MacPaint Interview and Demonstration with Bill Atkinson and Andy Herzfeld

Interviewed by:
Aimee Gardner and Al Kossow

Recorded: May 6, 2010
Mountain View, California

CHM Reference number: X5818.2010

Aimee Gardner: We're at the Computer History Museum on May 6th, 2010. I'm interviewing Andy Hertzfeld and Bill Atkinson about MacPaint and the early Mac days and QuickDraw and to get a demo of MacPaint. Anything else? Okay. Can either one of you begin by talking about the early Mac days when you were hired, and then the next one can answer? And what the applications software department was like at Apple at that time?

Andy Hertzfeld: You go first.

Bill Atkinson: Well I came on to Apple in 1978. My friend Jef Raskin encouraged me to come down there and sent me a couple of roundtrip airfare tickets said "Just come and visit for the weekend." And Steve Jobs took me around and introduced me to the 30 people. There was only 30 people at Apple. They were very intelligent people, interesting people, and passionate people. But what he did that got me to abandon my neuroscience career, I was almost finishing my Ph.D. in Neuroscience and come to Apple is he said "You know, up in Seattle where you're studying, you read about some hot new technology coming out in *Electronic Design News*. That technology really happened two years ago. And if you want to change the future, you have to be ahead of that curve and come down here at Apple where we're inventing the future." And he gave an analogy. He said "Look how fun it is to surf on the front edge of a wave and how not fun it is to dog paddle on the tail end. You go the same distance, but you got to work a whole lot harder, and it's really much more fun. Come down to Apple and surf." And so I did, and that- so when I got there, there was no Applications Software department. At the time I actually hired in as I was the Application Software department. And all we did, all I did at that time, was collect cassette tapes for the Apple II that were contributed software that users sent in and try to aggregate them together into a collection of useful little programs that people could run on their Apple IIs.

Hertzfeld: And that's how I first became aware of Bill through those very tapes that they distributed to the computer clubs. My story starts I was an Apple customer before I was an employee, so my romance with Apple began when I bought an Apple II in January, 1978. Got so obsessed with it that I was inexorably drawn there. I started as an employee in August 1979. But Bill's – on those discs you had the Lady Be Good [ph?] and Winston Churchill's [INAUDIBLE] at we called them high resolution because they were 280 pixels by 190 pixels. <laughs> Nowadays you'd call that an icon.

Atkinson: <laughs> But they were also only one bit, right?

Hertzfeld: Yeah, and only one per pixel. But yeah.

Atkinson: I was into graphics a long time ago because I actually modified a Western Union desk facsimile machine to become a scanner. I would tape a picture out of a magazine on there and spin it around and digitize it. That's how those came to be.

Hertzfeld: Yeah.

Gardner: So can you guys talk to me a little bit about how MacPaint started? Was there a concept at design meetings? Or-

Atkinson: No, no, no, no. Well first the predecessor-

Hertzfeld: I talk-

Atkinson: Yeah?

Hertzfeld: I talk about QuickDraw even before that because it's more fundamental.

Atkinson: Yeah.

Hertzfeld: You know which was-

Atkinson: That's true.

Hertzfeld: You know I don't know if you want to start but try to build a computer with a graphical user interface the most important part of software is the software that draws the graphics on the screen. So the very first part that Bill worked on long before I got involved with it, before I came to Apple was just the basic graphics routine starting with I believe drawing lines was the very first part.

Atkinson: Lines, filled areas, and text.

Hertzfeld: Yeah, text.

Atkinson: Actually drawing the text quickly was one of the most important things. It's also one of the reasons we wanted to do a graphical screen for the Lisa which turned out to be the precursor to the Mac. We didn't know it at the time. One of the reasons is we could do variable width text where an "I" didn't have to be the same width as a "W" and you could do much more graceful-looking text if you used variable width fonts. What we were using on all computers before then was a ROM [ph?] based character generator that every character on the screen occupied a fixed space, and it was generated, you would put a bytecode in memory and the ROM Would read that out as a set of dots that would make up that character, but all of those were a fixed width. It was kind of a brave experiment to see could we do a computer where there wasn't any character generated ROM. Where everything was done as graphics and where there was just a bitmap of all the pixels and in order to draw a character you had to turn on and off a whole lot of pixels, or in order to draw a line you had to turn on a lot of pixels, or to erase an area, scroll an area, these we were a little worried that there wouldn't be enough performance, or that the cost of making such a computer would be too much because you'd need more memory for that thing. So what QuickDraw I think the biggest contribution was in figuring out ways to do what needed to be done on that screen rapidly enough they could be efficiently used for real applications.

Hertzfeld: So everything you see on the screen of the Macintosh was drawn by the QuickDraw software that Bill started writing in 1979 and worked through all the way up until the Mac came out in 1984.

Atkinson: Now when you have windows you've got sort of one window and another one obscures parts of it and another one. So when you draw into one of the behind windows you have to clip it to some arbitrary area made by the shapes of the other ones that are overlapping it. And there were ways to draw clipped information of the SIGGRAPH core stuff you sort of pre-divided the line segments and things like that. But they were only about two orders of magnitude too slow to do a real word processor with, and I

had to come up with ways to efficiently draw into those, and it also kind of drew me into being kind of the window manager maker because you had to- these regions of where you would clip to had to be managed. When you drag the window around you've got to change the graphics context such that it will know okay, now I can draw on this area and now I have to subtract this other area off the others. So I ended up becoming QuickDraw and I ended up doing the window manager and the menu manager, even the event manager.

Hertzfeld: Should tell the funny story about how during the Xerox PARC demo, Apple got one demo-

Atkinson: Oh yeah.

Hertzfeld: –from the Xerox stuff, and Bill thought that he saw them drawing into behind windows, windows that weren't the topmost window. And so he knew it could be done and he worked really hard to make it. It turns out that was erroneous. They weren't doing it. He solved- he was motivated to solve the problem because he thought he saw an example but it really was a fundamental problem and they hadn't solved it but Bill did. <laughs>

Atkinson: Sometimes a little ignorance can be very motivating, and later I think one of the people from PARC said "How did you do that?" "What? I thought you were doing it?" "Oh, no, no. We were drawing the back we'd have to redraw all the other windows." "Oh, I missed that point."

Hertzfeld: It helps to be young to make breakthroughs because you don't know what's impossible.

Gardner: Do you want to make your point again?

Atkinson: So I was able to come up with a very efficient way that really didn't cost much more than drawing into a rectangle but could allow for very arbitrary shaped objects that are [ph?] clipping.

Hertzfeld: And so the very center of that in the heart of QuickDraw's capability was the status structure called a "Region" and since you guys are going to have access through the MacPaint source code through this website, you can look at the region code and see how lovingly it was designed and brilliantly it solves that problem.

Gardner: Were there any other user interface ideas that you guys really brought out?

Atkinson: One of the parts was how to do events. You've got this is really a different way, when you're doing a graphical user interface, you don't just sort of like before it was text. The computer would write some text out. Wait for the user to type something in and then take whatever they typed and respond to that in some way. With a graphical user interface, you draw things on the screen and you have to remember where they are and respond appropriately when they click on something. And I ended up there was a little bit of a disagreement about how to do event management and one person on the Lisa team wanted to do it a certain way and I thought we should have a sort of a polling loop where once the application receives the mouse down it can kind of hog [ph?] the processor to give very responsive feedback as you do. Like if you were drawing with a pen or pencil, you would press and then as you're going around you basically could hold on to the processor and then when you release then you can let go

of it. So I ended up involved with the window manager, the event manager and the menu manager. And the menu manager-

Hertzfeld: Talk about pull-down menus.

Atkinson: Oh yeah.

Hertzfeld: One of the innovations.

Atkinson: Yeah, I had a Lisa prototype in my home and there were only a few of them, and I would do user interface development at home and then ride my motorcycle in to Apple and show them show Polaroids to the Lisa team. I took Polaroids and I would bring them in. And so I actually have sort of chronicled in those Polaroids a step-by-step how we bumbled through the user interface for the Lisa which became pretty much what we had on the Mac.

Hertzfeld: Yep.

Atkinson: And at one point I had I moved the menus from being I think at that point they were at the top of the window, I moved them to the top of the whole screen so I could always have the full width if the window was stubby you didn't have to lose any menu titles. And also I'd always have the full height. If the window was down near the bottom, what do you do? Are you going to bounce it upwards or how do you get the menu to show right? By putting them up at the top and then making it so you could just move across the titles, they would kind of ruffle. It would flop down on a different one, a different one, a different one, so you could see kind of at a glance well here is the functionality that this application provides. And because the titles were short and the items were long it kind of multiplied your screen real estate by like three times but you sort of didn't realize it. They just appeared. A given menu item was always at the same place on the screen. It was sort of a kinesthetic thing. You could go and move to and start the action before you even sort of you know totally aware of what you were doing.

Hertzfeld: Because they were at the very top, you didn't have to aim. It would just- you would just go all the way up.

Atkinson: Yeah. When I first moved them from the top of the windows and the Lisa application writers screamed. They said "That's going to be so much father to reach because then we can't, you know, we can't just reach up to the top of the window. We'll have to go all the way to the top of the whole screen."

Hertzfeld: Luckily we had a tiny screen.

Atkinson: And well also I think – tell me if this is correct Andy – I think this is about the time we incorporated variable speed mouse scaling.

Hertzfeld: Yeah.

Atkinson: That if you moved quickly you got a different gear ratio. So millimeters on the table to millimeters on the screen, you'd move more millimeters on the screen if you were moving quickly and that

way you'd make sort of a quick upward move and it would move all the way to the top easily and pin there because there wasn't anywhere else to go.

Hertzfeld: Yes.

Atkinson: It was actually easier than the top of the window because you could overshoot the top of the window.

Hertzfeld: Yes, yes.

Atkinson: And so that also solved this problem of how do you point between two lowercase "l"s? You need sort of the mouse to go into compound low where a lot of millimeters on the table makes a few millimeters on the screen so you can very carefully point between them. Now I remember the Lisa team saying "Oh this is not going to work. The mouse is going to fall in your lap." You know if you move up quickly and down slowly, eventually it will fall in your lap. And it sort of does but nobody notices. Every so often they pick it up and reposition it and it worked, but there was a lot of skepticism about it at the time.

Hertzfeld: Yeah, yeah. Try, you can in the control panel turn off the mouse scaling and try using your Mac without it. I don't think you'll last more than a minute.

Gardner: Can you show us this in a demo?

Atkinson: Sure, sure.

Hertzfeld: Well yeah, let's see. If we can get at the control panel we probably can.

Al Kossow: Just quit MacPaint.

Hertzfeld: No, no, you don't have to quit it. Just control panel. It's one of the desk accessories. You should be able to- oh it wasn't even on the scaling. Now it's on.

Atkinson: Yeah.

Hertzfeld: Okay so now here you see how little I have to move the mouse to get to the top. Whereas if I do it this way, ahhhh. <laughs>

Atkinson: Yeah.

Hertzfeld: You know it seems like ten times as much work. So turn on the scaling and it's almost like just an extension of my will. I don't even have to think about it. Turn it off though…maybe for very fine work you may want it off.

Atkinson: Well you can't just always have it be the fast gear because then it can't do fine-pointing between two lowercase "l"s. This sort of variable speed gear box in there really saved the day there. I'm going to turn it back on.

Hertzfeld: We have the control panel back up there which is one of the things I wrote. I just remember from one of the early reviews of the Macintosh, a reviewer called it a “crib toy”.

Atkinson: Okay.

Hertzfeld: I was proud of that. <laughs>

Atkinson: Yeah, yeah, it’s fun to play with. <laughs> The Mac was an interesting combination of a toy and a tool. And particularly MacPaint was very much so. There are things in MacPaint that are strictly toy. You know you’ve got in Goodies you’ve got Brush Mirrors. You turn on lots of Brush Mirrors and now whenever you draw something you get a kind of a kaleidoscopic echo of it. Now that probably isn’t a productivity tool, but I saw a lot of people use it when they were on the phone with somebody and they were just doodling. And I felt that you could make something that was fun and useful. People made, you know, invitations to parties and you know memos and things like that with this tool, but it was also fun.

Hertzfeld: And to this day I think that concept is at the very heart of Apple, that work can be fun and fun can be useful.

Atkinson: Yeah.

Gardner: What sort of influence do you think this had on future software like PhotoShop? Were there any direct influences?

Hertzfeld: Oh yeah, just look at PhotoShop.

Atkinson: Well the first one, the first one would be see this thing over here called the Tool Pallete?

Gardner: Sorry, could you reanswer and say “Well the first”-

Atkinson: Ah. One of the ways that MacPaint influenced later software like PhotoShop, it was the first instance of Tool Palettes, and I know how this came to be because we were working on a graphics editor on the Lisa that was modeless. It would kind of stretch out a rectangle of marching ants, that’s a whole ‘nother story, but it would stretch out a rectangle of marching ants and then say put an oval and I was saying well this would be much better if we could see the oval when we’re stretching it out. And Larry Tesler had learned that modes are bad. You know the old text editors used to get into delete mode and now wherever you moved the cursor it would be like a machine gun. It would be like <machinegun firing sound> delete all that. And it would be better to select a range and then say delete that, or cut or copy or make it bold or do something else to it. That was modeless so select and then operate on. But that meant that the selection was kind of generic. You didn’t know what you were going to do to it, and so Larry actually had a license plate “NOMODES” [ph?]. California plate. So one day I came in and just trying to show people what I was thinking because it seemed to me that what we knew about “modes are bad” was creating a blind spot that any person who didn’t have that particular dogma would see immediately. You pick up a crayon and you’re drawing with a crayon for a while. You set it down. You pick up a pencil, now you’re drawing with a pencil. So I made a little mockup of structure graphics editor that I brought in to the Lisa team and showed them well you can dip in a rectangle tool and now stretch out rectangles. You can

dip in a select tool, now you can select things. Now dip in an oval and now you can drag out ovals. Well that was the first Tool Palette to my knowledge and it was in reaction to the problem of being in a mode, but if you're going to be in a mode, it's got to be visible; so one of those tools is highlighted big time to say "Okay, we're in paintbrushes now versus now we're in the pencil tool." So at least the mode that we're in is documented on this Tool Palette by it being lit up and also the cursor would change too, you know, which tool you were in. So a lot of programs later, you know, MacPaint was probably the first one, the first program while we used a Tool Palette like this, and other programs followed suit. MacPaint also was a very good example for people that were writing software. How do you do this new event-loop-based software where the computer is sitting there "Got anything for me? Got anything for me? Got anything for me? Oh, a mouse. Okay now what am I going to do with it?" and that kind of processing I believe Apple gave the source to MacPaint to several different developers to get them started in seeing how to make an event-based program.

Gardner: Can you explain what an event-based program is for people who might not know?

Atkinson: Okay so all computer programs, you type something into the keyboard, and then it would type something back at you. There wasn't a mouse, there wasn't a pointing device, and there wasn't any sort of cumulative history on the screen that you could point at. There were just some characters that would scroll off the top at the time called "dumb terminals" right? In that case the program really essentially was asleep until something got typed and then it got woken up and respond to the typing. Whereas in this new kind of thing, you might have to if somebody say I get the paintbrush here and I start drawing the program is saying "Did anything change? Did anything change? Oh the X and Y are different. Okay we'll make another impression of the brush." And there's this sort of this loop of the system says "Oh the mouse was pressed" and gives that to the application saying "The mouse was pressed and here's where it was pressed." The application takes it from there for a while and then hands it back.

Hertzfeld: So event-based programming is really a way of structuring your program to be very responsive to user input and to be able to apply sort of its full attention to the user's intimate interactions while that is necessary. So you can draw a beautiful picture or whatever.

Gardner: How do you think that what you guys developed helped artists? Or how did it revolutionize the way artists interacted with computers?

Atkinson: Well first of all with computer typing and typing back artists didn't use computers.

Hertzfeld: Yeah. <laughs>

Atkinson: They pretty much didn't. It made the computer an interesting tool for artists. With the one good display there's a limit to what you could do, but I saw some really fancy comics that were done by this guy in France called Gervais [ph?] and he made some really beautiful comic books on MacPaint. I think later as color was developed and shades of gray and tablets instead of a mouse, you know, the mouse was still a little bit like drawing with a rock.

Hertzfeld: And then- Oh I'm sorry.

Atkinson: Go ahead.

Hertzfeld: Your question makes me want to mention Susan Kare.

Atkinson: Oh yeah.

Hertzfeld: Who played a very big part in the development of MacPaint. In early 1983 we hired a fulltime artist because we were making a graphical computer and none of us were although Bill and Steve Capps are pretty good artists, they weren't as good as someone who's main job was that. Anyway we hired this brilliant great woman, Susan Kare right at about exactly the same time Billy started writing MacPaint, so she was his first real serious user who approached it from the perspective of an artist as opposed to a programmer. So every time Bill brought in developed a new feature he showed it to Susan and watched her use it, and I think a lot of the refinement of MacPaint came from watching an actual user, an actual artist use the program on a day-to-day basis.

Gardner: What were some of the-

Atkinson: I think I would credit Susan Kare as a co-designer of MacPaint because she used it as I was trying to write it. And I would just shut up and watch and when she would try to do something and it didn't work I'd kind of make a little note and think oh, maybe I should figure out how to make that work for her.

Hertzfeld: And she did some amazingly great art even with the MacPaint before it was fully refined.

Gardner: Wow, and what sort of insights do you think she in turn got from you guys? Did you guys mutually influence-

Hertzfeld: I think she discovered that you can make really, really interesting things just by turning on lots and lots of tiny dots on and off. At first she probably didn't think it was possible to make beautiful images so discreetly but she learned with great tools like MacPaint and with enough with small enough dots and enough of them, they can be just as expressive as any other medium.

Atkinson: She kind of found the haiku of making icons with maybe 32 pixels-

Hertzfeld: Yeah.

Atkinson: By 32 pixels. How can you draw a face in 32 pixels?

Hertzfeld: Yes.

Atkinson: So she made every pixel count and she would try here or there, move it around just a little bit. And for her part of the challenge was how can you be expressive with a very minimalist working set of just 32 pixels?

Hertzfeld: Yeah, an icon because it's 32 by 32, that's just a little over 1,000 dots, 10 24-dots and later Susan was so talented at designing those icons as a freelancer, she would charge about 1,000 dollars to design an icon so I figured out that was about a dollar a dot.

Atkinson: <laughs> But she had the right dots.

Hertzfeld: Yeah, the right dots.

Kossow: Do you want to give an example of FatBits just to show?

Hertzfeld: Oh sure, sure. FatBits will really show you the way pixels work.

Atkinson: Okay I laugh a little when I see the “Goodies” menu-

Hertzfeld: Yeah [INAUDIBLE].

Atkinson: Because I was- it wasn’t always named the Goodies menu. You know its various extra tools so I was going to call it the Aids menu, and somebody told me-

Hertzfeld: It was Susan, actually.

Atkinson: “Well, there’s this virus that’s starting to go around that could be nasty stuff. Maybe you don’t want to call it Aids.” And I had never heard of this virus but you know, a couple of different people told me that and I thought well okay, we’ll make it Goodies instead. FatBits, which actually I got license plates that read “FATBITS,” following in Larry Tesler’s tradition of having license plate to describe what you’re doing. Actually I had California license plate “GRAPHMAN” when I was working on QuickDraw, “FATBITS” when I was working on MacPaint, “HYPRCRD” when I was working on HyperCard, “GMMAGIC” when I was working on General Magic, and my current plate is “BAPHOTO.”

Hertzfeld: Yeah.

Atkinson: So my life in license plates. FatBits was a way of zooming in on the pixels and I used and we didn’t have a lot of processing power here so I used a trick that the zoom scan was always 8X, so 8X or not 8X so I could just drop in each FatBit which is eight bytes dropped into the screen so I could do that very quickly but you could scroll around with the hand and you could scroll around and you could work on things. Now I’m going to turn off Brush Mirrors here because we don’t want any of those right now. Then I’m going to take the pencil and you could set a pixel or clear a pixel and when you only have a black and white display that means you have full control. You can set a pixel or you can clear a pixel. Anything can be made. One of the challenges was when you only have black and white, how do you show that something is selected? And I came up with this solution and using animation, and so various people have called it a marquee, because it’s kind of like a theater marquee. Some people called it the marching ants because the ants seem to be sort of walking around this. And it’s actually done using an interesting technique that there’s sort of a striped mask moving behind the area of what is selected. If I were to select a full- how can I select a full area here?

Hertzfeld: Get out of FatBits would help.

Atkinson: Well I want to still see-

Hertzfeld: Okay because you can see out.

Atkinson: Right.

Hertzfeld: So get the selection tool.

Atkinson: Okay.

Hertzfeld: And drag out what you want to select.

Atkinson: Yeah, but I don't want just the edges to show. I want to actually see the diagonals moving.

Hertzfeld: Right, the last two will-

Atkinson: Let's find out what I get.

Hertzfeld: Yeah.

Atkinson: Okay now I can at least see them moving on here. And there it's kind of a funny story behind that is I was at a pub and I saw one of these Haam's Beer signs where the waterfall was flowing and I was looking at that, how does the waterfall flow? And I realized that behind it there was this pattern that was rotating around and it would just kind of meet with the picture of the waterfall and so that's how I ended up implementing what we call marching ants or the marquee.

Hertzfeld: Yeah, so stories of how the real world influences the software designer, interesting. Another one I remember that came from Steve Jobs was when Bill wrote some routines to draw ovals very, very quickly. Steve asked for "Well, that means you can draw rectangles with rounded corners." And he hadn't quite thought of that yet, but Steve was adamant. Bill says "No, no, I can't do that. That would be too hard." And Steve pointed out to him how many shapes in the real world had rectangles with rounded corners and I remember even took you for a walk outside pointed at the traffic signs and everything.

Atkinson: I think the killer was the "No Parking" sign. Okay. Well it wasn't that it was too hard. It's that it wasn't general enough to be included as a graphic primitive. But Steve convinced me and I had it working the next day.

Hertzfeld: Yep.

Gardner: Were there any other objects that you used in a graphic interface or anything like that, like a control key or any of those things?

Atkinson: Well we had an interesting discussion over the Command key, so on older computers, there wasn't the Command key. So you would type a character and the system would sort of have to understand was that getting you into some mode or was that actually typing that character because you wanted to enter that text. Right? And so on the Lisa we added a Command key so that when you hold that down and type a character, it's not typing a character. It's a whole different thing. It's a shortcut for doing some functionality. And interestingly enough on the Lisa we used the symbol on that key was an

apple, and Steve Jobs felt well that it was wrong to use the Apple logo that way, that we should come up with a better symbol and so she [sic] assigned that job to Susan Kare, come up with a different symbol. And Susan Kare looked through all her catalogues of iconography and she found a symbol used on Finnish travel maps for an outstanding feature. So we said “Okay, we’ll call it the Feature key and we’ll use this symbol.” It’s our little four-leaf clover thing. And everybody wonders what the heck is that you know? So it ended up on the Mac. Apple on the Lisa.

Hertzfeld: Yeah, I think it was Swedish.

Atkinson: Oh Swedish, okay. And then in later computers for a while you got both of them and then people would wonder was it a rocker switch? Do they press on the right or the left? Does it do something different? But the idea of the Command key and later and also the Option and the Shift keys is they are modifier keys. They don’t type something but they modify the meaning when you do do something. So when you hold the Shift key down then when you do type an “A” then you get a capital “A”. But likewise, Command A would mean a completely different thing. It would be “Select All” or something like that. Now how did we get to the command keys? I got to visit with Doug Engelbart. He’s one of my heroes. He invented the mouse back in 1968. And I got to use the mouse down in U.C. San Diego where I was doing my undergraduate work in biochemistry. I got to use a mouse there. Well I got later to visit Doug Engelbart and he had worked on this augment system which was definitely for a professional knowledge worker. It might take him several years to learn how to use this. It was pretty complicated but it had a lot of power to it. And what he said to me that kind of rung a bell he said “Look how- you know you’re all focusing on user friendly and you know, walk up to it, and intuitive. It just worked like you think it would and you don’t have to read a manual or anything.” And he says “The danger is that may end up limiting the depth that you can go or the power that you can get for the experienced user.” And he gave an example. He said “Look how hard it is to ride a bicycle. But look at the power that it gives you. It’s worth learning to ride a bicycle.” And that kind of stuck in my head and I think it was within a week we had command keys on the menus, so now like Undo has Command Z on it. Oh look. A little freeway interchange symbol. It means outstanding feature. Well these command keys shortcuts got added to the Lisa menus in direct response to this need for a way that an experienced user could get a little bit faster, and as you’ve done cut paste and copy a lot, you probably don’t go to the menu anymore. It always amazed me that for years Andy still went to the menu.

Hertzfeld: I still do.

Atkinson: You still do? Wow.

Hertzfeld: I might, not always.

Atkinson: I argued on the Lisa for actually having Cut, Paste, Copy, Undo, and Again keys on the keyboard and labeled like that, and never happened. I didn’t win that one.

Hertzfeld: For one thing Steve jobs is a minimalist when it comes to buttons.

Atkinson: But that was on the Lisa.

Hertzfeld: Yeah, but- <laughs>

Atkinson: Another whole area was Undo. If you want a user to be able to experiment and try something, you've got to have a way that it's safe. So whatever they do, and you do something like this and or you know you do something like that, "Oh, I didn't want that." There has to be a way to undo that. And on Lisa we didn't have that at first and I sort of went around to each of the application writers and said "Well what's your state like before and after you've done something? Is there anyway you could sort of remember what it was like just before you did it?" And you know ideally you want a complete track of all the changes made and back incrementally out of each of them, but the compromise that got every application to have it was that Undo would only be one level. It would Undo, Redo, Undo, Redo instead of Undo, Undo more, Undo more, Undo more, then Redo, Redo, Redo, Redo which would have been nicer. But would have had sort of unlimited storage potential if you didn't store everything differentially. So I think Larry Tesler was also a big proponent of Undo. He and I convinced the Lisa application writers that every operation you did, if it's possible, should have an Undo and that would make it safer for people to try something.

Hertzfeld: Yeah, that's probably where we didn't do a good enough job in the Mac Toolkit. Maybe it would have been impossible but we didn't support the applications in having a framework for Undo until much, much later.

Gardner: When was that?

Hertzfeld: In the original Macintosh. We asked all the applications to implement Undo but we didn't help them very much to do that. So it was one of the harder parts of writing a Mac application but then once you got into things like MacApp it did have a framework to help you at least a little bit.

Atkinson: There was another overriding design decision that was made early that was kind of a compromise that we wish we hadn't done. One way of working on a document, if you have a piece of paper and you write on it, those changes are there. And if you unplug the computer it doesn't go away. This idea of open a document into volatile memory, make changes to it, and then if you remember to write it out before things go bad or the software crashes, that was really dangerous. And I was a proponent of that as you're making changes, you're writing a little trickle, you knopw, writing what the changes are to the hard drive so if at any point you lose memory, or you lose power, it's all there. This "I've been editing for three hours and now I did something and the computer crashed, I lost three hours worth of work." That was really unacceptable. And I think it's taken a while to get more to a place where edits really are backed up as they go, and you can still revert. You can say "Go back to a previous version" so when you launch an application you can save the old copy and then be live-editing this one. And the first time I got that in was with HyperCard and that was because I could have stacks that had a million cards and now way would fit in memory of this little teeny computer. So in order to work on one, you had to be editing it in place on the disc, and I always made it so that every edit was such that you could pull the power plug and anything up to the last few seconds would all be safely on the disc.

Hertzfeld: One of the interesting things that reminds me of is just the basic challenge of developing the original Macintosh and applications for the original Macintosh was limited memory.

Atkinson: Oh my god.

Hertzfeld: We were really fighting against a very, very small amount of memory. <laughs>

Atkinson: <laughs>

Hertzfeld: That's because memory was one of the most expensive components and we wanted our computer to be affordable to ordinary people. So we worked really, really hard to get it to fit in a tighter spot than it really could. And so a lot of the fragility in Mac programs was caused by – we were really shoving more functionality in there than we could fit. It's so alien to today's world where as a programmer I never think about ROM very much unless I'm trying to do something very, very unusual. Just I mean nowadays there's literally what, I don't want to think about it but something like 100,000 times more memory-

Atkinson: I mean your iPhone has 128 megabytes on a small iPhone, and this had one-eighth of a megabyte.

Hertzfeld: Yeah. That's a thousand times the memory.

Atkinson: I know that MacPaint in its worst case-

Hertzfeld: Yeah. <laughs>

Atkinson: Had 134 bytes free. And I could drive it to that state by loading the right big font-

Hertzfeld: Yes, yes.

Atkinson: And all that and one of the challenges is how do you prove that you've found the worst case use? And what we had something called "Monkey Lives" [ph?] that's a bit you could set down in the operating system and it would when you turned that on, it would flail at it for you. It would like randomly type in lots of keys and choose menu items and drag across the screen with various tools. It was a really good way to just beat on it. You know like I think it came from a monkey typing for 1,000 years could they write the Bible?

Hertzfeld: Yeah, that was developed by Steve Capps and where it came from was he one of the jobs he had, he did a lot of different things. One of the things he had to write a journaling mechanism for our tutorials so we could make tutorials so we could sort of demonstrate the computer in action.

Atkinson: And play it back.

Hertzfeld: So make a mechanism where he could record events and feed them back. So he had the brilliant insight well, you don't have to record the events to feed them back.

Atkinson: You could make them up.

Hertzfeld: You could just make them up. And so that became a measure of robustness of Macintosh applications is how many minutes or hours could they survive the monkey. And eventually after a lot of work they could last all night. I don't think MacPaint ever lasted all night.

Atkinson: Oh yes it did.

Hertzfeld: I mean MacWrite.

Atkinson: MacPaint did.

Hertzfeld: MacPaint was-

Atkinson: MacPaint went two weeks once.

Hertzfeld: Yes. <laughs>

Atkinson: And the trick is anytime-

Hertzfeld: Oh and monkey would make interesting looking documents.

Atkinson: Drawings, yeah. You if you have any kind of imbalance allocation where you're allocating some memory and there's some path through the code where you escape without de-allocating it well then it will cumulatively build up inaccessible sort of abandoned pieces of memory, and the monkey would eventually eat up all the memory and you would crash. And so there was a really good way to test for that was can you make it through the monkey? Now I had to do some interesting techniques to deal with the small amount of memory. When code segments were loaded, you needed some code to do this job and that job. They would be loaded sort of at the first available place. But if you needed another code segment and this one would go out, it might leave a hole there that wasn't quite big enough for the next one that you needed but now you had sort of what we called memory fragmentation. That even though you had enough memory total, you couldn't load the pieces of code that you needed. And I developed a little technique for this which is setting a flat at the top of the event loop, saying that we failed and as I went to load code segments, if I failed to load one then I would beep and let it go back to the top of the event loop without doing anything. And it would say uh-oh, there's a failure here. Whereas if I succeeded and got to the right part of the code and got everything in, then I would set the flag to say we succeeded. The net result was the user would go to draw something and it would beep and they would try it again and it would work, and they'd shrug and they'd never know that they just avoided crashing the program. <laughs>

Hertzfeld: Yeah. <laughs>

Atkinson: Maybe I think maybe even the first time it didn't. It was the second time it beeped.

Gardner: I think I'll let Al go on, but I just wanted to ask you if there's anything more general we can give for the exhibit, maybe something like-

Atkinson: What MacPaint is.

Gardner: Yeah. Just maybe a really short broader kind of answer to what MacPaint is.

Hertzfeld: Sure.

Gardner: I'll leave it to Al to let you get into details.

Atkinson: MacPaint was the first widely distributed bitmap painting program, and to let people learn to use a mouse by drawing things. And it was not a structured editor, but it uses pixels and you could set or flair pixels and make drawings that way.

Gardner: Anything else you want to add?

Hertzfeld: I'd say it also was the very beginning of the thrust of development that led to what was called Desktop Publishing which was the killer app for the Mac that really got people that got the graphical user interface and the Macintosh firmly established. The fact that you can use the computer to make beautiful pictures, but then incorporate them in your word processing documents as well. That was the seed that led to all kinds- high schools, and-

Atkinson: Newsletters.

Hertzfeld: Yeah, newsletters, all kinds of organizations making beautiful documents that before required tens of thousands of dollars worth of equipment but they could just do on their desktop.

Atkinson: I remember pasting the tennis shoe into the MacWrite document.

Hertzfeld: Yes.

Atkinson: That was like the piece de resistance. See? We can mix graphics and text together.

Hertzfeld: Yeah and I have a funny story about that that's probably not worth getting into. With the bug and the clipboard stuff between there was that for a while it does to save the thing you cut as you switch applications is it basically keeps it on the stack in-between when they're transitioning and I was just subtracting however amount of memory it was I was pushing it on the stack but it could have been an odd number. Half the time maybe less than half the time because things tended to be even numbers but if it was an odd number-

Atkinson: It would crash.

Hertzfeld: They would crash unbelievably and so when they were pasting the shoe in as part of a demo and I hurdle the crashes-

Atkinson: <inaudible>

Hertzfeld: I suddenly realized what was going on. I said "Oh no." And went back and fixed it.

Atkinson: The clipper would always be an even number of bytes.

<break in recording>

Kossow: So the next thing I wanted to talk about were Steve Capps' programs that like Through The Looking Glass.

Hertzfeld: Right.

Kossow: And so I took that out of the archives.

Hertzfeld: Yeah, yeah, this is funny. There are a lot of funny stories about it but this was kind of Capps' ticket to joining the Mac team. He was still on the Lisa team when he developed – he started writing – Capps is an incredibly creative guy and he started writing all kinds of neat demos. Some of them recycled from things he did at Xerox like the famous Melting Clock or the Maze. And I don't know where he got the original idea for Alice but I thought it was such a clever idea to turn a turn-taking strategy game into a reaction-fast game. It's chess without turns. You can just kind of click it as quickly as you can. So he developed it on the Lisa and once we saw it on the Lisa we said "Oh we got to have it on the Mac." So we got him one of the prototype Macs which were pretty rare. And he ported it like instantly. It took him like ten hours or something once he got his Mac to get Alice running on the Macintosh. And of course Steve Jobs once he saw it he loved it and the first thing he thought was we got to get Capps on the Mac team so it took – but Capps was very important to Lisa. This was before Lisa shipped and Capps was the main guy writing the printing software so he actually couldn't come over until after the Lisa shipped. But there was some tension there before that. But then also Steve goes "Apple's got to sell this game." Even though it wasn't-- Apple was a little conflicted about selling games because one of the main criticisms of the Macintosh when it first came out "Oh it's a toy, it's game-like," so part of Apple wanted to sell it and the other part of him didn't but Steve promised to do deluxe packaging for it and to really promote it. And they kind of promoted it halfheartedly because they didn't really- and this was as far as I know the only game that Apple ever sold. Except nowadays of course in the app store they sell 100,000 different ones, but the only one developed by Apple I think in its long history. Do you know any other games that Apple- yeah, I don't know.

Atkinson: Well we used to have the puzzle game-

Hertzfeld: Yeah, the puzzle.

Atkinson: And you wrote that, right?

Hertzfeld: I actually wrote that. That was controversial too actually and I had to rewrite the puzzle. I developed the puzzle in Pascal just the excuse for developing it was just to show developers or to prove that you could write desk accessories in high-level language but because it was rooted in Pascal the linkers in those days weren't so good and it had to link with this runtime. It didn't know how to strip out code you weren't using. So even though the puzzle was a relatively small amount of code, it was linked with like a library that was six kbytes which sounds negligible today but was significant in those days so they were in early 1983 or so they were saying "Oh we gotta get rid" you know the marketing guys were

saying “We gotta get rid of the puzzle” because they didn’t like the Mac having something game-like but the excuse, the technical excuse was “Oh it’s too big.” You know “It’s eight kbytes,” and so I spent a weekend and I got rid of the Pascal and got it down to 680 bytes or something like that so they couldn’t use that excuse anymore and it actually shipped. Bill was actually the one who showed me how you could always win the puzzle. You showed me I think a general technique.

Atkinson: Did I?

Hertzfeld: Yeah.

Atkinson: I’ll be darned.

Hertzfeld: I wasn’t smart enough to figure that out but he did.

Kossow: So I was wondering if you could launch the game. Do you remember how to get to the Easter egg?

Hertzfeld: Alice? Probably not but-

Atkinson: Do you want to run it?

Hertzfeld: Yeah, I’ll look. Boy that launched fast. Oh <laughs>.

Atkinson: It’s in the Mac. It’s in the original Mac.

Hertzfeld: Yeah I know it’s hilarious because this thing the problem with this thing is it goes too fast even though it’s running in emulation. This thing doesn’t even have-

Atkinson: It’s just won because you were too slow. <laughs> You launched it and it won.

Hertzfeld: Yeah.

Atkinson: <laughs>

Hertzfeld: It’s unplayable. <laughs>

Atkinson: You need to insert a <coughs> a slowdown mode.

Hertzfeld: That’s a riot, though.

Kossow: When I reboot it I forgot to set the speed.

Hertzfeld: Yeah. What a lesson. So this is the Alice game we were talking about written by Steve Capps. The very first step here is to pick which piece you want to play at so if you’re a horrible masochist you could play as a pawn and you’ll get slaughtered. I always pick the queen because the queen is the most powerful piece. So at least you had a chance so you see what- what- oops! Boy it’s still pretty fast there.

Okay. So I'm Alice so I'm playing as the queen here so boy but it's still better than I- I haven't practiced Alice enough in the intervening years. Whoops it got me.

Atkinson: So unlike chess this is actually a race.

Hertzfeld: It's a reaction gave.

Atkinson: Right.

Hertzfeld: But as you- it has like a pseudo 3D and very impressive for its day, maybe not in 2010.

Atkinson: It doesn't really hold a candle to OpenGL.

Hertzfeld: Yes.

Atkinson: That didn't exist.

Hertzfeld: And one of the great things is Capps ported this to the iPhone last year so you can actually buy it in the apps store and play Alice on your iPhone. It just warmed my heart so much when I first saw that.

Atkinson: Come full circle.

Hertzfeld: To see, yeah.

Kossow: Can you bring up the-

Hertzfeld: Okay so you can bring up a menu here by clicking on the score and then maybe there's different Easter eggs on different parts of the cat here. I clicked on the cat's eye. Did that do anything? No. Do you know where do I click to get the most interesting Easter egg? I can't remember anymore.

Kossow: I don't either.

Hertzfeld: But there are built in a few different tricks. But I can't remember. Oh there we go. Now it's running upside down. Capps is very creative.

Atkinson: Is there a mode where the mouse works backwards? Where you have to move right for it to move left?

Hertzfeld: That would be good, yeah. That would make it hard.

Atkinson: That would be very Alice-like.

Hertzfeld: Yeah. But okay let me see if I can figure out how to get it back now. Maybe if I just start a new game it will flip back. Nope.

Atkinson: Oh you're stuck now. I bet if you quit and relaunched you could get it.

Hertzfeld: Now there's even a score but I can still click up there. Where did I click to make that happen?

Kossow: His teeth.

Hertzfeld: Oh his teeth. Okay I'll try that again.

Atkinson: The Cheshire Cat kind of fades in.

Hertzfeld: It's his smile. What I'm really clicking is his smile because that's the part of the- oh no, well.

Atkinson: Hee-hee-hee. Or the tail? Tip of the tail. Nope.

Hertzfeld: If I quit and start again I bet it will be okay. But anyway I really thought for one thing the team itself once we had Alice on here some people like Joanna was obsessively playing Alice. She got so good at it though that Capps claims that it ruined the game for everybody else because Joanna played Alice so much and she got so good so Capps started adapting it to make it challenging for her and he ended up making it a little too hard for other people.

Atkinson: He should have made [INAUDIBLE] so for a given user it would get tougher.

Hertzfeld: Yeah, yeah.

Kossow: The other fun thing is if you could start up the clock.

Hertzfeld: Yeah, so this is another one run by Steve Capps. Called the melting clock.

Kossow: It's a direct port of the one on the Alto.

Hertzfeld: Yeah, which I believe Capps also wrote when he was at Rochester.

Kossow: Right.

Hertzfeld: And so by today's standards maybe not that impressive, but was a great graphic demo for its time.

Atkinson: We've come a long ways.

Hertzfeld: Yes. <laughs>

Kossow: When was this designed, what year?

Hertzfeld: That was probably-

Atkinson: Eighty-three.

Hertzfeld: No it was before that because it was before he came to the Mac. So he probably wrote that in '81 or '82 but the Xerox version even before that perhaps in like 1980.

Kossow: So I did find a copy of Servant. [ph?].

Atkinson: Oh cool.

Hertzfeld: So this was a multitask game shell environment that I wrote in 1986 and 1987. I had written Switcher which was one of the first multi testing environments for the Mac. And this was my second iteration that had windows on the screen at the same time. I don't know if I -

Atkinson: I forgot about the Superman emblem on the system folder.

Hertzfeld: Yes.

Kossow: So I was wondering if you could talk about some of the features that you added to?

Hertzfeld: Yeah, well I had scrolling with momentum a little bit like the iPhone these days, like you know how you scroll through lists on the the iPhone. Here you give it a push and it keeps moving. So it lets you see that. It also had zoom functionality so you could step back and see lots of things at a smaller size or fewer things at a larger size. And then it also had these different- this different way of putting extra metadata at the bottom so I could say "Show me the sizes of the files right there. Show me the dates of the files." And then yeah, this was another zoom mode here where you could step back and see- and see the entire folder- let's see how did that work? When you hold it down you see everything no matter how big it is. We don't have a directory that's big enough here to really show it. But you typically could have had maybe 100 files in there and you have to scroll through there but this gave a way to say make it fit no matter what. And you can see now it's all fitting. We could see it better probably if we made a smaller menu. You could click on it and say okay now make it all fit and select the part you want to see. So it experimented with various viewing modes like that.

Kossow: Then there's the-

Hertzfeld: [INAUDIBLE] box.

Atkinson: How did the multiple apps work? The switching between apps?

Hertzfeld: Well I don't think there's multiple apps on this disc to run. But it works very much like MultiFinder today. They just come up in separate windows and you'd be able to switch between them.

Atkinson: It wasn't like the screen swap you did with Switcher.

Hertzfeld: Switcher, yes so this was the next evolution of it which was the more natural way of just letting the multiple windows-

Atkinson: App in a window.

Hertzfeld: Yeah.

Kossow: And then there's the integrated resource?

Hertzfeld: Yeah, yeah. And that was really one of the coolest parts I thought was taking which never really caught on. Possibly because I'd get it and Apple didn't want to use it because of that but you can just like you can open up a folder to see the files that are inside it. With Servant here, I allowed you to open up an application to see the active elements inside it. So I believe if you hold the shift key down and I double click it- no.

Atkinson: Here's the application Stuffit Expander.

Hertzfeld: Yeah. No but that's-

Atkinson: You didn't talk about the folder.

Hertzfeld: Yeah, yeah. If I open the Stuffit Expander it will probably crash. There was a way I think maybe in one of the menus. You could open it up and then I had icons representing different resources. If you double clicked on the resource like if it was an icon it would open up the icon editor.

<break in recording>

Kossow: It turns out we weren't recording the screen.

Hertzfeld: Okay.

Kossow: So could you run through just the-

Hertzfeld: Yeah, yeah. Okay so I'll just do it real fast. So this was Servant, a multi-tasking environment I worked on for the early Macintosh before the MultiFinder days. Ended up selling it to Apple. I started it in the fall of 1985 and was finished with it by spring of 1987 I think. And Apple used a tiny bit of code. You know one of the few pieces of code they used when you drag icons the outline is shaped like the icon. That wasn't how the original Finder did it, and I came up with a routine that took a bitmap and turned it into a region. That was one of the few things that Apple actually when they licensed it they paid me 150,000 dollars for it.

Atkinson: And that way you could select multiples-

Hertzfeld: And then yeah, right. That's right. But now I can't remember how to get a selection. Right shift?

Atkinson: Yeah. So shift back on another one. And another one.

Hertzfeld: Okay yeah, yeah.

Atkinson: And get a region for all three of them.

Hertzfeld: Yeah, yeah, yeah. So that was incorporated into MultiFinder and the coolest part of it, there's all these different experimentations with viewing modes like zooming here or whatever, but I thought the coolest part was extending the Finder metaphor to look inside of applications so I could select an application. This is just Stuffit Expander here but I could say instead of opening launch the application,

resource open would let me peek inside the application so we could see these little icons that represent the various parts of the application like this one here, Pics will show me different images inside this application so I don't know if they're interesting ones here or not. It's not- showing it or we can see the different strings [ph?] that are a part of it. You can see this little string install software into folder. I can edit that. Or look at the icons, etcetera. Or we can drag these resources between applications in exactly the same way you drag files between folders.

Kossow: That also turned out we didn't get any of Alice.

Hertzfeld: Oh that was another thing. Another one of the little funny little things I had in here was the watch. The hands- I made the hands spin, you can see that just briefly which I didn't audit early. There are lots of little acts.

Atkinson: Little touches.

Hertzfeld: Yeah. It's funny using, I mean that's software that I worked on 25 years ago basically.

Kossow: It's one of the things that very few people know about anymore. So I wanted to have--

Atkinson: Are you okay reaching across there?

Hertzfeld: Yeah, yeah, yeah, I'm fine. And then we just were seeing Steve Capps' brilliant game for the Macintosh, Alice, which we talked about it a little bit earlier, but the first thing you do is you get to select which chess piece you can make the moves of. I always pick the queen because I like queen's far and away the most powerful one. And then instead of chess which is normally a turn-taking game, this is a reaction-based game. I can make moves of the queen here. I can get that one before it gets me. See I'm already getting a little better at it. It got me. Anyway, it's hard to talk and demo it at the same time, but it's still kind of fun for a 25 year-old game. That's not too bad. It also shipped in this really, really nice packaging. And that was – also I think I talked about it before but it was the theme that got Steve Capps his job on the Mac team was writing such an impressive program.

Atkinson: You probably can't put this in your-

Hertzfeld: Oh and you wanted to show some of the Easter eggs like clicking on the score here opens up the Cheshire Cat is famous for fading in and out in *Alice In Wonderland* so it does here in the program. Clicking on various parts of the cat invoke various features in the program. I'll click on his ear here. I can't remember which ones do what, but I was able to make it turn upside down at one point here. But a fun program. Oh yeah, and it's great the way there's a trap door. You can go on a piece and then fall through. Every once in a while pieces of the board would disappear. And it was good if I can get that to happen so I can get in one of those you could see, there would be a nice falling animation when she fell through the board but I'm not going to try.

Atkinson: Steve Capps planted an Easter egg in the Lisa software. He was disgruntled with the architecture.

Hertzfeld: Yeah, yeah, I remember this one.

Atkinson: One time the Lisa main manager got this message on his Lisa we were pretty close to shipping and the message said “The Lisa print manager has/is fucked up.”

Hertzfeld: Yeah.

Atkinson: So then there was a code review. Everybody had to buddy up and Capps was really in the doghouse.

Hertzfeld: Yeah I wrote a part of the Macintosh ROM because we had all these managers, windows manager, menu manager. I wrote the part that handled errors that you couldn’t recover from I called the deep-shit manager, but they didn’t like calling it that in the documentation so [INAUDIBLE] changed that to the deep-sauce manager I think.

Atkinson: I thought it was dire straits.

Hertzfeld: Oh dire straits. Yeah.

Atkinson: Because they had all these [INAUDIBLE] in the software developer kit that was “ds” this, and “ds” that.

Hertzfeld: Dire straits is much better.

Atkinson: Dire straits was a little that was our- we all knew it meant deep shit. I mean you were out of memory, and you were so out of memory that you couldn’t even put up an alert saying you were out of memory. You were in deep shit.

Hertzfeld: Yes and the deep shit had special code that just relied on QuickDraw and nothing else. So even if the heap was corrupted it could still run.

Atkinson: Yeah.

Kossow: I’ll bring up HyperCard.

Atkinson: Oh my goodness.

Hertzfeld: Oh great.

Atkinson: Does it still run? <laughs> You know my wife still uses HyperCard on a G5. We have a whole set of HyperCard stacks that run my photography business and she keeps track of inventory at all the different galleries and equipment depreciation and property taxes and all that with a set of HyperCard stacks that she made and if that G5 ever dies we’re going to go buy two more of them.

Kossow: Did you want to give them some water?

<crew talk>

Atkinson: HyperCard actually wasn't going to be called HyperCard.

Hertzfeld: WildCard.

Atkinson: WildCard.

Kossow: There are even WildCard tee-shirts.

Atkinson: Yeah, and you know the reason Apple wouldn't go for the name is because there was a card that went in the Apple II and software with it that did nibble copy that defeated any copy protection and that was called the WildCard. Apple was in a suit with them and in order to get the name, WildCard for the software product they would have had to have settled the suit and they didn't want to. So I think Chris Espinoza [ph?] came up with HyperCard. And I didn't like it because "hyper".

Hertzfeld: Hype, yeah.

Atkinson: I was like you know this blowing your own horn. But that's what it shipped as.

Hertzfeld: And I think right it was Ted Nelson's [ph?] term hyperlinks.

Atkinson: And hypertext.

Hertzfeld: Hypertext, yeah.

Atkinson: Hypertext, yeah. So that's where it came from really, yeah. We had a marketing problem with HyperCard. Which is Apple didn't really know what they had on their hands and didn't know how to market it, and it was a software construction kit that would let laypersons make their own software. You didn't have to be so it kind of like opened up the priesthood, and you didn't have to be a professional software developer to make some little software that did the thing that you wanted. And we used to do that with Integer BASIC and Applesoft BASIC on the Apple II, but with the graphical user interface that sort of didn't cut it anymore. You didn't want something that just typed text out and you typed it in. And so what HyperCard was, was kind of like a software construction kit. Where you could put together pre-fab pieces. You could "I'll take this button here and that field here and this background," and you could wire them up. And it had cards that had graphics on them, text on them, links that could take you to another card of another stack, and it had this whole notion of a stack of cards so that you could have, that was sort of what an application was. It was sort of a stack of cards, and maybe it might be a suite of cards several different stacks that worked together in a suite. But it was actually, we didn't know it at the time, but it was actually a precursor to the first web browser which came six years later.

Hertzfeld: And the very first inkling I think of HyperCard I remember was the Rolodex application, remember? You wrote that at least a few months before HyperCard.

Atkinson: Oh long before that I think. I used to keep track of where things were. I had some big filing cabinets and I got tired of trying to organize things by categories or so finally I just started putting a number on a hanging folder, putting the stuff in there, typing the number on a card, and then some key words, any kind of tags I might want to be able to search for. And it would automatically keep a date on it. And that uh.. little Rolodex program which I think I got politely informed by the Rolodex Corporation that that was their valued trademark and could I use something else? And so it became I think it was named-- then it was named File--

Hertzfeld: I don't remember. I still think of it as Rolodex.

Atkinson: I remember there was we renamed it and that other name was also taken or something like that. So that was just a little program. I don't think it ever really went out. Maybe it was a giveaway.

Hertzfeld: Yeah.

Atkinson: But that was one of the ancestors of HyperCard. And-

Hertzfeld: MacPaint was obviously another ancestor.

Atkinson: Oh yeah. HyperCard had a MacPaint-like thing in it but it was-

Hertzfeld: The next generation.

Atkinson: Two layers. It had sort of a background and a card so you could actually draw card-specific stuff, or you could draw background that all the cards of the stack would share, so you could draw like the shape of a Rolodex card and then have every card use that.

Kossow: Are there any important features of the program that you'd like to demo?

Atkinson: Ah, okay. Well the first is linking. You would touch on something that would go to another- we're in another stack now. I think some of the most important things are that it's open source.

Hertzfeld: Scripted.

Atkinson: So this is kind of open source before open source was cool. You could take a HyperCard stack made by somebody else. You could open up each of the buttons and see how they worked. And in order to do that I have to set my user level. I think you bring up the message box.

Hertzfeld: You have- you mistyped it. You have two "E"s.

Atkinson: Oh, "set user"- this is sort of the access key. "set user level" maybe to five.

Hertzfeld: See I would have never remembered that.

Atkinson: Now I got more menus here. So now I can say you know get me the button tool, and open up this button and see what's in it. Here are the parameters and here's its script. You can see it says "Push

this card. Go to card documents of stack home." Now the HyperTalk language that I specified and Dan Winkler [ph?] implemented was designed so that somebody that spoke English could open up one of these buttons and sort of get the gist of how it worked, and they could modify it if you wanted to go to a different card you would change that, replace it with something else. And this was sort of a little more legible than the eventual Java Script which was the scripting language-

Hertzfeld: AppleScript which still is part of X today is very, very much like HyperTalk.

Atkinson: AppleScript descended from this and gone a lot further because AppleScript sort of sits on top of the other applications that can use each of the applications as a slave.

Hertzfeld: But it has that same English-like syntax [ph?].

Atkinson: Yeah. I had to invent for here tear-off menus.

Hertzfeld: Yeah, I remember that.

Atkinson: That way you could keep this palette around when you're using it, and you could put it away but then you could always get it back again and just tear it off again. With so we had browsing and the button tool and the field tool. How do you work on the shape of the container that you put text into as opposed to typing the text? And that's what this tool is about working with the fields and you can stretch one out. One of the problems Apple had, they sort of didn't want to tell people this is programming for the rest of us. And so they had one of their marketing campaigns was "HyperCard, but what is it?" I thought that was pretty lame.

Hertzfeld: Smart.

Atkinson: And another one had a little button that had a fire hydrant and a dog peeing on it the yellow ink where the dog was peeing and it said "HyperCard, freedom to associate." I guess I wasn't really happy with how Apple presented it to people, because they didn't quite get that normal people really cold make their own software. And millions of HyperCard stacks were created and I found a lot of really ingenious creative things came up out of it. Somebody that had a passion about some area, and it was newly empowered to be able to express that in a piece of software that they could make themselves, we come up with all kinds of wonderful stuff. One professor I think up at San Francisco, was a music professor and he made a whole course on music using Beethoven as an example, and he had the HyperCard could seek the audio CD disc to a given portion and he'd use that for his examples. I always thought it was easier to take someone who already had a pent-up passion and provide them a way to express themselves through code than it was to take somebody who was a good programmer and try to instill a passion into them. You know? HyperCard was definitely a populist program. It was designed to open up programming to the masses and make it the barrier of entry a lot less.

Hertzfeld: And the industry is missing that to some extent today even though the tools are orders of magnitude more powerful. I would love to see the analogous of HyerCard on the iPhone for example to allow an iPhone or an iPad user to create their own applications.

Atkinson: I'd love to see the same thing on the Web.

Hertzfeld: Yes.

Atkinson: Forget the iPhone and the iPad and the Mac and the Windows, it's the Web.

Hertzfeld: Yeah, sure, sure.

Atkinson: That's the platform today.

Hertzfeld: That's what Alan Kay I don't think I ever told you this but I met with Alan Kay a little bit after I started at Google and Alan Kay told me that's what you should work on Google is HyperCard for the Web. I didn't listen to him. I should have.

Atkinson: It was an authoring environment. It was a generative tool, an authoring environment whereas the Web browser that we got was sort of read-only only certain elite people could make a website, and now we've got sort of an in-between where at least you can type text into somebody- into a blog, a sort of premade website that you fill in the blanks.

Hertzfeld: Right but we don't have the end user script language.

Atkinson: Right.

Hertzfeld: I think is the essence of it.

Kossow: Unfortunately we're running out of time. I really wanted to spend more time on Magic Cap. And it follows right in with HyerCard.

Hertzfeld: Yeah, yeah, yeah. In a way Magic Cap was-

Atkinson: We should have brought a Magic Cap device.

Hertzfeld: He has one. He has one, but it's harder to demo. I remember getting beat up for the- for writing the flipping coin [ph?] because I wrote it at a time, the whole thing we wanted to have a game room because we wanted third parties to write games but the game room had to have something in it and but people thought I was fiddling while Rome burned.

Atkinson: Show them the room.

Hertzfeld: Yeah, yeah, so just the basic navigation metaphor Magic up here. You're starting at a desk. You're starting at what I call a physical metaphor to allow you real world experience to apply to the software on the screen so like if I wanted to look at addresses I'd click on the Rolodex. You can always step back, but you can step back beyond. This is just one set of functionality here on the desk. You can step back to a hallway of doors here. I always liked this picture that you can change to different things. In fact this idea of connecting the rooms with a hallway was Kevin Lynch's [ph?] idea who is now the CTO of Adobe. We had a great team at General Magic where a lot of them went on to do further things. But one

of the main things we were trying to pioneer with the General Magic software was the concept of the cloud which is kind of really finally happened I think in the last ten years this was before where we had online services powered by this interpretive language called Telescript so we need to express that in the metaphor. So just as we had the desk and you could step back to a set of rooms like here's a library where you can keep your books; I can click on a book browser or whatever. There are a variety of rooms here. You can step back further out of the house literally into here was your kind of house that represents all the stuff in your device but you could step out onto the street where there's other services like AT&Ts communication service.

Atkinson: That's out on the Web.

Hertzfeld: Or you can sign out very much like Bills current application PhotoCard, you could send these little postcards.

Atkinson: Well no we should do that on the desk.

Hertzfeld: Yeah we should probably. Sort of the key main part here, it might not because we're probably not personalized here but if you hold the Opt Shift key was it? I think the Shift key, hold that down. Yeah, it will automatically just pick a person. Then I can just click on this little postcard and notice the animations here. It reminds me of the iPhone really. In a way you can consider Magic Cap the iPhone version, you know negative five, you know, ten years before the iPhone.

Kossow: The notion of stamps?

Hertzfeld: Yeah, yeah, yeah. And the rubber stamps here where they could impart not just looks. Some of them are just decorative, but others had meaning.

Atkinson: Make a sound stamp.

Hertzfeld: Yeah a sound stamp. Let's work can we talk?

Atkinson: Hello hello hello? Hello?

Hertzfeld: In here? Yeah we can record the sound and then play it back. I'm not sure.

Atkinson: No, on a real Macintosh it will work.

Hertzfeld: Yeah, and some of the stamps my favorite ones were the animations where you can drop a little candle and work or we had sounds built in as well. So like we had songs here. See if this one works. Can play little sounds and stuff. So in away this followed on from HyperCard to build a system out where HyperCard was an application near this was the editing and stuff was ubiquitous. And Dan Winkler who wrote the scripting language for HyperCard wrote a scripting language for Magic Cap as well. We won't try to demo that now.

Atkinson: The central idea behind what General Magic was doing, one of the central ideas was the TeleCard.

Hertzfeld: Yes.

Atkinson: Which was a little digital postcard that would you could – we envisioned it as you take a picture and right on the back of it, flip it over, write something on the back of it, maybe add a little voice annotation and send it and it would show up on your daughter's pocket and unfortunately digital cellular hadn't been deployed 20 years ago.

Hertzfeld: Yes.

Atkinson: And there wasn't a really good way to get the messages around. Also the cost of the materials to make the devices was prohibitively expensive. The first ones I think were 800 dollars. I think they eventually got down to 300 but they never got down to 50 dollars so that each person in a family could have one. And then the meteoric rise of the Internet drew a lot of the top engineers that were contributed by the 20 partners drew them away to go work on the Web. But I think one of John Warnuck [ph?] was on our board and at one of our board meetings I remember him saying you know "You've got this rose, this brand new fresh idea of a TeleCard dropping out of the sky." And his argument is that you should do only that. Keep it really clean and simple. It's a TeleCard sender and receiver. And the problem was it cost too much. It cost as much as a computer and so if it was going to be a computer it had to have a whole bunch of other functions in it, be it, you know, personal information manger that kind of stuff. So we couldn't do the TeleCards back then but now 20 years later we could do it just with some software that runs on an iPhone. So Bill Atkinson's PhotoCard [ph?] is actually delivering a lot of what we had hoped to do and it has some of the DNA from HyperCard and from General Magic TeleCards. I'm pleased that people are starting to send lots of cards with it.

Hertzfeld: Yeah and you-

Atkinson: It's a dream we dreamt about for so long.

Hertzfeld: Yes.

Atkinson: And as the voice annotations and it has all the little stickers. So I remember we had something set up with this. It could send a fax.

Hertzfeld: Yes.

Atkinson: So we set up a burrito construction kit that would send a fax to the local Uno Mas [ph?] restaurant and it would have our order on it and then by the time we arrived there they would have our order ready for us. It would put together your burrito and then drive over to pick it up.

Hertzfeld: I thought I remember there was a stamp that you drop on that would make a fax but I don't see that now.

Atkinson: It's this stamp. So you've chosen a fax one. See this says "Fax".

Hertzfeld: Oh got it, got it, got it.

Atkinson: Versus-

Hertzfeld: Yeah, yeah that's the only communication service because we were not registered with AT&T.

Atkinson: They otherwise have you know send it electronically.

Kossow: That's the problem we have now with obsolete devices. There's nothing to talk to them.

Hertzfeld: Yes.

Atkinson: Yeah. The infrastructure that they counted on isn't there anymore.

Kossow: Uh-hum.

Hertzfeld: Okay, anyway, I gotta run. This was fun.

Kossow: Yep, thank you very much for coming by.

Hertzfeld: Yeah, yeah.

Kossow: Hopefully you can stop by another time and talk about Magic Slates [ph?].

Hertzfeld: Yeah, yeah, anytime.

Kossow: The whole history of it.

END OF INTERVIEW

