# Keynote: Paul Graham, YCombinator

HackersOnBoard · HackersOnBoard · 2013-07 · source

Interviewer: Another keynote this morning and then we'll take a break. Um, our next keynote is actually Paul Graham of Y Combinator.

Paul Graham: Stage is full of robots. Something's going on here. So a lot of you guys may be from out of town. Boy, this is a lot bigger than the last PyCon I talked at. Um, last time I talked at PyCon was in 2003, I think it was the second PyCon, and it was about sort of the front half of that bit over there, the True Believers, the early converts. So those of you guys from out of town are probably wondering, um, what Silicon Valley is like. Silicon Valley is a curious thing. It doesn't have an architectural center. There's no forum that is the center of Silicon Valley. The center of Silicon Valley moves around. It is wherever there is at this moment the greatest concentration of the people who are going to make the next generation of stuff. So you're probably, you're probably guessing where I'm leading this right now. The center of Silicon Valley, you are it. So I hope you're not disappointed. This is the center of Silicon Valley right now.

In this talk, I'm going to demonstrate a curious phenomenon I've observed in all these years I have been working on Y Combinator and not on programming languages: the frightening of very big startup ideas. Very big startup ideas are actually terrifying, and the way I'm going to demonstrate this is by frightening you. Uh, I'm going to give you a list of seven gigantic startup ideas. Any one of them is so big it could make you a billionaire. And at this point, those of you who are thinking maybe of one day starting startups are waiting expectantly. By the way, you don't have to take notes. I wrote this all down. It's going to be on the web as an essay. You can just listen. All the ideas are going to be in there. But if you're thinking of starting a startup, you're probably thinking to yourselves, "All right, here it comes." But you'll see, when I say them, you'll find instead of embracing them expectantly, you'll be thinking, "Maybe I should do that recipe site after all." Don't worry, it's not a sign of weakness. In fact, arguably, it is a sign of sanity.

The biggest startup ideas are frightening, and not just because they seem like a lot of work. The biggest startup ideas threaten your identity. There's a scene in Being John Malkovich—anybody know that movie? Yeah, I highly recommend that to hackers. We have a Malkovich room in Y Combinator, believe it or not. There's like this half floor up there that nobody knows about. It's the funniest thing. This is not like the—this is not like the elevator pass. It really exists. Well, there's a scene in Being John Malkovich where our nerdy hero—um, probably difficult for you guys to identify with, but imagine—he encounters this very beautiful, attractive woman, and she says to him, "Here's the thing: if you ever got me, you wouldn't have a clue what to do with me." And

that's what these really big ideas say to you. This phenomenon is one of the most important things you can understand about startups. You'd think really big startup ideas would be attractive, but actually, they tend to repel you. And so you need to learn how to tack into that wind, essentially.

All right, ready? Okay. So suppose you want to start, uh, the next Google. Let's start with the most obvious idea. Start the next Google. The best ideas are right on just the right side of impossible, and it's hard to judge at the point where you have them whether they're on the right side or the wrong side. Right? So I'm telling you, a lot of these ideas I'm going to suggest are going to seem kind of impossible. So making a new search engine to compete with Google, that sounds almost impossible. But here are encouraging signs—or there are discouraging signs, if you were a Google user. The point where it became clear to me that Microsoft had really lost their way was when they got into the search business. So like, that was not a natural move for Microsoft. Why did they do that? They did that because they were afraid of Google, and Google was in the search business. They got, "Think we got to get in the search business. Search is the future." Well, you probably noticed Google has been getting into this social network business lately, so maybe that's a sign of something.

Now, the mere fact that Google may have peaked doesn't by itself prove that there's room for a new search engine. However, just lately, like when I'm using Google search, I find I'm kind of nostalgic for the old days when Google was true to its own slightly aspy self and was like designed like a Unix utility that would just like give you the right answers fast, right? And now, like the search results seem like they're based on the Scientologist principle that what's true is what's true for you. And like the JavaScript, man, like I feel like if I put my cursor in the wrong place, anything might happen. Like I feel like I am being A/B tested on. Right? So how do you win here? The way to win here, the way to attack all these really gigantic ideas, is to find the tiny thing that turns into the gigantic idea, right? Find the dinosaur egg. The dinosaur egg is—I just like invented a meme on the spot, that was not in the talk—the dinosaur egg is to make a search engine that all the hackers use. If you just made a search engine that the top 10,000 hackers used and you got like the first 20% of them in this room, right? If you just made a search engine that the top 10,000 hackers used and nobody else, you'd have only 10,000 users, right? Talk about poultry. And yet you would be in a very powerful position, just as Google was when they had those first 10,000 users. So make a search engine for hackers. And the good news is, if you are capable of doing this startup idea, you are one of those 10,000 hackers, right? So the solution is make the search engine you yourself want. This is a case where it's good to be self-indulgent. You want to make your search queries Turing complete? Go for it, right? You want to make it like really good for code search at the expense of everything else? Fine. Don't worry about doing something initially that will constrain you in the long term, because if you don't win in the short term, there won't be a long term anyway. So call it Hacker Search or something like that, and like make the search queries, make the UI really hard to use. It doesn't matter. If you get those first 10,000 people, you're like 5% of the way to an IPO, just like Facebook was when they got all the Harvard

undergrads, even though they probably didn't realize it at the time.

Okay, number two. Replace email. By the way, a lot of these ideas, there are people already nibbling at the edges of any big idea. There's a bunch of people nibbling at the edges of it; they just don't realize how big the piece of cheese they're nibbling at the edges of is. Email was not designed to be used the way we use it now. It was designed for like guys in a one corridor at Bell Labs to send one another a message saying like, "Want to go to lunch? I hear they have, you know, sausages in the Bell Labs cafeteria." But email is not a messaging protocol; it's a to-do list, right? Or at least my inbox is a to-do list, and email is the protocol for putting stuff on it. Here's the problem: it is a shitty to-do list. Any one of you can put something on my to-do list, right? And I don't want that. So I'm open to different solutions to this problem, but I suspect tweaking the inbox is not going to be enough. You're going to have to make a new protocol. It can like degrade to the old protocol, but I think you should make a to-do list protocol instead of a messaging protocol. And as a messaging protocol, it ought to give more power to the recipient. I ought to be able to control who can put things in my inbox. When someone can't put something in my inbox, I should just—my server should just say, "Sorry, not accepting any more things," right? Not like I send another message back to them using the protocol saying, "Sorry, I'm away now," right? Like the away message is like so obviously a sign that something is wrong. Or how about like sending emails to yourself? Okay, so like I want to know more about this thing, not just like who it came from and when. I want to know like what somebody actually wants me to do. I want to give people different ability to tell me to do different things. I want them to tell me like, do they want me to do more than just read some text? Am I supposed to do something in the physical world? When does it have to be done by? At which point can this thing just disappear 'cause it's too late? You know, God knows how many like conference invitations I have from like 2007 in my deep bury down in my inbox. This is one of these ideas that's like an irresistible force meeting an immovable object. On the one hand, entrenched protocols are impossible to displace. On the other hand, I just do not want to believe that people in the future are going to be living in the same email hell that we do now. So if it's going to get replaced eventually, why not now, right? I'm ready.

If you do this, here's an interesting one. A friend of mine was an early employee at Google. You know, he's rich, that's what that means. Seriously, when in the valley, if you're not from the valley, like "early Google employee" is like a sort of code word for rich. So I asked him if he could still stand working at Google now that he was rich and he didn't have to, and he said actually he didn't really have any problems, except he got so much email, right? And this guy is like one of the most powerful people at Google. Like powerful people are in pain because of email, right? And whenever powerful people are in pain, that is an opportunity to make lots of money. So if you build this thing, some of the most powerful people in the world will be the first to use it. Like they don't care if it interacts well with other protocols. They'll say, "You want to put something on my to-do list? You're not sending me email. You got to use this." So you don't have to worry about a chicken and egg problem, right? There you have a dinosaur chicken, whatever. Whatever you

build, make it fast. Gmail has gotten painfully slow. By the way, the original version of that sentence was "Gmail is painfully slow." I owe my partner Paul Buchheit for correcting me there. Gmail...

Paul Graham: Has become painfully slow. If you just made something like the same as Gmail but a lot faster, you could start to pull users away from Gmail with just that. People will pay for this. I would have no problem paying $50 a month. In fact, when I think about how much time I spend in email, like that's probably one of the things I'm most in denial about. Um, like it would, I would probably be justified paying $1,000 a month for this if, you know, if it would make me better, if it would make my life better. That would be like the probably the lowest hanging fruit in making my life better.

Um, all right, number three: replace universities. Now, this, this is an idea people are all over lately. Um, boy, it just occurred to me this is an instance thereof, right? Um, hacker spaces, yeah, um, with [unclear] rooms. I am reluctant to suggest that an institution that's been around for over a millennium is going to disappear just because of some mistakes they have made in the last couple decades. But certainly in the last couple decades, universities seem to be heading down the wrong path. You know, they've, they're like not fun for the students or the professors. They've turned into like expensive country clubs where you don't learn a lot. Do you study abroad? I don't think universities will disappear. I think they'll, I don't think they'll be replaced wholesale. I think they'll be replaced by a whole bunch of little things that sort of serve a la carte what those monolithic four years did. Many of them will look different from what universities look like. People won't realize they're being replaced, like people will learn stuff, right? And that, that's how they will be replaced. Arguably, Y Combinator is one, one instance of this. This is, this is an instance of it. Learning is such a big problem that if you change the way people do it, it will also have a wave of secondary effects. For example, universities are now, for better or worse, sort of a credential, like where you went to college. People ask you that fairly early on in conversations. If universities go away, like maybe credentialing has to be supplied separately. You could replace high schools, that would probably be even better. But with high schools, uh, you face all kinds of bureaucratic obstacles, and you don't want to do things in a startup, you don't want to deal with stuff that's going to slow you down. So don't go for high schools first, go for universities.

Okay, number four, uh, kill Hollywood. I'll give a few more, here's a few more details about what that would involve, 'cause by the way, that whole kill Hollywood thing, that was Paul Bucheit's idea. That was not my idea, really. You guys should give him credit for that. You guys should give him credit for that. Um, and if there's any hit men from the RIAA, like definitely Paul Bucheit, that's credit for that one. Um, I just wrote the RFS. In fact, PB was kind of amused 'cause we were like having lunch one afternoon, and he said, "You know, we should like have an RFS for like people to replace the, the big studios." And I said, "All right." And so I wrote this RFS, and he's like, the next day, he's like, "Jesus Christ, that was just an idea yesterday and now it's a meme." So

Hollywood, Hollywood's big mistake, well, one of many, um, was to be slow to embrace the internet. And that was a big mistake because I think just that, you know, you can call a winner early sometimes in elections. I think we can now call a winner in the delivery mechanism for entertainment, and it is the internet, not cable. A lot of the reason is the horribleness of cable clients, uh, what are popularly known as televisions. I did not wait for Apple TV. The last TV we had was such a piece of [unclear] that I just like got an iMac and bolted it to the wall. Like, I don't know what, how much better Apple TV is going to be. Like all I need is like some way to see the screen while I'm driving it with a wireless mouse. It's a little inconvenient, but my God, it's so much better than that, like it was from Samsung. You know, they should really not have put their brand on it because like every time I was using it, I was seeing that Samsung on the bottom of the screen thinking, "Oh, I hate this company." It was like it was designed by the same people who designed my thermostat, but they had a lot more screen real estate to [unclear] up with. All right, some of the attention that people currently devote to watching TV can be stolen by completely different things, like social networks. Some can be stolen by similar things, like games. But there's always going to be this residual demand of people who want to sit and watch some story unfold. That part's not going away. So what it comes down to, like kill Hollywood, comes down to how do you deliver drama via the internet? I'm not sure. Whatever you make, whatever you make will have to be on a larger scale than YouTube clips, though. 'Cause when you think about it, when you sit down to watch TV and movies, as we probably all still do, um, you, you either want to watch a TV show where you have like some characters you already know, you kind of want to know what you're going to get. It has to be a sort of predictable length, you know what you're going to get, you know what the characters are. Either that or it's some big movie, and you know something about the movie in advance, right? Um, there are two ways delivery and payment could play out. Either some big company like Netflix or Apple will be the app store for entertainment, I guess that would be the entertainment store, and you'll reach audiences by just getting your stuff into them, or they'll be too overreaching and inflexible, and companies will spring up to supply this kind of stuff a la carte to the producers of this, this drama. If that happens, there will be a need for infrastructure companies too, so we'll see.

All right, speaking of Apple, um, number five: a new Apple. Now, I was talking, I was talking recently to someone who knew Apple well, um, I won't give any more details, and I asked him if the people now running the company would keep being able to like come up with new things the way they used to when Steve Jobs was running it. And he said, "No." Now, I sort of worried already that that was going to be the answer, and I asked him more to see how he would qualify it than what the actual answer was, but you will notice there were no qualifications. He just said, "No, there will be no, no new stuff," at least not that's good, um, beyond whatever is currently in the pipeline. And even that they'll argue about what to call it. No, actually he didn't predict that. So, um, so how do we do this, right? Um, if Apple is not going to make the next iPad, who is? Well, um, empirically, the none of the existing players are going to make it, right? Because none of the existing players are run by product visionaries. And empirically, the only way to get a product

visionary as the CEO of a company is for him to start it and not get fired. So what that means, like unbelievable as it sounds, this is like one of these results where you follow some logical chain of reasoning and end up with this bizarre, bizarre consequence. What that means is the next Apple is going to have to be a startup. And that sounds preposterously ambitious, but no more ambitious than it was for Apple. They became as big as Apple, right? And someone taking on the problem now has an advantage that Apple didn't have. They have the, the example of Apple, which is valuable in two ways. One, um, Steve Jobs showed us, like Roger Bannister did, what one person can do. If you really try, you can do a lot better than some suit running some electronics company. And like you have, like you have the product manager to decide what the features are, and then you give it to a designer. No, no, no, you can make things much better than that, right? The other way he showed us what's possible is like the way Augustus did it, if you know what I mean. Um, okay, Steve Jobs, like there is definitely like a gap around here in Silicon Valley. Steve Jobs like unrolled the future like a carpet. I don't know about you guys, but I had sort of unconsciously signed up for just like whatever they were going to make was whatever I would be using in the future, which is why it was so shocking when he died, because now, like who's going to make the future, right? Well, whoever it is, I don't know, but please apply to Y Combinator. Now that Steve is gone, there is a vacuum we can all feel, which means if a new company led the way, just like said, "Okay, we're going to lead the way into the future of hardware," people would follow, because they're used to following now. The CEO of that company, the so-called next Steve Jobs, might not, you know, he might just not quite live up to Steve Jobs, but, um, he wouldn't have to. He would just have to be better than Samsung and HP and Motorola, right? That doesn't sound quite so hard, does it?

Okay, number six. All right, you guys are going to hate this one, except for the like 30 or so of you who are like, "Oh my God, this is what I've been working on." All right, number six: bring back the old Moore's Law. The last 10 years, um, have reminded us what Moore's Law actually said. It didn't say computers were going to get twice as fast every 18 months. It said circuits were going to get twice as dense. I can remember when it used to seem pedantic to point that out, right? But not anymore. Um, well, this Moore's Law is not as good as the old one. The old Moore's Law used to mean that if your software was slow, all you had to do was wait, and the, I remember this, it was the greatest. We'd say, "Should we work on optimizing it?" It's like, "Whatever, we'll just wait for the next generation of processors." You know, hardware would just like solve software's problems. Well, now, now we have to work, right? Now, if your software is slow, instead of just waiting, you have to rewrite it to do more things in parallel, and that is a lot more work. No one knows better than I do. Um, it would be great if a startup could give, I mean really great, if a startup could give back something of the old Moore's Law by making a lot of CPUs look to the developer like one real big fast CPU. There are several ways to do it. The most ambitious, and I'm not even sure this is possible, we'll see, it's, it's all going to be a matter of degree, but the most ambitious way is to do it automatically by writing a compiler that like takes programs written for, written the old way, and turns them into programs for the new hardware. I mean, that is what compilers are supposed to

do, right? Um, so problem is, there's a name.

Interviewer: For this compiler, um, this efficiently smart compiler, and it is a byword for impossibility. Um, so is it really impossible, though? Is it really the case that no configuration of the memory, of the bits in memory of an existing, of a present-day computer is this compiler? Because if you really think that, you should, by formalizing certain aspects of it, you should try to prove it, because that would be an interesting result. And if it's not impossible but simply very hard, the expected value of working on it might be pretty high. Um, the reason the expected value is so high as web services, if you could write software that gave programmers the convenience of the way things were in the old days, you would get all the users, 'cause programmers like convenience. Um, just suppose there was a processor manufacturer that had continued to be able to translate increasing circuit densities into increasing clock speeds, right? Like Intel would be worried, they would take all of Intel's business. Well, now in the world of web services, you don't see your processors anymore, which means if you had the software, it would be indistinguishable from the user's point of view from you being that manufacturer, right? That's what's at stake. Um, okay, the less ambitious way of approaching this problem is to start from the bottom, um, and instead of like having the compiler do everything for you, you build programs out of more parallelizable Lego blocks like Hadoop and MapReduce. Um, that's obviously doable, it's happening. Um, then the programmer still does most of the work, and there's probably lots of startup ideas even there. Um, there is an intriguing middle ground where you build a semi-automatic weapon, um, where there's a human in the loop. You look, you build something that looks to the user like the sufficiently smart compiler, but actually there's humans in there optimizing [unclear]. Um, these people might be your employees, but it might be even more exciting if you created a marketplace for optimization. Now, of course, suppose there was a marketplace for optimization—you guys are probably already, some of you guys are probably already thinking about what I'm going to suggest next—people would start to write bots, right, to do the optimization. And if you ever got to the point where it could all be done by bots, you would be in the very curious situation of actually having created the sufficiently smart compiler, but no one would have a copy of it, right? Um, that would be pretty interesting. I mean, that would be worth doing just because it would be so interesting.

Paul Graham: All right, so I realize, believe me, I realize how crazy all this sounds. Um, there was division among the Y Combinator partners whether I should even mention this one. Robert Morris thought I would sound like an idiot if I even brought it up. Paul Buchheit and Aaron Iba said, "Eh, what the hell, you know, maybe it'll work." But what I like about this idea is all the different ways in which it's wrong. The whole idea of focusing on optimization is counter to the general trend in software development for decades, right? Trying to write the sufficiently smart compiler is by definition a mistake. And even if it weren't, compilers are supposed to be the sort of software that's written by open source projects, not startups, right? So the forum troll, which I have by now internalized, um, doesn't even know where to begin in raising objections to this idea. But I'm sure

I'll find out this afternoon on Hacker News. Uh, that is what I call a startup idea. But wait, there's one that will face even greater resistance. This is my, the last one, the list number seven, ongoing diagnosis. One of my many tricks for generating startup ideas is to imagine the ways we will seem backwards to people in the future. And I am pretty sure that 50 or 100 years from now, it will seem barbaric to people that we had to wait, um, until we had symptoms to be diagnosed with things like heart disease and cancer, for example. In 2004, Bill Clinton found he was feeling short of breath, um, and went to his doctors, and they said, "Your arteries are over 90% blocked," and 3 days later he had a quadruple bypass. Now, it seems reasonable to assume that Bill Clinton has access to the best medical care available, and even he had to wait until his arteries were 90% blocked to learn that the number was 90%, right? Um, that can't be that hard to figure out with some kind of non-destructive testing. I mean, maybe version one could do destructive testing, I don't know. Um, you know, version one, maybe I shouldn't be funding software startups doing medical devices, um, just get something out quickly, launch fast and iterate. You could do heart disease for pigs first, um, and you could like run a sausage company on the side to monetize it. This is, this is not in the talk, um, but like surely at some point in the future going to know this number the way we know our weight, right? Um, so why don't you start a startup to do that? Cancer too, it seems crazy that you have to wait until you have some lump and you go to the doctor and you say, "I have the lump," and they say, "Oh, it's cancer," right? Um, I think what'll happen is pretty soon we'll have some sort of radar screen that cancers will show up on much more quickly. It might not be cancer as we now think of it. I wouldn't be surprised if at any given time we all have tens or even hundreds of micro-cancers going on at any given time and we just don't know about it, but like we'll know, right? But it won't be the end of the world.

Paul Graham: So one of the obstacles, the big obstacles to come from this—well, there's so many obstacles, some of the big obstacles to come to this idea will come from the medical profession. This is not the way the medical profession has traditionally worked, right? Traditionally, you feel symptoms and you go to a doctor, and the doctor figures out what's wrong. A lot of doctors are rather alarmed at the idea of doing what lawyers call a fishing expedition and going to look for problems that you don't even know are there, um, problems you don't even know what kind of problem you're looking for. They have a name for the things that you discover when you do this: incidentalomas. And doctors regard them as kind of a pain in the ass. For example, a friend of mine once had her brain scanned as part of a study and was horrified to discover what appeared to be a large tumor in her brain, and, uh, after further testing it turned out to be a harmless cyst, but she had a couple weeks of terror there. A lot of doctors worry that if you start testing people all the time, this is what you'll get, um, a lot of terrifying false alarms that make patients panic and require expensive and possibly dangerous tests to resolve. But I think that that is just an artifact of current limitations. I think the way it's going to play out is if people are scanned all the time, they will know about these things early and they will realize, "Oh, that thing in my brain is just like a birthmark, it's always been there." There's room for a lot of startups here. The obstacles they face will be horrific. In addition to the technical obstacles all startups face and the bureaucratic

obstacles all medical startups face, you'll be going against thousands of years of medical tradition, but it will happen and it will be a great thing, um, and people in the future will feel as sorry for us as we for generations before who didn't have anesthetic or antibiotics.

Paul Graham: All right, so now I'll give you a little tactical advice. How much time do I have left? Anybody know? Oh, I, um, how much, enough. I, um, how much, enough. I, um, how much, enough. All right, um, I'm going to conclude with a little tactical advice. If you want to take on a problem as big as the ones I have been talking about, do not make a direct frontal attack on the problem. Don't say, for example, that you're going to replace email, um, or your employees and investors will constantly be saying, "Are we there yet?" and haters will be lined up waiting to watch you fail. Just say you're building to-do list software, what could be more harmless? They can notice that you've replaced email when it's a fait accompli, just as people will notice about, uh, iPads replacing Windows. Empirically, the way to do really big things seem to be to start with small things and, um, grow them bigger. Want to dominate microcomputer software for decades? Start by writing a BASIC interpreter for a machine with a couple thousand users. Want to make the universal website a giant vacuum for people's time? Start by building a website where Harvard undergrads can stalk one another. Empirically, it's not just for, uh, other people's sake that you need to start small, you seem to for your own sake, too. Neither Bill Gates nor Mark Zuckerberg knew how big their companies were going to get. All they knew was that they were on to something. Maybe it's a bad idea to have really big ambitions initially, 'cause the bigger your ambitions, the longer they're going to take to realize, and the longer you're projecting into the future, the more likely you're going to be wrong. There are plenty of examples of that, too. So I think the best way to do these big ideas is not to try and identify a precise point in the future and say, "How do I get from here to there?" like the popular image of a visionary. I think a better model is Columbus, who thought, "There's something to the west, I'll sail westward." Start with something that works, that you know works, that's small, and then when the opportunity comes to move westward, right? Um, the popular image of the visionary is someone with a very, very precise view of the future, but empirically it's probably better to have a blurry one. So, um, that's my whole talk. Am I done? Do I, am I out of time, or should I take questions, or what? How much time do I have left?

Interviewer: You can do whatever you want.

Paul Graham: No, I don't want to do whatever I want. I want to do what you want. What do you want, right? Um, California, yeah, whatever you want, whatever is true for you. All right, you've got nine minutes. Nine minutes, they say. That's what I wanted, a number. Okay, if anybody has any questions, I will now answer them. Anybody have any? Yes, yell, I'll repeat the question as long as I can hear you. It's cool. All right.

Interviewer: Uh, in one of your recent, uh, things online, you said the answer for the music

industry, for example, is probably to give up insisting on payment for recorded music and focus on licensing and live shows. Yeah. All right, I would like, I've been talking to Vint Cerf and some people about this. Licensing is exactly what, when this licensing is exactly what, when copyrights mean nothing, you know? The internet and software are in a viable business terrain only because software enjoys these copyrights. So how can you just sort of throw away the music industry's product and copyright and yet tell all these people to work on things that must be protected by copyright, trademark, a form of copyright and trademark, a form of copyright, and process patents?

Paul Graham: Well, I don't think, um, you necessarily have to protect your stuff. I don't think if you build technology you have to protect it with...

Interviewer: Copyright, I mean Amazon Web Services isn't protected with copyright. Um, I think selling copies of stuff, like if you simply do that, like you make a copy of stuff and you sell lots of it, it's just not practically going to work. Um, you know, the funny thing is I was going to write an essay about this, but maybe I'll just like make it up on the fly. Um, the funny thing is like what is property is actually historically has been somewhat defined by what it's convenient to be property, right? So like in the days of hunter-gatherers, it was not convenient for land to be property, right? But now it is, and so now land is property. If you imagine that we lived like on the moon and everything, and you know, we had to get like air in pipes and paid for the air, right, it would—people could charge for smells, people could charge for good smells, right? And so it would seem reasonable for like smells to be property. But now, like you walk by a restaurant and you smell this like delicious smell, you get like this free boost for nothing. And like someone coming—I think the record labels are like these people who are from the moon, right? And they used to be able to sell these things because it was—that was the only way you could get them was through their channel. But now, like files move around like smells, and it's just not convenient to charge for them. Ultimately, this stuff is pragmatic. I realize that doesn't sound very principled, but historically it seems to be the way things work. You just can't charge for copies of stuff anymore, you know? It just doesn't practically work anymore.

Paul Graham: So you say you cannot charge for copies of software?

Interviewer: It's the same copyright protection.

Paul Graham: Yeah, I know. I don't think—I don't think the same copyright protection—so you can't just say that's too inconvenient. I don't think it works as well as it used to. I mean, there are some people who pay for copies of music, right? And then a lot of people don't. And there are some people who pay for copies of software, and a lot of people don't, right? So maybe you will be able to make some money by making copies of stuff, but you won't be able to print money the way the labels used to back in the 80s.

Interviewer: Music, this is also taking over books and movies.

Paul Graham: Well, you know, as a writer of books, I feel like writing books is—I mean, it never made that much money. People did it mostly out of vanity, right? And so now the ratio of vanity to money is even higher. All right, do we have another question? Yes.

Interviewer: Okay, my question was in regards to your third or fourth idea about killing the universities.

Paul Graham: Yeah, replacing universities, not killing them. They're just going to go to sleep, I feel like.

Interviewer: Um, your objections to the university as it stands are missing some of the actual value they provide to society instead of just being like certificate mills that you're paying for. Um, I feel like it's going to be very hard to like make, you know, a group of hackers who are going to sit down and parse like all the data from LHC for years and years, or work on theories of postmodernism, both of which are, you know, equally valid areas of research. And also, I feel like a lot of universities, their value is in the social networks that you create in real life and not just the, you know, the Facebook ones you have. And I was wondering, you know, wouldn't a world without physical universities kind of contribute to a, I don't know, culture of alienation by means of technological abstraction? Wouldn't a future without physical universities contribute to a what?

Paul Graham: So a lot of my elite friends are complaining about how, you know, no one talks to anybody else anymore 'cause you're always on Twitter and, you know, texting or whatever. So the problem is, would it be bad if people don't physically meet? Is that what you're talking about, right? It's the physicalness of universities, that people all get together at the universities, and Twitter is not a substitute for that. Well, let me tell you, I'm all in favor of people physically meeting. Every 6 months I make 150 people move to California and they can't get out of it. So I definitely agree with you, things have got to happen in person, right? We like to fund startups here, and like all you guys have moved here in person, right? You're not just talking to one another on Twitter. You can like shake hands with the person next to you. So definitely I agree with you, people have to meet in person, but you can make things that involve people meeting in person. All right, do you have any more questions? You want to yell? Just yell. Okay, yes.

Interviewer: This works. So, there is a very interesting phenomenon in universities at the PhD level where doctoral education is essentially a distributed phenomenon. It's not so much which university you received your PhD from, it's who signed your dissertation, the three or four people that were on your PhD committee. Yeah. So, a very disruptive idea may in fact be—and I guess I'm also giving a free idea to everybody in this room—is it possible to make baccalaureate, so lower

level of—lower levels of university education, is it possible to make that a distributed phenomenon as well, where your credentials are from your adviser, from your lab, from your research projects?

Paul Graham: Yeah, that's an interesting idea. That's an interesting idea. The idea is instead of getting a degree from an institution, you would get it from a person, which is actually, if you know about the history of universities, that's how it originally used to work. I mean, universities had a guild system, and you would be certified as a teacher by an existing teacher, just like you could be certified as a woodworker by an existing woodworker. So, since it was the way they used to work, maybe universities or their replacements will be returning to their roots. I'm not saying like get rid of universities, just like eat away at that big lump of cheese. Any more questions? Yes, shout and I'll repeat it.

Interviewer: So, I know you're joking earlier, and I read your work and I'm inspired by a lot of your essays, but you're in a position to fund some things that very well might change the entire world, about medical devices and things like EEG reading and ability to manipulate very complicated devices off a series of—I mean, the technology is still in its infancy. My concern is, what are your principles on funding something that very well may significantly affect like our genetic code, our, you know?

Paul Graham: Wow. Okay, so the question was, what are my principles on funding something that could affect our genetic code? So that's definitely a fundamental question. Honestly, I hadn't thought about it. I try and fund founders who seem like they're going to succeed, and if the idea seems overtly evil, then we don't do it. But like affecting our genetic code, I mean, we're probably already affecting our genetic code because we make all these people move here and they end up like getting girlfriends and procreating, so a very small number of them. I don't know. I don't know. We haven't really thought about that before. I mean, we haven't funded a lot of people doing medical type stuff. We would just—we don't get many applications to do that kind of thing. So I don't know, we're just trying to play it by ear. But we have Paul Buchheit. Paul Buchheit like invented "don't be evil," right? So Paul Buchheit is our official morality consultant, and he tells us when things are evil. Boy, you would be surprised. You'd be very surprised. Sorry if that's a bad answer. Yes, question.

Interviewer: So does it make it easy to outsource your evil detection?

Paul Graham: Does it make it easier to outsource my evil detection? Well, when I said Paul Buchheit was our morality consultant, I was actually kind of joking. So, I mean, but you know, although it does turn out, somehow with moral questions, you have to ask other people's advice more than technical questions, right? I don't know, maybe I'm morally incompetent and technically competent, but with technical questions you can just see what you got to do a lot of the time, and moral questions you have to ask other people. There's probably an essay there too. Too

bad I don't write them anymore. Yes.

Interviewer: So, just an easier question than that one, and an easier question going the direction of technical things. Why does news.ycombinator.com return lines delimited by CR instead of CR LF?

Paul Graham: Oh my God, 'cause like I wanted to write an HTTP server and I didn't read the spec. I just like did stuff until my browser could get pages back, right? So it's all some early version of Firefox's fault, essentially. When I could see pages on my browser, I'm like, "Well, it seems to work." So, I mean, this may actually get fixed. We hired somebody to work on making Hacker News faster, you'll be happy to hear, and they'll probably fix that in the process. There's some line somewhere where I should like add a line feed or a carriage return that's not there or something, but like it's never been on my top 100 things to do to go and look in the source and fix that because it like works on my browser. I'm sorry.

Interviewer: Thanks a lot. Well, it won't be the first. All right, yes. So you talked about the next Apple, and one of the things we've been seeing is Apple have been winning in, partly in the price war for the delivery of hardware, particularly their tablet screens, because they have such buying power and they can buy up the manufacturing capacity. And particularly for small startups looking to make hardware, to make innovative and groundbreaking hardware, manufacturing has to be a really difficult problem to solve. So how are you going to compete with Apple when they have such scale? Is that what you're saying? How are you going to—how do you think—is there an opportunity for this sort of monolithic process? When I say compete against Apple, I mean I'm not saying the startup's first product should be like a competitor to the iPad. You should probably not—even the startup should probably not even do consumer stuff initially. They should do something that's like some thing for hackers, you know, like the Arduino, if those guys had startup ambitions, right? They make this thing that like wouldn't even seem like a threat to Apple, right? And it's just something, it's for hackers. They solder this stuff together themselves and ship them out, and when they get the money back, they buy more parts.

Interviewer: So you don't think a startup can make consumer devices?

Paul Graham: It depends on what the device is. If it was cheap enough, maybe you could make some kind of like headset thingy. You know, Jawbone as a startup, right?

Interviewer: Think there's an opportunity to interface to the sort of the monolithic manufacturing processes and make it—I mean, we've seen some of the like 3D printers, some of the Kickstarters, the way they're making—there is a startup in the current YC batch doing this. There is a startup in the current YC batch that's like [unclear].

Interviewer: The print button for your circuit designs, um, like you just hit print and like stuff arrives from China. Finally, this the stuff printer. So yes, there is an opportunity there. They haven't launched yet, so I can't see what they're called.

Paul Graham: No time for further questions? I understand. We done?

Interviewer: I just wanted on a point of information to let you know that you were the first keynote speaker at the very first PyCon.

Paul Graham: Oh, was I? In 2003? Wow. Thank you for doing that. Thank you for coming back. This one was way more interesting. Thank you, ladies and gentlemen, Paul Graham.

---