

## Crafting a Revolution

*Aza Raskin talks about The Humane Environment, his father (inventor of the Macintosh), and challenging the status quo.*

*This apple doesn't fall far from the tree.*

Aza Raskin left high school after 11th grade to attend the University of Chicago as a math-physics double major, where he is also principal horn in the University's orchestra. He is a lead programmer in The Humane Environment project ([humane.sf.net/the](http://humane.sf.net/the)) and a researcher at Tokyo University conducting dark matter experiments. For recreation, he is co-authoring a physics textbook because he's not old enough to buy alcohol.

**UBIQUITY:** Since we've already interviewed your father, let's start by having you to tell us about Jef Raskin from your point of view. Introduce him. We've got to refresh people's memory.

**AZA:** The range of Jef's accomplishments is truly remarkable: He was the inventor of the Macintosh, a professor of art and photography, a bicycle racer, an orchestral soloist, a model airplane designer, a published mathematician ... the list goes on for as long as one wishes to talk. But, Jef does not reminisce or dwell on the past. Instead, he focuses his energy on moving forward. For Jef, that he invented the Macintosh is nice, but he realizes that he could have done better. And does so.

What makes Jef unique is his inability to accept the status quo. When something does not make sense to him -- whether it be in computer science, mathematics or musicology -- he presses until he either understands why it is the way it is, or until he knows enough to realize that it is wrong. This is how he was able to formulate the ideology behind the Macintosh: That computers should make tasks easy for people, not the other way round. Jef's talent is in realizing when something is flawed, challenging it, and inventing something that's significantly better. His genius is being able to generalize his inspirations and create a rigorous theoretical framework in which he situates his inventions.

**UBIQUITY:** How many people have you found that have the same kind of talent he has for giving an entirely fresh look at things?

**AZA:** There are other people in the field who share Jef's talent, but not his genius. No one has ever gone back and started from the basics of human psychology and information theory to derive a quantitative means for analyzing and designing user interfaces. He is turning a guru-istic field into a science.

**UBIQUITY:** Do you have any way of describing that insight in words? It's a nice trick. How is it done?

**AZA:** I don't know that I can. I suppose that it's a matter of introspection -- You have to step back and watch yourself use a computer and ask, "What problems/errors am I having?" and "How can fix them?" Answering the first question is a matter of practice. Answering the second question requires a flash of inspiration coupled with the ability to throw out entrenched precedent.

**UBIQUITY:** What sort of classroom teacher is he?

**AZA:** Jef is a dynamic teacher: He forces the class to interact. He does not passively lecture rather he forces his students to think through the problems on their own to arrive at new solutions. Like any good teacher, he shows his students the door but they have to step through it on their own.

**UBIQUITY:** Pause for a moment and tell us about the operating environment that he created, The Humane Environment, or THE. How do you say it? T-H-E?

**AZA:** We pronounce it like the word "the." There are currently two genres in interface design: graphical user interfaces and command line interfaces. Neither is exemplary. GUIs are slow to use and CLIs are hard to learn. THE synthesizes the best parts of these two ideas into a framework that creates an interface which is both easy to learn and efficient to use. But, as such, THE represents an entirely new paradigm ... it does not look like any other system. We started by redesigning the basic building blocks of computing. The magic is that all computing tasks can be accomplished through the use of the fundamental blocks so that if one keeps scaling

in mind, the same solution that works for the restricted domains works for the entire range of computing. The first building block that we started with -- the thing that people do most often -- is text manipulation.

**UBIQUITY:** Is that true for everybody?

**AZA:** For almost everyone. There are a few niche markets, such as video editing and graphics manipulation, where the use of a GID necessarily dominates. But even people in those markets write email and surf the web. The basis of computer interaction is via text. Our current implementation of THE, therefore, focuses on the text aspect of our larger design.

**UBIQUITY:** Walk us through the process.

**AZA:** How do you move around text? Although this seems like a trivial question, none of the current interfaces have a good solution. In most GUIs, the user must take their hand from the keyboard, move it to the mouse, move the pointer to a small target, click, and return their hand to the keyboard. This takes awhile. In fact, from experimenting we know that an average on-screen cursor move takes 3.5 seconds (starting with hands on the keyboard). This may not seem like a large amount of time, but cursor motion is a frequent activity. If you move the cursor a hundred times in the course of an hour then you have spent almost six minutes, roughly 10% of your time, getting nothing productive done. Even seemingly small interface inefficiencies add up.

**UBIQUITY:** What's your alternative?

**AZA:** We have a mechanism called "Leap." Leap is an incremental search, meaning that when you want to move the cursor, you find where you want to go, hold down the Leap key, and type what you see. The cursor seems to just appear where you want it to be. Once people sit down and use our system, they never want to leave because Leap is that addictive. When I have to work outside of THE, I find myself reaching for the Leap key all of the time.

**UBIQUITY:** And so you've shown that this is the fastest way to search text?

**AZA:** Let me give you a comparison. In the same experiment I mentioned before, it took users an average of 1.5 seconds to move the cursor using Leap, yielding a 43% increase of speed when compared with the mouse. With the same hundred-cursor-moves-in-an-hour scenario, you would spend only two and a half minutes positioning the cursor, wasting only 4% of your time.

**UBIQUITY:** And you're also saying that it's the more pleasurable way to do it.

**AZA:** It's also more pleasurable. You spend less time wrestling with the cursor and more time working.

Computers are tools. When you are futzing with a tool, you get no work done. Therefore, our goal is to make computers as transparent as possible so that you don't think about using the computer, you think about the completing your task. Leap removes the burden of cursor motion, allowing you to get to any position in the text almost instantaneously. To anyone watching, it seems like magic. To a user, it becomes indispensable.

**UBIQUITY:** Explain how Leap fits into the general conceptualization of THE.

**AZA:** The fundamental idea that differentiates THE from everything else is that the world consists of exactly two distinct objects: Content and operations on that content. Things like text, images, and movies are content. Things like printing, converting to MP3, and translating to Japanese are operations. When you wish to complete a particular task you select the set of content you wish to operate upon and then issue the appropriate command to invoke the operation. If you have text selected, you can do things like move, copy, make italics, or send as e-mail. If you have an image selected, you can change its contrast, remove scratches, or send it as e-mail. You can use any command on any content at anytime. If you try to motion blur text, the system is smart enough to realize that it should be treating the text as an image and does the conversion automatically. Thus, we have done away with the idea of "applications" and replaced them with a more powerful and more humane paradigm.

**UBIQUITY:** Explain that a little bit more. How would you define an application and how would you define this as different from an application?

**AZA:** In current interfaces, tasks are needlessly compartmentalized. Say you are putting together a presentation and want to place it on your website. You need PhotoShop to edit any images you use, Excel to do a financial spreadsheet, PowerPoint to compile the presentation, Dreamweaver to create the appropriate web pages, Mozilla to check it, and an FTP client to upload it when you are done. A substantial portion of your time is flat-out wasted when you are moving content from one application to another: You are fiddling with the tool and not being productive. To make matters worse, there is the time loss and frustration from errors caused by the significant mental overhead required to switch applications, each of which has its own idiosyncrasies -- A keyboard shortcut may make text bold in PowerPoint but create a bookmark in the Mozilla; You may be able to spellcheck in Dreamweaver, but not in Photoshop. A user should not have to worry or think about what application they are in, and any habits they form using the system should not be betrayed.

If you use a web-based email account, how many times have you wanted to spellcheck your email only to be forced to either use an awkward web-interface, or transfer the text to a different application, spellcheck it there, and transfer it back? Why can't you just use a full-functioned spellcheck command right there? The same spell check (and same code) that you use everywhere else. If you want to edit an image on your website, why should you have to download it with one program, edit it with another, and upload it again? Why can't you just edit the image in the browser? There is no good reason but for the inherent limitations of the concept of applications. A user should be able to issue a command (like spellcheck or change contrast) anywhere, at anytime, and have it always do the same thing. This is truly humane. It cannot be achieved with our current application-centric computing model. Thus, in THE there is exactly one workspace in which you keep and view your content and instead of standalone applications we have command sets.

There is an interesting bonus you get with the content-operator model of computing: A decrease in system and code size. Everyone is familiar with "bloatware" -- applications keep getting larger and slower. Bloatware is due, in part, to sloppy

coding induced by ever increasing computing power, but there is a more fundamental cause. The compartmentalization of tasks into applications forces code redundancy among applications because tasks rarely fall completely within a single compartment: Word has an underpowered drawing package, CAD packages have underpowered text engines, and even Google has a calculator. Thus we arrive at the modern monolithic application mired in mediocre implementations of subtasks. When application compartmentalization is removed, so is the unnecessary code overlap: Disk and memory footprint drops, development time decreases, usability and reliability goes up.

**UBIQUITY:** Let's try out an analogy, to see if helps to formulate a question. A surgeon looks at a patient lying on the table and the patient is analogous to content. The surgeon theoretically could use one knife to do the job but usually surgeons seem to have all kinds of different little instruments. What's wrong with having lots of different instruments?

**AZA:** Nothing at all, and in fact that's what we're doing. We are placing all of those different and necessary instruments at the surgeon's fingertips so that they can be used at the precise moment they are needed (without even having to ask the nurse). To use your analogy, current interfaces look like this: If a patient needs to have their appendix removed then instead of being able to just do the operation you first have to move the patient to the appendix removal area of the hospital, and then if you want to wash and dress the wound you have to move that patient to the dressing area, etc. Every time a new procedure is undergone, the patient must be relocated to the designated part of the hospital; whereas, in our system, the surgeon is able to pick up the appropriate tools and use all of them right there. For time-critical situations, our system can save lives.

**UBIQUITY:** How far along is THE now, both in terms of conceptual development and then actual implementation?

**AZA:** We have a large specification document that outlines many of the essential parts of THE and I would say around half of them have been implemented. The core is being solidified and the API has been crafted. I am now working on creating the killer command sets (apps). As of now, THE can do things like text edit, send and

receive e-mail, compile code, etc. Currently I'm implementing the "undo" system. THE is almost to the point where it can be released to the public.

**UBIQUITY:** Eventually, it will be an environment that includes everything? In other words, will it include spreadsheet software?

**AZA:** Yes it will; Everything that you do on your computer now will be doable in THE, but faster and more easily. The goal is that once you enter THE, you will never have to leave it. And never want to leave it.

**UBIQUITY:** Now, you don't call this an operating system, correct? Or do you?

**AZA:** It doesn't necessarily need to be an operating system. Eventually, we would like for a computer to be able to start up into it, but I, for one, do not want to write drivers. For now, it piggybacks on top of other operating systems. We do not redefine any low-level structures. But it doesn't matter whether it's an operating system or not, as long as THE works, it doesn't matter to the user. For the user, the interface is the product.

**UBIQUITY:** What hardware and software do you use personally?

**AZA:** Well, we're writing the code for THE to be as cross-platform as possible, so it runs on Windows, Macs, and Linux boxes.

**UBIQUITY:** Is this an open source project?

**AZA:** This is open source available from [jefraskin.com](http://jefraskin.com). The core is free; Companies can sell command sets.

**UBIQUITY:** In terms of what will this replace, it will eventually replace almost everything, right? How long will it take to do that?

**AZA:** Yes, that is the goal: it will eventually replace everything. I would hope the text portion will be complete within 2 years and the full product be complete within 5 years. After that it is anyone's guess for the adoption rate, but judging by user

response it will be fast -- it's just a matter of having third parties write the killer command sets.

**UBIQUITY:** At the University of Chicago, where you're a student, you're a double physics and math major, not a computer science major. Explain that for us.

**AZA:** The University of Chicago is a theory-based school, so our computer science department looks like any other university's math department. If I am going to be doing math then I think that I should do it properly and pick up computer science as an application. But, to be fair, math and physics are both passions of mine; It bothers me when I do not understand how something works. When I came to Chicago I had the fear that I would take lots of classes and like them all. That fear came true. But given a strong background in math and physics I can go into any field.

**UBIQUITY:** Do you see your future to continue to be inextricably tied up with THE?

**AZA:** It's not the only thing that I think I will be inextricably entwined with, but yes, I do foresee that. I will continue and remain as one of the lead developers of THE. It is a unique thing to help craft a revolution.

*Source: Ubiquity, Volume 5, Issue 21, July 21 - 27, 2004*