Computer Science

CREATING A VIRTUAL REALITY SIMULATOR OF THE ENIAC

Kristin Briney, Computer Science Department, DePauw University, IN 46135 (kbriney@depauw.edu)
Kevin Mercurio, Computer Science Department, Siena College, NY 12211 (kevin.mercurio@siena.edu)
Nate Nichols, Computer Science Department, DePauw University, IN 46135 (nnichols@depauw.edu)
Paul Raskin, Computer Science Department, DePauw University, IN 46135 (praskin@depauw.edu)
Douglas Harms(*), Computer Science Department, DePauw University, IN 46135 (dharms@depauw.edu)

The ENIAC (Electronic Numerical Integrator And Computer) was the world's first electronic computer. While modern computers are much smaller and more efficient, the ENIAC was pivotal in the development of computing as it is known today. Despite its antiquated nature, the ENIAC is still useful as a teaching tool for computer organization. Through our simulation, we hope to give users a general view of how early computers operated.

Aside from educational applications, our simulation is useful for historical purposes. Creating a working virtual simulation of the world's first electronic computer using a modern system will help people connect to the roots of contemporary computing.

Containing 18,000 vacuum tubes and weighing over 30 tons, the ENIAC was a comparative giant of the computer world and occupied a 30 by 60 foot room. It was programmed by connecting cables between units and setting individual dials on each unit. This process was the most time consuming operation on the ENIAC, requiring up to a week to complete. The ENIAC was a digital computer, though it was based on a decimal numbering system, rather than a binary one.

We coded the simulation on a Red Hat 9 machine, chosen because of its availability and standard programming software. The programming was done in C++ because of familiarity with the language. We used the Open Graphics Library (OpenGL) for graphical effects, along with the OpenGL Utility Toolkit (GLUT).

We researched and implemented several ENIAC units, including the Master Programmer unit, responsible for handling subroutines on the ENIAC; the Divider-Square Rooter, which calculated divisions and square roots; and the High-Speed Multiplier, used for multiplication. Features added to the Accumulator units consist of the ability to connect two units for double accuracy and a usable significant figures switch. Other components that were implemented include: cables, shifters, and deleters. Photographs of the original units helped to graphically implement them in the simulated room. We enhanced the graphical representation of the Accumulators as well as completely representing the Master Programmer.

In addition, we also added a user interface, allowing the user to actively program the ENIAC. Users can manipulate dials and cables in order to fully program the ENIAC machine. When a user approaches a unit, a crosshair is displayed on the screen which is controlled by the mouse. When a user points the mouse at a dial it becomes highlighted in red, and the user can issue an "increment" or "decrement" command to the dial, and see it change its position in real time. The user can also issue a "cable" command to two plugs, and a cable is inserted into the simulation with those plugs as its endpoints. Finally, the interface was developed to support the use of a 3D head-mounted display and a data glove to navigate the virtual world.

This simulator can be useful as an educational tool. Users donning the head mounted display (HMD) can view the ENIAC full scale; of course, users not fortunate enough to possess a HMD can view the simulation on a normal monitor. This graphical simulation is advantageous for grasping concepts of the inner architecture of computers. With the components of the computer on a large scale compared to modern hardware, individual parts of the system are clearly visible. This facilitates explaining the basics of hardware; for example, the concept of a "bus" is transformed from microscopic etches in silicon into a eight foot long tray with visible wires. Users can create a program, observe its execution, and modify it without ever leaving the simulation.

Acknowledgements: NSF REU grant numbers EIA-0242293 and EIA-0215834 supported this work
DePauw Science Research Fellows also supported this work