The Impact Variable Image Transformer

by Dave Anderson, former systems software engineer for Microtime, Inc.

Digital Video Effects machines (DVEs) provide creative transitions from one video source to another, rather than making an abrupt cut. DVE effects include things like fading, shrinking or dissolving frames, colored borders, wiping from one frame to another, tumbling and emitting trails. More complex effects might include exploding images, drop shadows, and bending or morphing images, depending on the system's capabilities. DVEs also allow multiple video sources to be positioned side by side, or on top of one another for picture-in-picture effects. Along with graphics creation workstations, DVEs are the workhorses of television advertisement production, and they're commonly used for creating industrial videos.. Every television studio and editing suite has at least one DVE.

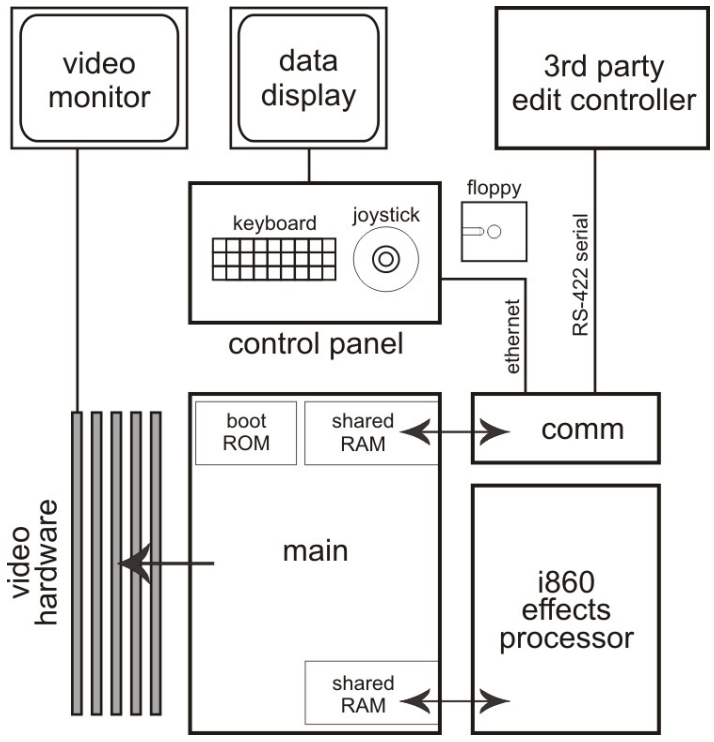
From 1989 through 1995, I worked as a software engineer at Microtime Inc., a television products company in Bloomfield, Connecticut. Being mostly an electronics manufacturer, they had a small software staff – three people. One of them had recently left, and I was hired to fill the vacancy, as they had a big project on the drawing board: the Impact "variable image transformer."

The Impact was a DVE on steroids. What made this system special was its ability to map live television onto 3D graphical "objects" – sets of polygons that could form cylinders, cubes, spheres, and panels that could be bent, warped and shattered, all under joystick control. When they started this project, Microtime was the only company in the world that could do this kind of effect live, in "real time," with no preparation whatsoever – a big factor for sportscasts and newscasts.

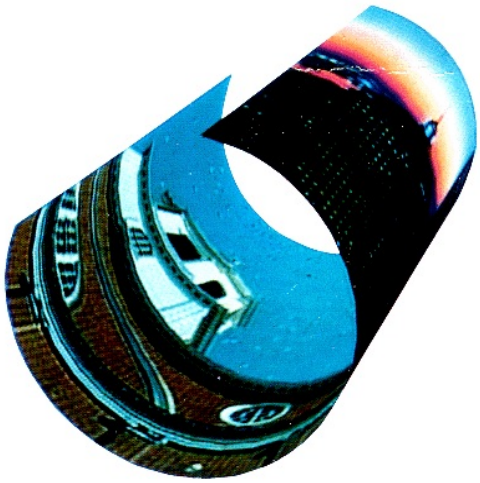
The Impact was a box about the size of a small refrigerator, with a dozen or so custom circuit boards inside. Up to three video feeds were routed into the box, and one video line came out with the processed picture, ready for tape recording or on-air broadcast. A separate control panel allowed the Impact to be run from a comfortable air-conditioned control room, while the main box of electronics (the "mainframe," as we called it) was located somewhere where the noisy fans wouldn't disturb anybody.

The senior software engineer was a man named Uri Thier, a very bright fellow who had previously programmed aircraft guidance systems for the Israeli air force. Uri had the software vision on this project, and he worked closely with the senior electronics engineers to develop the system to create animated polygons with live television pasted onto their surfaces.

My job was to develop the operating system for the Impact. The various effects, or "video objects" as we called them, were C programs downloaded from a floppy disk into a separate CPU. For each frame of video generated, the operating system fed the object program a set of parameters indicating the object's position, and the object program calculated where the corners of each polygon were located within the television picture.



The Impact had four CPUs which needed to work synchronously – a 16-bit Intel 80186 as the master processor in the mainframe, an 80188 (the 8-bit sibling of the 80186) running the control panel, another 80188 to handle communication with other video equipment, and an Intel i860 (an experimental RISC CPU) for crunching the math necessary to locate the polygons in virtual space. The figure on the left gives a schematic view of the Impact. The "video hardware" consists of a number of custom circuit boards mounted inside the mainframe.



In practice, the user watches the video he or she is creating on a television monitor, and uses the joystick on the control panel to manipulate the video and run the system. The joystick acted something like a mouse, allowing the user to select control parameters on eight or more data screens. Then, with a click of the button on top of the joystick, those parameters were linked with the joystick's three axes, allowing direct manipulation of the picture on the television monitor. There was also a keyboard with dedicated keys for common functions, allowing users to work with both hands.

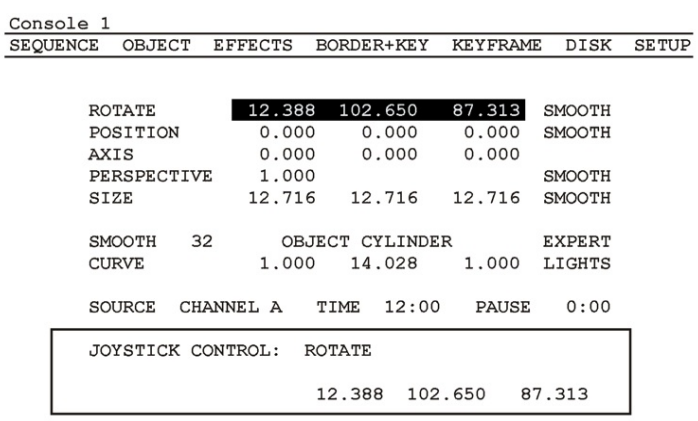
Like most DVEs, the Impact was a "keyframe" system. The user would create a configuration on the television monitor, and then press a key to record the system's current state into a keyframe – a snapshot of all of the settings at that moment. Then the user would make changes, such as moving the video object across the screen. Pressing the record key again would capture this new state into a second keyframe. Then by pressing a "run" key, the Impact would smoothly transition from the first keyframe to the second. Up to 20 keyframes could be recorded into a "sequence," and up to 20 sequences could be retained in the Impact's memory at once. The sequences don't record the content of the video input, just the behavior of the polygons used as display surfaces for the video.

The user could save the sequences to floppy disks for later use. The system came with several basic effects built in – a page turn, a cylinder, a cube, Venetian blinds, and some other useful 3D shapes. Additionally, Microtime periodically released software libraries of new effects, which could be downloaded to augment the basic shapes.

The Mainframe

The primary function of the mainframe software was to manage the creation and running of sequences. As a sequence runs, the mainframe software would interpolate all of the system's parameters for each frame of video in realtime. The current parameters for the object being used (a page turn, for example) would be inserted into common memory between the mainframe and the i860. Output from the previous frame's calculations would be taken out of the shared memory and fed

to the appropriate video control boards, which were memory mapped into the mainframe's RAM space. The system parameters currently showing on the video monitor were routed over an ethernet cable to the control panel, where they were displayed in numerical form, an example of which is shown here.. The control panel display, shown at right, was a monochrome text monitor that could also generate straight lines, which kept the user displays simple. The Impact's video output was always displayed on a video monitor, which allowed us to keep the control panel display simple.

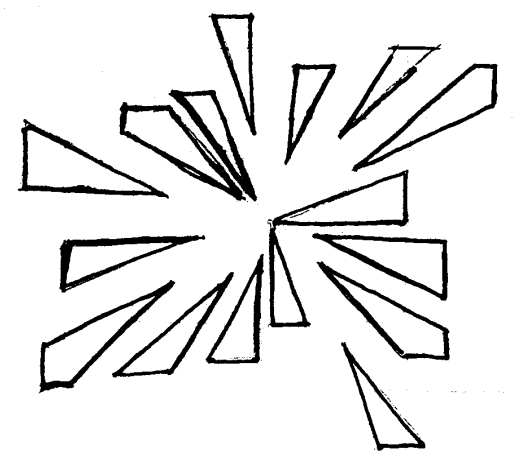


The first Impact had a maximum of 32 patches, which could generate simple curved surfaces, provided the curves weren't too tight. Later versions of the Impact had up to 256 polygons, suitable for much more complex shapes. This Impact could run in either real-time or non-real-time modes, as required. If the i860 couldn't complete crunching the numbers for every frame of a sequence, the mainframe would spool the i860's output internally, after which the sequence could run smoothly.

The Control Panel

The control panel resembled a poor man's Windows (for performance reasons, controlling this kind of system with a desktop computer in 1989 was not an option). The figure above shows a typical display screen. The joystick moves the highlight from one parameter to another. When the user clicks on a parameter (or a group of two or three), the joystick's axes become attached to that parameter, and the video display responds to the joystick accordingly.

The control panel had only 64K of ROM space, which meant that my display system had to be very efficient. All of the screens were tables of data representing the location of the parameters on the screen and in memory, the kind of data, a text label, and various other bits of information. Locations of adjoining parameters were hard-coded into the tables, so that nudging the joystick immediately shifted the highlight to another parameter. Some labels were in fact gateways to other screens,



so that clicking on them brought up other sets of parameters. Pop-up menus were handled this way.

Interfacing with the floppy disk was one of the more interesting aspects of developing the Impact. User sequences were saved to and loaded from the floppy. A number of other people wrote effects programs for the Impact on desktop computers, in C. This executable code was then packed onto floppy disks, and could be downloaded to expand the Impact's effects capabilities. We had briefly discussed using a file system compatible with MS-DOS, but I elected to create my own, for reasons of development time and the size of the code.

My file system allowed files to have up to three "forks" – separate components, any one of which could be handled separately. Effects programs were composed of three forks. One was the executable code, transmitted to the mainframe, which then loaded it into the i860 memory and linked to external routines. Another fork was the table needed for the control panel screen associated with this effect. The third fork was a schema, identifying the number and types of parameters, which was used by both the control panel and the mainframe.

Linking the downloaded code with external routines was an interesting process. The default effects code for the i860, loaded from ROM at powerup, contained a number of frequently used functions. Pointers to these functions were placed into an array in low memory within the i860's RAM space. When new effects code was written, we tricked the linker/locator into assigning phony addresses to these common routines. The high word of the address was a unique pattern of bits, and the low word was an index number specifying the cell number in that array that would point to the required function. When the mainframe loader routine saw the unique bit pattern, it replaced that and the following index number with the corresponding address from the array.

The Communication Unit

Remember that the Impact didn't store any video, just the behavior of 3D effects, as sequences of keyframes. In order for these effects to be used in a production, the

running of the Impact's sequences needed to be timed correctly with the video input, which often came from a tape deck.

Like many commercial television devices, the Impact was designed to be controlled by a device called an edit controller. This is a device which controls tape decks and various other paraphernalia, rewinding or fast-forwarding tapes and running them in sync. To an edit controller, we looked like a tape deck. The Impact user could download sequences from a floppy disk, and then use an edit controller to automatically play the sequences, inserting the desired video into the sequence.

Emulating a tape deck was one of the main reasons we had a separate CPU to handle communication. Commercial video tape decks have a complex communication protocol, consisting of various commands from an edit controller, and various responses from the deck, indicating its status and the position of the tape.

Any sequence the user had created in memory could be run by the edit controller, by "positioning" the Impact to its start, and issuing a "play" command. Being a digital device, the Impact didn't have to rewind or fast-forward tape to do this. But if the Impact didn't *pretend* to do this, the edit controller would assume that something was wrong, and abort the operation.

For that reason, I wrote "ballistics" emulation software in the communication module. This would respond to a tape positioning command as though there was a large, sluggish reel of tape being manipulated. I had studied the actual performance of video tape decks in a previous software job, and could do an acceptable job of emulating a variety of common tape decks.

Programming

The mainframe, control panel and communications unit of the Impact were programmed in a language called PL/M. Created by Gary Kildall of Digital Research, PL/M was the standard language of Intel in the 1980s, and was also the language CP/M was written in. It resembled a simplified Pascal, was quite pleasant to work with, and the Intel version was an extremely efficient compiler. The only complaint I had was the segmented address scheme used by the 80186, which limited the size of my code modules, and forced me to keep track of addresses as two-word pairs. But this was the norm at the start of the 1990s.

In order to create code for embedded systems, we had to use a device called an in-circuit emulator, or ICE. The ICE substituted RAM memory for ROM, on a printed circuit board that had pins on the bottom which would fit into the socket for the ROM. It also had a cable to hook into the back of a desktop computer. Software on the computer would load compiled code into the ICE. I also used a version of the Turbo C debugger, configured to work with PL/M.

Using Turbo C, I wrote a scripting language which would specify control screens for new effects. There were three or four people developing effects software, and they needed customized controls for each one. The script processor would read an effect control script and generate a mock-up of the Impact's control panel screen, as it would look when this effect was selected. It also generated the schema and control forks of the effects libraries.

Conclusion

In 1989, when I started working at Microtime, we were the only people in the world who could map live television onto the sides of a 3D polygon object. By the end of 1995, when I left the company, you could do this on a well-equipped desktop computer. The Impact sold reasonably well during its heyday, but as each unit cost between $40,000 and $100,000, it was no longer economical.

Microtime had gone through two changes of ownership while I was there. In 1993 it had been sold to a California company called Digital F/X, which had acquired a number of high-end graphics companies in a short period of time. But they had bitten off more than they could chew, and went out of business within a year. In 1995, Pinnacle Systems bought Microtime, mostly for a character generator which Microtime had acquired from Digital F/X. It was clear they had no interest in the Impact, nor any of the people who wrote the software for it. It was time to move on.