Coding for

by Johnny Ho

SHUTTERSTOCK

My pursuit of computer science started small. Bored in my middle school math classes, I fiddled around with my graphing calculator. At first, I looked for patterns in numbers. Next, I found ways to compute square roots, construct Pascal's Triangle, and solve counting problems. As my interest in math transferred almost completely to computer science, I started programming games, learning recursion, and exploring graphics. Seeing how far I progressed on a clunky calculator, it's not hard to imagine how captivated I became when I finally started learning C++ from my dad and found out about programming competitions.

Learning about various famous algorithms online, I was baffled by how they worked and how I could construct them myself. Once I got started, however, the thrill of thinking through problems and refining my thought process consumed me.

At the beginning of eighth grade, my brother introduced me to the intense world of competitive programming. For high school students in the U.S., this centers on the USA Computing Olympiad (USACO). Starting with the USACO training pages, competitors learn the basics of computer science. Each year, there are several monthly contests, dividing competitors into Bronze, Silver, and Gold divisions. At the end of each competition season, 16 students are invited to attend the USA Invitational Computing Olympiad (USAICO), a training camp where the top four competitors are then selected to represent the U.S. at the International Olympiad in Informatics (IOI).

Gold

The USACO and the IOI

I soaked up all of this information with curiosity, never expecting that I would reach further than the first few steps of this process. Based on what I saw on the website, I found the path to success to be incredibly daunting. Many of the top competitors had ranked internationally in math, others had started programming at a young age, most had completed the USACO training pages—none, it seemed, had gotten started by playing around with a calculator. Nevertheless, I had found a source of inspiration.

In the Company of Cows

My first year of competition was incredibly eye-opening. Although the first problems were conceptually easy, I still found C++ programming syntax difficult to decipher. At this point, I was also introduced to the idea of an algorithm. In theory, an algorithm is simple: command the computer to perform some operations on memory to translate input to some desired output. There is much more involved, however, with actually creating an algorithm and then implementing it in code. Learning about various famous algorithms online, I was baffled by how they worked and how I could construct them myself. Once I got started, however, the thrill of thinking through problems and refining my thought process consumed me. Exploring further, I found an inexhaustible variety of problems to practice on.

The first time I attended the USACO training camp, the summer before my freshman year, I was still fairly intimidated by the top competitors. Getting on a plane to Wisconsin, I was mostly concerned with the idea of traveling, living on my own, and meeting other students. Fortunately, at camp, competitors are divided into two groups: Holsteins and Guernseys (lovingly named after two breeds of cows). The Holsteins are put through a grueling course of competition, whereas the Guernseys are given easier contests and more lectures. At the end of camp, only Holsteins are eligible for IOI team selection.

Even as a Guernsey, I struggled to compete against other campers. On each contest, I found myself consistently unable to fully solve problems. Instead, I spent most of my time making simple optimizations, known as "hacks," in order to eke out partial credit. What I found far more interesting were the lectures, which are essentially entire computer science courses packed into a couple of days. Although I'm sure none of the campers could claim that they completely understood

the contents of the lectures, the key concepts stuck with me. Of particular interest were data structures, which are techniques for quickly storing, accessing, and querying data. As I trained over the next few years, the techniques taught in these lectures would reappear and steadily become easier to understand.

Interacting with the other competitors was just as rewarding. I found myself immersed in an entirely different culture. I'll never forget the experiences we had, including staying up past midnight to play card games despite having a contest the next day (yet still doing well!). Even today, my friends and I still joke about mistakes we've made over the years, argue about different styles of code, and discuss the merits of various data structures.

Guided by the dedicated organizers and brilliant coaches of the USACO, I have reached heights that, four years ago, I could not have imagined. I have met competitors from around the world who share my interests and, just as important, are fun to interact with.

Focus and Patience FTW

After two summers attending camp as a Guernsey, I was finally promoted to a Holstein. Doing fairly well among the Holsteins, I was invited to the IOI in Thailand, where I placed among the top 20 and received a gold medal. Although I was proud of my achievement, I was still struck with a sense of dissatisfaction. At camp, I had resorted to using numerous hacks to get easy points, and, at the IOI, I missed several key observations, resulting in suboptimal solutions. I left Thailand excited by the experience and eager to improve.

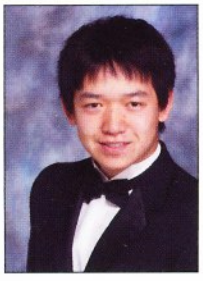
This past September at the IOI in Italy marked the culmination of all of these efforts. After four years of experience, this was the first time I could honestly say that I had familiarized myself with most of the techniques that could appear. There was, however, an issue: over time, the IOI has shifted toward using more ad-hoc problems—unpredictable problems that require intuition and logic more than technical knowledge. Many of these problems have no perfect solution, requiring creativity and flexibility to solve.

On the first day of competition, I put my well-practiced skills to use, patiently working through one problem after another. The most difficult problem of the day was an ad-hoc problem that introduced an "odometer" programming language reminiscent of Karel, which is typically taught in introductory Java classes. The hindrance, however, was that the language supplied only the most primitive programming operations. The contestants were restricted to navigating a 2D grid and moving stones around to act as counters. Without traditional programming constructs like variables, loops, and functions, the tasks became tremendously more difficult. Further, the problem was composed of five independent subtasks and graded using a partial-credit system. Strategically, I decided to leave it for last.

Fortunately, with focus and patience, I managed to solve each of the problems with a small amount of time to spare. Relieved at being able to eke out a perfect score in the last few minutes, I walked out preparing to be chastised for my last-minute submissions. Instead, I was greeted with cheers and warm smiles as the only perfect scorer of the day. Incredibly, I had managed to unseat the existing three-time champion, but it remained to be seen whether I could repeat the feat on day two.

On day two, faced with incredible pressure to perform, my mind was flooded with thoughts and concerns. When I finally managed to focus my attention on the actual problems, I found that they were less time-consuming than the previous day's. The problem statements, however, were unusually lengthy, which exacerbated my anxiety. Further, tripped by simple mistakes in my thought process, my programming took much longer than it should have. Ultimately, after re-reading the problem statements and sorting out my code, I walked out as one of three perfect scorers on day two. The previous day and its seemingly unremarkable odometer had proven to be the deciding factor—I had won.

From there, my experiences were fairly anticlimactic, since the full rankings of the IOI are released as soon as the contest ends. Given a few days to tour Italy and its scenery, I had time to reflect on my experiences. Thinking about the contest, I realized that the problems were of comparable difficulty to those presented at the USACO training camp. Guided by the dedicated organizers and brilliant coaches of the USACO, I have reached heights that, four years ago, I could not have imagined. Through two years of the IOI, I have met competitors from around the world who share my interests and, just as important, are fun to interact with. It's hard to say where I'll stand in this next year of competition, but I know that the skills and friendships I've developed through the USACO will last even as my participation comes to an end. i



Johnny Ho is a senior at Lynbrook High School in California and will be attending Harvard University in the fall. In his spare time, he enjoys playing cello, tossing a Frisbee around, and browsing Quora.

Magical &

Research in Computer Science

You might think your computer is fast and that your smart phone is smart. You might feel pretty confident that you've secured your digital information with an unbreakable lock and key. But at universities across the country, there are computer science students who think there's plenty of room for improvement in these and many other areas. In their research, they envision the path from what *is* to what is *possible*. Here, four of these graduate students share their work, what led them to it, and how computer science is helping them create the future they imagine.

HUMAN-COMPUTER INTERACTION

BY JULIA SCHWARZ

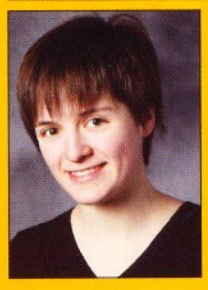
When I entered college, I was certain that I would never study computer science or anything computer-related. Then, as a sophomore, I grudgingly took an introductory computer science course at my parents' urging. To my surprise, writing programs felt like solving puzzles, which I've always enjoyed, and the immediate feedback of running a program and seeing results was quite rewarding. The following summer, I attended Microsoft's TechFest, where researchers show off their coolest inventions, and students and professionals get a glimpse of the future. One project in particular drew me in: Soap (www.patrickbaudisch.com/projects/soap), a mouse that could easily be controlled in midair. The mouse was wrapped in a fabric hull, and to control the pointer, the user simply needed to move the surface (the fabric hull) along the mouse. The cleverness and elegance of this device captivated me and planted a seed of interest in my mind that has grown into a passion for Human-Computer Interaction.

Sitting at the intersection of several disciplines—computer science, cognitive science, social science, and design—Human-Computer Interaction (HCI) involves studying how people use computers and designing tools to improve that interaction. My work is closer to the computer science side of HCI and falls into two categories. First, I am working on developing new sensing systems and interaction techniques for mobile devices. For example, I helped develop a sensing technique called FingerSense that determines which part of your finger you are using when touching a mobile device. This adds a secondary input to touch, similar in function to right and left mouse buttons. For example, you could knuckle-tap on an image or email message to bring up a menu of options, eliminating the need for press-and-hold or a menu bar.

The second part of my research involves developing systems for handling different types of user input. Consider touch input: When you touch a screen with your finger, most systems treat that contact area as a single point, but it's really larger than that. There may be better ways—such as modeling the intended touch as a probability distribution—to more precisely infer what the user is trying to touch and therefore make an interface that more accurately and reliably responds to the user's intent.

What draws me to HCI is also the thing that makes it most difficult: Because HCI combines many disciplines, it's not enough to be a good programmer. A good HCI practitioner must be a programmer, a designer, a statistician, and an artist. HCI practitioners are the Renaissance men and women of the Information Age. And because we all interact with computers and information on a daily basis, innovation in this field is incredibly important. Just think about how much of an impact the iPhone—a device that popularized innovations in Human Computer Interaction—has had on our daily lives.

I plan to continue pursuing research on new interaction techniques for touch and in-air gesture. In the future, I hope to build and perhaps bring to market some of the interfaces we see in movies (think *Minority Report* and *Cloud Atlas*) in a way that is both magical and practical.



Julia Schwarz is a PhD student studying Human Computer Interaction at Carnegie Mellon University. She has worked as an associate researcher at Microsoft Research and on the Xbox NUI Platform team, and is co-founder and research director of Qeexo. She participated in several CTY programs as a teen and credits CTY for several key turning points in her education.

Practical

COMPUTER SECURITY

BY J. AYO AKINYELE

My interest in computer science was already well established by the time I got to high school. My older brother had majored in computer science and taught me how to write programs when I was about 10 years old. I found programming very exciting, primarily because I was able to give the machine "instructions" and it faithfully executed them. The experience was empowering, and to me, the possibilities with computer science seemed endless. I decided to major in computer science in college.

During my sophomore year, I took a course on computer security and learned about viruses, worms, and rootkits—and their potentially devastating effects. These types of malicious software can wreak havoc on a computer's defenses, rendering it completely useless or turning it into a zombie, hiding any trace of the software while giving the hacker remote access to the computer. The more I learned, the more I wanted to pursue research in this area. In particular, I wanted to learn how to protect computer systems from cyber attacks and how to protect user data on systems that are under attack. This led me to pursue research in information security in graduate school.

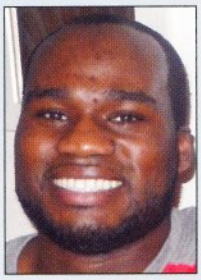
One of the central tools for providing information security is cryptography. Through encryption, a message or information is transformed into something that appears random and meaningless; only a person with a key can decode and read the message. An analogy is a locked mailbox: anyone can put a letter into the mailbox's slot, but only the person with the key can open that mailbox and access those messages. This type of encryption, known as public key encryption, is widely used today. Online activities such as logging in to a bank account or accessing email rely on public key encryption behind the scenes to secure data.

One of the strengths of this type of encryption is that each encrypted message can be opened by only one key, but that can also be a limitation: What if you wanted to allow several recipients to access the same data but didn't want to encrypt it separately for each recipient?

In my research, I have explored ways to encrypt data such that access is determined based on whether the recipient satisfies certain criteria. One form of this type of encryption is called attribute-based encryption (or ABE). For instance, say Alice wants a confidential report at her university to be accessible only to faculty in the history, computer science, and psychology departments. Alice can use ABE to specify which faculty members will be able to access the report, and those criteria are cryptographically enforced: The private keys issued to faculty members will work only if they are members of the departments Alice has specified, so if they were to forward the report to someone outside their department, that person would not be able to read it.

I am interested in using this type of encryption to protect electronic medical records (EMRs), which are becoming widely adopted as a way to both reduce costs and improve patient care. While EMRs must be accessible to caregivers, they contain extremely sensitive information that must be protected. I developed software that successfully implements ABE and a prototype iPhone app that uses ABE to provide protections for EMRs. For example, a patient's records could be encrypted so that credentials from the patient, doctor, or nurse are required to read the records.

Advances in cryptography like these will be critical in combating evolving information security threats in the near future. Modern cryptography is only part of the solution, but I am excited about the diverse opportunities to utilize it to solve practical, everyday problems.



J. Ayo Akinyele earned his BS in computer science from Bowie State University and his MS in software engineering from Carnegie Mellon. He is now a fourth-year PhD student at Johns Hopkins, focusing on applied cryptography research. Ayo enjoys playing the drums, reading, and watching movies, football, and basketball.

SHUTTERSTOCK

COMPUTATIONAL BIOLOGY

BY KATHERINE ROMER

Biologists have long dreamed of understanding the genome: the DNA that controls an animal's body plan, development, and behavior. In 2003, scientists finally managed to get a nearly complete sequence of the human genome (3.2 billion base pairs), at a cost of 13 years and 3 billion dollars. Now, we can get 3.2 billion base pairs of data from a single sequencing machine in a week—and that's from a cheap machine having a bad week.

This fast, cheap sequencing allows biologists to answer questions about the genome more quickly, and on a much larger scale, than we previously could have imagined. Discover a new organism? We can sequence a large part of its genome in a couple of weeks. Identify an interesting mutation (a fly with white eyes, for example)? Sequencing can help us rapidly identify genes that might be responsible.

However, to reap the benefits of sequencing technology, we have to deal with some interesting computational challenges. Modern sequencers produce enormous quanitites of data—a single machine churns out almost a terabyte of data each week. Simply storing and processing our data requires clever use of data compression and parallel processing. Then, once we've generated and stored our sequence data, we have to assemble it. A sequencer doesn't spit out the whole sequence of a genome at once; it can sequence only a tiny bit at a time—often, each sequence piece is only 20 base pairs. We get billions and billions of these tiny pieces, and develop algorithms to put them together into one complete sequence. Finally, once we have completed sequences, we can use a variety of statistical techniques to compare them to one another.

During my PhD research, I've used sequencing-based approaches to study the mouse reproductive system, with hopes that my work can be used to develop new birth control methods and infertility treatments. I'm particularly interested in new applications of sequencing technology to understand the regulation of genes in developing egg and sperm cells. Every cell in a mouse has the exact same genome sequence, so there's no genetic difference between a developing egg and a developing sperm. But during development, different sets of genes are turned on in these cells, resulting in their different shapes, sizes, and functions. It turns out that sequencing is a powerful tool to figure out these regulatory differences. In particular, we can isolate the active genes in sperm and eggs, sequence those genes, and analyze which genes are significantly different between the two cell types. Once we understand how a cell activates genes to become a sperm or an egg, we might be able to copy this gene activation program, and thus produce sperm or eggs outside the body.

I really like being able to apply my computer science skills to solve biological problems. When I started college, I knew I loved computer science, but I wasn't sure where I wanted to apply it. I stumbled into computational biology in my sophomore year of college when I got a part-time job designing a website for a biology lab, and I got hooked. In graduate school, I enjoy dealing with computational challenges of large-scale DNA sequence data, and I love that I'm contributing to our understanding of human health and disease.

Katherine Romer studied biology and computer science as an undergraduate at MIT and is now working on her PhD in computational biology there. In her free time, she enjoys gardening, church, and square dancing.

CHIP DEVELOPMENT

BY MAX SHULAKER

When I first arrived at Stanford as an undergraduate, I thought that if you wanted to have a tangible impact on technology, industry was the only choice. Academic research seemed impractical to me. Of course, I've since learned that I was wrong. Now I'm a PhD candidate and spend the majority of my time on research.

Over the course of my undergraduate education, I noticed that engineering classes typically follow the same outline: a week or two of introduction and background information, followed by the bulk of the class material, ending with a weeklong overview of "what's coming up in the future." In classes focusing on computer chips and transistors (the devices that make up the bulk of computer chips), the "what's coming up in the future" always had the same punch line: we don't know.

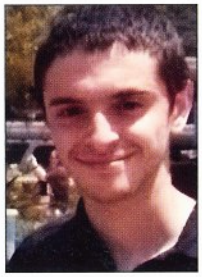
Over the years, computers have become increasingly powerful, as you have probably noticed. This increased power is achieved by physically shrinking the transistors. The smaller the transistors, the more we can fit onto a chip, and thus the more computing power we get. For the past several decades, Moore's Law, which states that the number of transistors in integrated circuits doubles every 18 months, has been driving the entire semiconductor industry. But we're fast approaching a physical limitation that makes scaling smaller increasingly difficult.

At the end of one my undergrad classes, Professor Subhasish Mitra (who is now my advisor) discussed promising emerging technologies that might be able to extend Moore's Law for years to come. Specifically, he was talking about the great promise of carbon nanotubes, tubes of carbon with a diameter approximately 10,000 times smaller than that of a human hair. It has been shown that carbon nanotubes can be used to make nearly ideal transistors and might be a way to keep scaling smaller to increase computing power.

The carbon group at Stanford, led by Prof. Mitra and Prof. Philip Wong, includes about 15 students. Our research on carbon nanotubes ranges from modeling the properties of a single transistor to modeling variations in circuits with billions of these transistors. One of the biggest challenges of working with carbon nanotubes is that they tend to grow randomly on a wafer, looking much like spaghetti thrown on a plate. As you can imagine, it's nearly impossible to build a billion-transistor design with such a random configuration. We've developed a process that allows us to grow the nanotubes so they're almost perfectly aligned, and we've also created layout designs that allow circuits to function despite any remaining mispositioned carbon nanotubes.

Using everything the group has learned over the years, we recently built a subsystem entirely out of carbon nanotube transistors. We've even hooked up the carbon nanotube circuit to a motorized arm to build a hand-shaking robot. It's a fun demonstration, and it illustrates that we're reaching a point of maturity in this technology when we can have live demonstrations of working circuits.

While we still can't answer for certain what's coming up in the future, actively engaging in this research is one way to help shape what that will be. i



Max Shulaker is a PhD student at Stanford University. When he's not in the lab, Max is either playing jazz trumpet, knocking around a tennis ball, or enjoying the beautiful California weather.

In February, Wiley Publications released *Minecraft for Dummies, Portable Edition*, a guide for beginners and those who want to explore some advanced features of the game. Here, the 16-year-old author of that guide shares what got him hooked on *Minecraft* and how it has inspired him to new levels of creativity.

The Creative, Collaborative Universe of MINECRAFT

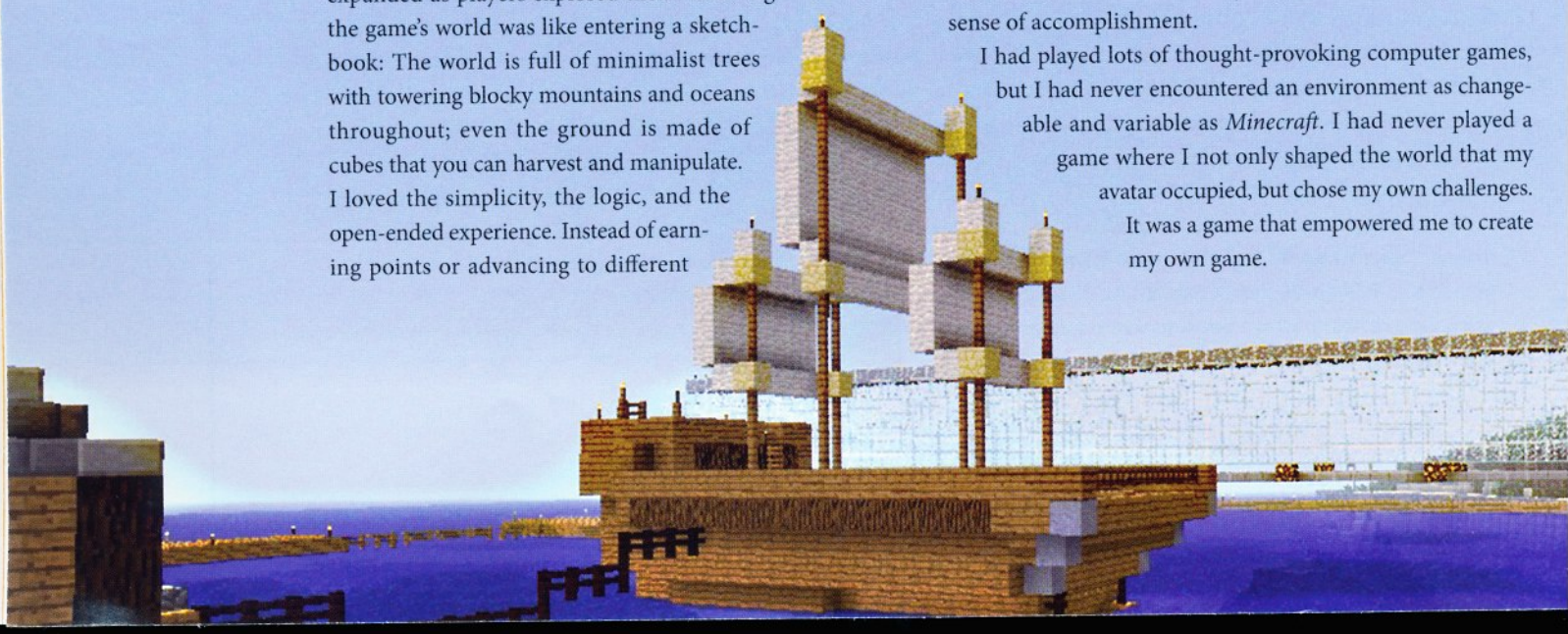
by Jacob Cordeiro

If you enjoy games about building, survival, engineering, and adventuring, Minecraft *is for you. Having attracted more than 8 million players,* Minecraft *is a loose-ended yet adventurous sandbox game that becomes whatever you make of it.* —from *Minecraft for Dummies, Portable Edition*, by Jacob Cordeiro

I started playing *Minecraft* a few years ago when it was an under-construction sandbox game. Even then, it was immediately appealing. Instead of offering a static setting where everything was pre-made and pre-determined, the game's algorithm generated detailed, randomized landscapes that expanded as players explored them. Entering the game's world was like entering a sketchbook: The world is full of minimalist trees with towering blocky mountains and oceans throughout; even the ground is made of cubes that you can harvest and manipulate. I loved the simplicity, the logic, and the open-ended experience. Instead of earning points or advancing to different levels, the only goal was to build what you liked.

For example, players could dig through underground caves, breaking the rock into efficient formations so they could build structures or extract resources to build powerful gear. My first buildings—a wooden fort and a ladder up the face of a cliff—weren't exactly majestic, but they gave me a sense of accomplishment.

I had played lots of thought-provoking computer games, but I had never encountered an environment as changeable and variable as *Minecraft*. I had never played a game where I not only shaped the world that my avatar occupied, but chose my own challenges. It was a game that empowered me to create my own game.



Creativity and Survival

After *Minecraft* Classic—as that early version of the game is now called—the game offered two modes that allowed players to interact with the world in two different ways. In Creative mode, which retains the sandbox characteristics of Classic, players could continue to build and explore the world according to their own goals. The addition of redstone, a collectible "mineral" that can be placed like a block and arranged in such a way that it powers mechanisms, allows players to build everything from elevators and computers to automated improvements such as self-managing farms in their worlds.

In Survival mode, however, manipulation of the environment is essential to your avatar's survival. In this mode, the game is a challenge that entails collecting resources such as food, armor, and tools, and maintaining the health and safety of your now-mortal avatar. Collecting and moving blocks now requires time and the proper tools, and luxuries like large houses and precious metals have tangible value. Along with the addition of limited resources such as food, tools, and minerals, Survival mode introduced the challenge of fighting blocky monsters that roam the landscape. This mode not only encourages creativity but requires it, as players must become increasingly devious in the face of adversity.

Sharing Ideas, Mods, and Adventures

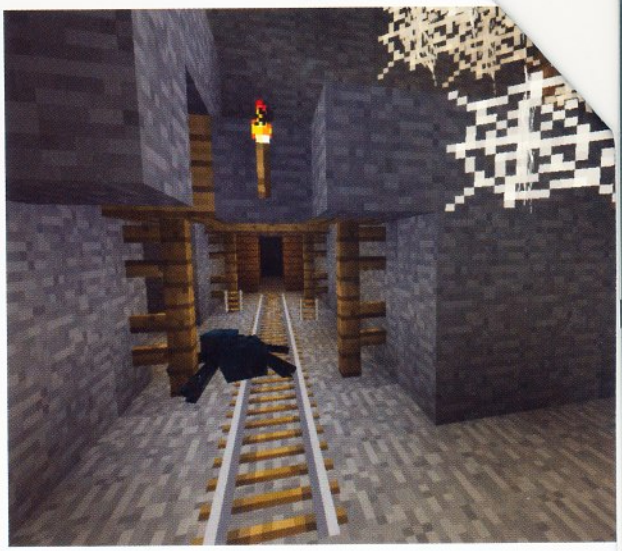
Activities like building a house and crafting a collection of survival items open up opportunities to collaborate and share interesting ideas. Some players build entire adventures and challenges that incorporate their own assortment of monsters or economic difficulties, which other players can then download and attempt. I've attempted some ingenious user-made adventures, puzzles, and challenges for surviving in harsh conditions.

One of the biggest catalysts of this collaboration is Multiplayer mode, which allows multiple players to create, survive in, or adventure through the same shared world. Players can share their ideas in a way that directly benefits their friends, while experiencing the added challenges of sharing resources and building a solid online economy. Residents of a shared world can trade, team up, duel, or divide a list of tasks to make their world grow much faster.

Minecraft's rich gameplay derives from the intricate balance of the world's resources and the ingenuity of the players. Even in the Classic version of the game, players had the ability to program new features into the world and share these modified or "modded" games with others. That spirit of invention and sharing remains pervasive, and there are huge communities and forums where players share ideas, worlds, mods, and other relevant content.

Inspired Education

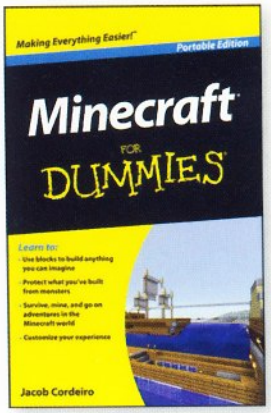
For me, *Minecraft* provides inspiration that extends beyond the game itself. I've always had a passion for world-building, which I have pursued through writing, drawings, and sketching maps, but *Minecraft* is one of the best ways I've found to express myself. Building in both Survival and Creative modes has given me the mindset to build worlds in much more detail; in fact, my first major creative writing project is a trilogy set in the *Minecraft* world.



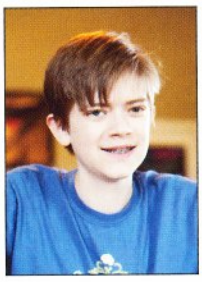
Cave spiders are mobs (mobile entities) that inhabit the *Minecraft* world. Neutral in daylight, these spiders turn hostile and dangerous in the dark.

I've also gained skills I can use to build my own games. *Minecraft* allows you not only to play a game, but to design and analyze your game experience at the same time. This helped me realize that I really enjoy designing games, especially games that teach or are built on interesting ideas in mathematics and science. I've built some simple games like a block-sliding puzzle and a shooter with a counter-intuitive spell system, and I've sketched many others, such as a defensive survival game that involves controlling the evolution of hostile creatures.

In building games based on grid formats or based on the clever employment of resources, I've drawn from some of the characteristics I most admire in *Minecraft*. The various in-game challenges I've encountered in *Minecraft* (such as building a bridge from an arrangement of limited-capacity pistons) have inspired me to new levels of technical mastery in my own games.



Minecraft has provided me with both a creative outlet and a community. I think this is what makes *Minecraft* such an enriching and rounded experience. It is an entertaining, open-ended game based firmly on logic and analysis, and it has attracted a community of creators who are passionate about exploring its unlimited possibilities.



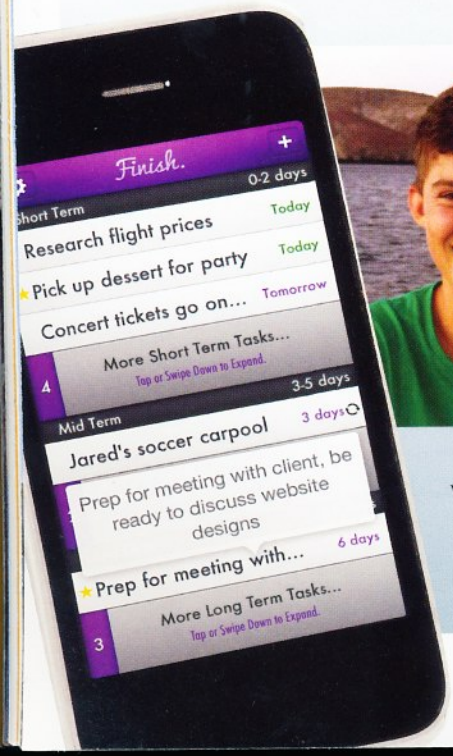
Jacob Cordeiro, 16, attends the Stanford University Online High School. A math and gaming enthusiast, he was a panelist at the Games for Change conference in 2012. Jacob wrote *Minecraft for Dummies* out of his love for inventive games, and he hopes to design interdisciplinary games in the future.

Going Mobile

by Amy Dusto

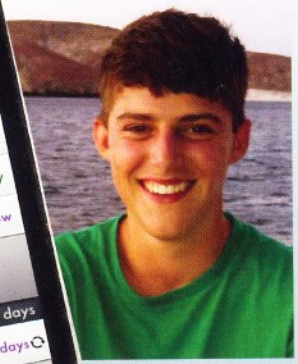
Cool Apps Built by Teens

When you browse Apple's App Store or Google's Play Store, it might seem that there is an app for everything. But when you need an app to solve a specific problem in your everyday life, you might find, as these high school students did, that sometimes the best way to get the app you need is to develop it yourself.



FINISH

Last year, Ryan Orbuch decided that the planner that his Boulder, Colorado, high school distributed to students wasn't helping him organize his busy life. So he came up with a more useful and convenient solution, a planner app for the iPhone that automatically pushes more pressing tasks to the front of the line and reminds users when their deadlines are approaching. Called "Finish," it went on sale in January for 99 cents. So far, it's been downloaded more than 24,000 times, Ryan says.



Finish is designed to organize tasks according to timeframes the user sets. For example, a student could indicate that short-term tasks must be complete within two days, medium-term tasks within two weeks, and long-term tasks within a month. Then, as time goes by, the app slides the tasks into the right categories, keeping the more pressing ones at the front. The app also lets users set notifications to remind them to, say, finish their homework.

On the first day the app became available in the Apple App Store, it reached No. 2 in the productivity category; by the next morning, it was No. 1. He sells the app under the label of the company he founded while building it, Basil Ltd. Beyond updates and expanded features that Ryan says are in the works, he's planning to develop other education-related technology projects through Basil.

The 16-year-old 11th grader had never built an app, started a business, or done much coding before, he says. But he knew where to look for those skills: Ryan asked his friend and classmate Michael Hansen, a more experienced programmer, to build the prototype he designed. They started early last fall, with Ryan handling the layout and look of the app and Michael writing the code that directs how it works.

During the project, the two realized they each needed help. On the coding end, Michael turned to online app development forums to figure

THINKSTOCK

out how to fix snags in his work when he wasn't sure what to do. Ryan, in addition to asking all his classmates for feedback—and later asking them to test a not-quite-finished version of the app before submitting it to Apple—began reaching out to experts for advice. "I'm a big fan of Twitter. It's an incredibly valuable platform to learn who's involved in things and ask them questions," he says. He also began attending any kind of technology event or meetup he could find in his hometown—Boulder is known for its strong community of technology entrepreneurs—to get help from other, more experienced creators.

"You really start slowly, which is frustrating," Ryan says. "It took a lot longer than expected." But once he and Michael started to get the hang of what they were doing—and learned where to find help when needed—the only thing left to do was practice.

CLIPPED

Tanay Tanden's "Clipped" app allows users around the world to keep up with news they are interested in without reading through entire articles. The free app, which works on Apple and Android phones and as a plugin for the Google Chrome web browser, analyzes text to extract the most relevant information and present it in easy-to-digest, bullet-point summaries.

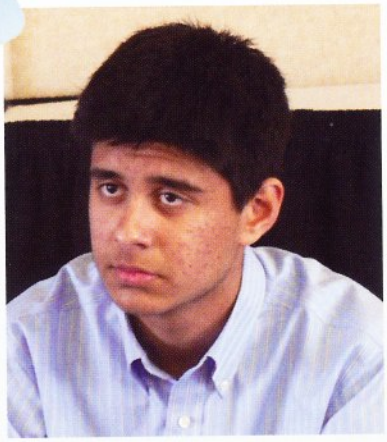
SHUTTERSTOCK

Tanay's idea for making Clipped originated from a personal need before he realized it could be useful to other people, too. "As a Lincoln-Douglas debater, I was tired of reading through dozens of evidence files and highlighting bits of information to include in my written case," he says. "I wanted to write a computer program that could extract the most relevant information from the news article or document and immediately show me the important stuff."

To begin, Tanay, a high school sophomore, needed to write an algorithm that could search for grammatical patterns and keyword frequencies to identify the key parts of an article. He already knew the Java programming language from reading a book his father gave him. To build the algorithm, he taught himself two more languages, PHP and JavaScript, by reading library books and following online tutorials.

"I've learned that making an app is a really long process—it doesn't just happen overnight," he says. The biggest challenge was learning how to design a user interface, the part of the app used to search for news by keyword or topic, also known as the front end. Previously, all Tanay's programming experience had been with the back-end coding—the part of the app that users don't see but that allows the app to function. "I had the algorithm mostly coded, but I wasn't sure how I wanted users to use the mobile application," he says.

So Tanay looked at similar apps, including Flipboard and News.me, to help him formulate the best layout. He chose to create a news feed, a web page that constantly adds more recent articles to the top of the screen as they become available. This



way, Clipped users always see the most recent news stories from around the world right away. "By simply scrolling through the app, it is possible to quickly get an idea of what is going on in the world," he says.

Tanay also had to meet the challenge of coding the app to display properly on various screen sizes. "Since there are so many different types of Android devices, I did my best to make the app fit on all of them, and this was a tedious process," he says. Now he's working on adding features so the app can analyze text from documents and research papers in addition to news articles. He'll also soon release an update that allows users to customize the app to show them news from select sources only, like a particular website.

"It was a great feeling to see Clipped come alive when the app started to deliver summaries," Tanay says. So far, he says the app has more than 35,000 downloads and has translated 2.8 million news articles into bullet points. "Getting into coding is much easier than it seems," he says. "Just picking up a book or following a tutorial online is enough to get you started, and then it's really fulfilling to be able to create something of your own."

SHUTTERSTOCK

PRONUNCIATION CHECKER

Fifteen-year-old Richard Nehrboss built an Android app that allows people to speak into their phones and get feedback on how well they are pronouncing a particular word. Called "Pronunciation Checker," the app helps when learning a foreign language, but Richard was inspired to make it to address another need.

"My little brother and a few other people I know have difficulty pronouncing words with sounds such as 'ch,'" he says. But gauging how well people are pronouncing a word can vary, depending who is giving the feedback, he says. An unbiased computer algorithm, though, would give them a consistent, definite measure of their progress. When he couldn't find an existing app that did this as well as he wanted it to, Richard decided to build one himself.

He'd begun learning computer programming from his father at age seven, but he didn't start learning Java, the programming language needed to make Android apps, until last year in his AP Computer Science classs. He also watched YouTube tutorials to learn how to start using Android development software. Once he began building Pronunciation Checker, he asked for feedback from friends with speech impediments to make sure the app worked well and was easy to use.

Despite his coding background, creating the app was still a challenge. "Sometimes even the simplest things take forever to accomplish, but it is vital that you don't get discouraged and give up," he says. For example, after spending tens of hours starting the project, a particular file Richard was using suddenly stopped working and he was forced to redo everything. "I was tempted to give up because I couldn't stand losing everything I had done."

But he kept going, and the app launched in the Google Play app store in January. It checks pronunciation for more than 30,000 words in English, French, Spanish, and German.

Looking back on the process, Richard says that he wishes he'd defined a clearer plan of action about which features he wanted to be in the app from the start, rather than adding new ones as he thought of them. Although the project eventually succeeded, he says he could have done it much more efficiently with a better plan. Then he might have had a single database of all the app's settings, for example, rather than having to create a new one for every feature he added.

"I have long been interested in computer programming, but it was always purely for recreation," he says. "I never knew just how rewarding it is to watch other people use and like something you develop. I got hooked."

While he's not sure what his college major will be, Richard says he's confident he'll be doing some kind of programming. For now, he's trying to come up with his next app idea.

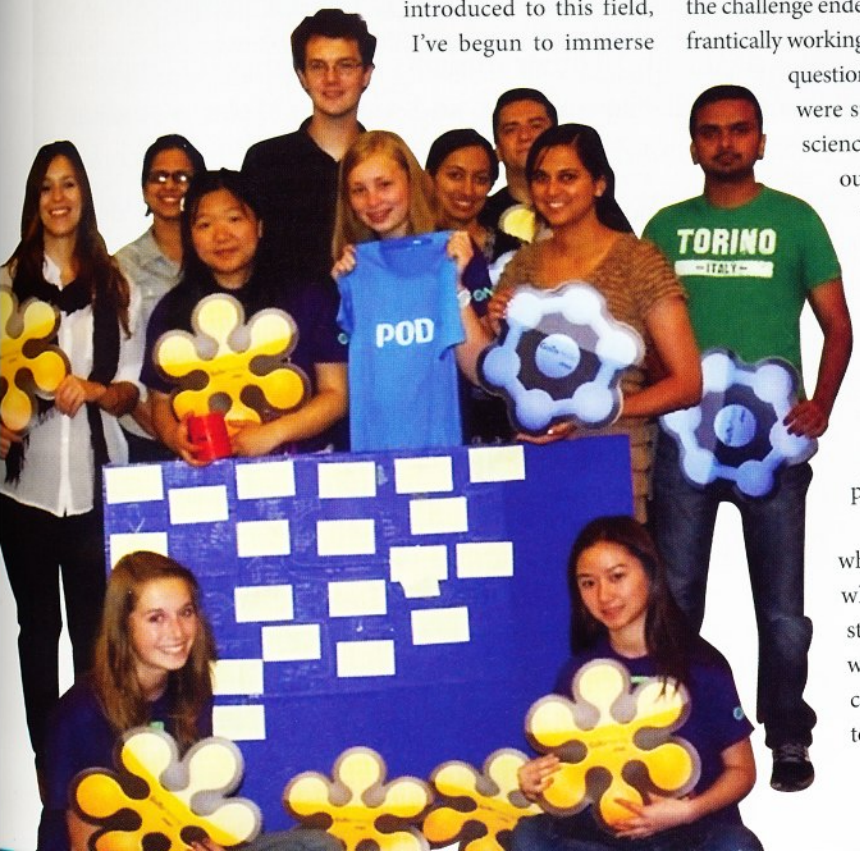
STÜDI-US

Studying for AP science tests can be a drag—at least according to members of team CoffeeBeans, who built an app called "Stüdi-US" to make it more fun. Stüdi-US provides flashcards, a game, multiple choice and free response practice questions, and test simulations for AP science courses. The app also tracks users' correct and incorrect answers so that the next time they study, it can show them more questions related to topics they need extra practice answering, says team member Wendy Li.

Last year, Wendy and her classmates Lisa Illes, Siyao Ma, Elena Prioreschi, and Denise Leung entered a 10-week app development contest called the Technovation Challenge. They won, earning a $10,000 prize to help them finish the app, which they are doing with the help of professional programmers. Stüdi-US will be available first for Android and later for Apple phones.

The girls entered the contest with little to no coding experience. Wendy, who was enrolled in a computer science class at the time, took the lead on programming the app, while Siyao helped gather data and Lisa helped with the design. In addition to learning about coding, team CoffeeBeans also learned how to create a business plan and design their app to appeal to a large audience, Wendy says.

"I really didn't comprehend all the elements that go into app development, like developing a full business plan," Siyao says. "Now that I've been introduced to this field, I've begun to immerse myself in everything it has to offer." A junior, she has begun an AP computer science class and says she loves it. Lisa, currently a sophomore, also plans to take a coding class in high school and enter the business and technology field as a career. Wendy, a senior, plans to pursue a major in electrical engineering and computer science in college.



All the girls say that time was the biggest obstacle they faced. "We had ten whole weeks—at the beginning of the program, that is," Siyao says. "Ten weeks turned to seven, then five, before we realized we had to take time out of our already busy schedules to complete our task." The last two weeks before the challenge ended were stressful for everyone. Wendy recalls frantically working to fix a bug in the code—the multiple choice questions she'd plugged in weren't loading when they were supposed to—until she asked her computer science teacher to help her troubleshoot. It turned out she was missing an "if, then" clause, which, when she added it, solved everything.

"Looking back, I think that the most important element in making an app is teamwork," Siyao says. "This was most apparent near the end when the business side of app development was introduced; we had to be able to communicate and rely on each other to create a cohesive business plan and effective pitch."

The girls also needed to decide collectively what to include in the app in the first place—which ate up a lot of the team's time getting started, Lisa says. "I learned a lot about how I work independently and as part of a team in a creative setting, which I think will be helpful to me for the rest of my life." i