

Beginnings of PowerPoint

Dennis Austin

A Personal Technical Story

© 2009 Dennis Austin

This is the story of PowerPoint's earliest days told from my own point of view as designer and programmer. I tell the general story including trivia perhaps memorable only to me. In particular there is some design and engineering detail, as I hoped to capture some of the thinking behind PowerPoint as a new product.

These days PowerPoint itself needs no introduction. It succeeded beyond my wildest dreams. It has become a synonym for presentation slides. Although loved by many, it is also reviled by audiences condemned to sit through boring presentations seemingly stretched by an endless series of visual aids. Whatever PowerPoint's contribution to communication, it could only magnify a presenter's skill—or lack of it.

PowerPoint was the first presentation program of its kind, but it was certainly not the first application for making presentations. Offerings like Harvard Graphics (from Software Publishing) came earlier and many competitors in the PowerPoint style followed—Aldus Persuasion and Lotus Freelance most notably. Competition was keen, but PowerPoint eventually succeeded as Microsoft Office succeeded. Heavy consolidation in the industry left room for only one championship competitor, and PowerPoint was fortunate to be part of the brilliant Office suite.

In the present PowerPoint continues as the standard of presentations everywhere, but that position is never secure. New competition appears regularly.

1984

In September 1984 I found myself out of work for the first time in my career. Gavilan Computers laid me off. The company closed its doors that month after a two-year whirlwind existence creating the world's first battery-powered laptop computer of the modern mold. Modern mold in a way, but no 1984 computer was much like today's. Computers with graphics displays, mouse pointers, and internet communication have become so ingrained in consciousness that we have to concentrate to visualize or remember 1984.

Low cost personal computers were still a new phenomenon, but they had been around long enough to prove some value to business. Mainstream machines were the Apple II (1977) and the original IBM PC (1981). The IBM PC AT, with its sixteen-bit bus and high-powered 286 processor, debuted in the spring of '84. Apple was trying to introduce the Apple III to catch up to IBM. Lotus 1-2-3 was the newest sensation because it out-performed VisiCalc, but it only ran on the IBM PC.

Up the price ladder a different and more futuristic competition was taking place. In the late 1970's, the Xerox Alto demonstrated a new way of interacting with computers using a mouse to manipulate images on a graphics display. Text was treated as just another kind of graphics, re-establishing a connection that had broken down after the invention of the typewriter.

Xerox didn't commercialize these ideas from its Palo Alto Research Center until the Xerox Star of 1981. By that time, a host of start-up competition was circling. Energetic young brains, inspired by the work at PARC, founded Three Rivers Computer, Apollo Computer, Sun Microsystems, and Silicon Graphics.

Apple Computer aspired to be part of the bright future promised by these workstation companies, and built its own competitor in 1982, the Apple Lisa. The Lisa cost twice as much as other personal computers, but half what the workstation market was charging. It brought the vision of a graphics-based user interface to a slightly larger audience.

1984 was famously the year the first Macintosh computer was announced in an Orwellian Superbowl spot. Although much less capable than the Lisa, its low price finally brought a graphics interface to the masses. Well, at least to the well-off, somewhat geeky masses.

The Macintosh introduced not only these new ideas in user interface, but a new kind of computing appliance. Unlike the IBM PC, and in fact every prior Apple computer, the Mac wasn't an open system.

It came complete with sound and networking, which PC's didn't, but you couldn't plug in cards or use a different monitor. You could write software for it, but it wasn't a capability designed for users. There was no equivalent of the BASIC language, and writing anything required complex interaction with the operating system—something PC programmers of the day were unaccustomed to. You could write in an elegant version of Pascal, but the compiler ran only on a Lisa—hardly the home-brew tinkerer's realm.

Although Apple could change things over time, most potential competitors saw the future as marrying the successful, open IBM PC with a software environment providing a graphics user interface. Personal Computing, the maker of VisiCalc, led the field for a time. It shipped its VisiOn software framework and application suite in 1983. Microsoft was working on a system of its own called Windows.

Forethought, Inc.

Two former Apple employees, Rob Campbell and Taylor Pohlman, formed Forethought in 1983. They planned, essentially, to bring a graphics user interface environment to PC's using a high-resolution monochrome video system (Hercules, at the time). They planned to build a multi-purpose application that would integrate word processing, graphics, and spreadsheets, all based on a foundation database storage system. They would build everything using object-oriented technology, the latest trend in software. They started building the software system from the ground up: operating system, graphics, user interface toolkit, and even fonts.



Robert Campbell, president of Forethought during the development of PowerPoint. Rob and Taylor Pohlman founded Forethought in 1983.

Rob and Taylor hoped to make their product fit naturally into an office environment and be much easier to use than current PC's. They wanted broad distribution, so they were intrigued when introduced to Darrell Boyle. Darrell wasn't from the computer industry, but he had real consumer marketing skill beyond the range of typical Silicon Valley people. Darrell soon became the third company executive, vice president of Marketing.

Product development proceeded without any experienced management and it was difficult to measure its real progress. Forethought started a search for a manager of development, but it went slowly and months rolled by. After a scheduled initial demonstration to investors went poorly, the company needed a big fix.

A few months after Darrell had joined, Apple announced the Macintosh. Although an Apple product, it was similar in price to the PC's they were developing for, and it was targeted for a mass market. It seemed it should be part of a company re-start strategy. They decided to enter the Macintosh market by publishing software. The software publishing business could provide a revenue stream while the company's flagship product continued development. Getting a reputation and a full company built up early would spring-load Forethought for success when its own product was ready to go. Darrell created and trademarked the MacWare brand and designed his strategy. Rob used his Apple industry contacts to look for application developers in need of a publisher.



Darrell Boyle, Director of Marketing at the time I joined Forethought. He created the MacWare brand and mapped the strategy for Macintosh.

The concept of publishing software, common at the time, has now mostly disappeared. The idea was that programmers were like novelists. They didn't require a lot of resources during the creation phase, preferring to create works of genius in the basement at night. But to make money they needed the help of a large organization that could leverage its resources to properly refine, print, market, and sell the product. In other words, a publisher.

The publishing plan might keep the company going, but it still needed accountability in its own development group. The company had been talking to Bob Gaskins for a number of months about a position managing product development. In the summer of 1984 the company found its fourth executive, vice president of software development Bob Gaskins.

Bob Gaskins

Bob was a highly literate man with wide interests and knowledge. He spent ten years as a UC Berkeley graduate student studying English literature, linguistics, and computer science. He finally left Berkeley to join Bell Northern Research, an arm of Northern Telecom, where for five years he led a small research group in computer science and communications. This gave him experience with photo typesetting, Donald Knuth's TEX typesetting system, the SmallTalk language, and other advanced ideas of the day. Like other computer futurists at the time, he was a strong advocate of the personal computer and of graphics user interfaces. He came to Forethought to stop researching and start building.



Robert Gaskins, Forethought's Vice President of Development, at his desk. PowerPoint began with Bob's idea.

While Bob was getting settled at Forethought, he met a couple of times with Darrell Boyle to discuss products. Bob and Darrell learned a lot from each other during their time at Forethought and were destined to work together, on and off, for years to come. At this time they were discussing future product directions. Bob came prepared with list of possible products that he felt would fit the vision of the company. One of the ideas was a product for preparing visual aids for presentations.

The presentation idea captured Darrell's imagination. He liked it so much that he eventually focused his Silicon Valley career on the market for presentation software. (After Forethought, Darrell founded Trailblazer Consulting, where he helped Apple, Microsoft, and other companies.) For Bob, visual presentations were in his blood. His father's career had been in the photographic and audio-visual industries, and Bob was well acquainted with the needs of presenters. At Bell Northern, his group had awkwardly produced gorgeous presentation materials using a DEC 20 and a phototypesetter. After the discussion with Darrell, Bob produced a brief but memorable two-page proposal for a future presentation product.

The pressures of the present crisis, though, took precedence over new product ideas. Bob had to wrangle the errant development group. After a period of evaluation, he decided that the best course forward was to salvage whatever was useful at the application level and migrate it to a new foundation—Microsoft Windows. Windows was his bet for an eventual standard, so it was the only reasonable choice for the future.

Bob fired most of the development team, retaining only Peter Bishop. Peter was an extremely capable programmer who had worked at Xerox. He had a deep understanding of the software architecture in place. His continued presence assured that not everything would be lost from work already complete. If anything positive lay in the future, though, a new team needed to be built.

In October Bob interviewed me for a position in development. We had met before through mutual friends, and I was interested in the new company he had joined. Did I mention that I was also out of work? New Enterprise Associates financed both Forethought and my now defunct employer, Gavilan Computers. C. Woody Rae, a partner with NEA, nevertheless steered me away from Forethought. Not a promising company in his opinion. I had great respect for his judgment, but the Forethought mission was seductive. I joined Forethought in spite of its cloudy prospects. My brief unemployment was over, and I didn't worry much about job security. This was Silicon Valley.

Dennis Austin

On my first day at Forethought, the company had a party to celebrate the release of its first MacWare product, FactFinder. FactFinder was a simple free-form database. A second product soon followed, Typing Intrigue. It was a mystery game designed to help you learn to type. The business was moving, but Darrell's plan called for seven applications, over a two-year period, in order to generate sensible profits. And that was assuming strong sales of Macintosh computers.



The author during PowerPoint's his days at Forethought.

I came with high hopes for great work at Forethought. There was no specific course mapped, but I was a true believer in Forethought's mission. My first task, with Peter Bishop, was to salvage the existing software by moving it to Microsoft Windows. We worked on this for nearly six months. We wanted to build a demo that would convince the investors, and us, that we had a promising product. It was a hard six months for me as I gradually became disillusioned by Foundation and even the design work underlying it. Our demo was poor, and we decided to abandon Foundation in favor of some new plan yet to be devised.

The good thing about this period was getting to know Bob. We found that we saw eye-to-eye on many issues and we could communicate well. Although we were previously acquainted and had common ideas, I could now be confident that we would be productive partners.

While Peter and I were porting Foundation, the MacWare publishing line got its third product. This was a database application from a Massachusetts group called Nashoba Systems. Nashoba sold a PC version of the database called Nutshell, but believed Forethought would improve their fortunes on the Macintosh. The product was renamed FileMaker and rapidly gained market share against competition from the more established companies: Microsoft File and PFS:File from SPC. FileMaker was a well-designed product and was reviewed well. In spite of its data-oriented core, it took full advantage of the Macintosh graphics approach and produced beautiful forms and reports. An improved version, FileMaker Plus, was produced the next year and the sales definitely lifted the profile of Forethought.

The company was teetering on the edge nevertheless. Our big presence on the Macintosh platform wasn't enough because the platform itself was a small market. We hoped that the Macintosh would catch on and improve our fortunes, but in the meantime we needed to conserve capital. Darrell was advising Rob that the best course was to hunker down and reduce the size of the company to make our money last longer. Rob was holding on, hoping our money could last until Macintosh sales took off.

Finally it was clear that Darrell should indeed take down marketing and sales—including his own position. Rob could handle those duties himself. (Pohlman had already left the company prior to the FileMaker success.) Engineering (Peter Bishop and I) were retained as the hope for a distant but brighter future.

The company went down to fewer than ten employees. Soon even Peter Bishop was gone. Every Monday morning at our company meeting I feared that the closing of the company might be announced. Rob Campbell was always honest about our prospects, but his optimism and leadership were unflinching.

Presenter

We began to research ideas for a new product we could create ourselves with modest resources. A number of proposals were written up and discussed. The most promising seemed to be the presentation slide idea that Bob had worked up the year before.

It was such a natural idea for Bob. He was well acquainted with the equipment and processes of formal presentation graphics. He was deeply interested in the skills and techniques of communication. He remembered his BNR experience producing typeset transparencies for overhead projection: the Bell Northern executives had adored the results.

He observed that good quality presentation graphics make speakers look better. Speakers sense this and are motivated to create the best possible graphics. If only they could *do it themselves* instead of relying on a company art department that never fully understood what was being communicated.

Darrell was confident a product that allowed speakers to create their own high-quality graphics would have a ready market. And speakers are ideal sales targets. They are often people with some authority over company budgets, and they can expedite the purchase of such a tool without lengthy approvals.

We all knew that only a product based on a graphics user interface could change the world of presentation material. Only a graphics environment could allow a user to manipulate the drawings and text directly, seeing the results just as it would eventually appear to an audience.

My interest was in preparing text and graphics for printed output, but I immediately liked the idea of doing the same for presentation graphics. I liked it especially because the graphics involved were actually fairly simple, and that would allow us to build a product relatively quickly.

We decided to proceed, and we adopted the working name *Presenter*. Starting from Bob's proposal from a year prior, I sat down to write a specification for a product. Bob and Darrell worked on a business plan for it. Bob became the product visionary and expressed the product concept. Darrell defined the market, identified competition, and requirements for success, but this was to be the last of his contributions. Rob finally agreed to take the company down in scale. Marketing and sales were eliminated along with some more minor functions. The company was soon down to fewer than ten people, and I became the only engineer. We missed Darrell, but he returned in a consulting role later in the project, and he eventually rejoined Bob and me at Microsoft.

The product design details required a lot of invention. Bob was able to spend many hours with me hashing over ideas. It was a productive process and the quality of my designs reflected Bob's support and feedback. I have compared our collaboration to a building design project: Bob wanted to build a dream house and I was his architect. My spec went through several revisions and was ready for its first distribution in August of 1985. I listed both our names as authors.

The business plan, the spec, and some enthusiastic exposition of their contents convinced our venture backers that Forethought might yet recover their investment. We were ready to move on to the next phase.

The Platform

An odd thing had happened during the design phase. We started by assuming that we were targeting Microsoft Windows. That seemed like the path to long-term success because Microsoft ran on the



Target hardware: 512k memory, floppy disk only

mainstream, open system and was likely to form an eventual standard. I had acquired a good bit of experience with Windows during the previous effort with Foundation, though, and I didn't really think Windows could meet our needs. Its graphics and typographic support were weak and it was immature in all respects. We decided that the better strategy was to build for Macintosh first and to move to Windows when it was ready.

It may seem ludicrous to view the Macintosh, at 16 months old, as being mature. But, perhaps owing to earlier experience with Lisa, Apple had this simple machine quite well buttoned-down. It had a simpler programming model and a more complete and well-documented toolkit. It had an effective graphics system that was easy to control precisely. More important, it had a well-designed text subsystem. All this is not to mention that the system was already proven to be efficient for the kind of work we wanted to do.

The Macintosh had a key feature that Windows didn't even appreciate—the Apple LaserWriter. Unlike laser printers available for the IBM PC, LaserWriters were based on Adobe PostScript. PostScript could produce the graphics we needed and, more particularly, the text. Beautiful fonts in any size, and printable in “landscape” orientation. Some companies were already buying a Macintosh and a LaserWriter for the sole purpose of preparing presentation slides—in MacWrite or MacDraw!

What's more, Apple's admittedly small market was genuinely excited about using the machine. Microsoft's customers were extremely skeptical about Windows and you couldn't even install it without upgrading your configuration and buying a mouse. It was our opinion that the Macintosh was the machine that would break the ice and eventually prepare a market for Windows.

Then there were our contacts at Apple. Rob had been working his connections and promoting the idea that presentation software could be the next big category for Macintosh. After some initial market confusion, so-called desktop publishing was emerging as a new category that could fuel the success of Macintosh. “Desktop presentations” might be another. After considerable dealing, Apple agreed to make an investment in Forethought—its first in another company. The new money gave us the breathing room we needed to finish Presenter, and promotional help from Apple as well.

A Software Category

We viewed Presenter as a revolutionary product despite the fact that visual aids for presentations were not at all new. Even the idea of making those aids on a personal computer was not new. An established, if small, software category called “presentation graphics” addressed that need.

(The term “software category” has lost currency since consolidation standardized on one or few successful applications from each category. In the 80's, competition was intense in every category: word processing, spreadsheets, database, graphics, etc.)

Presentation graphics was mainly about building charts and graphs based on input data. It also included “word charts” with classic bulleted list of points, but they were simply a special case of a graph with the words being the input data. Output was generally produced on a pen plotter, so results were slow and looked like engineering drawings. Color was possible, but only a few different inks could be used. Previewing color on the computer monitor was limited because of the low resolution available.

Top quality presentation graphics weren't produced on personal computers at all. Companies like Genographics made them on expensive specialized workstations, and their output was produced on film recorders and projected as 35mm slides.

Operationally, speakers themselves did not produce their own visual aids. The skill required to operate the software was too specialized. Companies had art departments that could create the needed graphics, either by computer or on a drafting table. Individuals making their own material usually drew them by hand or typed with an IBM Selectric Orator type-ball on transparency material.

We didn't want Presenter to be positioned in this category, as it existed, so we had to define a new category—"desktop presentations". We drew attention to our new way of approaching presentations:

- Graphics, direct-manipulation interface. What you are editing looks exactly like the final product.
- Simple enough so that speakers could produce high quality slides for their own use.
- Focus on creating and editing an entire presentation. Other products produced slides individually.
- Use the content-integration capabilities of the Macintosh to incorporate graphics from specialized sources.
- Output to laser printers, only recently available. This was the only reasonably priced high-resolution output on the market.

On the flip side, Presenter eschewed some key features of the category:

- No graphs at all. We reasoned that Excel had a perfectly respectable graphing component and anyone interested in producing graphs from data would already own Excel. Why not simply copy the graphs from Excel and paste them into PowerPoint? That is the integration Apple was marketing, so we might as well leverage it.
- There was no color at all. After all, laser printers couldn't produce color and the Macintosh had only a black and white monitor.
- There would be no output to plotters. Plotter drivers were available for the Mac, but we were believers in bit-mapped graphics and there was no way to adequately print bit maps on plotters. What could be plotted was too low in quality for our target audience.

All this seemed risky, but necessary to draw the right kind of attention.

The Concept of Presenter

Forethought was founded to build graphics user interfaces and both Bob and I had joined to create that kind of software. We conceived Presenter in that thinking and the product could not be separated from it. We viewed the creation of presentations the same as page preparation for printing, and the advantages of the interface applied even more clearly to presentations. The invented category name of desktop presentations was obviously borrowed from the new category of desktop publishing for precisely that reason.

Bob Gaskins had a clear vision for the kind of product he wanted to create and there were several key insights that would guide the design.

Presenter was to create *presentations*—not simply slides. It was to provide a means for structuring, writing, and reviewing a presentation as a whole. That would also make it easy to re-sequence old slides and to re-use parts of old presentations to make new ones.

It would construct a consistent layout for all slides in a presentation. The layout included items like a graphics border, corporate logo, running heads, and slide numbers. The Presenter package would also use templates with such features to support corporate graphic standards.

It should allow the production of various presentation materials from a single master file. This would include both presentations slides (overhead transparencies, but also on-screen presentations, and perhaps eventually 35mm slides) and printed handouts. The potential high quality of output should be matched by high-resolution graphics and typeset-quality text.

Bob was good at communicating his vision to the management of Forethought, to potential sources of financing, and to me. When I sat down to design the details of Presenter, the goals were crystal clear. To get a more concrete idea of the output we wanted to create, we accumulated a small library of good-quality presentation slides from real conferences. From these, we chose what graphics would be created directly in Presenter and what we expected to be integrated from other sources.

To understand our new target environment more deeply, I switched to using a Macintosh at the office. Although limited in scope, it was a delightful machine to use with interesting surprises around every corner. I wrote specifications in MacWrite and I drew up sample screen images in MacPaint.

The Design of Presenter

In the presentation graphics category, the conventional model was to describe an intended slide and then have the program produce it. You described the slide by filling in forms. You selected a type of slide and then detailed its precise characteristics. Then you could preview the results.

This was obviously not the way Presenter would work. Our model was the what-you-see-is-what-you-get paradigm, popularized at Xerox using a phrase borrowed from the popular 60's TV show, Laugh-In. Presenter displayed the "preview" at all times, and the user interacted with the finished appearance.

All this seems pretty obvious to modern thinking. To Bob and me and pretty much anybody else at Forethought, it seemed obvious even in 1985. But the general industry thought much differently.

Anyway, even with our fervent devotion to the graphics user interface model, I needed some design rules for this particular effort. I took ideas from the wisdom of the day:

The principle of least surprise: You might not be able to make things obvious, but at least you could design them so that they made sense once discovered. Once you realize that software is designed with this principle in mind, you can sometimes guess how something works.

Make easy things easy and hard things possible: this idea, later sloganized by Eric Wall, emphasized the commonplace activities. Make sure that the everyday stuff is straightforward. Other stuff needs to be there, but don't worry about how easy it is.

More narrowly, though, I was aiming at a particular audience—an audience identified by Bob Gaskins. Our users were familiar with computers, but probably not graphics software. They were highly motivated to look their best in front of others, but they weren't savvy in graphics design. It was my job to make Presenter easy to learn and use for this audience specifically.

Design Points: Slides

Apple was selling a vector drawing program for the Macintosh. Named MacDraw, it was patterned after their earlier effort, LisaDraw. I had used LisaDraw during my time at Gavilan. I made my own presentations with MacDraw and printed them on the LaserWriter, and this influenced my design thinking. Very early I decided that drawing vector "objects" would be the basic metaphor for Presenter. I respected the precedents being set in MacDraw, but still thought the details of drawing would need to be a lot different in Presenter. I could simplify the usage considerably by targeting precisely the features needed for creating typical presentation slides—not elaborate drawings.

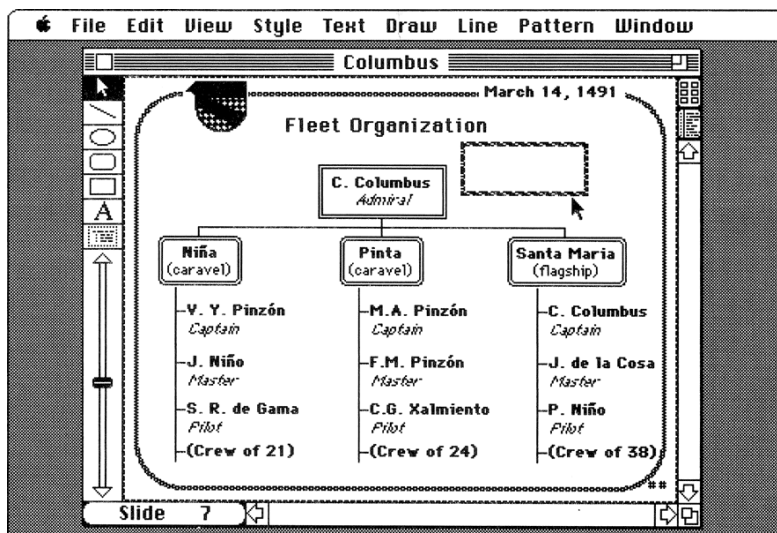
The Presenter drawing surface, a single slide, had known bounds. These could be adjusted, of course, but the output of the drawing had a set real-world size. For MacDraw and similar programs, the

drawing was abstract and expanded as needed. Scaling and fitting to an output medium was a separate problem and output might even spread across multiple pages.

The default slide size I chose was 7.5 by 10 inches. This seemed ideal, because it had a 3:4 aspect ratio that was the same as computer monitors of the day, and also fit US letter paper with exactly .5 inch margins. In contrast to every other application I'd seen, my default page (slide) had a horizontal or landscape orientation. This was appropriate for on-screen presentations, obviously, but I thought it was also best for overhead projectors where the bottom part of vertical slides was often hard for the audience to see. The Macintosh system software accommodated landscape printing, but it didn't support that orientation as a default. Unsupported software tricks were necessary to accomplish my ends, along with a lobbying campaign at Apple to make the feature a standard capability.

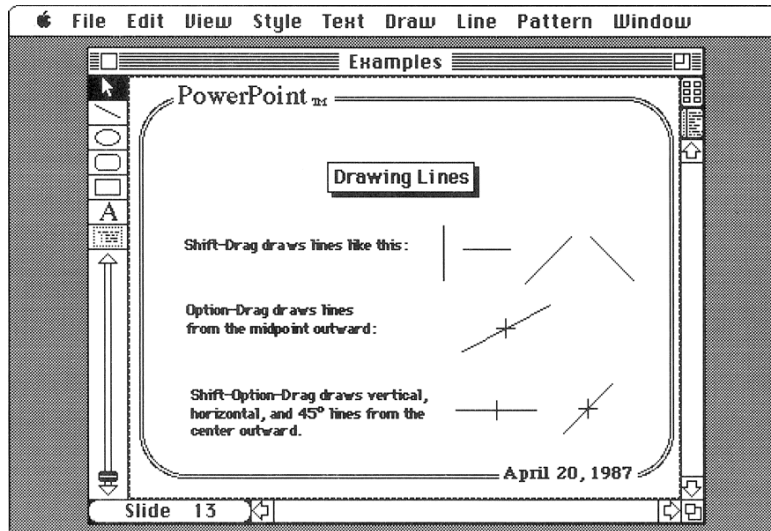
Drawing

Presenter users would build slides using the drawing metaphor, but the "drawings" would be only simple diagrams and text. As already mentioned, that simplified the interface considerably and also permitted specialization in the areas we thought important. I omitted many features offered by MacDraw, including dimensioning, arbitrary polygons, and freehand curves. More interestingly, we largely obscured the significance of stacking order. Most objects would have no fill (we had no color, remember) and we were trying only to build diagrams.



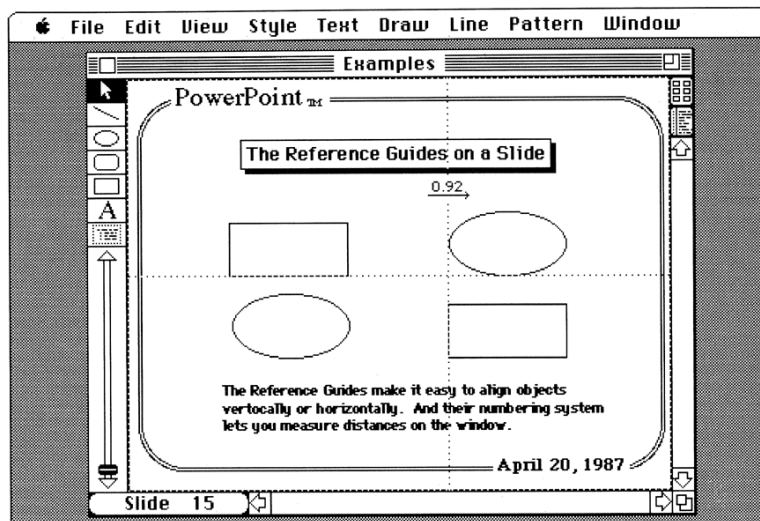
This tongue-in-cheek organization chart, from an early sample file, illustrates the level of drawing complexity we were aiming for.

MacDraw had other complexities to be reckoned with. You could click on an object to select it, but you could also simply drag the object to reposition it. Since a click sometimes entailed a slight, accidental mouse movement, the difference was one of timing. It was confusing to new users and sloppy clickers because an attempted selection often resulted in unintentional movement of an object. I addressed the problem by inventing a different kind of selection highlighting and requiring that you could only move or resize a selected object. You couldn't select and move in one mouse motion.



As an aid to symmetry, Presenter introduced a method for resizing objects starting at their center point.

For graphic balance it was typically appropriate to arrange objects symmetrically, or at least carefully, with respect to the slide area. For that reason, I adopted a measurement system having the origin at the center of the slide rather than the usual upper left corner. That way it was easier to place content in a pleasing design. I added movable guidelines, at first called T-squares, to aid in the positioning of objects. (The name T-square was borrowed from FileMaker, Forethought's hot product at the time.) When a T-square was being dragged, its measured position appeared next to the cursor. By holding down the Macintosh Option key, the measured position was shown in relative numbers so it was easy to measure relative distances without resorting to rulers and doing arithmetic. The T-squares appeared, by default, crossing at the center of the slide and thus again allowed easy positioning of objects at the center, horizontally or vertically, of the slide. T-squares were "magnetic" in that objects being dragged with the mouse would snap to positions relative to the guideline. The edges would snap, but so would the object centers—another aid to centering and symmetry.



Presenter used guide lines (inspired by a similar idea in FileMaker) to aid in alignment of objects. When dragging an object, its center and sides would "snap" to the guides for accurate placement.

I noticed that objects often needed to be resized without changing their position. That gave rise to the Option key resize modifier that resized the object but anchored it at the center. Of course it could be

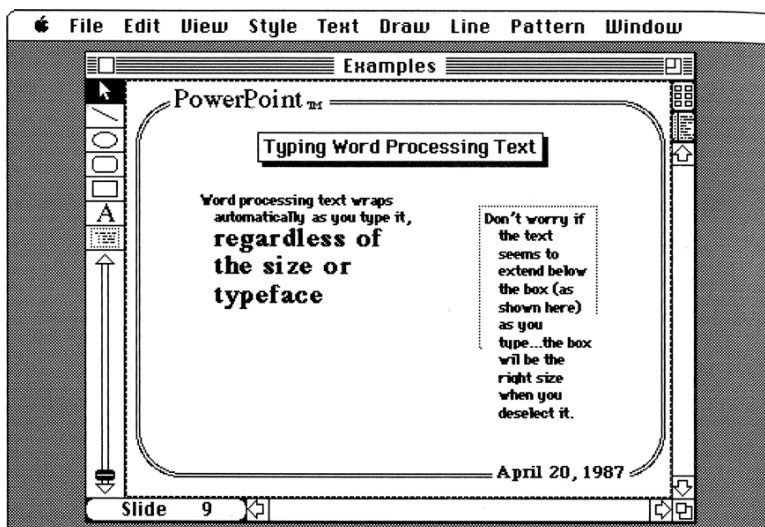
used in combination with the already-traditional Shift modifier that maintained aspect ratio while resizing. (MacPaint inspired the Option key idea, but MacPaint didn't have the concept of resizing.)

One additional problem of object placement was accuracy. The Macintosh screen was small, but, luckily, slides typically contain items large enough to be seen from the back of the room. Consequently, you could usually work with the entire slide on view—50% scale. We provided other scales, of course, for detail work. I selected a limited set of scales with simple multiples. It allowed use of integer arithmetic (i.e. fast arithmetic) in calculation, and, at the same time, preserved perfect registration of pixels. Two points placed coincidentally at one scale would be coincident at all scales.

Text Boxes

Unlike MacDraw, beautiful text was a centerpiece of Presenter. Instead of simple labels with a single text style, Presenter had to provide full typeset-quality text as a word processor might. Unlike word processors, though, text appeared in graphics elements that I called text boxes. A text box was like other graphic objects in that it could have a line border or a fill, but its contents were a miniature document. It was a “word processor in a box”, including not only typefaces, sizes and styles, but word wrap, line and paragraph spacing, margins, tabs, etc.

Presenter's text was even more specialized. Presentations often included bulleted lists, and sometimes sub-lists with their own bullets. I chose this as the default format of Presenter's text box. A bulleted list consisted of paragraphs with a hanging indent, i.e. a paragraph where the margin of the first line is left of the subsequent lines, and with paragraph spacing that separates the items in the list. The indentation of the paragraph was fully adjustable, but this was the default style.



Presenter was at its heart a drawing program. Unlike other drawing programs, text was our most important product.

To cater to sub-lists, enabling an outline of sorts, paragraphs had a “level”. Each level had its own margins of indention, so you could easily type in a simple bulleted outline without fussing with formatting. Full formatting was, of course, available if the defaults weren't suitable.

Label Text

Text boxes weren't the only kind of text needed. For diagrams, it was important to be able to create simple text labels. In MacDraw labels were pieces of text in a single font and style. Presenter had these simple labels (although the text could be fully styled), but it also allowed graphic objects to *contain* text.

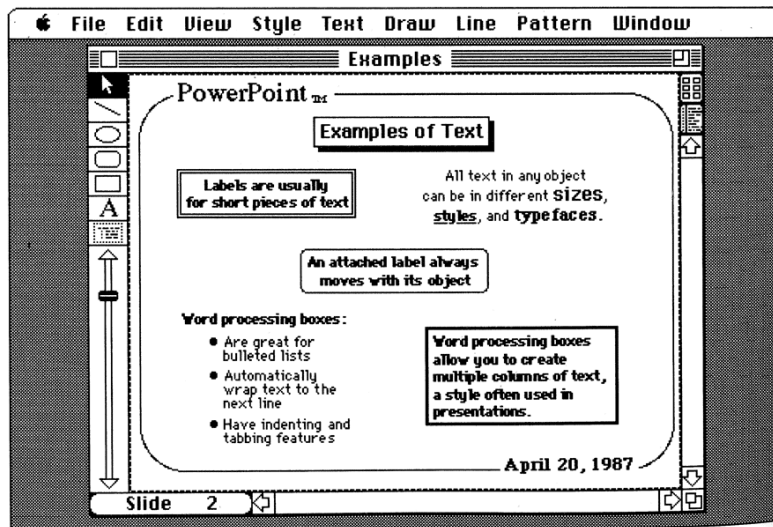
Text contained in an object (rectangle, oval, etc.) was centered in the object both horizontally and vertically. This made it very convenient to construct typical diagrams and simple tables out of labeled boxes. If needed, it was possible to anchor the text as needed instead of centering it.

Labels were created with the label tool and had, by default, left alignment. If you edited the text, the upper left corner of the object on the slide did not change. Here too the anchor point could be changed if needed.

The distinction between simple label objects and graphic-objects-containing-text was made by an attribute called size-to-text. The size of label was determined by the text it contained. Even if you later added a frame or a fill, these objects automatically resized when their text was changed.

Graphic objects that contained text, however, had a set size and were re-sized by adjusting handles in the usual way. The text display was centered in the object. If the object wasn't big enough to contain the text, it simply stuck out of the frame.

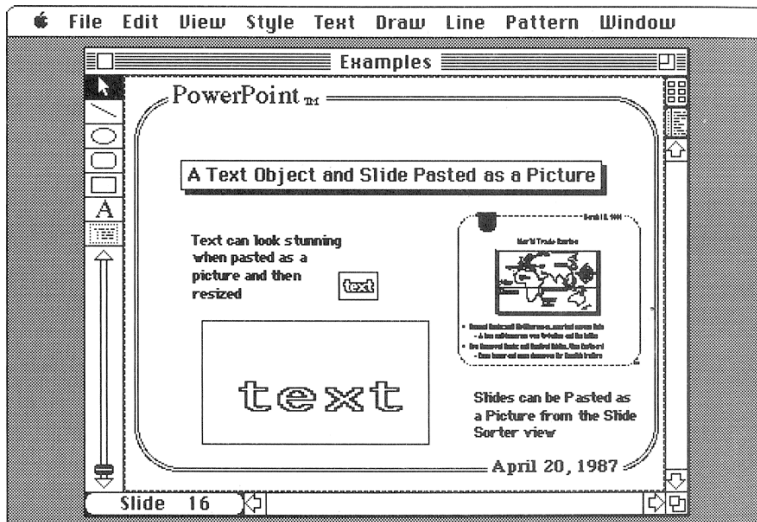
By the way, text boxes (the full word-processor kind of text) also were size-to-text. Because their text word-wrapped, the width of a text box was fixed. The bottom of the frame, though, was automatically set by the size of the content.



Types of text objects
Presenter produced.

Pictures

With simple drawing tools plus the elaborate text tools, Presenter was ready to create most of the content of presentations. It was our intention to leverage the rich world of other Macintosh applications to provide more specialized content. Doing so made our job easier, of course, but it also made it easier for the user. Content could be created using the tools most appropriate for the job by experts in those particular tools, and then pulled into the presentation where needed. It seemed far better to create graphs in Excel for example, than to introduce our own set of tools for the same purpose. And those who missed polygons or other features from MacDraw could create their complex drawing there and then paste them into a presentation.



A page illustrating how Presenter drawings, slides, and text, could all be turned into pictures that could be pasted onto another slide.

Presenter achieved this with deep support for the Macintosh concept of a picture. Picture was a standard format of the platform and could contain vector drawing, bitmap art, and text in any combination and arrangement. Arbitrary visual content could be captured in a picture, placed on the clipboard, and pasted onto a slide. Pictures scaled well, so a picture could be resized on the slide and placed as needed.

Design Points: Presentations

Building presentations out of slides demanded some sort of structural similarity among slides. There was to be a graphics theme uniting slides, but in terms of content, the main structuring elements were a title and a body. Titles were to be text, of course. A body could be text or a graphics element, but text was the unifying theme Presenter could provide.

Slides had two distinguished objects: a title and a body. The titles could be extracted for help in organizing the slides. Body items, text or otherwise, weren't used for organization in the initial version of PowerPoint.

Model Slide

My main idea for unifying the look of slides was to provide a "model slide" whose attributes were copied for other slides. The model slide had two functions. It provided background art—a border, logo identification or any other consistent information—and provided the location and format of the title and body objects. Background art was the same for all slides (it could be omitted for exceptions), but title and body formatting was only the default. Any slide could change after it was created.

The background art capability was also the vehicle for providing slide numbers, dates, and similar text substitutions in a standard location.

About the same time, Aldus Corporation invented a very similar device for their new product, PageMaker. They used the name "master page", though, so by the time Presenter got to market we decided to change the name of our model slide to master slide.

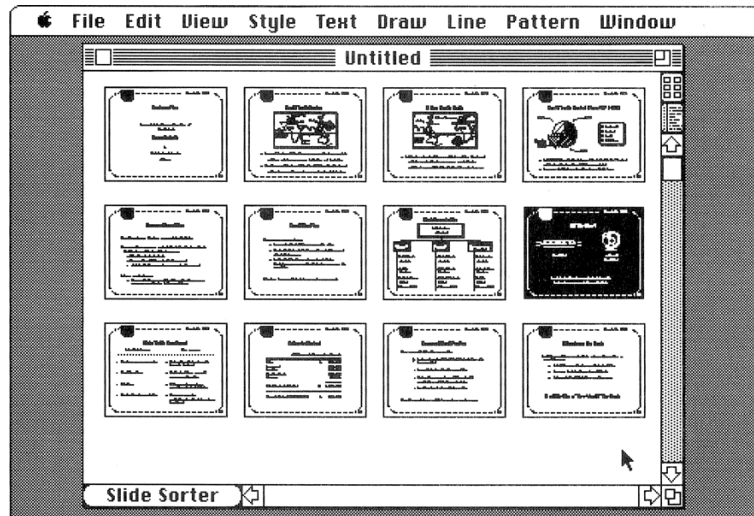
Titles

Every slide had a title. It could be empty, or invisible, but it existed. The title provided a name for every slide and Presenter gathered those names to display a list of slides by title. The view was called

Title Sorter in the original version of the product. It eventually became the Outline view when the text in the body object (if any) was included.

Slide Sorter

Thanks to a surprisingly capable graphics system in the Macintosh tool kit, it was possible to display images of slides at greatly reduced scale. That allowed a special view of the presentation showing slides as thumbnail images. The effect was much like sorting slides on a light table, an extremely novel feature at the time. The slide images could be copied, deleted, or dragged on-screen to reorder to the presentation.



The slide sorter. We considered this an essential feature of Presenter and went to great lengths to assure adequate performance.

Making this feature practical required that the program construct thumbnail bitmaps of the slides to optimize rather slow slide drawing times. The approach was practical, though, and never failed to impress new customers.

Slide Show

It was sometimes useful to give a presentation with the slides displayed directly on the screen with no other clutter. It could be used in for a one-on-one presentation and, with bigger displays in the future, for larger groups.

Scaling slides to fill the screen and hiding the menu bar were not too difficult, but Apple didn't officially support it. In fact, it was a violation of the Macintosh user interface rules. Presenter was the first Macintosh application with a convincing reason to hide the menu bar and break out of the windowing environment. I showed this to the user interface experts at Apple and lobbied them for support.

Once the menu bar and all controls were removed from the screen, the user was in a mode from which there was no obvious exit. The mouse button was used to advance the slide, so that didn't help. We settled on using the ESC key as the emergency exit. It didn't have much precedent in the Macintosh UI, but it was the standard shortcut to cancel modal dialog boxes. That seemed like a precedent.

Notes & Handouts

It was easy to take advantage of Macintosh graphics scaling to produce handout pages from the slides. Using a model handout page, analogous to the model slide, you could set up the background graphics, page numbering, etc. and then Presenter could produce handout pages with two, three, or six slides printed at reduced scale on each page.

Reusing the same idea, we created notes pages. Each notes page showed a single slide at half scale leaving the other half of the page for notes. You could flip through notes pages the same as you could slides, adding notes as appropriate. The notes pages were actually a type of “slide” in themselves, so anything you could draw on a slide you could draw on a notes page.

Both handouts and notes pages were printed at the opposite orientation from the slides. The default, then, was to print them vertically. This gave an efficient use of space on the page, and vertically oriented pages were more convenient as handouts in any case.

The Implementation of Presenter

Apple made available a native development environment for the Macintosh, but there were no compilers. There was only a crude text editor and an assembler. A superior, though expensive, option used the development environment on the Lisa. It had a reasonable command-line interpreter with a reasonable set of commands—certainly not Unix, but far better than MS-DOS. I think there was a C compiler, but the featured language was a very good version of Pascal. All the interfaces were documented in Pascal, so this was a clear favorite. I knew the language well and, in fact, had developed two different Pascal compilers in the past. Bob remarked once that he liked the idea of developing in Pascal because it felt like relying on a 1968 Mercedes-Benz.

Luckily, cash-poor Forethought already owned a Lisa. I dragged the Lisa down to my office, joining the PC and the Macintosh already on my desk, and started learning the ropes. The Lisa took up a lot of space, especially since its hard disk was external. I was lucky enough to have *two* Apple ProFile external disks, each with five megabytes, connected to the machine’s parallel port. I was crowded, and, as there were plenty of empty offices, I tore down the separation to the next cubicle and expanded my quarters. The company hierarchy, such as it was, did not exactly condone this, but I was thrilled with the extra room. It came in especially handy in later months when groups were constantly dropping in for demos.

The Apple Strategy

The Macintosh was not only a new kind of personal computer; it was a new kind of approach to personal computers. It was open to (and desperate for) third-party software, but for the system to succeed applications needed consistency in user interface. That required evangelizing the independent software developers and selling them on the benefits of standardization. It helped that Apple had developed a fairly large toolkit of services that developers could rely on and avoid building a lot of their own software. Using the toolkit saved time and it also provided a lot of the standardization automatically. The downside was that developers were accustomed to doing things exactly as they pleased. They didn’t like following rules or standards.

We, on the other hand, were not from the personal computer world and we embraced this new way of doing software. We hoped to not only save time, but to leverage the Apple marketing machine to our ends so they could use in promotions of how to write a great Macintosh application. Eventually this hope was more than realized in the form of an investment by Apple—their first in another company.

Target Hardware

As I remember it, the first Macintoshes weren’t really bad performers. Well, there was that 128K version that didn’t really work, but the “fat Mac”, the 512K version, was decent. That memory belies just how modest the hardware was by modern standards:



Apple Lisa with a ProFile hard disk on top

8 MHz processor

512 K main memory

No hard disk at all

9 inch black & white screen (no gray), 512 x 384 pixels

The core operating system was scarcely more advanced than DOS, but it had a very nice user interface toolkit and enough programming at the graphics level that there was no command line system at all.

The Lisa development system wasn't exactly a powerhouse either.

5 MHz processor

1000 K (1 Mbyte) main memory

5 MB hard disk connected to the parallel port

12-inch black & white monitor, 720x360 pixels.

I took all these limitations seriously. The 1.0 product had to succeed or there would be no second chance. Performance had to be there.

Tom Rudkin

I began development of Presenter by myself. I implemented a basic application structure and simple drawing. After that, I tackled text because it seemed like the area most difficult to get right. To my surprise and pleasure, the implementation work went fairly well and the performance of drawing and text seemed excellent.

There was obviously too little time to complete the entire application on my own, however, so, with the company finances temporarily stable, we hired another programmer. Tom Rudkin had worked for Bob back at Bell Northern, and had since been employed at VisiCorp. He and I worked very well together and our two-person team was at least twice as fast as I was working by myself.



Tom Rudkin, ace programmer, in his Sunnyvale Forethought office. The picture was taken after we had eliminated the Lisas in favor of native MPW development. (See text.)

We acquired a second-hand Lisa for Tom to use, but we had to solve the source control problem. There was no software solution at hand, but we soon invented a practical alternative.

We bought one of those card racks that hangs on the wall intended to hold employee time cards for punching in and punching out. In each of the slots we put a 4x6 card corresponding to a single source

file. When you wanted to modify a file, you pulled the card out of the rack and took it to your office. To check in, you made a dated notation on the card about what had been done and then returned the card to the rack. The actual file was, at this point, copied to our network repository.

This crude system worked remarkably well for us. We also had neighboring offices, so it was well supplemented with other communication. This was the only time in my career where I was called upon to work in such intense cooperation with a single other individual, and it will always be a highlight. Tom was a consummate professional as well as a warm, spirited and dedicated compatriot.

Presenter took about eighteen months to develop, and there were some remarkable improvements in the development system during that time. When the Macintosh Plus was announced, it brought a SCSI port for hard disk connection and a full megabyte of memory. That kind of hardware was sufficient for the new Macintosh Programmers Workbench (MPW) that ran on the native hardware. Hard disks were expensive, though, and we didn't have much cash. Glen Hobin, vice president of sales, managed to trade some FileMaker product to get a couple of 20-megabyte SCSI hard disks so Tom and I could abandon the slow Lisas.

Alas, the schedule seemed to slip away from us nevertheless. We finally employed the help of contract programmers, Bear River Associates, to help us complete the printing portion of the code. Tony Meadow and his partner had worked with Macintosh printing before, and they could work independently of our main code to help us complete the product on time. They did their work well and are fondly remembered for it.

Integer Arithmetic

I was determined to avoid the use of floating point arithmetic in the graphics calculations. I was sure that the processor was too slow to permit the constant calculation needed for real arithmetic. QuickDraw itself, the Macintosh graphics engine, worked in integer arithmetic and that certainly seemed good enough for the resolution of a ten-inch slide.

This decision led to some slightly odd features of the design, such as fixed scaling factors for the view. It did give the intended result, though: Lightning speeds for hit testing and drawing.

Compact file size

The Macintosh had no hard disk. The single floppy disk had to contain the operating system, the application program, and the data files. Compact files were de rigueur. Luckily, presentations don't generally require large data volumes. Descriptions of the graphics are generally small. The text is perhaps the bulkiest part of the data, and that isn't much compared to a word processor. The main consumers of space were pictures pasted into the presentation. The Mac format for vector graphics was very compact, but bitmap graphics take more space. Pictures bulk was unavoidable, but we did our best to keep the storage of the rest extremely compact.

To avoid wasting space with pictures used more than once, pictures were kept in a reference-counted pool. That saved memory during operation as well as file space, but introduced the possibility of reference-count errors, especially given the tricky aspects of copies on the clipboard, in the undo buffer, etc. We did have a few such errors, of course.

While open, a presentation was kept completely in memory. Main memory was fairly commodious compared to disk, so this seemed obviously right. It also made for snappy response. The problem was in managing the available memory.

This was not a virtual memory system. The OS could manage data segments, but the application controlled which segments were movable or discardable. Data had to be locked during access lest the OS move it while the app is holding a pointer to it. But data needed to be unlocked most of the time or the heap would get too fragmented.

The user could set heap space. There was a lot of data that Presenter could not allow to be discarded, but could be moved around as needed. We wanted to take advantage of more heap space, though, if it

were available. One good use was the thumbnail images displayed in the slide sorter. These took a while to draw, but we could retain a bitmap of the each thumbnail whenever memory was available. We devoted a lot of resources to optimizing use of memory for the thumbnail images.

Code segments were completely managed by the operating system, but it was up to us to decide the segmentation. We carefully planned the segments so that uncommon code could demand space when more common code could be unloaded. Printing is an archetype for this kind of analysis. Presenter's printing code needed to be loaded, and a print driver as well, but lots of UI code could be discarded while printing.

Text

Text for slides was a special problem. It was critically important that text on the screen be a faithful preview of text on the slide. This was manifested particularly with respect to line lengths and word wrap. Users would line up their graphics with their text, and that alignment had to remain the same at different scales and at different resolutions.

Amazingly, even in its primitive form, QuickDraw measured text in fractional pixels so that measurement and spacing could, in theory, be retained at different resolutions. Presenter was the first application where the theory had to work in actual practice, and it was a continuing source of difficulties.

CoreEdit

The Macintosh toolkit handled styled text, but only in the simplest form of a single run of characters in a particular font and size. There had been ambition to provide more complete toolkit support with word-wrap, mixed styles, and so forth, but it never made it to production.

The basic tools for complete text processing had been written into a package called CoreEdit, but it was used only by MacWrite and was not part of the toolkit. Apple made the source (assembler) available for licensing, but there was no support for it.

Presenter needed to have extremely high-performance text and it obviously needed to be written in machine language. CoreEdit seemed like it was the shortcut to our goal. It was indeed fast, but I found that it played fast and loose with the Macintosh rules to gain that speed. It took heavy modification to adapt it to the Presenter graphics environment and change it to follow the Macintosh standards more closely. It might have been easier to write a new text module from scratch, but I learned a lot from reading the code. In the end, PowerPoint 1.0 ran on much later Macintosh systems than MacWrite did!

Slide show

It was obviously important for slide shows to snap quickly between slides. The only way to do that was have the next slide ready in an off-screen bitmap. There was only enough memory to keep one or two off-screen bitmaps as large as the entire screen, but we could rely on each slide remaining on display long enough for us to draw the next one in the background. Backing up to the previous slide could be a problem, but there was sometimes enough memory to keep that bitmap available for a while.

Slide Sorter

To implement the Slide Sorter, we drew thumbnail images of slides off-screen and saved them in bitmaps. Presenter tried to prepare thumbnails in the background so that the slide sorter would pop quickly into view when needed. There was no multi-tasking in the operating system, of course, so this was do-it-yourself multiprocessing. When a slide was being drawn into an off-screen bitmap, it constantly checked for user input and abandoned the drawing whenever demanded.


The thumbnails also consumed valuable memory. They were simply discarded when memory was needed for another purpose, and drawn anew when memory became available again.

Shipping

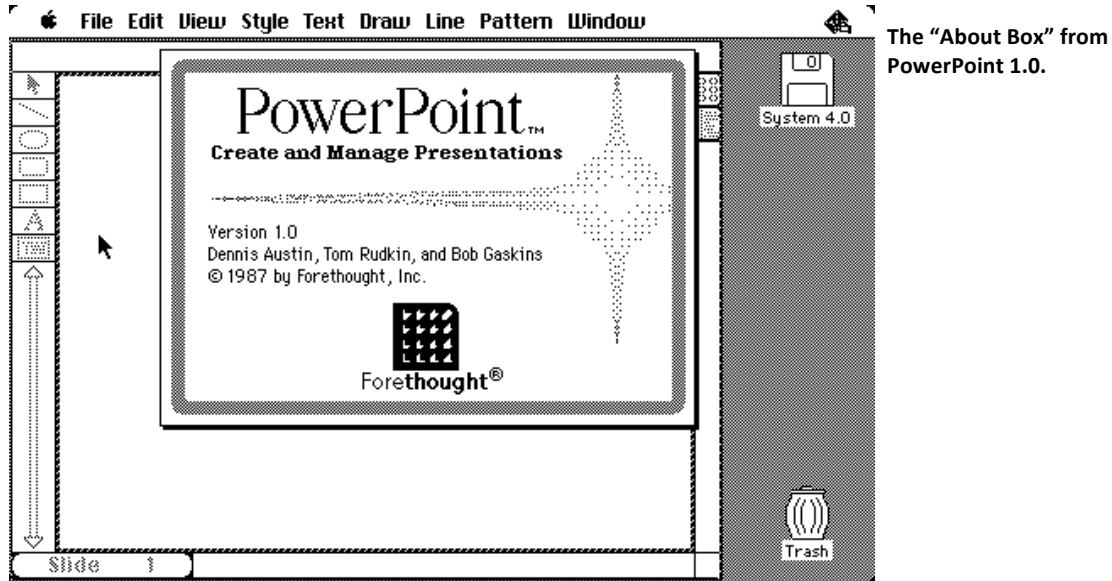
Thanks to sales of FileMaker and our investment from Apple, Forethought was able to hold together. We moved the company to more practical quarters in 1986, making such interesting compromises as offices without doors. Yes, we had individual offices. But we saved a bunch by skipping the doors.

Experience in the publishing business did indeed come in handy as we prepared to complete PowerPoint. We had good packaging already designed. We only needed new art work and, of course, a user's manual. We had the manual written by a local firm called Publishing Power, and used their expertise for many other projects over the years. The manual was a beautiful hardcover book that was bound in such a way as to lie open on a table. It gave the product a feel of permanence and quality.

We searched for a real name for Presenter. On a business trip, Bob looked out the airplane window and noted the "Power Point" runway sign just before takeoff. The name cleared the trademark search and before long we were putting the finishing touches on an About PowerPoint window for the new product.

The Macintosh style introduced a standard "About" command (in the  menu, back then). It displayed a box that was supposed to give details of the application, but invariably added some stylistic flair and gave credit to the programmers that created it. We didn't have any artists on staff, so I concocted an image myself. I drew a fixed bitmap (using MacPaint) that we could display when needed. Only the exact version number was inserted dynamically.

I tried to capture the spirit of the box art in black and white. My artistic efforts didn't draw any raves, but time was short. I was only too proud to put my name and Tom's name in the credits. On reflection, though, Tom and I realized that Bob Gaskins' name deserved to be added too. Tom and I may have created the program, but Bob created the idea and shepherded it through the violent storms of management and financing that made it happen. We built his vision.



Late in the evening on April 20, 1987 we finalized the golden master disk for PowerPoint 1.0. The official time of release was nevertheless set at 4:30 pm, and that time stamp became a tradition that was maintained through many releases to come.

Microsoft

Forethought was finally looking like a company with assets. We had a sales staff built by Glenn Hobin that was doing well with FileMaker. Our brand new PowerPoint product was loaded with promise. For our investors it looked like a great time to sell. They had been trying to put together a deal for some time. But then Microsoft heard about PowerPoint and knocked on our door.

Microsoft was a much smaller company than it is today, and they sold mostly operating systems and compilers. Their applications weren't terribly successful on the PC, but they were strong on the Macintosh. They had the best selling Mac word processor, Microsoft Word, and the innovative new spreadsheet, Excel. (Neither of these was available yet for Windows.) They also had FileMaker's toughest competitor, Microsoft File. Recently the company had become interested in the market for presentation graphics and was doing some internal concept studies.

When Microsoft learned about PowerPoint, they saw a way to get into this market instantly. Acquisitions were a new thing for the Redmond powerhouse, but something they knew they needed to learn. What's more, an acquisition in Silicon Valley was an opportunity to gain a foothold in a geographic area of great interest. The PowerPoint team would remain in Silicon Valley.

Microsoft took control of Forethought in August of 1987, only four months after PowerPoint's debut. The right to publish FileMaker did not come with the deal, so PowerPoint and its developers were the core of the purchase. A few of our people took jobs with Microsoft in Redmond, but many didn't want to move and left the company. That included the president, Rob Campbell. Bob Gaskins took over leadership of the remaining group that comprised the new Graphics Business Unit of Microsoft.

We were apprehensive, but becoming part of Microsoft was a delightful experience. We were warmly welcomed with a trip to Seattle for the annual company meeting and tours of the home campus. Our offices were outfitted with real Ethernet and other conveniences we were previously unable to afford. It was like receiving an inheritance from a rich uncle you never knew you had.

We got a new employee, Aniko Somogyi, to act as an office manager. She had been an employee of Microsoft until only recently when she and her husband had moved to the Bay Area, but now she was back in the fold. Aniko single-handedly brought us into the Microsoft culture.

Version 2

During the acquisition period I was occupied primarily with the design of PowerPoint 2.0. We had cut a number of features to meet our schedule on the first version, and those could now be completed. The most important new feature, though, was color.

A lot had happened to the Macintosh since my first spec was written. The Macintosh Plus and the Macintosh SE models were big improvements in speed and memory. More important, perhaps, they brought hard disks (SCSI interface) to the Mac world.

In early 1987 the Macintosh II was announced. This was a whole new kind of Macintosh with open hardware. It had a bus architecture called NuBus and six slots to add cards. It had the 16 MHz Motorola 68020 that brought performance to equal the new Intel 386 standard. For PowerPoint, though, the key new feature was *color*. There were color and grayscale monitors, and the video card could display 256 colors simultaneously.

The Mac II presented a big opportunity for PowerPoint because we were in a position to really showcase its color capabilities. The video card could display only 256 colors at one time (eight bits per pixel), but the card's color table could be loaded with any 256 colors. By loading the table with, say, 50 slight shades of a single color, that allowed us to display color shadings. This was unprecedented in personal computing.

I started design by picking up a few art books about color. It quickly became clear to me that ordinary users, i.e. non-artists, would have a difficult time choosing colors for a presentation. To help such

users, I designed the color system around the use of color “schemes”. The product would ship with a set of pre-configured color schemes so you could just pick a scheme instead of starting from scratch. You could still pick arbitrary colors for anything, but the schemes were a big help getting started.

By this time, we were forming a relationship with the Genigraphics Corporation to provide a way for customers to produce color slides using PowerPoint. Genigraphics was a spin-off from General Electric that specialized in production of high-end presentation graphics. They made their own workstations and film recorders and had their own staff of artists to create slides for their customers. Our joint project with the company was a special print driver that would send the output directly to Genigraphics by telephone and modem connection. Finished film would be returned to the customer by mail from the closest Genigraphics center.

To give our color schemes the marketing sheen of artistic quality, we had them designed by Genigraphics artists and we advertised them as such. We also had background designs created as a further aid. PowerPoint 2.0 did not ship until a year after the Macintosh II, but it was the first application to take full advantage of its color capability.

Back to the Future—Microsoft Windows

While I worked on completing PowerPoint 2, Tom Rudkin took up our other obvious mission—porting PowerPoint to Microsoft Windows. Thanks to the new Microsoft backing we could hire developers and expand our resources considerably. The Windows version was supposed to work just like the Macintosh and not introduce new features. Nevertheless, the job wasn’t easy.

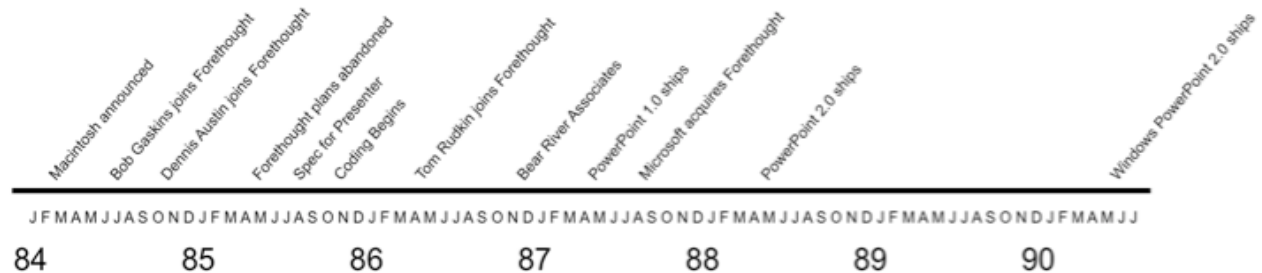
The first challenge was the programming language. PowerPoint was written in Pascal, and a lot of the newer code was written in Object Pascal. Microsoft offered a Pascal compiler, but it wasn’t very good. We thought it best to translate the entire code base into C. That wasn’t really difficult, but it is a lot of code to hand-translate.

The more serious challenge was Windows itself. Windows was now at version 2, but it was still primitive in many respects. This was particularly true for its graphics subsystem that hadn’t been stressed by applications yet. The text system and the fonts were also problems.

Working as an internal project, our team was able to provide a lot of feedback for the Windows team and it eventually resulted in Windows 3 finally containing the features we needed to imitate Mac PowerPoint. Windows PowerPoint 2.0 was announced in May, 1990, on the same day as Windows 3. That was two full years after the Mac version of 2.0 had shipped, and three years after 1.0 had shipped.

Even with more programmers and a working example, it took longer to build the first Windows PowerPoint than the equivalent Macintosh version. In terms of man-hours, it took a total of 60 to build Mac version and Mac version 2, and it took 185 to build the Windows version. (That doesn’t include the man-hours for Microsoft Graph that shipped along with it) The programmers working on the Windows project were outstanding. The additional effort was all about dealing with Windows 2.0, which just wasn’t ready for prime time.

This time line graphically shows our progress over the six years leading up to it.



From that point on, though, Windows was the lead platform. Windows 3 was much more capable and we were over the learning curve. When the Microsoft Office product was formed, PowerPoint got just the kind of market lift it needed. A great many people got a “free” copy thrown in with their Word and Excel. Slowly, people started to figure out what PowerPoint could do.

Life at Forethought

I have many memories of the years at Forethought, and lots of them don't flow well into a narrative. Here are some of those thoughts.

Bob & Rob Show

Like any startup company in Silicon Valley, people arrived at work late, but more than made up for it by staying even later. At home my family worked on a more traditional schedule. School hours were the same as anywhere else. To be sure I remained part of the family, I tried to work earlier hours than most Forethinkers.

By 6:00 or 7:00 in the evening, I would be trying to finish up a few things before heading home. About the same time, after the day's meetings and scheduled business was over, people would start wandering in to my office to see how Presenter was coming along. Two common visitors were Bob Gaskins and Rob Campbell, and I was always glad to see either of them. The problem was when they visited together.

Bob and Rob had a pretty good working relationship, but they never fully agreed on *anything*. We would start with a friendly and joking three-way conversation, but invariably ended up with Bob and Rob in some sort of two-way debate. I never wanted to take sides and alienate one of my bosses, and, besides, my goal was either to get some work done or to get home to my family. Eventually, I'd have to squeeze in a word of apology and slip out the door. The show repeated from the beginning every few of days.

Parties

Silicon Valley culture demands hard work, but hard partying is a tradition as well. My first day at Forethought had an office party to celebrate the shipping of our first Macintosh product, FactFinder. But Forethought was truly impoverished and the Christmas party that year was held at the home of Rob Campbell. Rob knew something about parties, though, and fun was generously supplied. He even showed up as Santa Claus.

In later and better times, our Christmas parties were in San Francisco. One was at the Sir Francis Drake hotel and one at the Hyatt Union Square. These were memorable occasions, but our best party celebrated the launch of PowerPoint. We took over an interesting art gallery in the Potrero Hill area, and the food and entertainment were top notch.

Later in the summer, after Microsoft's deal was firm, we had a company picnic as a final celebration of Forethought's past and Microsoft's future. It was a lavish affair that we all enjoyed immensely, but our new partners in Redmond were rather surprised when they got the bill.

Making Golden Masters

I learned a lot from Bob Gaskins, but few lessons come to mind more pointedly than the Golden Master procedure. "Golden Master" was a term widely used for the disk delivered to the duplication house that made disks for shipping. (The general culture used the word disk to refer to floppy disks that IBM insisted should be called diskettes. This minor dispute ended when distribution moved to CD's, which, because of their earlier use in audio, have always been called discs.)

Bob's experience in shipping software was recent, but he had learned the danger of trivial errors. The software may be debugged, but one little error in setting up the disk could mean the re-manufacturing of hundreds or thousands of copies. When you make a golden master, you better be absolutely certain it is perfect.

The key to perfection was twofold. First, plan and document every detail of the disk to be created and the precise procedure for creating it. Second, use redundant care in following procedure to the letter.

Once the steps were documented, we made a huge ceremony out of the actual procedure. We first recognized that *two* people were required to do the work. One would follow the steps on paper and the other carry out the instructions. The two would monitor one another's actions to be sure no one slipped up. (The two people were generally Bob and I.) We donned special robes to signify that we were engaged in a solemn rite. (Our robes were actually some old purple T-shirts from Bell Northern Research, but it's the idea that's important.) Step by step we soberly followed the instructions to guarantee the perfection of our master.

We practiced this routine several times building FileMaker releases, so we wasted no time on the night of PowerPoint's Golden Master.

The Laser Printer

Laser printing was still a new technology in 1984, but it was being commercialized at a rapid rate. Hewlett Packard introduced its first LaserJet printer that year. It was no more graphically capable than the "daisy-wheel" printers it replaced, but it made clearer letters—quietly! HP gradually improved the line by adding other fonts (bitmapped fonts) and limited graphics capability, but, for our purposes, the first *real* laser printer was announced in 1985...the Apple LaserWriter.

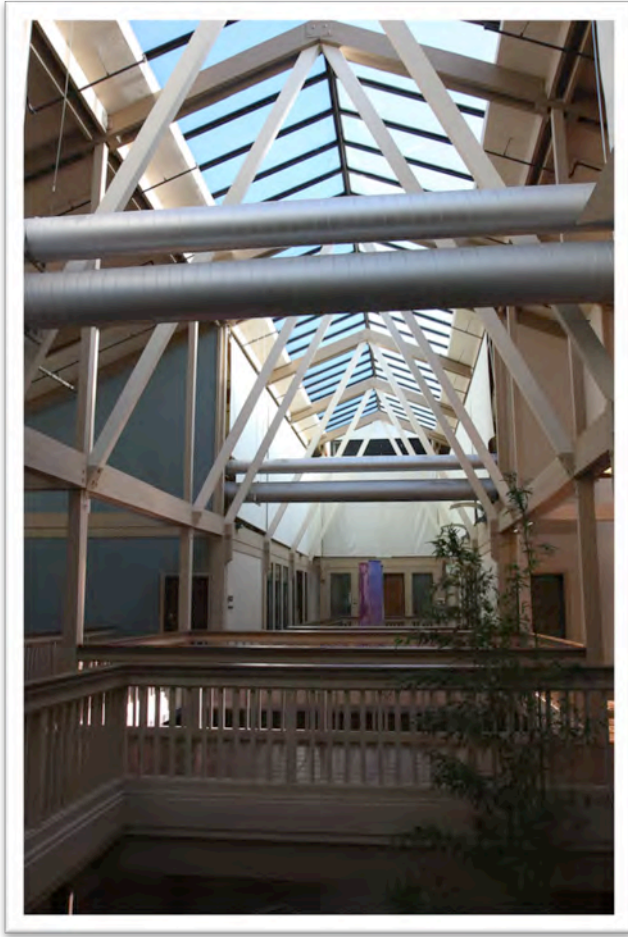
It used the same Canon print engine that the LaserJet relied on, so it was no clearer or faster. And yet it cost twice as much. The magic ingredient was PostScript. The LaserWriter had more processing power than the Macintoshes connected to it. It even had built-in networking so that all the computers using it could share without needing a separate print server.

The PostScript engine enabled the LaserWriter to print complex high-resolution graphics. Most important for us, it could render beautiful outline fonts in any size and any orientation on the page. The LaserWriter was a technological advance on a par with the Macintosh itself.

Forethought was proud to obtain one of the first LaserWriters. It was not hidden away in a utility area like most printers, but displayed prominently on a table at the entrance to the engineering area. Everyone shared it, but Bob Gaskins and I had a focused interest in it. We joked about the old days when companies displayed their huge mainframe computer systems behind glass for visitors to admire. The LaserWriter was silent (at least compared to printers of the day) and no glass was needed, but we showed it off in the same spirit.

Buildings & Offices

Forethought was originally housed in a suite at 1973 Landings Drive in Mountain View. The second floors of these buildings had big, open spaces and high ceilings. There were clear skylights that brought the sun in and made it a cheery place to work. My favorite feature was the windows that could be opened to the bay breezes. They even had screens! My view looked out to the Mountain View landfill, unfortunately, but that was in the process of being closed down. The new Shoreline Park and Shoreline Amphitheater were being built on what were once hills of garbage.



The upper floors of the Landings complex that was Forethought's first home. Forethought occupied the suite directly ahead at the end of the "hall". (Picture is from 2009.)

I liked these quarters, but they had practical problems. Offices under the great ceiling were just cubicles, of course, and sound leaked everywhere. The ceiling was a giant acoustic reflector. Not a good environment if you need quiet to be productive.

Then there were those skylights. Sunlight is nice, but it can make it mighty hard to see a computer display. There were some large pieces of cloth rigged up to ward off the direct light, but they weren't entirely successful and they looked a bit raggedy. At least the space was "interesting".

In 1986, the company was able to save some money by moving to Sobrante Way in Sunnyvale. This was a small conventional tilt-up building devoid of architectural drama. The most interesting features were a few curved interior walls we added in the build-out. The building was much more conducive to work with mostly private offices and an acoustic ceiling. The expense of all that was slightly offset by omitting the doors on the offices!



Forethought offices on Sobrante Way in Sunnyvale. Pictured in 1987 after the Microsoft acquisition.

We remained in place for a year after the Microsoft acquisition, but were soon planning an elegant new home in the Quadrus complex on Sand Hill Road, Menlo Park. We temporarily occupied some available space there while our new building was gutted and completely rebuilt to our specification.

The completed building had its quirks, but it was a beautiful place. It had rows of offices, of course, but every part of the building had its own unique character. The facility housed the Anderson Collection, one of the largest private collections of modern art. Pieces were displayed throughout our building, and the building was designed with appropriate spaces for it. We traveled between floors on an elegant central stair that helped unite the three levels. It was the best office space I have ever occupied. It was created chiefly through the efforts of Bob Gaskins, and he was rightfully proud of the achievement.



The most beautiful offices Microsoft ever occupied. Note Anderson Collection art in the lobby, including a huge three-dimensional piece by Frank Stella at center.

People of Forethought

Bob, Tom, and I continued building new versions of PowerPoint at Microsoft. But there were quite a few others at Forethought who also remained involved with PowerPoint or stayed with Microsoft in some other capacity. Here are some of those people, noted here at least partially because I had pictures.

Glenn Hobin



Glenn joined Forethought in 1986 to build our sales department. He hired a number of good people and his efforts kept FileMaker moving and Forethought in business. The sales team worked in a large room across from my office, and there would be cheers of success when a good sale was closed. The sounds of a happy sales team are music to the ears of anyone in a startup company!

Microsoft already had a sales organization, so Forethought salespeople were offered jobs in Redmond. Glenn left the company, but rejoined the PowerPoint team at Microsoft in 1991, this time working in the marketing group with Darrell.

Kathy Baker



Kathy became our receptionist at Forethought shortly before PowerPoint was completed. We hadn't had a good person at the front desk in a long while, but Kathy handled it superbly. She also kept the pulse of most other activities in the building. Part of her job became planning parties, and she was born to the job. She later played that same role at Microsoft where she made sure that fun was never in short supply.

Keith Sturdivant



Keith served as product marketing manager for PowerPoint 1.0, and did an admirable job. After the acquisition, however, he was eager to make the jump to Washington where the real estate prices beckoned.

Robert Lotz



Robert was our primary quality assurance engineer for PowerPoint 1.0 and he kept that role for the Macintosh version of PowerPoint 2.0. Then he, too, was seduced by the attractions of the north and he moved on to Redmond.

Nate Dixon



Nate was the mainstay of telephone support, and later became a QA engineer. He loved to work with computers and he worked hard. Like Keith, he took his family to Washington as soon as the acquisition was final.

Harris Meyers



Harris joined Tom and me as a developer in June after PowerPoint 1.0 shipped. This was still prior to the Microsoft acquisition. He contributed to PowerPoint 2.0 and remained with Microsoft through the next several versions.

Aniko Somogyi



Aniko doesn't quite belong in this list because she never actually worked for Forethought. She started almost on the day of the acquisition, however, and completely orchestrated our assimilation into Microsoft. She deserves to be an honorary Forethought employee.

Epilogue

By the time PowerPoint 2.0 shipped for Windows, our Graphics Business Unit was a strong organization with lots of wonderful new talent. (The people of the GBU deserve a separate story.) We knew there was a lot to do for our next version, though, including addition of a new tool bar like all the fashionable applications were sporting. We began to get pressure from the Microsoft Office group to standardize user interface elements with them. It was easy to see the value in such a standard, but there was a constant tension in allocation of resources. Is it more important to standardize or more important to remain competitive in the market race?

PowerPoint 3.0 for Windows was finished two years later. It was a tough two years, and we were glad to have it behind us. Bob Gaskins decided that the moment was right for him to bow out of the Microsoft rat race, and he retired to other pursuits.

PowerPoint was doing quite well on Windows, especially with the help of Microsoft Office. Our Macintosh customers, however, hadn't seen an update in four years. Many were defecting to Aldus Persuasion. The announcement of PowerPoint 3.0 for Windows, with a Macintosh version not *quite* ready, left many convinced that Microsoft considered them second-class customers.

Internally, though, we did take the Mac seriously. We, as well as the Word and Excel groups, were committed to a new "core code" policy where we would develop both Windows and Macintosh versions from the same source code. Every programmer had both a PC and a Macintosh, and new code had to run on both. We presumed that this would eventually pay off in time saved compared to writing two separate programs and keeping them compatible. In the end, though, the Macintosh did tend to get treated as the ugly step sister.

I remained as manager of PowerPoint development for two more versions. By that time my department alone had grown to thirty-five people. We were forced to leave our Sand Hill Road offices for more space. Without Bob's aesthetic perseverance, though, we wound up in much simpler Cupertino quarters.

In closing, I have included a chart showing the growth in man-hours devoted to releases through PowerPoint 7 (Office 95). I found the growth over ten years quite astounding. It reflects the growing

importance of PowerPoint, of Windows, and of the personal computers business. By 1995, the process of software design and development was hardly recognizable compared to our modest effort of 1985. The man-months shown are incremental. Each line reflects only the man-months for that particular version which is, of course, built starting with the work of the previous version. The 4.0 Mac version was developed simultaneously with beginning the 7.0 Windows version, which explains the small team.

Release	Date	Man-months	Developers	All depts
1.0 Mac	Apr 87	34	2	6
2.0 Mac	May 88	26	3	16
2.0 Win	May 90	185 (+Graph)	9	34
3.0 Win	May 92	300	15	60
3.0 Mac	Sep 92	50	15	61
4.0 Win	Feb 94	220	17	68
4.0 Mac	Aug 94	60	6	74
7.0 Win	Aug 95	377	21	75