was made of individuals of varied skills and backgrounds. It was up to the group to organize itself.

1.4 The Kaypro II

The Kaypro II was one of the first luggable computers. It was also one of the first computer "clones." The computer was self contained, and included a screen, keyboard, storage devices, CPU and peripherals.

1.5 Group Activity

It was up individual groups to research their assigned machine. Our group utilized the Internet to gather most of the information. We obtained complete schematic diagrams, and made some very valuable contacts.

Although we never formally organized ourselves, our group members ended up being organized into the following areas:

- Project manager
- Technical lead
- Spokesman
- Maintenance programmer
- GUI programmer

1.6 Accomplishments

The project was delivered within the given timeframe. All class objectives were met. The emulation literally "fell together." The project proved that proper software design techniques really do pay off. The class was extremely difficult, and the project was very ambitious. It pushed many team members to their limit.

The real payoff was at the end of the project. After the final presentation the group met at a nearby restaurant. They closed the restaurant down. The conversation spilled into the parking lot. As it turns out the synergy of the group was such that nobody wanted to let go of the moment. Perhaps the greatest lesson learned was that of teambuilding.

2. Simplified Project Requirements (Specific)

The assignment was to produce a working emulation of a Kaypro II computer. The Kaypro II was designed in the 1970's. It was one of the first computer systems to be self-contained. This put great demands on the design team. Every inch of the computer needed to be researched, documented, and implemented in emulation.

The essence of the assignment was the decomposition and recomposition of a computer system. The Kaypro II was reproduced so faithfully that even the original font, characters and green screen were duplicated. To produce such an accurate emulation, it was necessary to examine and duplicate the entire Kaypro II architecture at a hardware level.

The Kaypro II contains internal software. This software is located on internal ROMs. To emulate the Kaypro II from a hardware level, it was necessary to extract data from the Kaypro II's ROM chips. To do this, the ROM chips were extracted from a working Kaypro II computer and read with an EPROM reader. The binary data from the EPROM reader was converted into static tables and inserted directly into the emulator software. The result is that the emulator runs the actual Kaypro II internal ROM code. The same was true of the floppy disks. Because the Kaypro II reads boot data from it's internal ROM and floppy disks, it was necessary to extract binary images from floppy disks. This meant reading individual tracks and sectors from actual Kaypro II diskettes. To do this, the group located software (via the Internet) that allowed Kaypro II diskettes to be read via a PC. Unfortunately, modern disk drives were incompatible with this software. The group obtained an old 486 computer with a 360K-diskette drive. The Kaypro II diskette images were extracted, converted into static tables and inserted directly into the emulation software. The tables were used by the floppy disk controller emulation routines.

Integrated circuits were analyzed and reproduced in hardware. System RAM and ROM were reproduced down to clock cycle reads and writes. The group was fortunate to obtain schematic diagrams of the Kaypro II computer system. The entire architecture was analyzed down to the chip level. Data books for individual chips were obtained. Each major chip was analyzed and emulated in software. In addition, discrete circuitry was analyzed and simulated. For example: the Kaypro II screen is a bank-switched, memory mapped architecture. The Kaypro II keyboard shares the system serial port, and the characters that are displayed are generated via discrete circuitry (as opposed to modern character generator chips, etc). This discrete circuitry was analyzed and duplicated in software.

All major portions of Kaypro II hardware were emulated in software. A sampling of Kaypro II applications were extracted byte-by-byte from native media. This software was transferred to the Kaypro II emulation. The software that runs on the emulator runs without modification. The software "thinks" that it is running on an actual Kaypro II. This means that the emulation is capable of running a greater base of software.

2.1 Functional Requirements

- The emulation program shall emulate the Kaypro II model of the Kaypro product line
- The user shall view the emulated version just as they would the original
- The emulator shall support the same character set as the original. Limitations in keyboard may apply. It is acceptable to substitute functionality when necessary.

2.2 Technical Requirements

- The emulated RAM shall be 64K bytes
- The emulated ROM shall be 12K Bytes
- The emulated video RAM shall be 4K bytes
- The emulator shall support 2 virtual floppy disk drives
- The emulator shall support an 80 character per line by 24-line text screen with graphic extensions.

2.3 Platform Requirements

- The Kaypro II emulation shall be implemented in Java.
- The Implementation shall be pure Java (e.g. no Microsoft extensions)
- The Kaypro II emulation shall be implemented as an applet.
- The Kaypro II emulation shall be made available via the World Wide Web.

2.4 Emulation Requirements

- Emulation shall be at the hardware level
- BIOS ROM shall be extracted from the original Kaypro II and inserted into the emulated version as executable code.

2.5 Peripheral Requirements

- Parallel printer output shall be supported
- Screen output shall resemble the original Kaypro II as closely as possible
- Keyboard input shall be supported. Keyboard input shall resemble the original Kaypro II as closely as possible
- The emulator shall support the emulation of the Z-80 processor
- The Kaypro II emulator shall not support serial port operation

2.6 Operating System Requirements

- CP/M 2.2 shall be supported
- The emulator shall support loading of the OS from floppy drive A
- The CP/M operating system shall actually be run at the software level via an obtained copy of the CP/M operating system

2.7 Port Requirements

- Parallel support shall be supported
- Serial ports used for keyboard control shall be supported
- Disk drive ports shall be supported

2.8 Bank selection

- The emulator shall support banking
- The emulator shall support memory-mapped video

2.9 Interface Requirements

- Keyboard input shall be made directly into the Java page. The user shall be able to type directly into the emulated Kaypro II.

2.10 Debug

- The developer shall be able to select (or deselect) a debug mode

3. Working The Assignment

3.1 Group involvement

Because the group was diverse, it became necessary to partition the project into pieces appropriate for each team member. Some team members had not come up to speed in Java. Others were highly aggressive, others were passive.

The team divided itself in such a way that each member utilized their own particular strength. Although each team member concentrated on their strength point, all were involved in each design decision. Decisions were not made in ignorance. Before some issues could be decided upon, the problem needed to be understood. Many a late-night was spent "schooling" each other in different parts of the system.

3.2 Important Decisions

The group needed to decide how to attack the problem. Two methods were possible:

1) Organize around a central technical leader. This person would do most of the work. The rest of the group would provide support for this single individual. The advantage was a potentially quick development cycle. The disadvantage was that the overall group learning would be minimized.
2) Organize the group into "specialists." Each specialist would take one piece of the project. A piece could be technical, such as a CPU implementation, or it could be operational, such as a presenter or information gatherer. The advantage was a greater learning potential; while the disadvantage was a higher risk that one individual would not come through with their piece of the project.

The decision was made to go with option 2, mostly because of the learning potential. As it turns out option 2 was a winner for more than just learning.

By organizing into "specialists" the team was forced to depend on each other. Interdependence forced adherence to a formalized design methodology. The group learned that a well-documented project meant less confusion, and a more cohesive group. By relying on each other the group gained synergy. Group synergy has immeasurable benefits. Much of the benefits manifest themselves in work ethic. Team members don't want to let each other down. They are driven by the motivation of themselves and the team. When one is down, the others filled in the slack.

What is interesting is that this same situation applies to the computing industry. The team synergy is sometimes overlooked. At times documentation is seen as unnecessary, or wasteful. In fact, documentation is a form of communication between internal and external group members. Many corporations view such documentation as wasteful only because they fail to see the true purpose of the process. The same is true for many of today's programmers.

The CSI 426 project was an invaluable demonstration of a well-communicated process. The results were tangible. The evidence was a well-orchestrated project, and most importantly: working, well thought out, reliable software.

3.3 Research

The first step was researching the assigned computer. That involved searching the Internet for tidbits of information. Along the way, we made some really useful contacts. One source provided us with complete documentation and schematics. Still others helped bring the group up to speed on the operating system (CP/M). These same people were involved in testing the preliminary and finished product (see Software Beta Feedback, below).

3.4 Documentation

The documentation for this project was enormous. It filled three 3" binders.

The group didn't write documentation just to fill-up binders, however. The goal was effective communication. Every shred of information obtained was committed to paper. The result was a rather aggressive documentation set.

One benchmark of good documentation is how often it is used, and how often one finds the answer they are looking for. As the project progressed, the documentation set became the bible. Diligent updating of the documentation became imperative. 80%+ of the questions asked by individuals could typically be found in the documentation.

The software produced was actually derived from the documentation. It would have been possible to simply write the software and back-fill the requirements (a practice that all-to-often is exercised in industry). Instead, the group used this documentation to enhance their internal and external communication.

The real payoff came at implementation. The group had met weekly to create the design documents. Once all research was complete, the implementation was easy. The group decided to skip the next meeting and concentrate on writing code. The implementation was modeled from the documentation.

As it turned out, the documentation was very much like an instruction manual. Every programmatic function was described in the design document. The expected inputs, outputs and function were spelled out. Most ambiguities were ironed out long before implementation.

3.5 Implementation

The two weeks spent implementing was extremely productive. In the two weeks that the team did not meet, each member produced code as per the design. In addition, each unit tested their part of the project. The result was individual program modules that worked independently. The question was how they would work together.

Two weeks later the team met to join all the pieces. An intermediate system had been constructed via constant email communication. The latest pieces were brought together and a beta system built in less than two hours.

An amazing thing happened. Much to the surprise of the team, the system ran the first time! There were a few issues that were discovered while testing the integrated system, but those were minimal. Most dealt with performance and usability issues.

The team worked together to resolve the usability and performance issues. In the end, the team found time to enhance the original target product. When the system was presented, it ran flawlessly. Performance met or exceeded the original Kaypro II.

3.6 Interesting notes

The finished project ran so well, and was tested so rigorously that only a small number of bugs were discovered. After troubleshooting these problems it was discovered that the problems were actually in the original Kaypro II software.

Upon running the finished project a number of features were “discovered” in the emulation. Upon further inspection, it was found that these features resided in the original hardware as well.

The pride of a job well done was evident in the preparation of the final presentation. A problem was encountered divvying up the presentation. The issue was that each person wanted to present too much. Each person wanted to “be in the spotlight” with the project. When the project was presented, the entire team was closely bunched around the computer. I guess everyone wants to be associated with a winner.

3.7 Thanks

Special thanks to Dr. Howerton for running this class. He provided an atmosphere that closely mimicked real-life. The project was extremely difficult, yet one of the most rewarding. He will be missed at MSCD.

4. Software Beta Feedback

Perhaps one of the best ways to test a product is to solicit real-life users. That's what our group did. Many of the contacts made were via the CP/M users group (on the Internet).

We decided to publish a beta version of the project on the Internet. We created a web site, and published the URL to the CP/M users group. To our surprise, we received quite a bit of feedback. Typical comments are shown below.

…..

"Excellent job! I've been putting off upgrading to IE4.0 at home, but after playing with your emulator in some free time here at work, I think I just found a reason to upgrade! (However, I find it ironic that I need to upgrade to the latest technology in browsers to emulate something as old as my Kaypro II! <g>)

I own a Kaypro II and a Kaypro 4, but I haven't dug them out in years. From what I remember, your emulator is practically identical. Well done!!"

…..

…..

"I saw a message on comp.os.cpm describing your Kaypro II emulator (on-line yet!) and I have to say I am flabbergasted. (Talk about retrocomputing). I have two Kaypro II's (but upgraded to DSDD drives) and two Kaypro 10's (one the with original 10 mB drive, the other with a 20 mB drive) and really love the machines. Glad to see that someone else finds them of interest. Anyway, I played a game of Hunt the Wumpus on-line, and it was just like the real thing -- I am definitely impressed. Assuming this is, as I understand, a class project, I hope you all get A's, because it really is a fine piece of work."

…..

….

"I'm speechless. (More to the point, I hope my wife doesn't find this site. If she does, she'll want me to get rid of my Kaypro's, since from her point of view there's no reason why I can't play with an on-line emulated version as easily as a real one!)"

….

5. Online Access

Currently the Kaypro II emulation is located at: www.yoy.org/kaypro. The email address is: kaypro@yoy.org. Because Java is an emerging standard, the emulation works on only a narrow band of browsers. See the web site for details. Complete documentation and user guides are located in PDF format on the web site as well.

Plans to incorporate the Kaypro II emulator, and others, on the MSCD web site are in the works. Please contact the MSCD computer science department for further information.

The group is currently entertaining the idea of releasing the source code to the public domain. Comments or suggestions are greatly appreciated (use the email address above).