HACKERMONTHLY

Issue 15 August 2011

Curator
Lim Cheng Soon

Contributors
Josh Nimoy
Kirill Zubovsky
Philip Kaplan
Jason Freedman
Jason Cohen
Kristof Kovacs
Peteris Krumins

Proofreader
Emily Griffin

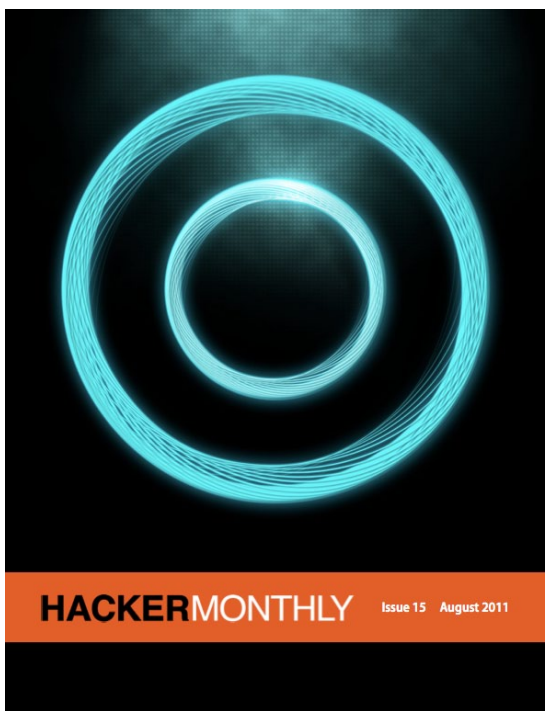
Printer
MagCloud

HACKER MONTHLY is the print magazine version of Hacker News — *news.ycombinator.com*, a social news website wildly popular among programmers and startup founders. The submission guidelines state that content can be "anything that gratifies one's intellectual curiosity." Every month, we select from the top voted articles on Hacker News and print them in magazine format. For more, visit *hackermonthly.com*.

Advertising
ads@hackermonthly.com

Contact
contact@hackermonthly.com

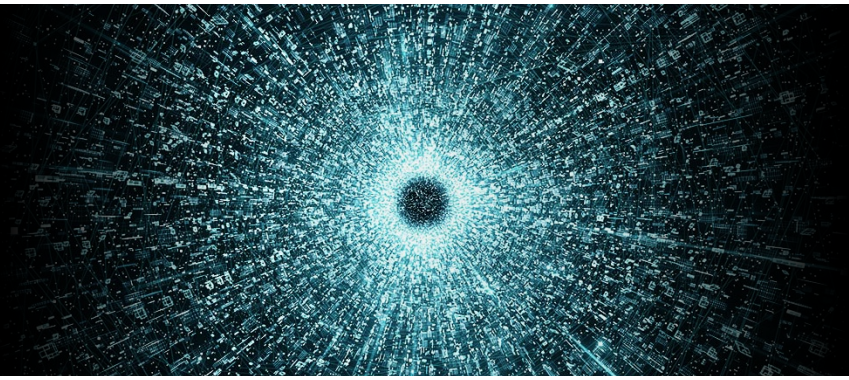
Published by
Netizens Media
46, Taylor Road,
11600 Penang,
Malaysia.



Cover Illustration: Simon C Page [excites.co.uk]

Hacker Monthly is published by Netizens Media and not affiliated with Y Combinator in any way.

Contents



FEATURES

04 **Tron Legacy: How the Special Effects Were Done**
By JOSH NIMOY

STARTUPS

14 **3 Lessons From a Coffee Entrepreneur**
By KIRILL ZUBOVSKY

16 **Getting Users For Your New Startup**
By PHILIP KAPLAN

20 **You Don't Get Shit You Don't Ask For**
By JASON FREEDMAN

24 **Never Say "No," But Rarely Say "Yes"**
By JASON COHEN

PROGRAMMING

28 **A Collection of Cool, but Obscure Unix Tools**
By KRISTOF KOVACS

32 **Low Level Bit Hacks You Absolutely Must Know**
By PETERIS KRUMINS

For links to the posts on Hacker News, visit *hackermonthly.com/issue-15*. All articles are reprinted with permission of their original author.

Tron Legacy:
How the Special Effects Were Done

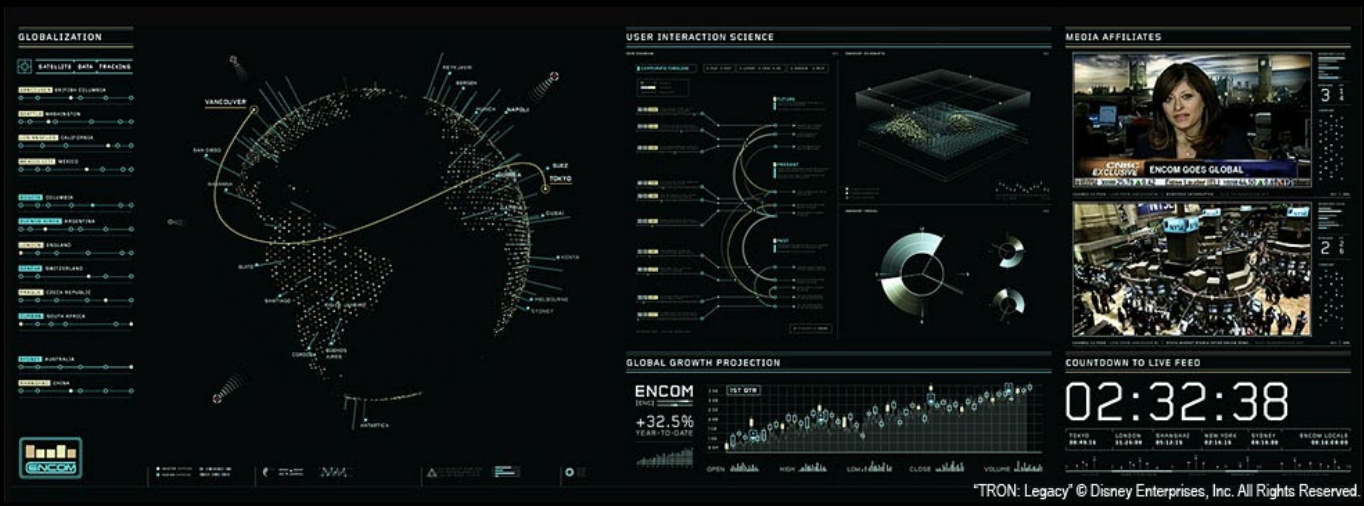
By JOSH NIMOY

"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

I SPENT A HALF year writing software art to generate special effects for Tron Legacy, working at Digital Domain with Bradley "GMUNK" Munkowitz, Jake Sargeant, and David "dlew" Lewandowski. This page has taken a long time to be published because I've had to await clearance. A lot of my team's work was done using Adobe software and Cinema 4D. The rest of it got written in C++ using OpenFrameworks and wxWidgets, the way I've always done it with this team. Uniquely, however, Digital Domain's CG artists were able to port my apps over to Houdini for further evolution and better rendering than OpenGL could ever provide. Special thanks to Andy King for showing me that what seasoned CG artists do at DD is actually not so far off from what's going on in the Processing community.

"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

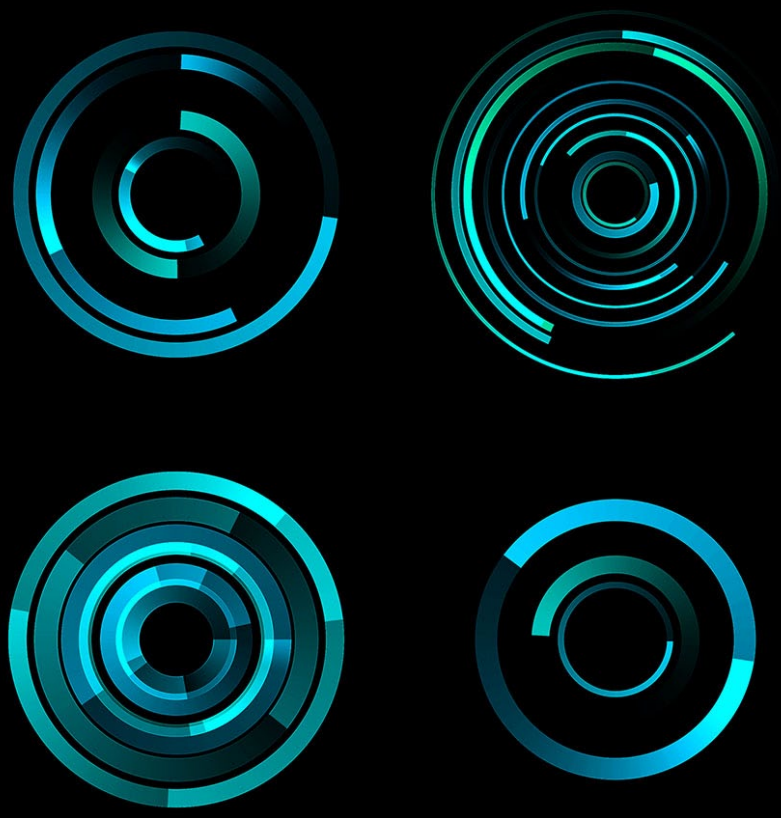
In addition to visual effects, I was asked to record myself using a Unix terminal doing technologically feasible things. I took extra care in babysitting the elements through to final composite to ensure that the content would not be artistically altered beyond that feasibility. I take representing digital culture in film very seriously in lieu of having grown up in a world of very badly researched user interface greeble. I cringed during the part in Hackers (1995) when a screen saver with extruded "equations" is used to signify that the hacker has reached some sort of neural flow or ambiguous destination. I cringed for Swordfish and Jurassic Park as well. I cheered when Trinity in The Matrix used nmap and ssh (and so did you). Then I cringed again when I saw that inevitably, Hollywood had decided that nmap was the thing to use for all its hacker scenes (see Bourne Ultimatum, Die Hard 4, Girl with Dragon Tattoo, The Listening, 13: Game of Death, Battle Royale, Broken Saints, and on and on). In Tron, the hacker was not supposed to be snooping around on a network; he was supposed to kill a process. So we went with posix kill and also had him pipe ps into grep. I also ended up using emacs eshell to make the terminal more l33t. The team was delighted to see my emacs performance — splitting the editor into nested panes and running different modes. I was tickled that I got emacs into a block buster movie. I actually do use emacs irl, and although I do not subscribe to alt.religion.emacs, I think that's all incredibly relevant to the world of Tron.



"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

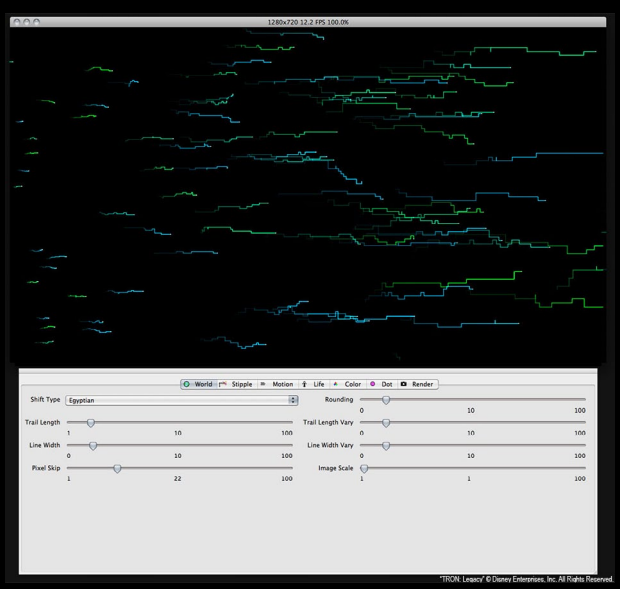
HexVirus is a spherical map of the globe that features vector outlines of the continents. These continent vectors are slowly eaten away by a more hexagonal representation. Algorithmically, this is a path stepping function which looks ahead for the closest matching 60-degree turns. The HexVirus globe was used in the executive board meeting scene, and also inside the grid as a visual aid in CLU's maniacal plan presentation. In the board room interface, the globe element is surrounded by the lovely work of my team.

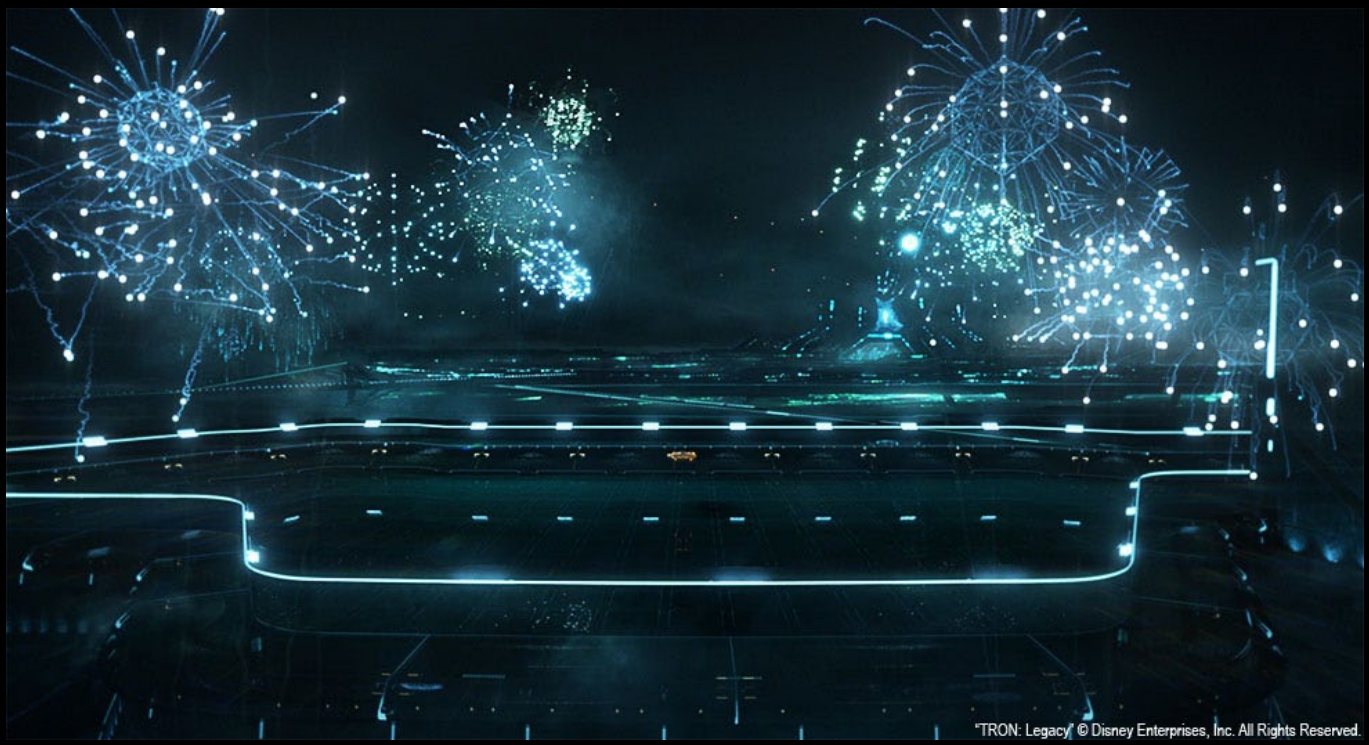
"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.



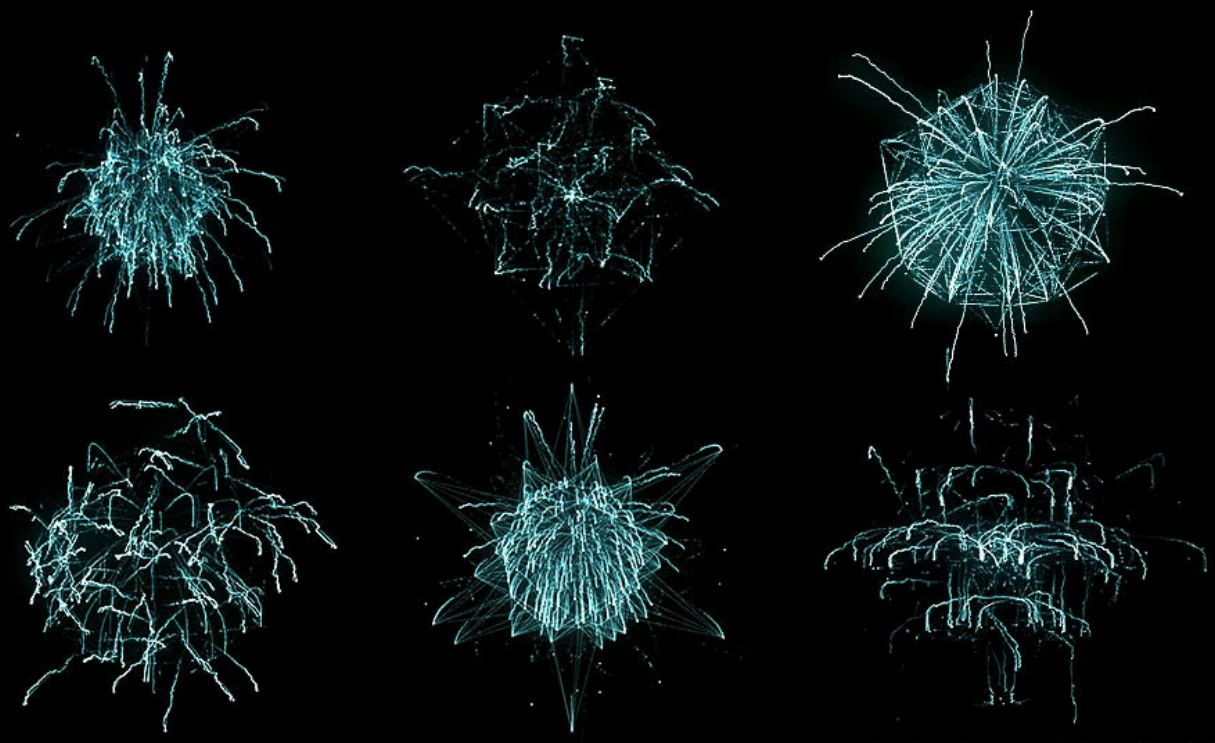
"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

The scoreboard was the first element I worked on. I created a line-generator that produced bursts of lines which turned at adjustable angles. The line generator had "radial mode" which arranged the geometry in concentric circle form. This line generator was used to generate generic elements and layers of style in different things, and is a GMUNK favorite. At this point, I found myself moving to multi-sampled FBOs because the non-antialiased polygons were just too ugly to work with, and we needed to make film-resolution renders. In fact, this is the highest res I've ever seen my apps render.

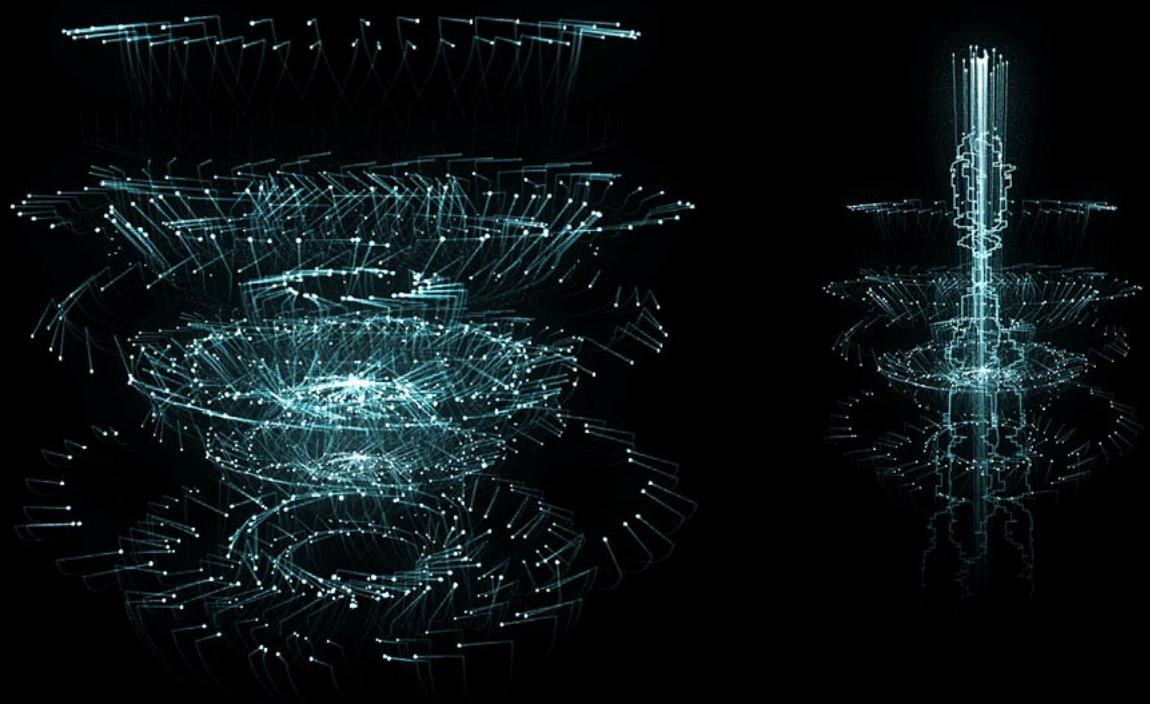




Fireworks, mmmm. I started with a regular physics simulation where a particle has an upward force applied at birth, sending it upward while gravity pulls it back down resulting in a parabola. I then added particle-children, followed by various artistic styles, including what our team has called "Egyptian" across several jobs — which is a side-stepping behavior. We were trying to create fireworks that looked enough like real fireworks but had interesting techno-aesthetic. As a homage to the original Tron character Bit, we used icosahedrons, dodecahedrons, and similar. I was disappointed that Bit isn't in this one. After doing this simulation, I've grown more aware of how often fireworks are used in movies.

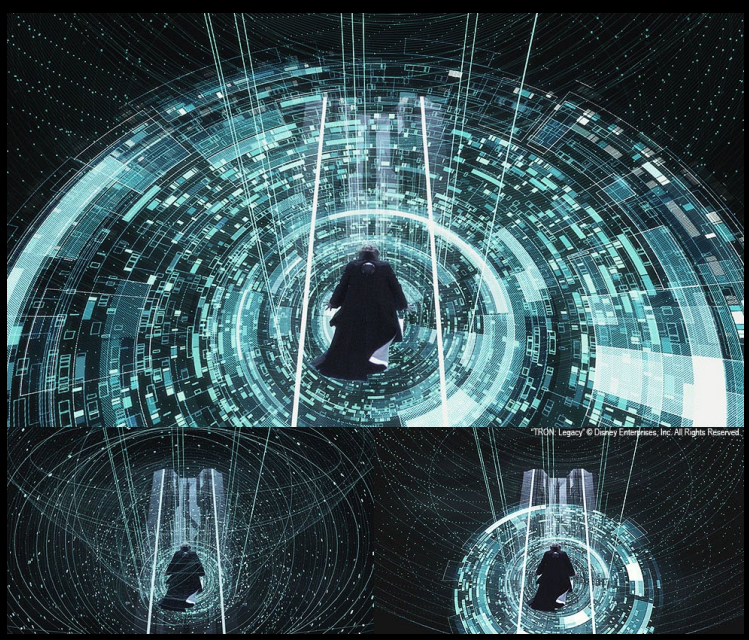


"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

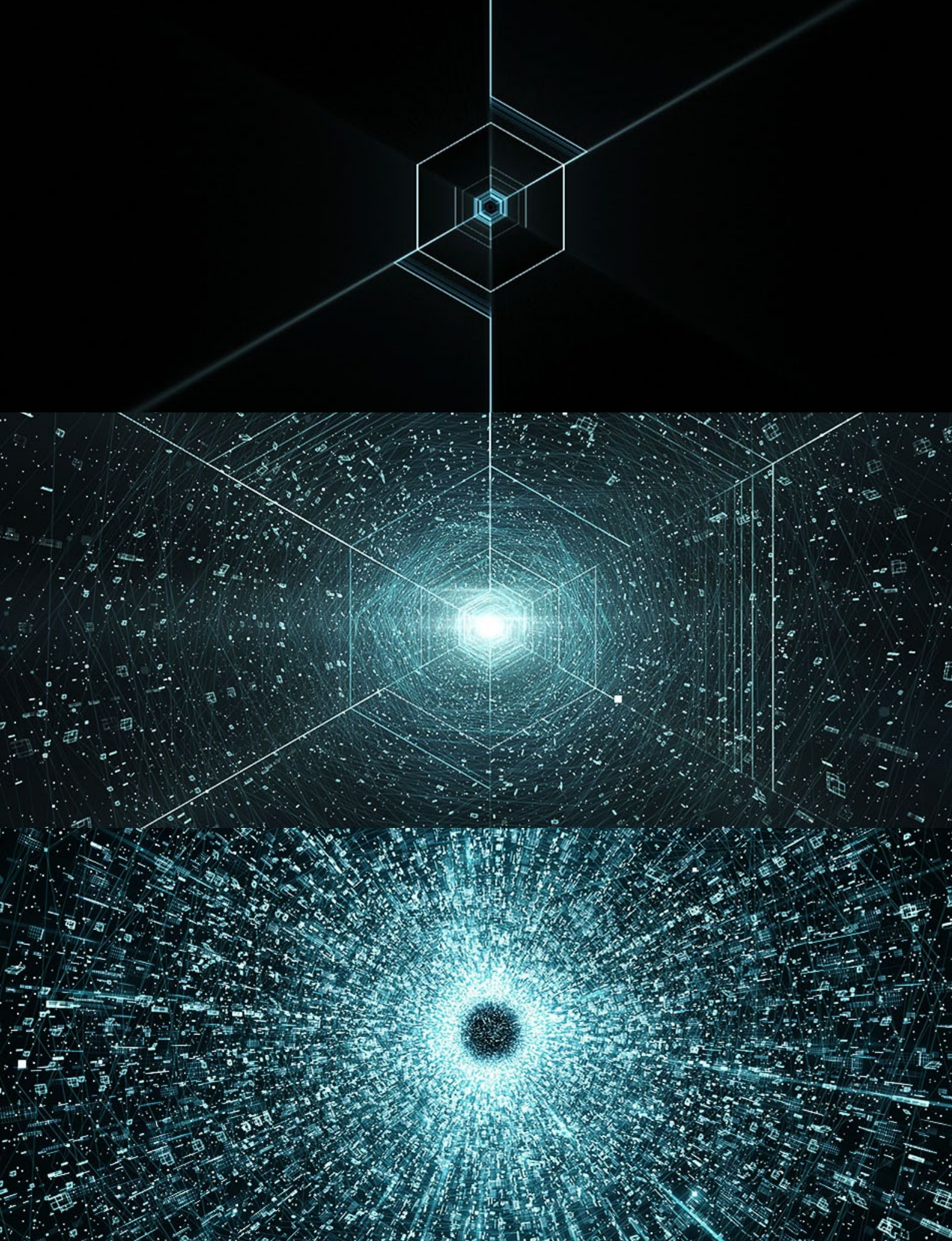


"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.



For the portal climax, the Tron-Lines app was used, but also apps like "Twist" from our team's previous jobs. Once the look was mocked up by GMUNK, a Houdini artist recreated the rig for deeper control.

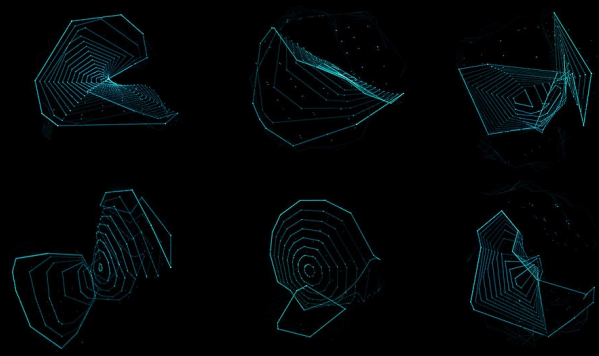


"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

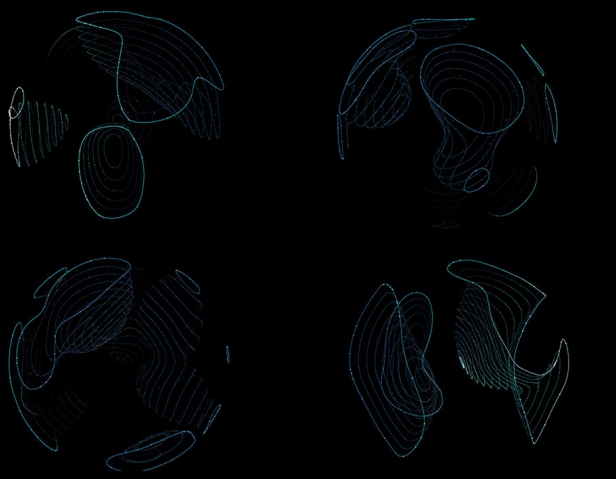
"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.

I wrote a particle renderer that could make the head holograms slurp in and out of the data discs. Special thanks to Keith Pasko for CLUing me in about using exponential functions to create a sliding-gooey sort of delay.

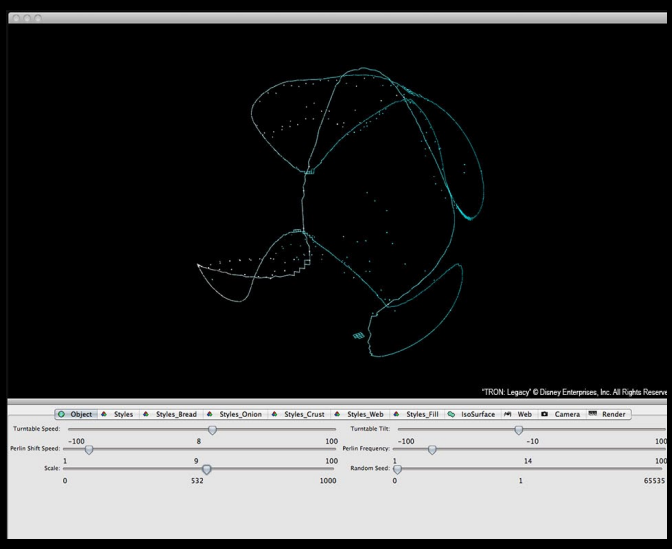
"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.



"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.



"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.



When fixing Quorra, there was an element in the DNA interface called the Quorra Heart which looked like a lava lamp. I generated an isosurface from a perlin-noise volume, using the marching cubes function found in the Geometric Tools WildMagic API, a truly wonderful lib for coding biodigital jazz, among other jazzes. The isosurface was then drawn along different axes, including concentric spheres. The app was mesmerizing to stare at.

After this project, I was fed up enough with wxWidgets and Carbon that I was ready to author my own OpenGL based UI. The most important thing I could use was a floating-point slider. I also got irritated with the way the Carbon sliders would not slide all the way to the minimum and maximum values. It totally messed with my zen thing. Also, after a job like this, it's clear that a member of the Processing community working within a CG community is greatly restricted by the differences of real-time graphics rendering engines, and that probably messes with an art director's zen thing. ■

"TRON: Legacy" © Disney Enterprises, Inc. All Rights Reserved.
Reprinted with permission of the original author.
First appeared in *hn.my/tron* (jtnimoey.net)

Josh Nimoy is a new media arts hacker who has worked in several industries, and shown art in major galleries/museums. Recently, Nimoy's software art renderings for Hollywood have been widely visible in mass media. Currently, Nimoy is a senior media and visual designer at Motorola Mobility in Silicon Valley.

3 Lessons From a Coffee Entrepreneur

Story of how John turned a problem into an opportunity.

By KIRILL ZUBOVSKY

JOHN IS THE owner of Canal Street Coffee. He is sixty-six years old, and he spent most of his life being an employee. After years of working in blue collar jobs, John finally opened this cafe 5 years ago. For the first time in his life, he was at the top, and he was learning on the fly. The business was doing well until the recession came along. He had to lay off a few staff and get behind the counter himself. This is a story of how John turned his problems into opportunities — and what you should learn from his experience.

Lesson ❶ Reduce Overhead

Life was going steady, until one day the credit card processor raised its fees in the same month that the computerized register crashed. John had a decision to make: either go ahead and buy a new computer and keep paying credit card fees, all to spend more and more money, or to drop credit and go cash-only. Just like any good entrepreneur, he made a decision quickly and got rid of the extra overhead. What do you know, all of a sudden there was no computer to worry about nor were there any feeds to the processor. That meant John could sell less every month, and still remain profitable.

Lesson ❷ Increase Conversion

For the first two months, the sales plummeted. With a "cash only" sign on the door, customers would come up to the building, walk up the stairs, get to the door, and then leave. "We couldn't have that! I needed to convert them," said John. Notice, he simply gets what needs to be done.

To fix the conversion rates, John moved the "cash only" sign inside, and on top of that , he started accepting checks, foreign currency, and even the IOUs. The latter was a big surprise. Since he started this gimmick, the store actually gained more loyal customers. People haven't been skipping on payments and have always come back. Perhaps they actually like to be trusted!

Lesson ❸
It's Not The Customers' Job to Know What They Want

The store was up and running, the customers were coming, the economic situation was getting better, but not all was well at Canal St. Coffee. When it comes to baked goods, every coffee shop operates at a loss. John, too, used to order $70 worth of baked goodies every day, but only sell about half. One day the bakery raised their prices, and that was the tipping point.

A coffee shop is expected to have baked goods, and although John wasn't sure what he would do, he was not going to pay a cent more. Shortly after, John started making home-made banana bread; John's gardener baked some cookies. What do you know? The customers ate it up, quite literally!

Although on a winning spree, John wasn't quite satisfied with the choices. Something else was needed on the menu, and so one day he drove by Fred Meyer and purchased a box of donuts. His only employee was skeptical about selling those 'manufactured' donuts, but they tried anyway.

Believe it or not, the donuts were sold out. In fact, locals started to come in just to get these donuts, while of course also buying coffee, beer, and whatever else was available. Nobody could turn down 25-cent pastry! With the bread, cookies, and donuts, John was now operating with virtually no loss on the baked goods, and managed to attract more stable customers.

Conclusion

I didn't plan this story, but after listening to John tell me about his experiences, I couldn't help it. The lessons above apply to any company: big or small, tech or not, the concepts are the same. To apply them, you don't need to be a developer ninja, or a rocket scientist, or a mad biz-dev guy. All you need is some courage to try things out. Start by making fast decisions, listen to your customers, get them in the door, experiment, and, whatever happens, look on the bright side. ■

Kirill Zubovsky is a business savvy industrial engineer and a wantrepreneur, living in Seattle. He is most passionate about solving challenging problems. You should follow him on twitter at @kirillzubovsky

Reprinted with permission of the original author.
First appeared in *hn.my/coffeejohn* (geekatsea.com)

Getting Users For Your New Startup

By PHILIP KAPLAN

The most frequently asked question I get from new entrepreneurs is, "*How do I get users?*"

Here's most of what I know.

Preface

The best sites seem to take off magically by themselves. Truth is, every site needs a little kickstart to get to its first 10,000 to 100,000 users. Consider this a list of kickstarters. But keep in mind the saying, "nothing kills a bad product faster than good marketing." You have been warned.

Start controversy

In April, 2000, I launched *Fuckedcompany.com* — a blog that chronicled the dot-com bust. It had 5 million readers per month, a lot of revenue, and I eventually sold it.

Marketing started when I joined a large online community of web developers, not unlike Hacker News. "*Hey, look at this asshole, what a jerk*," I wrote, with a link to my own site. This started a heated debate about the site. It seemed like hundreds were participating — much love, some hate. I gracefully ducked out of the conversation and watched 24,000 new registered users join that week. There's a line between contributing and spamming — I think what I did was okay.

With *Blippy.com*, we also chose to go the controversial route. Instead of telling people "this is a good site to tell friends the restaurants you're going to" (which Blippy can do), we went with a more controversial message along the lines of "broadcast your credit card statements." This resulted in a huge amount of PR and attention (more on that in the "press" section, below).

Keep in mind that your product doesn't have to be inherently controversial to stir controversy. If you're an electronic music artist, hang out with a folk music crowd. They'll hate you, which is good. If there's one new user for every ten haters, I'll take it.

Almost anything can be controversial. If there's nothing that can be controversial about your product, it might be boring.

Viral tricks

Address book importers, auto tweets, "send this to 5 people and get special access," etc. I've never been a huge fan of this kind of thing, but it's popular.

One of my sites, Fast140 [fast140.com], sends a tweet the first time you use it (with disclosure that it's gonna do that). Another one of my sites, *Mobog.com*, has an address book importer that lets you invite everyone you know to join. Address book importers stopped being effective around 2007, I should actually just get rid of it.

If you're building Facebook apps or something using Facebook Connect or Twitter, these kinds of tricks can still be effective, though the window is closing on that.

Affiliate programs

Pay people a commission to send you users. This only works, obviously, if you're running a pay site.

One of my sites, *HitMeLater.com*, has an affiliate program. It works a'ight. Running a successful affiliate program is more work than it seems. You can't just put up an affiliate link & wait. Instead, you need to find affiliates, nurture them, run specials and promotions. The best way to get an idea of how a well-run affiliate program looks, is to join one like this [hn.my/zappos] or this [hn.my/friendfinder].

It's possible to find an affiliate manager who will work for you for commission only.

Search Engine Optimization (SEO)

The great thing about search engines is that you don't have to do anything — if you have a good service or good content, Google will generally make sure people find it.

SEO tweaks can increase these numbers. But in the early days, SEO is generally a bad way to spend your time; gains will likely be insignificant.

My free email newsletter service, TinyLetter [tinyletter.com], is on the first page of Google search results for "free email newsletter". That generates a fair amount of business. All I had to do was give the site a good <title> tag & Google did the rest.

Press

Press is one of the best ways to get users. The easiest way to get an article written about you is to tell a writer about a good story idea you have. Don't write a press release. Don't hire a PR firm.

Instead, think of a compelling story you would like to read. Pick a writer who you like, and who you think might want to write about it. Tell them your story idea in 2-3 sentences. Contact them via email. If you don't know their email address, guess. Also ping them on Twitter and Facebook.

If the writer doesn't respond, try a different one. But only one at a time — they won't like you if you give the same story to their competitor.

Also, don't just pitch stories about your company. Ingratiate yourself with writers by selflessly giving them scoops and ideas for stories unrelated to you. Karma.

- Bad story idea: "I just launched a thing and you should write about it."
- Good story idea: "Facebook fucked something up. Oh and it happens to relate to my thing."
- Good story idea: "I just raised $20 million to launch a thing."
- Good story idea: "My competitor raised $20 million, yet my thing is better."
- Good story idea: "My competitor's thing is unsafe and could possibly kill you."
- Good story idea: "I'm not sure if my thing is legal."

Celebrity endorsement

Ask someone famous (real-famous or internet-famous) to use your site. Have them tweet about it or make a YouTube video.

Get in touch with them through their manager, PR firm, lawyer (a little googling goes a long way), Facebook, Twitter, MySpace page (still effective), whois info, and so on.

Consider offering them some combination of nothing, money, and/or equity.

"Biz Dev"

Get a bigger company to promote you. At Blippy, we got Sephora to promote us on their Facebook page to 1M+ fans for free, because we did a special Sephora thing on our site.

BillShrink got T-Mobile to say "if you don't believe that we're the cheapest, go check BillShrink.com, an independent third party."

Think of something cool you'd like to do with a bigger company and contact someone who works there on LinkedIn, or Facebook via BranchOut [branchout.com], which is actually better and freer for this (disclaimer: I'm an investor in BranchOut).

Offline events

Throwing parties is grueling, but has helped launch a small number of companies successfully.

Offline events work best when your product has strong ties to the real world. Evite got started by throwing big parties around the country, with the catch that you had to RSVP using Evite. Yelp did something similar.

The majority of launch parties (and SXSW events) are not effective for getting new users.

Get creative

Read about Twitter's famous flat panel displays at SXSW [hn.my/sxsw].

I know a company that promoted a game called "Shrooms" by passing out bags of (innocuous) mushrooms at a gamer conference, almost getting arrested.

Richard Branson drove a tank through Times Square to promote Virgin Cola in front of a giant Coke billboard (actually not sure if that worked). Hugh Hefner bought a mansion in LA and invited celebrities to hang out with naked people.

Groupon does a boatload of ROI-positive online advertising (okay, not that creative but it works). YouTube let you embed semi-illegal content into MySpace pages. Facebook and Digg made widgets that got thousands of sites pointing back to them.

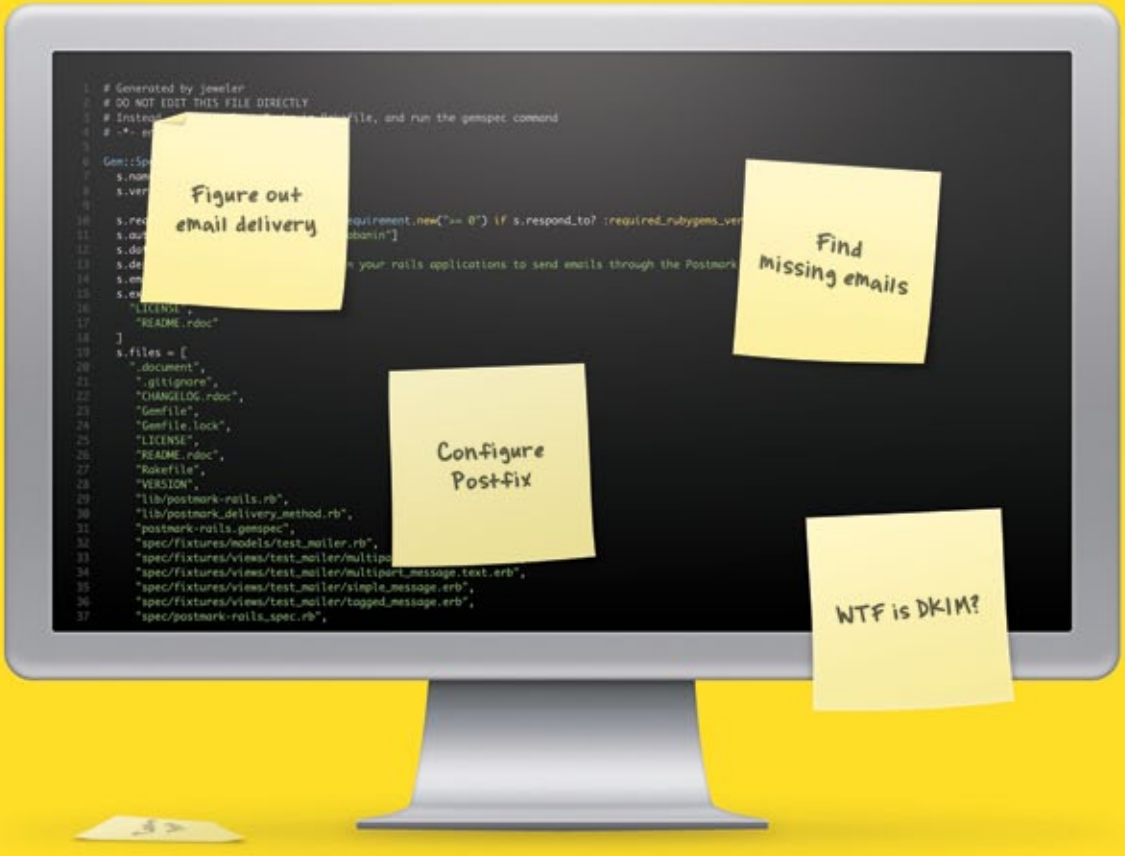
Conclusion

I hope you enjoyed this list of ways to promote your website. One final tip: As entrepreneurs, we obsess over our products. Try to forget about your product for a couple of weeks, and instead obsess over how to promote it. ■

Philip Kaplan is an entrepreneur in San Francisco, California. He co-founded several companies including Blippy, a venture-backed social shopping company, and AdBrite, the largest privately-held Internet ad network. In 2010, Philip was entrepreneur-in-residence at Charles River Ventures.

Reprinted with permission of the original author.
First appeared in *hn.my/users* (blog.pud.com)

If you enjoy configuring Postfix and Sendmail, ignore this.



We'll deal with that so you can get back to coding.

Postmark

hacker.postmarkapp.com

Transactional email delivery for web apps.

You Don't Get Shit You Don't Ask For

By JASON FREEDMAN

When we were raising money for FlightCaster in the fall of 2009, we met with dozens of investors, both VCs and Angels. Most of them turned us down. The investors that did put money into FlightCaster provided us with a ton of value, both in terms of their capital and all the support/guidance/networking.

And what about all those investors that had turned us down? They actually ended up providing a lot of value as well, and that's what this article is about. I had noticed a funny trend from each one that had turned us down. They all closed with a variant of this statement:

> *"If there's ever anything I can ever do to help, please let me know."*

At first, it sounded like a standard pleasantry, in the same vein of "I wish you the best of luck." But then, I started thinking about why it was that every investor was saying it. I realized that there are two forces at work here:

> *The first is that many investors are genuinely nice guys that do root for entrepreneurs. Many made it as entrepreneurs and remember the pain and the hustle. If (and this is an important if) an action is not at odds with their fiduciary duty to their limited partners, most investors will do what they can to support you.*
>
> *The second force at work is pure incentives. Investors know that relationships matter. An investor that passed on a company may want back in at a later point if that company takes off. Or she may want to fund a future company of that entrepreneur. Doing favors without an investment is a cash-free way to prove to the entrepreneur that they have real value to provide in addition to their capital.*

> **“I literally contacted every investor that turned us down and asked for a concrete favor.”**

So, back to our story. I decided to take them up on their offer. All of them. I literally contacted every investor that turned us down and asked for a concrete favor.

We had just launched the FlightCaster product and were working hard to meet people in the industry. My co-founder Evan Konwiser was not yet recognized as the thought leader of the travel industry that he is today. We had a big conference coming up where every leader of the travel industry would be in attendance. And we knew no one.

With a few hours on LinkedIn, I was able to see which investors knew someone that knew someone we wanted to meet. I sent a personalized version of this email to all those investors that offered to help:

Thanks for offering to further support us as we build FlightCaster.

Right now, we're preparing for the PhoCusRight conference next week, the travel industry's largest get together. We're trying to make some connections with people at the conference. Can you help us connect with leaders in the Business Travel divisions of Expedia or Orbitz? Or any other introduction that you feel would help us?

Thanks for the help!
Jason

And you know what happened? A huge number of those investors responded within 48 hours, and most of them were able to help in some way. And it was the most impressive investors that responded the quickest. Here's the response I received from Ron Conway the next day (using his classic all caps):

> **“Don’t play games, don’t posture, don’t hint. Just ask for what you want.”**

I KNOW THE CEO OF EXPEDIA…send us an email template to send to him

Ron

So, yes, we got introduced to the CEO of Expedia because Ron Conway, who had previously passed on investing in us, was willing to help. Booyah!! Why did he do it? Because Ron Conway is also one of those guys that is fundamentally rooting for the entrepreneur.

The other reason he did it? Because we asked. You don’t get shit you don’t ask for.

A few tips on asking for help:

Ask for what you want

This is a common Paul Graham statement to YC companies. Figure out exactly what you need and just ask for it. Don’t play games, don’t posture, don’t hint. Just ask for what you want.

Make your request very concrete

We didn’t ask for general advice. We did our homework and made very specific requests. It’s much easier for people to respond to concrete requests. Even if they can’t provide for that direct request, the specificity of the request helps them find an alternative way to help.

Don’t use and abuse

I only wanted to ask these incredibly busy investors for help once or maybe twice. Obtaining industry introductions was one of the most important necessities of our startup. I made sure to ask for something that they could provide with minimal effort and risk.

Pay it forward

We’re all part of the same innovation community. This whole entrepreneurship thing is hard for everyone, but fortunately, everyone has something to offer. Don’t procrastinate on giving back.

Say thank you

Investors love knowing the outcome. For both personal and professional reasons, they’re interested in long term relationships. Do your part by keeping them up to date with your successes. I like to email everyone that helped me immediately prior to something showing up on Techcrunch. It helps me communicate to them that I appreciate everything they did to be a part of our success.

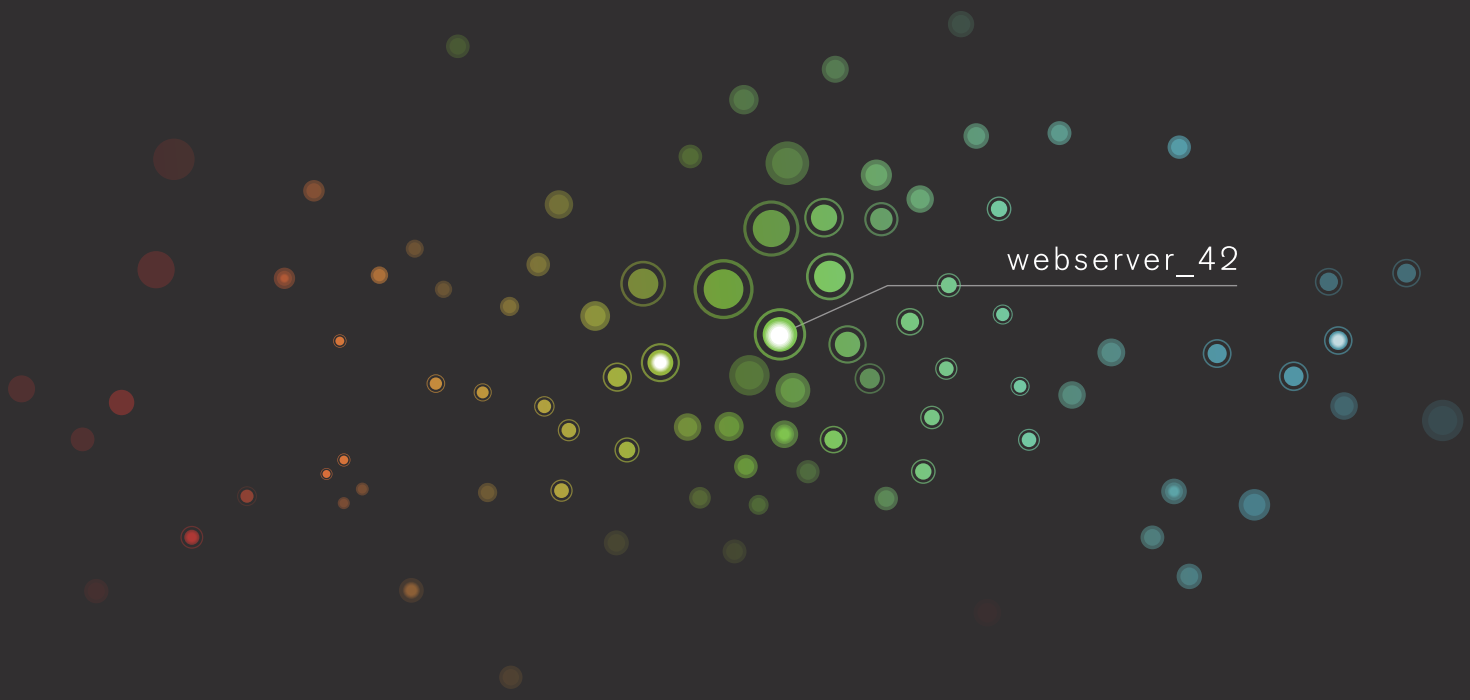
Give it a shot. The worst that can happen is that they say no. And even that’s not so bad — investors love knowing that you hustle.
So hustle. ■

Jason is the co-founder of the recently-acquired FlightCaster. Previously, Jason co-founded and worked on several other startups that crashed and burned. Jason continues to stubbornly insist that everything he learned in life came from summer camp. You can find his non-apologetic, self-righteous blog posts at *humbledMBA.com* and on Twitter at @jasonfreedman

Reprinted with permission of the original author.
First appeared in *hn.my/ask* (humbledmba.com)

These are your servers

These are your servers on Cloudkick



Any questions?

cloudkick.com
415.779.5425

support for 8 clouds + dedicated hardware

cloudkick

the best way to manage the cloud

Never Say "No," But Rarely Say "Yes"

By JASON COHEN

Everyone says small startups require focus. Say "no" to anything that distracts from your goal, your vision, your strategy, tempting though it is to explore all opportunities, hoping each time that this is the one that will catapult you from Mixergy listener to Mixergy interviewee.

Lack of focus results in half-assed initiatives, each interrupted by apparently greener pastures before you've invested the time and devotion it deserves. Learn to say "no!"

Ah, but then again you must also experiment with new ideas. Fail fast! Pivot! Test! Doubt! Always be collecting evidence that you're wrong, always be trying new things in case you've been blind. Never pass up an opportunity to change, learn, grow.

So…how are you supposed to explore other ideas if you're also supposed to be saying "no" to anything that diverges from The Plan?

Here's what I do: I never say "no." But I carefully qualify "yes."

I learned this trick while still in high school. In the mid 90s, it was clear that Apple had lost the personal computer battle, and all their developers were fleeing like rats off a sinking ship into the ocean of opportunity that was Windows 95. As a maven of the Macintosh API and still willing to admit it, I landed lots of small contracting jobs fixing up code that other developers wouldn't touch.

My typical rate was $25 per hour, which when you're 17 seems like a lot of money.

One day I got a call from some poor schleps I didn't want to help. They had just completed a new product written in Java and it was broken on a Mac, and could I help? None of their customers used Macs, so they didn't think Macs were important. But then it turns out the main investor is keen on seeing the demo on a Mac, and when they tried it, it didn't work. (Yay investors!)

I wanted no part of this. Java was new and known to be full of bugs, and anyway I was a C/C++ kind of guy, and I didn't want to get involved in an academic fad language like Java. (So yeah, I simultaneously decided that (1) Java is a fad and (2) I'm sticking with the Macintosh Toolkit; five years later one those platforms had zero developers and the other had one million, and I picked exactly wrong.)

I could have said "no." Given my specialty and my goals, traditional career (or startup) theory says I should have said "no."

But instead, on the advice of an older, wiser friend, I showed up at their office and said I'd do it for $100 per hour.

I fully expected them to laugh in my face. Maybe I would receive a condescending talking-to about the audacity — nay, the impudence! — of someone of my age and experience walking in here and demanding such outrageous compensation, someone who, let's be clear, is technically too young to even enter into a legal consulting agreement in the first place.

And then I would have slinked out of there embarrassed, but ultimately no worse for wear.

But that's not what happened. They looked me up and down, their faces revealing both incredulity and surrender. They said OK. An hour and a half later, everything was working. The difference between saying "no" and getting $150 for about two hours of my life was all in how I phrased "yes."

And I have stories that went the other way, which are just as important.

At WPEngine [wpengine.com], for example, we're constantly talking to large bloggers who want to move to our system. These are folks with big requirements — tens of millions of monthly page-views, traffic spikes, custom code, perfect up-time, and 24/7 support.

Should we take on those clients? Maybe not — after all, the stated goal of WPEngine is to serve the "middle market" — the folks who have outgrown free blogs, don't like maintaining their own servers, but aren't so large that they have extreme hosting demands. That's our profitable niche.

So we think. But if we just say "no" to these big bloggers, maybe we're closing the door on big, important orders. Perhaps the entire company should pivot — maybe it's easier or more profitable to serve 100 large blogs than 1000 medium ones. But how do we know if we say "no?"

Then again, if we say "yes" we might really be screwed. If we can't provide them the human and technical service they expect, now we've hurt a blogger, we get bad press, and we've wasted a bunch of time. Or even worse, we hold on for dear life but it's extremely unprofitable, and now we have this expensive, time-consuming albatross around our necks.

So we've said "yes" by quoting high enough that we know for certain we will make good money on the deal, so much so that it will partially fund something else we want to do. Maybe that means a big new advertising campaign, or hiring another WordPress expert for our staff.

There was one especially large customer where we literally thought of it like this: this deal needs to be big enough to not only make a reasonable profit on the operating expenses, but to pay for an entire developer's salary (assuming bootstrapped, put-in-elbow-grease-for-stock low salary), because we know this new customer will occupy a lot of that person's time, but all the remaining time we get "for free."

So we've given a lot of qualified yeses, and many were rejected.

At Smart Bear [smartbear.com] I used this principle yet again. Companies would fly me out to help them implement a peer code review process, which half the time actually meant that "management" wanted me to convince everyone else that code review was a good idea, and invent a process painless enough they might actually do it.

From a business perspective, this was a poor use of my time. These folks had already bought our software, so it didn't sell more seats. When you counted a travel day on either side of the engagement, the time I lost could easily have been spent landing just one additional customer or make some important changes to the code, either of which almost certainly makes us more money.

Therefore, initially I just said "no." But of course that's wrong. Eventually I said "yes," but the price was $2500/day including travel days, which for these sorts of engagements is unheard of. (Typically you get reimbursed for travel expenses but not paid for that time.)

This immediately cut out most trips, but some remained. Of course on those trips I'd haul in $10,000 for a week of easy work, which I'd often combine with a long weekend with my wife. And anyway those people really wanted me there, which made the work that much more enjoyable.

So the principle is easy: Set the conditions of "yes" such that:

- If they say "yes," you're happy, because the terms or money are so good it more than compensates for the distraction, possibly even funding the thing you really want to do.
- If they say "no," you're happy, because it wasn't a great fit anyway, so it's not worthwhile for a small return on your time and effort.

So that's the punch-line, but before you go I'd like to over-emphasize the idea of "funding the thing you really want to do."

This can take many forms, but it's the single best way of figuring out how to qualify your "yes." Examples:

- "Yes" if it pays for an entire additional person.
- "Yes" if this extends the runway of our startup by at least three months.
- "Yes" if it completely funds development we'd like to do anyway.
- "Yes" if it means one of the co-founders can quit her day-job.
- "Yes" if it will completely pay for three new marketing efforts.

Think of it like another form of funding. Funding is always a distraction from actually running your business, so the amount of money you get must be transformative to the business. Each of those bullet points is transformative, in that each has the potential to move your company from "hobby" to "real business."

And if they say "no," you're fine with that, because it would have been a distraction which wouldn't have moved the needle. ■

Jason is the founder of three companies, all profitable and two exits. He blogs on startups and marketing at *blog.ASmartBear.com*

Reprinted with permission of the original author.
First appeared in *hn.my/sayyes* (blog.asmartbear.com)

Seattle's Fastest-Growing Startup Wants You!*

*Especially if you love helpful & cuddly robots

We dare you to name another Seattle startup on Inc's 500 Fastest-Growing Private Companies list that provides

- generous vacation
- flex time
- travel to conferences globally
- weekly lunch catering
- 401k plans
- free home Internet
- web-streamed pie-eating contests

SEOmoz is looking for

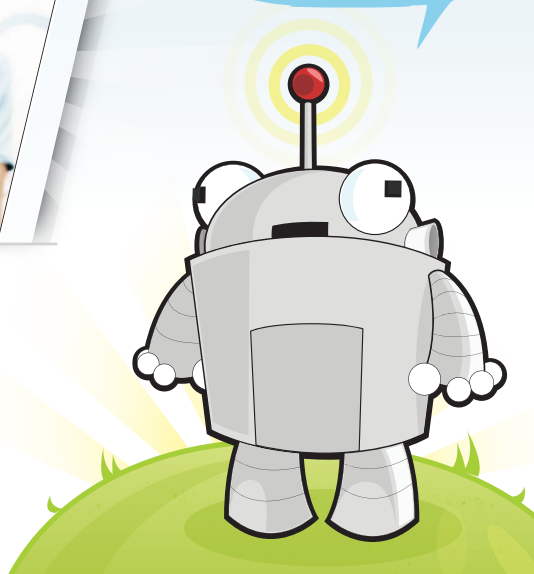
- brilliant software engineers
- out-of-the-box marketers
- customer service experts
- creative graphic designers
- savvy developers

See **www.seomoz.org/jobs** for more information

JOIN OUR FUN TEAM

Know a talented engineer?

Tell your friends - we're paying a $12K referral bonus!



If you're ready to share your passion with the software-building geeks at SEOmoz, we're excited to hear from you. **www.seomoz.org/jobs**

SEOmoz
SEO Software. Simplified.

A Collection of Cool, but Obscure Unix Tools

By KRISTOF KOVACS

JUST A LIST of 28 tools for the command line. Some are little-known, some are just too useful to miss, some are purely obscure. I hope you find something useful that you weren't aware of yet! Use your operating system's package manager to install most of them.

dstat & sar

iostat, vmstat, ifstat and much more in one.

slurm

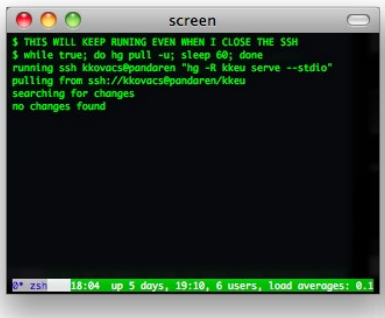
Visualizes network interface traffic over time.

vim & emacs

The real programmers' editors.

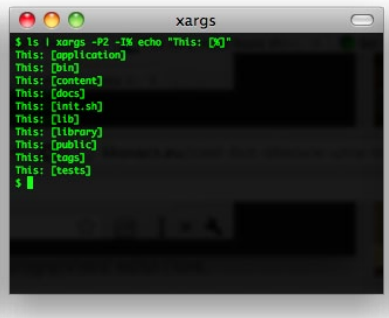
screen, dtach, tmux, byobu

Keep your terminal sessions alive.



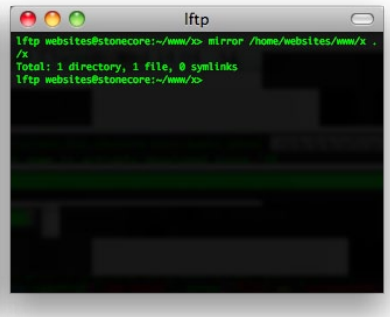
xargs & parallel

Executes tasks from input (even multithread).



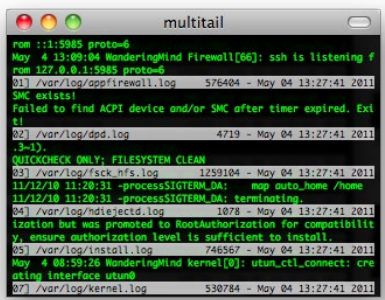
lftp

Does FTPS. Can mirror, kind of like rsync.



multitail

See your log files in separate windows.

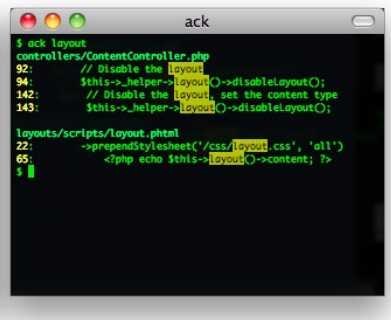


duplicity & rsyncrypto

Encrypting backup tools.

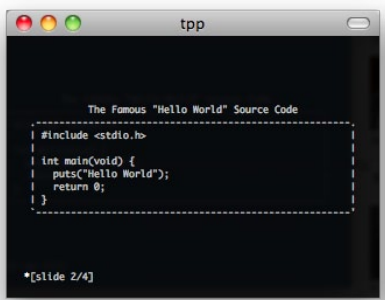
ack

A better grep for source code.



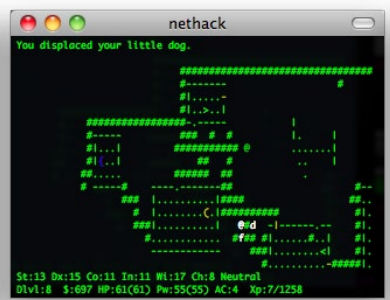
tpp

Presentation ("PowerPoint") tool for terminal.



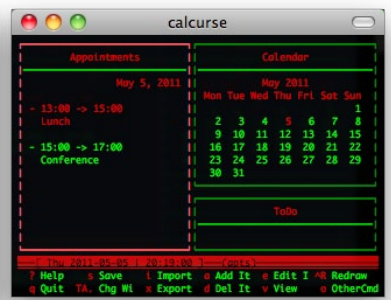
nethack & slash'em

Still the most complex game on the planet.



calcurse & remind + wyrd

Calendar systems.

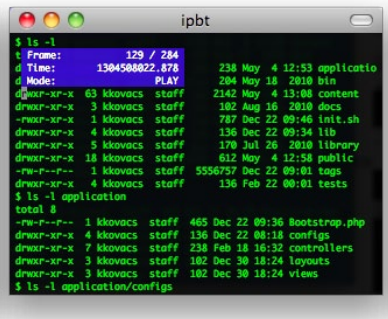


newsbeuter & rsstail

Command line RSS readers.

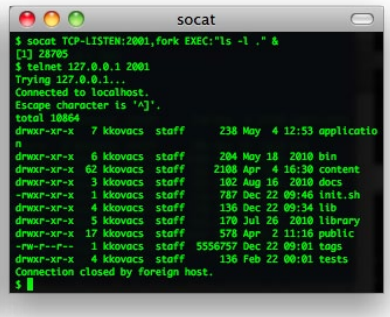
ttyrec & ipbt

Record and play back terminal sessions.



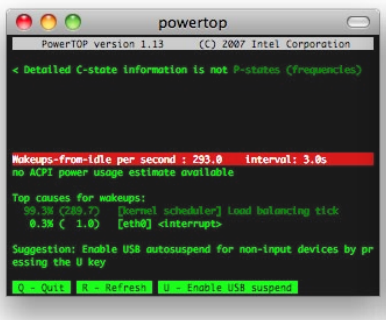
socat & netpipes

Directing stuff easily in and out of sockets.



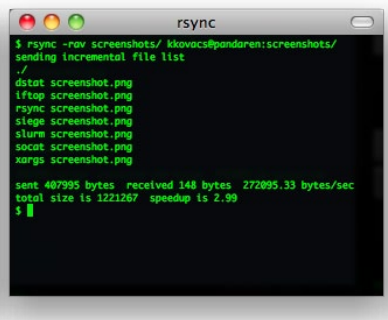
powertop

Helps conserve power on Linux.



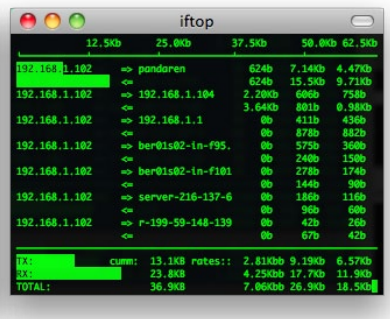
rsync

Keeps filesystems in sync over SSH.



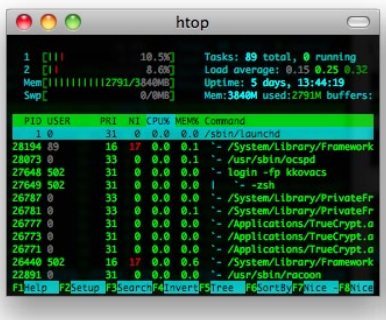
iftop & iptraf

To see where your traffic goes.



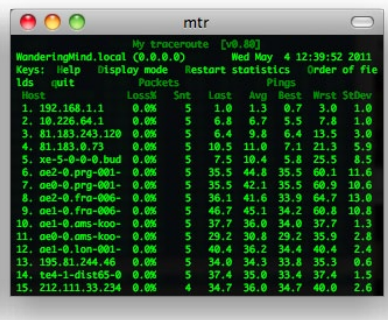
htop & iotop

Process, memory, and io monitoring.



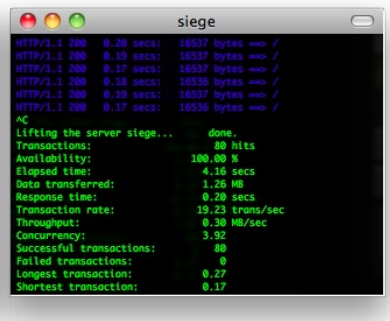
mtr

traceroute 2.0.



siege & tsung

Command line load test tools.

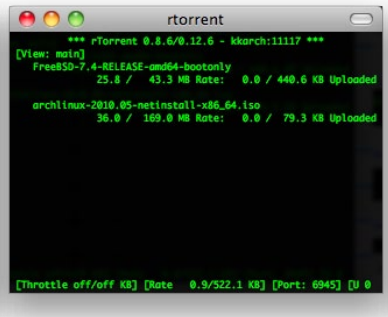


ledger

Terminal-based accounting package.

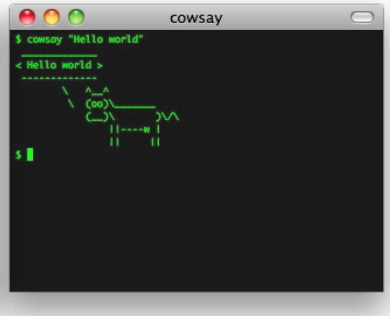
rtorrent & aria2

Command line torrent downloaders.



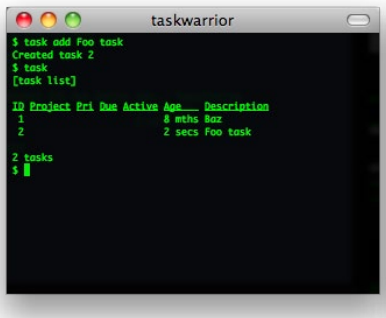
cowsay & sl

I just couldn't resist. :o) ■



taskwarrior

Todo management in the terminal.

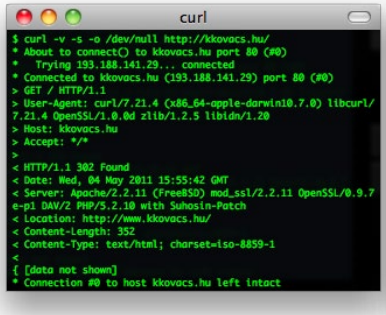


ttytter & earthquake

Nice trendy Twitter clients.

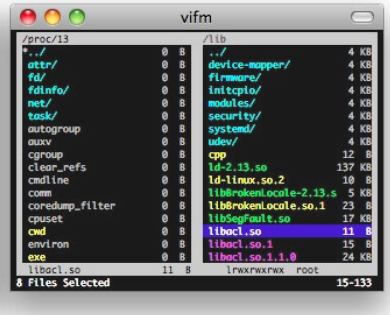
curl

Everybody's favorite HTTP toolbox.



vifm & ranger

Alternatives to the midnight commander.



Kristof is a software architect and consultant from Europe. He works on high-availability, high-security systems in real-time stock exchanges and in the energy industry. You can learn more about him at *kkovacs.eu*

Reprinted with permission of the original author. First appeared in *hn.my/unix* (kkovacs.eu)

Low Level Bit Hacks You Absolutely Must Know

By PETERIS KRUMINS

I DECIDED TO WRITE an article about a thing that is second nature to embedded systems programmers: low level bit hacks. Bit hacks are ingenious little programming tricks that manipulate integers in a smart and efficient manner. Instead of performing some operation (such as counting the 1 bits in an integer) by looping over individual bits, these programming nuggets do the same with one or two carefully chosen bitwise operations.

To get things going I'll assume that you know what the two's complement binary representation of an integer is and also that you know all the bitwise operations.

I'll use the following notation for bitwise operations in the article:

```
&   -  bitwise and
|   -  bitwise or
^   -  bitwise xor
~   -  bitwise not
<<  -  bitwise shift left
>>  -  bitwise shift right
```

The numbers in the article are 8 bit signed integers (though the operations work on arbitrary length signed integers) that are represented as two's complement and they are usually named "x". The result is usually "y". The individual bits of "x" are named b7, b6, b5, b4, b3, b3, b2, b1 and b0. The bit b7 is the sign bit (the most significant bit), and b0 is the least significant.

I'll start with the most basic bit hacks and gradually progress to more difficult ones. I'll use examples to explain how each bithack works.

Here we go.

Bit Hack ❶ Check if the integer is even or odd.

```
if ((x & 1) == 0) {
  x is even
}
else {
  x is odd
}
```

I am pretty sure everyone has seen this trick. The idea here is that an integer is odd if and only if the least significant bit b0 is 1. It follows from the binary representation of "x," where bit b0 contributes to either 1 or 0. By AND-ing "x" with 1, we eliminate all the other bits than b0. If the result after this operation is 0, then "x" was even, because bit b0 was 0. Otherwise "x" was odd.

Let's look at some examples. Let's take integer 43, which is odd. In binary 43 is 00101011. Notice that the least significant bit b0 is 1 (in bold). Now let's AND it with 1:

```
    00101011
&   00000001
    --------
    00000001
```

See how AND-ing erased all the higher order bits b1-b7 but left bit b0 the same as it was? The result is thus 1, which tells us that the integer was odd.

Now let's look at -43. Just as a reminder, a quick way to find negative of a given number in two's complement representation is to invert all bits and add one. So -43 is 11010101 in binary. Again notice that the last bit is 1, and the integer is odd. (Note that if we used one's complement it wouldn't be true!)

Now let's take a look at an even integer 98. In binary 98 is 1100010.

```
    01100010
&   00000001
    --------
    00000000
```

After AND-ing, the result is 0. It means that the bit b0 of the original integer 98 was 0. Thus the given integer is even.

Now the negative -98. It's 10011110. Again, bit b0 is 0, after AND-ing, the result is 0, meaning -98 is even, which indeed is true.

Bit Hack ❷ Test if the n-th bit is set.

```
if (x & (1<<n)) {
  n-th bit is set
}
else {
  n-th bit is not set
}
```

In the previous bit hack we saw that (x & 1) tests if the first bit is set. This bit hack improves this result and tests if the n-th bit is set. It does this by shifting that first 1-bit n positions to the left and then doing the same AND operation, which eliminates all bits but n-th.

Here is what happens if you shift 1 several positions to the left:

```
1       00000001    (same as 1<<0)
1<<1    00000010
1<<2    00000100
1<<3    00001000
1<<4    00010000
1<<5    00100000
1<<6    01000000
1<<7    10000000
```

Now, if we AND "x" with 1 shifted n positions to the left, we effectively eliminate all the bits except the n-th bit in "x". If the result after AND-ing is 0, then that bit must have been 0, otherwise that bit was set.

Let's look at some examples.

Does 122 have 3rd bit set? The operation we do to find it out is:

```
122 & (1<<3)
```

Now, 122 is 01111010 in binary. And (1<<3) is 00001000.

```
    01111010
&   00001000
    --------
    00001000
```

We see that the result is not 0, so yes, 122 has the 3rd bit set.

What about -33? Does it have the 5th bit set?

```
    11011111      (-33 in binary)
&   00100000     (1<<5)
    --------
    00000000
```

Result is 0, so the 5th bit is not set.

Bit Hack ❸ Set the n-th bit.

```
y = x | (1<<n)
```

This bit hack combines the same (1<<n) trick of setting n-th bit by shifting with OR operation. The result of OR-ing a variable with a value that has n-th bit set is turning that n-th bit on. It's because OR-ing any value with 0 leaves the value the same; but OR-ing it with 1 changes it to 1 (if it wasn't already). Let's see how that works in action:

Suppose we have value 120, and we wish to turn on the 2nd bit.

```
    01111000    (120 in binary)
|   00000100    (1<<2)
    --------
    01111100
```

What about -120 and 6th bit?

```
    10001000   (-120 in binary)
|   01000000   (1<<6)
    --------
    11001000
```

Bit Hack ❹ Unset the n-th bit.

```
y = x & ~(1<<n)
```

The important part of this bithack is the ~(1<<n) trick. It turns on all the bits except n-th.

Here is how it looks:

```
~1        11111110  (same as ~(1<<0))
~(1<<1)   11111101
~(1<<2)   11111011
~(1<<3)   11110111
~(1<<4)   11101111
~(1<<5)   11011111
~(1<<6)   10111111
~(1<<7)   01111111
```

The effect of AND-ing variable "x" with this quantity is eliminating n-th bit. It does not matter if the n-th bit was 0 or 1, AND-ing it with 0 sets it to 0.

Here is an example. Let's unset 4th bit in 127:

```
    01111111    (127 in binary)
&   11101111    (~(1<<4))
    --------
    01101111
```

Bit Hack ❺ Toggle the n-th bit.

```
y = x ^ (1<<n)
```

This bit hack also uses the wonderful "set n-th bit shift hack" but this time it XOR's it with the variable "x." The result of XOR-ing something with something else is that if both bits are the same, the result is 0, otherwise it's 1. How does it toggle n-th bit? Well, if n-th bit was 1, then XOR-ing it with 1 changes it to 0; conversely, if it was 0, then XOR-ing with 1 changes it to 1. See, the bit got flipped.

Here is an example. Suppose you want to toggle 5th bit in value 01110101:

```
    01110101
^   00100000
    --------
    01010101
```

What about the same value but 5th bit originally 0?

```
    01010101
^   00100000
    --------
    01110101
```

Notice something? XOR-ing the same bit twice returned it to the same value. This nifty XOR property is used in calculating parity in RAID arrays and used in simple cryptography cyphers, but more about that in some other article.

Bit Hack ❻ Turn off the rightmost 1-bit.

```
y = x & (x-1)
```

Now it finally gets more interesting! Bit hacks #1 - #5 were kind of boring, to be honest.

This bit hack turns off the rightmost 1-bit. For example, given an integer 00101010 (the rightmost 1-bit in bold) it turns it into 00101000. Or given 00010000, it turns it into 0, as there is just a single 1-bit.

Here are more examples:

```
    01010111    (x)
&   01010110    (x-1)
    --------
    01010110

    01011000    (x)
&   01010111    (x-1)
    --------
    01010000

    10000000    (x = -128)
&   01111111    (x-1 = 127 (with overflow))
    --------
    00000000

    11111111    (x = all bits 1)
&   11111110    (x-1)
    --------
    11111110

    00000000    (x = no rightmost 1-bits)
&   11111111    (x-1)
    --------
    00000000
```

Why does it work?

If you look at the examples and think for a while, you'll realize that there are two possible scenarios:

1. The value has the rightmost 1-bit. In this case subtracting one from it sets all the lower bits to 1 and changes that rightmost bit to 0 (so that if you add 1 now, you get the original value back). This step has masked out the rightmost 1-bit and now AND-ing it with the original value zeroes that rightmost 1-bit out.
2. The value has no rightmost 1 bit (all 0). In this case subtracting 1 underflows the value (as it's signed) and sets all bits to 1. AND-ing all zeroes with all ones produces 0.

Bit Hack ❼ Isolate the rightmost 1-bit.

```
y = x & (-x)
```

This bit hack finds the rightmost 1-bit and sets all the other bits to 0. The end result has only that one rightmost 1-bit set. For example, 01010100 (rightmost bit in bold) gets turned into 00000100.

Here are some more examples:

```
    10111100  (x)
&   01000100  (-x)
    --------
    00000100

    01110000  (x)
&   10010000  (-x)
    --------
    00010000

    00000001  (x)
&   11111111  (-x)
    --------
    00000001

    10000000  (x = -128)
&   10000000  (-x = -128)
    --------
    10000000

    11111111  (x = all bits one)
&   00000001  (-x)
    --------
    00000001

    00000000  (x = all bits 0,
               no rightmost 1-bit)
&   00000000  (-x)
    --------
    00000000
```

This bit hack works because of two's complement. In two's complement system -x is the same as ~x+1. Now let's examine the two possible cases:

1. There is a rightmost 1-bit bi. In this case let's pivot on this bit and divide all other bits into two flanks: bits to the right and bits to the left. Remember that all the bits to the right bi-1, bi-2 ... b0 are 0's (because bi was the rightmost 1-bit). And bits to the left are the way they are. Let's call them bi+1, ..., bn.

   Now, when we calculate -x, we first do ~x which turns bit bi into 0, bits bi-1 ... b0 into 1s, and inverts bits bi+1, ..., bn, and then we add 1 to this result.

   Since bits bi-1 ... b0 are all 1's, adding one makes them carry this one all the way to bit bi, which is the first 0-bit.

   If we put it all together, the result of calculating -x is that bits bi+1, ..., bn get inverted, bit bi stays the same, and bits bi-1, ..., b0 are all 0's.

   Now, AND-ing x with -x makes bits bi+1, ..., bn all 0, leaves bit bi as is, and sets bits bi-1, ..., b0 to 0. Only one bit is left, it's the bit bi - the rightmost 1-bit.

2. There is no rightmost 1-bit. The value is 0. The negative of 0 in two's complement is also 0. 0&0 = 0. No bits get turned on.

We have proved rigorously that this bithack is correct.

Bit Hack ❽ Right propagate the rightmost 1-bit.

```
y = x | (x-1)
```

This is best understood by an example. Given a value 01010000 it turns it into 01011111. All the 0-bits right to the rightmost 1-bit got turned into ones.

This is not a clean hack, tho, as it produces all 1's if x = 0.

Let's look at more examples:

```
   10111100  (x)
|  10111011  (x-1)
   --------
   10111111

   01110111  (x)
|  01110110  (x-1)
   --------
   01110111

   00000001  (x)
|  00000000  (x-1)
   --------
   00000001

   10000000  (x = -128)
|  01111111  (x-1 = 127)
   --------
   11111111

   11111111  (x = -1)
|  11111110  (x-1 = -2)
   --------
   11111111

   00000000  (x)
|  11111111  (x-1)
   --------
   11111111
```

Let's prove it, though not as rigorously as in the previous bithack (as it's too time consuming and this is not a scientific publication). There are two cases again. Let's start with easiest first.

1. There is no rightmost 1-bit. In that case x = 0 and x-1 is -1. -1 in two's complement is 11111111. OR-ing 0 with 11111111 produces the same 11111111. (Not the desired result, but that's the way it is.)

2. There is the rightmost 1-bit bi. Let's divide all the bits in two groups again (like in the previous example). Calculating x-1 modifies only bits to the right, turning bi into 0, and all the lower bits to 1's. Now OR-ing x with x-1 leaves all the higher bits (to the left) the same, leaves bit bi as it was 1, and since lower bits are all low 1's it also turns them on. The result is that the rightmost 1-bit got propagated to lower order bits.

Bit Hack ❾ Isolate the rightmost 0-bit.

```
y = ~x & (x+1)
```

This bithack does the opposite of #7. It finds the rightmost 0-bit, turns off all bits, and sets this bit to1 in the result. For example, it finds the 0 in bold in this number 10101011, producing 00000100.

More examples:

```
    10111100  (x)
    --------
    01000011  (~x)
&   10111101  (x+1)
    --------
    00000001

    01110111  (x)
    --------
    10001000  (~x)
&   01111000  (x+1)
    --------
    00001000

    00000001  (x)
    --------
    11111110  (~x)
&   00000010  (x+1)
    --------
    00000010

    10000000  (x = -128)
    --------
    01111111  (~x)
&   10000001  (x+1)
    --------
    00000001

    11111111  (x = no rightmost 0-bit)
    --------
    00000000  (~x)
&   00000000  (x+1)
    --------
    00000000

    00000000  (x)
    --------
    11111111  (~x)
&   00000001  (x+1)
    --------
    00000001
```

Proof: Suppose there is a rightmost 0-bit. Then ~x turns this rightmost 0 bit into 1 bit. And so does x+1 (because bits more right to the rightmost 0 bit are 1's). Now AND-ing ~x with x+1 evaporates all the bits up to this rightmost 0 bit. This is the highest order bit set in the result. Now what about lower order bits to the right of rightmost 0 bit? They also got evaporated because x+1 turned them into 0's (they were 1's) and ~x turned them into 0's. They got AND-ed with 0 and evaporated.

Bit Hack ❿ Turn on the rightmost 0-bit.

```
y = x | (x+1)
```

This hack changes the rightmost 0-bit into 1. For example, given an integer 10100011 it turns it into 10100111.

More examples:

```
    10111100  (x)
|   10111101  (x+1)
    --------
    10111101

    01110111  (x)
|   01111000  (x+1)
    --------
    01111111

    00000001  (x)
|   00000010  (x+1)
    --------
    00000011

    10000000  (x = -128)
|   10000001  (x+1)
```

```
    --------
    10000001

    11111111  (x = no rightmost 0-bit)
|   00000000  (x+1)
    --------
    11111111

    00000000  (x)
|   00000001  (x+1)
    --------
    00000001
```

Here is the proof as a bunch of true statements. OR-ing x with x+1 does not lose any information. Adding 1 to x fills the first rightmost 0. The result is max{x, x+1}. If x+1 overflows it's x and there were no 0 bits. If it doesn't, it's x+1, which just got rightmost bit filled with 1.

Bonus stuff.

If you decide to play more with these hacks, here are a few utility functions to print binary values of 8 bit signed integers in Perl, Python and C.

Print binary representation in Perl:

```
sub int_to_bin {

  my $num = shift;
  print unpack "B8", pack "c", $num;
}
```

Or you can print it from command line right away:

```
perl -wle 'print unpack "B8", pack "c",
shift' <integer>

# For example:
perl -wle 'print unpack "B8", pack "c",
shift' 113
01110001

perl -wle 'print unpack "B8", pack "c",
shift' -- -128
10000000
```

Print binary number in Python:

```
def int_to_bin(num, bits=8):
 r = ''
 while bits:
  r = ('1' if num&1 else '0') + r
  bits = bits - 1
  num = num >> 1
 print r
```

Print binary representation in C:

```
void int_to_bin(int num) {
  char str[9] = {0};
  int i;
  for (i=7; i>=0; i--) {
    str[i] = (num&1)?'1':'0';
    num >>= 1;
  }
  printf("%s\n", str);
}
```

Have fun with these! ■

Peteris Krumins is 25 years old and he's from Riga, Latvia. He loves mathematics, physics and computer science. He currently works on his own startup called StackVM [stackvm.com], which puts virtual machines in web browsers.

Reprinted with permission of the original author.
First appeared in *hn.my/bithacks* (catonmat.net)

Dream. Design. Print.

MagCloud, the revolutionary new self-publishing web service by HP, is changing the way ideas, stories, and images find their way into peoples' hands in a printed magazine format.

HP MagCloud capitalizes on the digital revolution, creating a web-based marketplace where traditional media companies, upstart magazine publishers, students, photographers, designers, and businesses can affordably turn their targeted content into print and digital magazine formats.

Simply upload a PDF of your content, set your selling price, and HP MagCloud takes care of the rest—processing payments, printing magazines on demand, and shipping orders to locations around the world. All magazine formatted publications are printed to order using HP Indigo technology, so they not only look fantastic but there's no waste or overruns, reducing the impact on the environment.

Become part of the future of magazine publishing today at www.magcloud.com.

25% Off the First Issue You Publish

Enter promo code **HACKER** when you set your magazine price during the publishing process.

Please contact promo@magcloud.com with any questions.

MAGCLOUD