EPITA SUP

Promo 2017

Technical English Package

Max TESTEMALE: testem_m@epita.fr

Table of content

I. **Syllabus.**

II. **Planning of the year.**

III. **Articles.**

- How Do I Get Started in Game Programming
- From Sand to Silicon: Manufacturing an Integrated Circuit
- Android Ads Could Attack
- The IPhone has passed a key Security Threshold
- Car to Car Communication Systems
- TED: The Future Race Car
- A Smarter, Stealthier Botnet
- TED: All Devices Can Be Hacked
- Talk to the Machine
- Where Speech Recognition Is Going
- TED: Gaming Can Make a Better World

IV. **Guidelines for the Major Project.**

V. **Rules about the Major Project report.**

VI. **Rules about the Major Project Presentation.**

VII. **Preparing for *"Presentation 4" of your IT project.***

VIII. **Technical English Report front page.**

Technical English Syllabus

Teacher: Max TESTEMALE
Email: testem_m@epita.fr

Classroom Rules :

1. No French will be spoken in the class, neither by the students nor by the teacher.
2. No electronic items will be tolerated in the classroom (i.e., laptops, cell phones, etc.).
3. Eating is not allowed in class.
4. Arriving late will not be tolerated.

Objectives:

The objectives of this class include: improving vocabulary, written and oral comprehension as well as expression of technical English. It will be divided into different activities such as translation of various articles about IT related subjects and short presentations given by the students.

Content:

Comprehension:
Translation of IT related articles.
Focus on vocabulary used in articles.
Understanding of the subjects.

Oral expression:
Presentation in groups of 3 or 4 students + questions.
Discussion on the various subjects dealt in the articles and presentations.

Major Project:
Preparing for an hour long presentation to be given in groups of 3 or 4 students

Grades overview:

- QCM's: All questions of the QCM's will be based on the articles that we would have studied in class.
- *Contrôles* et *Partiels:* Translation, QCM, etc...
- Group presentations in Class: Two presentations per student
- Major Project

Technical English
Planning of the Year 2012/2013

Week	Program	Date
Week 1	Welcome presentation To Tech English	Week of Sept 24th
Week 2	How do I get Started in Game Programming	Week of Oct 1st
Week 3		Week of Oct 8th
	QCM 1 Tech	
Week 4	From Sand to Silicon: Manufacturing an Integrated Circuit	Week of Oct 15th
Week 5		Week of Oct 22nd ← general QCM
Week 6		Week of Nov 12th
	QCM 2 Tech	
Week 7	Android Ads Could Attack	Week of Nov 19th
Week 8	The Iphone has passed a key Security Threshold	Week of Nov 26th
	QCM 3	
Week 9	Car to Car Communication Systems	Week of Dec 3rd
Week 10	TED: The Future Race Car	Week of Dec 10th
	QCM 4	
Week 11	Presentation Prep 1	Week of Dec 17th
Week 12	A Smarter, Stealthier Botnet	Week of Jan 21st
Week 13	TED: All Devices Can Be Hacked	Week of Jan 28th
	QCM 5	
Week 14	Talk to the Machine	Week of Feb 4th
Week 15	Where Speech Recognition Is Going	Week of Feb 11th
	QCM 6	
Week 16	Presentation Prep 2	Week of Feb 18th
Week 17	Presentation Series 1	Week of March 18th
Week 18	Presentation Series 2	Week of March 25th
Week 19	Presentation Series 3	Week of April 1st
	QCM 7 (includes every articles) from previous QCM	
Week 20	Presentation Series 4	Week of April 8th
Week 21	Presentation Series 5	Week of April 15th
	QCM 8 (includes every articles)	
Week 22	TED: Gaming Can Make A Better World	Week of April 22nd
Week 23	Preparation for the IT project Presentation	Week of May 13th
Week 24	Course Evaluation	Week of May 20th

How do I get Started in Game Programming.

When you first begin to learn about game programming, it can be difficult to know where and how to start. Every game programmer has their own views, derived from their own, first-hand experiences, about the best way to learn about game design and about the best way to design a game. If you tried to follow all of their advice, and gave each programmer's opinion equal weight, you would quickly discover that you were going nowhere. You wouldn't even be able to pick a programming language to write in.

The very first thing you need to learn as a programmer is that **it is up to you to decide what kind of game you are going to make**. Ultimately, you are going to have to make all of these decisions yourself (or with the aid of your teammates), and it is the aim of this article to give you the confidence and direction you need to make those decisions.

Contents

- 1 Rule One: Know yourself
  - 1.1 Programmer stereotypes
- 2 Rule Two: Know your objective
  - 2.1 Programming paths
- 3 Tips on designing your first game
- 4 Choosing a programming language
- 5 Choosing a library/API
- 6 Educating yourself
- 7 Creating game media
- 8 Coding tips
- 9 Future considerations
  - 9.1 Deploying your game
  - 9.2 Selling your game
- 10 Getting more help
- 11 External links

Rule One: Know yourself

Before you type one word in your design document you have to stop and ask yourself a question: **What are you willing to do to make this game?**

prêt à faire

Not all game programmers are created equal, so it's natural that different programmers will be willing to do different things to get the job done. But these differences in temperament can lead to drastically different design paths. Before you start working on your game, think about the following types of programmer stereotypes and see how you fit in.

champs de conception

HUD = affichage tête-haute

Programmer stereotypes

s'attaque aux problèmes de programmation obscures

tinkerer
Bidouiller

- **The hardcore techie:** the techie is the kind of programmer who seems like they were born to program computers. They love to tackle arcane programming problems and thrive on making computers do things that other programmers wouldn't even consider possible. Their main objective is to master the technology and pioneer new fields of research. This kind of programmer is good to have on your team, if you're not one yourself, but they tend to get bogged down in details and veer off on unrelated tangents. They are also prone to a tendency to reinvent the wheel merely for the pleasure of doing it. If you are a hard core techie, you may need to team up with a visionary who can keep you on track or learn to nurture your own creative vision through creative writing, music or visual art.

  le technicien inconditionnel
  prospérer
  initier
  to master = maîtriser
  bogged = embourbé
  veer off = obliquer/dévier
  wheel merely = roue simple
  nourrir

- **The visionary:** the visionary is the kind of programmer who can see the end results of a program vividly in their mind's eye, but not always the means to get there. They can see every menu option, every HUD element, and every feature of a game clearly, exactly how it should look, and how it should be used, and how much fun it would be to play. They tend to be good writers, artists, or musicians as well. This kind of programmer is good to have your team too, but is more likely to contribute through ideas and enthusiasm than through original code. If you are a visionary yourself, you will probably want to team up with a hard core techie who can solve the kinds of unusual problems that your ideas will inevitably create or force yourself to spend some time every day learning more about computer science.

  clairvoyant
  AFH
  le visionnaire
  IDE

- **The tinkerer:** the tinkerer is the kind of programmer who likes to take programs written by other people and see how they can be made to work together, or how they can be expanded and improved. They are firm believers in the adage that there is no use in reinventing the wheel, and tend to stick to improving on the work of others. They often know a lot about a lot of different programs and file formats, though they may not have written any complete programs themselves. These kinds of programmers are especially good to have on your team in modern game design where so many powerful and robust solutions to common problems have already been written. The problem with tinkerers is that they less likely to implement a truly original idea themselves, preferring to restrict themselves to existing technologies. A lot of modders fall into this category. If you are a tinkerer yourself, you may want to gather a team of like-minded enthusiasts and get into game design via the modding game design path.

  le bricoleur

gamme
humeur

Every programmer falls somewhere on this spectrum from techie to tinkerer to visionary, and their position on this spectrum probably changes every day depending on their mood and their level of sleep deprivation. The important thing to understand is that these are all valid and necessary approaches to game design. The more fully that you can embody the positive aspects of each of these stereotypes and curtail the negative, the more effective you will be as a game programmer.

privation de sommeil

réduire

Rule Two: Know your objective

The next thing you need to learn is that **there is no right way to design a game**. Every game makes its own demands and defines its own limits.

The important thing you need to keep in mind when you are designing your game, and throughout the game's entire development cycle, is that **you need to design around the core features that make the game fun to play**. If fun gameplay is not your first priority, then you might want to consider a career in an alternate field, like 3D imaging or artificial intelligence. The point is, you have to **be clear about your real goals and intentions**. Creating video games isn't the only way to contribute to this field. If your sole motivation in designing a game is to demonstrate some new technical feature, cut the gameplay elements and focus on the feature instead.

Here are a couple of different paths to game design that you may want to consider before beginning your own project.

Programming paths

- **Climbing the mountain:** this is the 'starting from scratch' approach to game design, and the one most likely to appeal to hard core techies. This approach takes a look at the tools that *could* be needed to create a game and sets about creating them. It does not necessarily look at the specific features required for a particular game, so much as the kinds of tools that would be required for games in general. This approach will give you the most control over your final product, but at the expense of a much longer development time. These days, a powerful, robust 3D game engine usually cannot be created by a single person, so be prepared to shop around for a lot of talented teammates.

method descendants

- **Game first, technique second:** this is the end-product, or top-down approach to game design, and the one most likely to appeal to visionaries. This approach takes a look at the specific gameplay features that are needed to make a fun game and designs the game engine around these features. This path is indifferent to the actual tools that are employed in creating the final product and will use off the shelf products or custom, in-house solutions depending on time and budget constraints. This approach will give you the best compromise between control and fidelity to your vision.

- **Standing on the shoulders of giants:** this is the 'no use reinventing the wheel' approach to game design, and the one most likely to appeal to tinkerers. This approach takes a look at existing game technologies and tries to find the best fit to the original concept with the minimal amount of development time and effort. This approach gives you the least amount of control over your final product, but gives you the fastest development time and the best features with the least amount of effort. This is probably the easiest approach to take for a new game programmer (though the learning curve is still quite steep).

Quite very rapid

Like the programmer stereotypes, each of these paths to game design has its own strengths and weaknesses and you will probably find yourself preferring different paths at different stages in the design phases of your game development cycle. Remember, there is no one right way to make a game, but the path you ultimately end up choosing will have a profound impact on the difficulty and length of the development cycle. Here is where you need to take another good hard look at yourself and decide what you are willing to do to bring your vision to life.

Tips on designing your first game

- **Caution!** Yes, caution is definitely called for when designing a game. Avoid newbie mistakes! Follow this advice:

- **Start small.** Most newbie game programmers dream of making the next hit game. If you're just starting out, this is an impossible goal! It will only lead to frustration and defeat. You must start small; the only way to *learn* game programming is to *be* a game programmer. Try making a simple Tetris-style game, a Breakout clone, or something of that nature. Perhaps pick your favorite and really flesh it out with all the bells and whistles and fantastic features you can think of. You'll run into enough trouble trying to design these simple games if you're just starting out. If you have a little more experience.

- **Plan.** If you have nothing more than some vague notion of making a "really cool RTS game" you are likely going to run into trouble about halfway through your project, when you realize that your initial assumptions were incorrect. Try to be as thorough as possible in the designing phase; it'll save you countless hours in the end. Create a design document for your game, and include a structured version of all of your gameplay and technical design notes. Think through your game from beginning to end, and document every gameplay feature, and how you plan to implement it technically. Having said that, I'll now suggest that no design document is perfect. Something unanticipated will always come up, no matter how thorough you've been. Just do your best!

- **Wait.** I've always found that my ideas seem brilliant to me when I first come up with them, but a few weeks down the road they don't always seem quite so interesting. Be sure that you've given yourself ample time to mull an idea over, before committing yourself to it. You don't want to get bored with your game after investing hours in its development!

- **Get feedback.** Find a group of gamers and pitch your idea to them. If they're not excited and interested by your idea when you're enthusiastically describing it to them, they'll surely not be excited and interested by the end result. Get feedback, and re-work your idea.

- **Know your options.** If you are the type of person that would like to see results quickly and are interested in avoiding as many complicated and frustrating scenarios as possible (they will always be there), it could be ideal for you to take up programming by working with a pre-made game engine, as you would likely get faster results that way than you would with a traditional programming language/gaming library combination.

Choosing a programming language

If this is your first time programming, then the first thing you need to do is take a look at the programming languages and programming environments you will be using. The specific language that you use will, of course, depend to some degree on your background, and on the needs of the game.

There are a couple of different ways you can go about choosing a language:

- You can begin by choosing which language you wish to program in and then find a suitable development environment for that language.
- You can begin by choosing a development environment and work with the language options your chosen development environment gives you.

The reason why your choice of a language may depend on the development environment is that not all languages support equally robust environments on all systems. If you are a new programmer, the last thing you want to worry about is setting up a complex command line build environment! Even though you may think that some obscure little language would be cool to learn, do yourself a favor and start by using a popular and well-supported environment. Unless, of course, you're the hard core techie type who finds that kind of trouble-shooting appealing. attract

If you are already proficient in a particular programming language, it is likely best to begin there since game programming is difficult enough on its own; learning a whole new language will only compound the difficulty.

If you are not familiar with any particular programming language, you'll need to decide which one you'd like to learn, but you are also in a unique and fortuitous scenario; you can pick the programming environment and language that is best for you.

When researching languages, you should consider:

- How easy is the language to learn?
- Is the language suitable for making games? (Is it fast and powerful enough?)
- Does the language work with suitable game development libraries or APIs?
- What platforms (Operating systems/processor architectures) will the language run on?
- Will I be able to find support when I run in to problems? Is there adequate documentation available?

Choosing a library/API

Once you've decided on a language, you'll need to choose a game development library or API. Game libraries may provide you with functions for displaying graphics, playing sounds, getting input, and more. See the Wiki's Libraries for a list of libraries from which you can choose. This is another huge decision. Consider the following:

- Is the library compatible with my language?
- Is the library suited to my skill level?
- Will the library work with the kind of games I plan on making?
- Is adequate documentation available so that I can learn the library?
- Do I understand and agree to the library's license?

Educating yourself

Now that you've got a language and library in mind, it's time to read tutorials! This is the most important step in the whole process. Immerse yourself. Read everything you can. If you don't understand something, ask for help. Above all, **do not give up!**

The languages page will lead you to a number of game programming tutorials for your language of choice, and your library should have documentation for you to read. When you're ready to try writing some code of your own, you'll need an IDE, or "Integrated Development Environment," which is a program that supplies, among other advanced features, a place to write code and a way to compile it. With many languages, a full-blown IDE is not strictly necessary, but it still might be easier for a beginner to use one.

Whatever you do, don't get overwhelmed, or frustrated. Scope out some of this information, then roll up your sleeves and try some of it out. There's certainly something here that will fit your needs and your personality.

Creating game media

For many game developers, this is the hardest part! After perusing the content tools page and deciding on a set of tools, it's time to get down to the business of creating your game's media. If you aren't an artist, you may need to search around for partners who are skilled in this area. For beginner games, there isn't anything at all wrong with so-called "programmer art," however. You can also try some of the freely available art resources that can be found on the Internet. (See the Game Content Resources page.)

Coding tips

- **Code flexibly:** changes happen, no matter how well you plan. If your game's structure is too rigid, you may find that a single oversight in the design phase could necessitate a re-write of some of your game's engine.

- **Prototype:** try to get your game to a quick-and-dirty playable state as soon as possible. The graphics need not be complete, and the bells-and-whistles need not be implemented, but it is important to get an early idea of how your game will play. It's possible that you'll try it and say, "Wow, this is no fun at all". Thankfully, if you've made this discovery early enough, you'll be able to re-work the gameplay and try again without much trouble. Seeing your game in action, even if in a rudimentary state, could also motivate you very effectively.

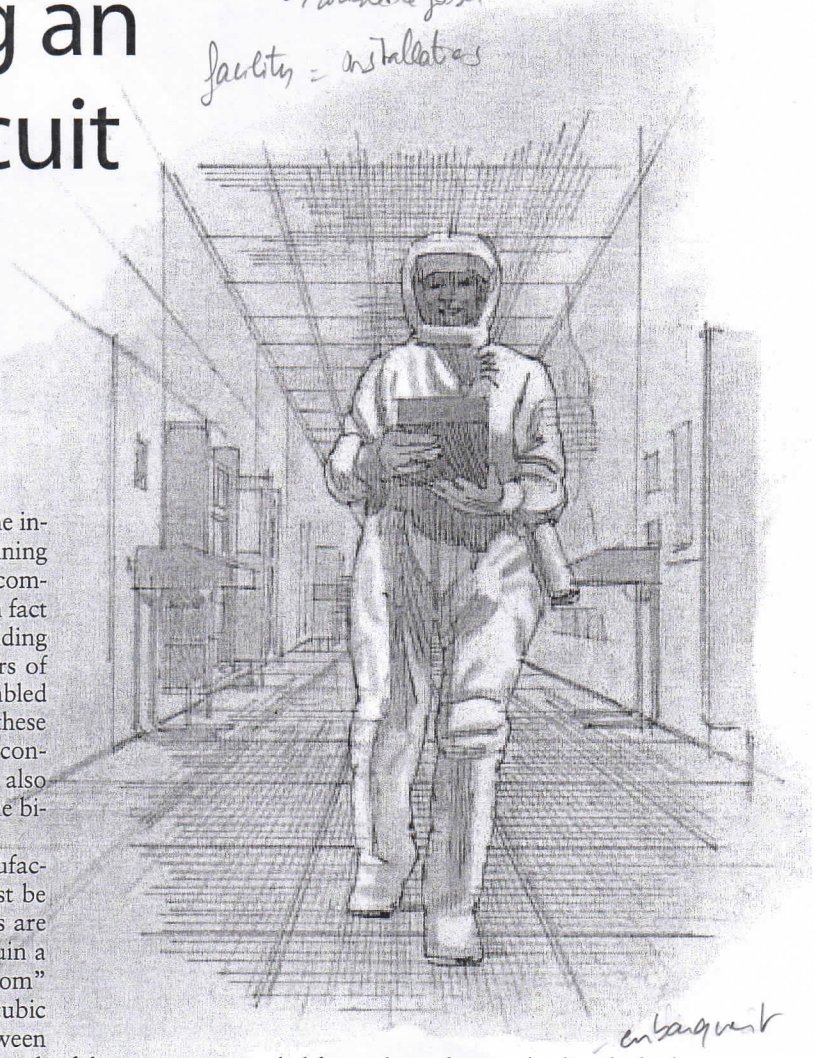
- **Don't optimize early:** spending your time optimizing code before you even know how the game as a whole will perform is pointless. Code flexibly, yes, such that future optimization will be possible, but early optimization can be a big time waster. You won't know where your bottlenecks will be until the game approaches its final state. If you spend days honing your blitting routines, only to find out later that your pathfinding routines are the bottleneck, you've wasted your time! (in the words of Donald Knuth, "Premature optimization is the root of all evil".)

- **Try not to throw away well-tested code:** the principle of code reuse is very important for large projects. You may think throwing away all of your code and starting from scratch is a good idea, but it may not be. Think of the weeks, months, and even years you have spent writing and testing that code.

Tiny silicon chips make modern digital technology possible. Here's how the chips themselves are made

From Sand to Silicon: Manufacturing an Integrated Circuit

by Craig R. Barrett



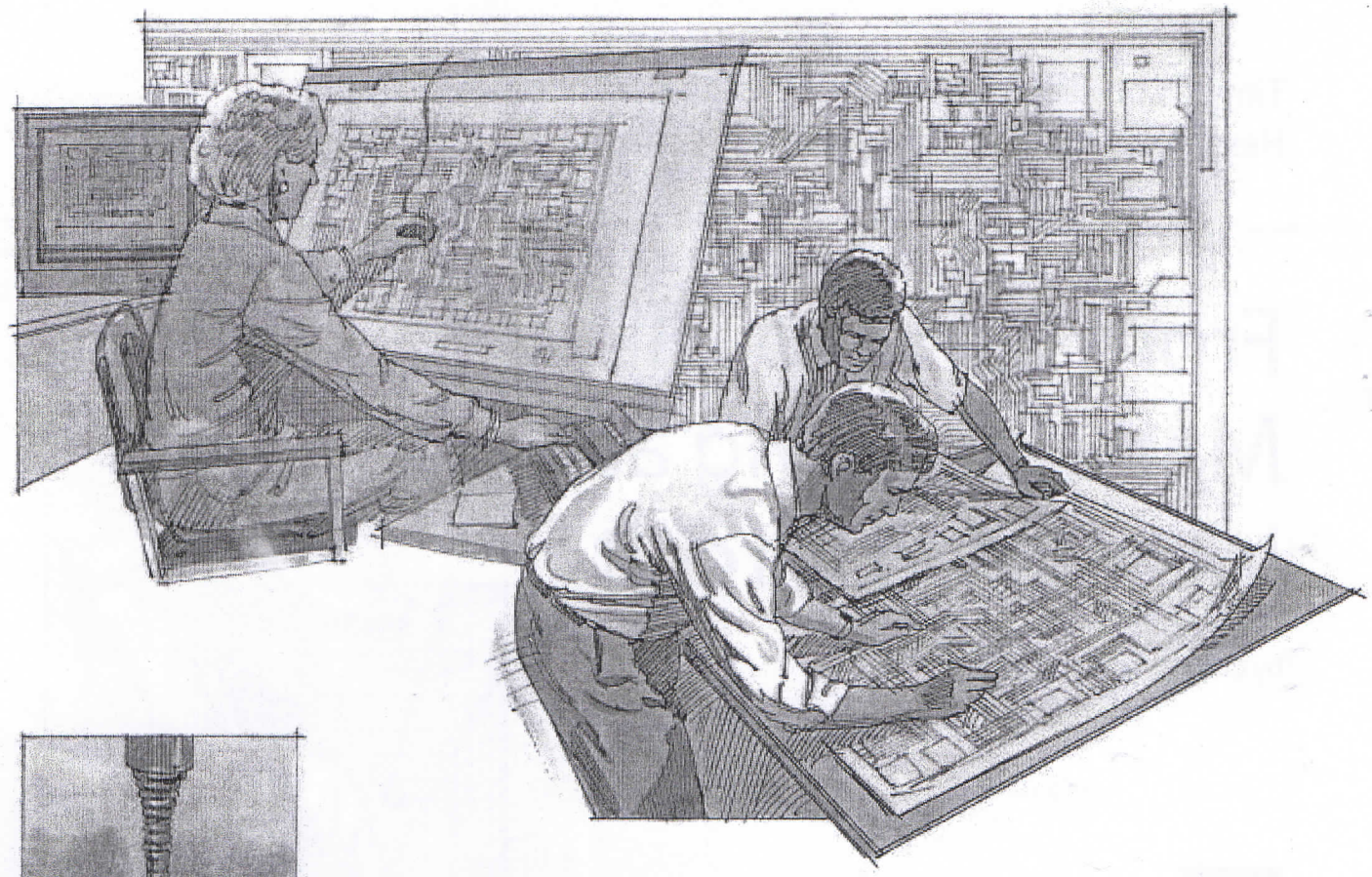
The fundamental device of the digital world is the integrated circuit, a small square of silicon containing millions of transistors. It is probably the most complex of man-made products. Although it looks flat, it is in fact a three-dimensional structure made by painstakingly building up on the silicon base several microscopically thin layers of materials that both insulate and conduct electricity. Assembled according to a pattern carefully worked out in advance, these layers form the transistors, which function as switches controlling the flow of electricity through the circuit, which is also known as a chip. "On" and "off" switches manipulate the binary code that is at the core of what a computer does.

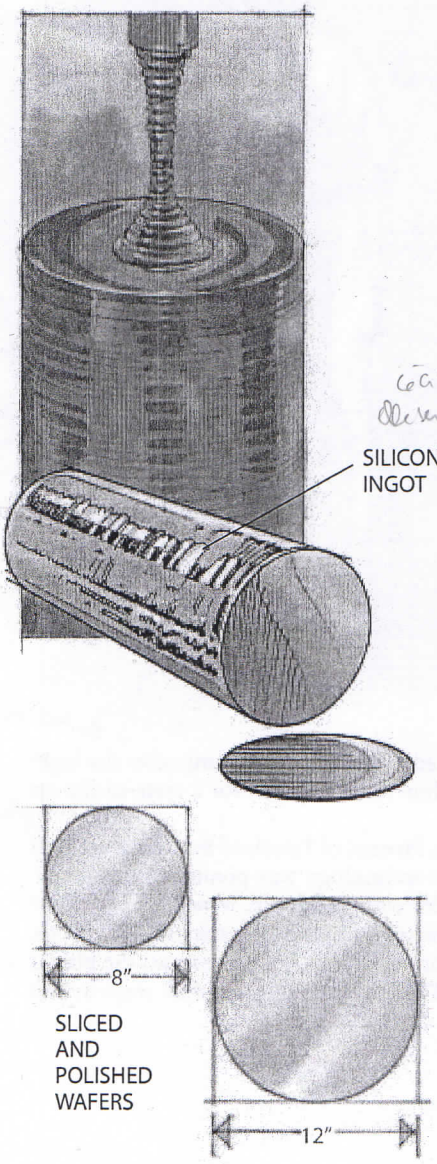
Building a chip typically requires several hundred manufacturing steps that take weeks to complete. Each step must be executed perfectly if the chip is to work. The conditions are demanding. For example, because a speck of dust can ruin a chip, the manufacturing has to be done in a "clean room" containing less than one submicron particle of dust per cubic foot of air (in contrast, the average living room has between 100,000 and one million particles per cubic foot of air). Much of the equipment needed for making chips embodies the highest of high technology, with the result that chip factories—which cost between $1 billion and $2 billion for a state-of-the-art facility—are among the costliest of manufacturing plants.

A basic technology of chipmaking is the "planar" process devised in 1957 by Jean Hoerni of Fairchild Semiconductor. It provided a means of creating a layered structure on the silicon base of a chip. This technology was pivotal in Robert N. Noyce's development of the integrated circuit in 1958. (Noyce later became co-founder with Gordon E. Moore of Intel Corporation, the company that invented the microprocessor and has become the world's leading supplier of semiconductor chips. An article about Moore appears on page 62.) Bridging the gap between the transistor and the integrated circuit, the planar technology opened the way to the manufacturing process that now produces chips. The hundreds of individual steps in that process can be grouped into a few basic operations.

CRAIG R. BARRETT is president and chief operating officer of Intel Corporation.

Copyright 1997 Scientific American, Inc.





Chip Design

The first operation is the design of the chip. When tens of millions of transistors are to be built on a square of silicon about the size of a child's fingernail, the placing and interconnections of the transistors must be meticulously worked out. Each transistor must be designed for its intended function, and groups of transistors are combined to create circuit elements such as inverters, adders and decoders. The designer must also take into account the intended purpose of the chip. A processor chip carries out instructions in a computer, and a memory chip stores data. The two types of chips differ somewhat in structure. Because of the complexity of today's chips, the design work is done by computer, although engineers often print out an enlarged diagram of a chip's structure to examine it in detail (*above*).

The Silicon Crystal

The base material for building an integrated circuit is a silicon crystal. Silicon, the most abundant element on the earth except for oxygen, is the principal ingredient of beach sand. Silicon is a natural semiconductor, which means that it can be altered to be either an insulator or a conductor. Insulators, such as glass, block the passage of electricity; conductors, such as copper, let electricity pass through. To make a silicon crystal, raw silicon obtained from quartz rock is treated with chemicals that remove contaminants until what remains is almost 100 percent silicon. This purified silicon is melted and then formed into cylindrical single crystals called ingots (*left, top*). The ingots are sliced into wafers about 0.725 millimeter (0.03 inch) thick. In a step called planarization they are polished with a slurry until they have a flawless, mirror-smooth surface. At present, most of the wafers are 200 millimeters (eight inches) in diameter, but the industry is moving toward achieving a standard diameter of 300 millimeters (12 inches) by 1999 (*left, bottom*). Because a single wafer yields hundreds of chips, bigger wafers mean that more chips can be made at one time, holding down the cost per chip.

DRAWINGS BY BARRY ROSS, AFTER INTEL AND SEMICONDUCTOR EQUIPMENT AND MATERIALS INTERNATIONAL

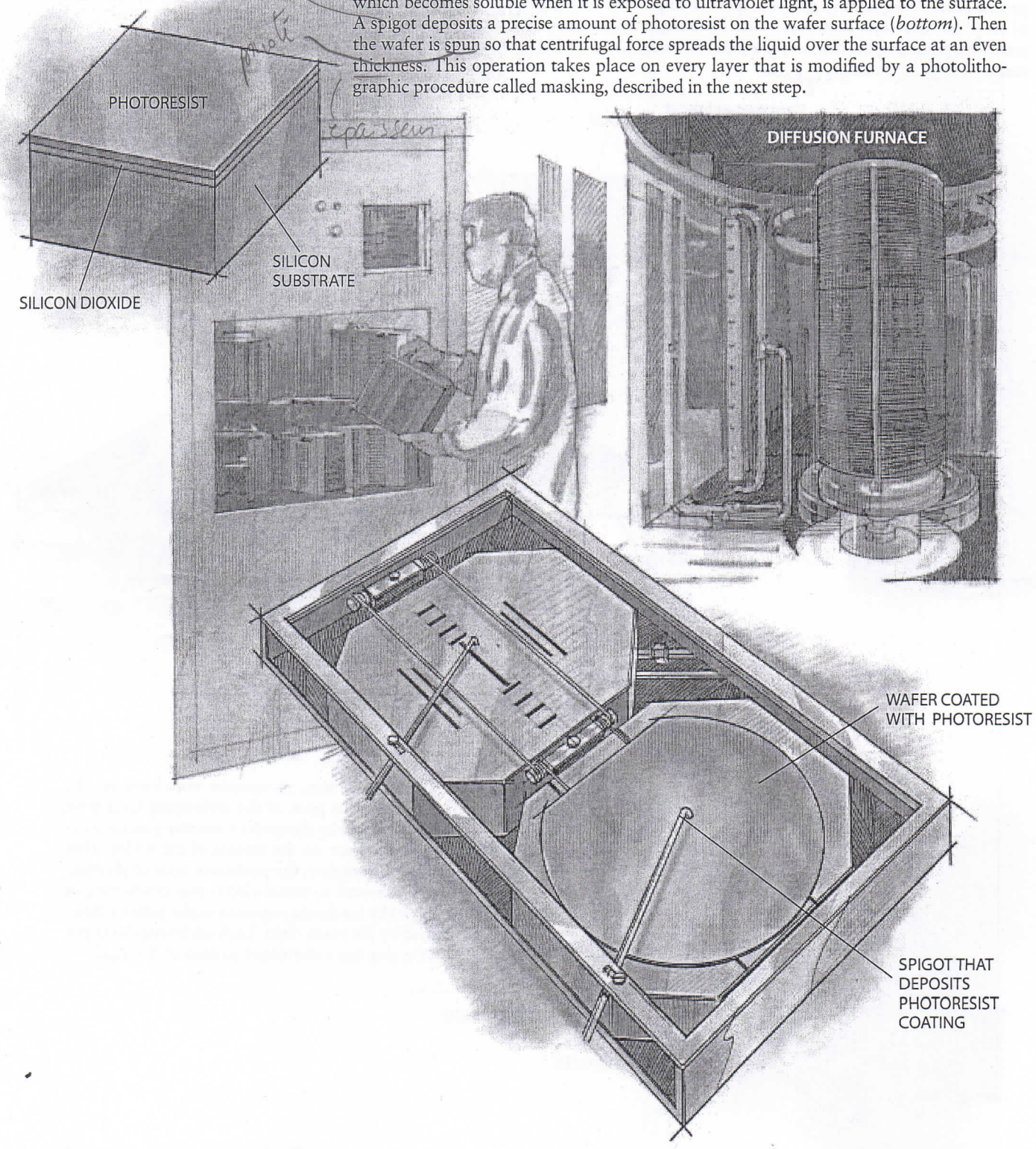
Copyright 1997 Scientific American, Inc.

The First Layers

With the wafer prepared, the process of building the chip's circuitry begins. Making the transistors and their interconnections entails several different basic steps that are repeated many times. The most complex chips made today consist of 20 or more layers and may require several hundred separate processing steps to build them up one by one.

The first layer is silicon dioxide, which does not conduct electricity and therefore serves as an insulator. It is created by putting the wafers into a diffusion furnace (*top right*)—essentially an oven at high temperature where a thin layer of oxide is grown on the wafer surface.

Removed from the furnace, the wafer is now ready for its first patterning, or photolithographic, step. A coating of a fairly viscous polymeric liquid called photoresist, which becomes soluble when it is exposed to ultraviolet light, is applied to the surface. A spigot deposits a precise amount of photoresist on the wafer surface (*bottom*). Then the wafer is spun so that centrifugal force spreads the liquid over the surface at an even thickness. This operation takes place on every layer that is modified by a photolithographic procedure called masking, described in the next step.

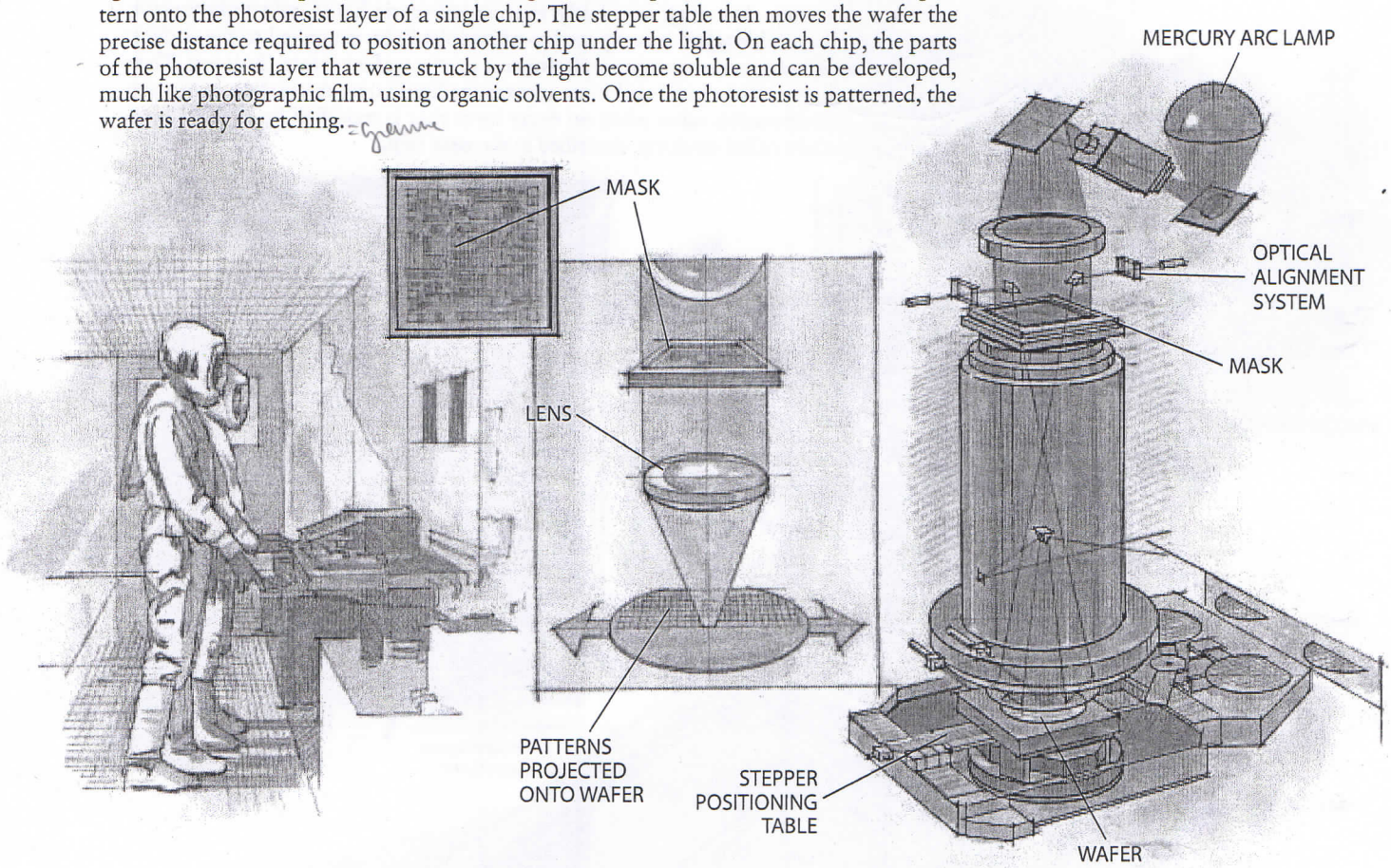


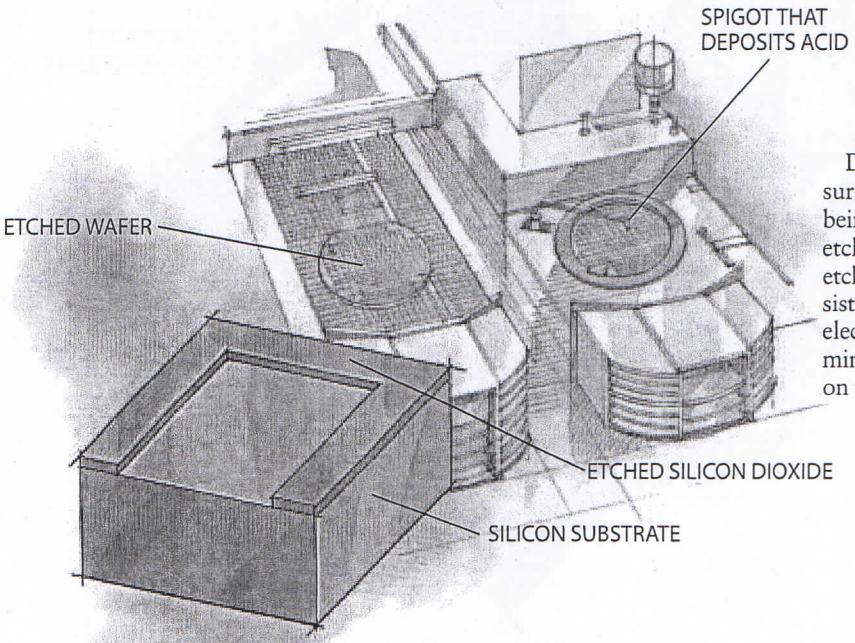
Copyright 1997 Scientific American, Inc.

Masking

A mask is the device through which ultraviolet light shines to define the circuit pattern on each layer of a chip. Because the pattern is intricate and must be positioned precisely on the chip, the arrangement of opaque and transparent spaces on a mask must be done carefully during a chip's design stage.

The mask image is transferred to the wafer using a computer-controlled machine known as a stepper. It has a sophisticated lens system (*below*) to reduce the pattern on the mask to the microscopic dimensions of the chip's circuitry, requiring resolution as small as 0.25 micron. The wafer is held in place on a positioning table below the lens system. Ultraviolet light from an arc lamp or a laser shines through the clear spaces of the mask's intricate pattern onto the photoresist layer of a single chip. The stepper table then moves the wafer the precise distance required to position another chip under the light. On each chip, the parts of the photoresist layer that were struck by the light become soluble and can be developed, much like photographic film, using organic solvents. Once the photoresist is patterned, the wafer is ready for etching.





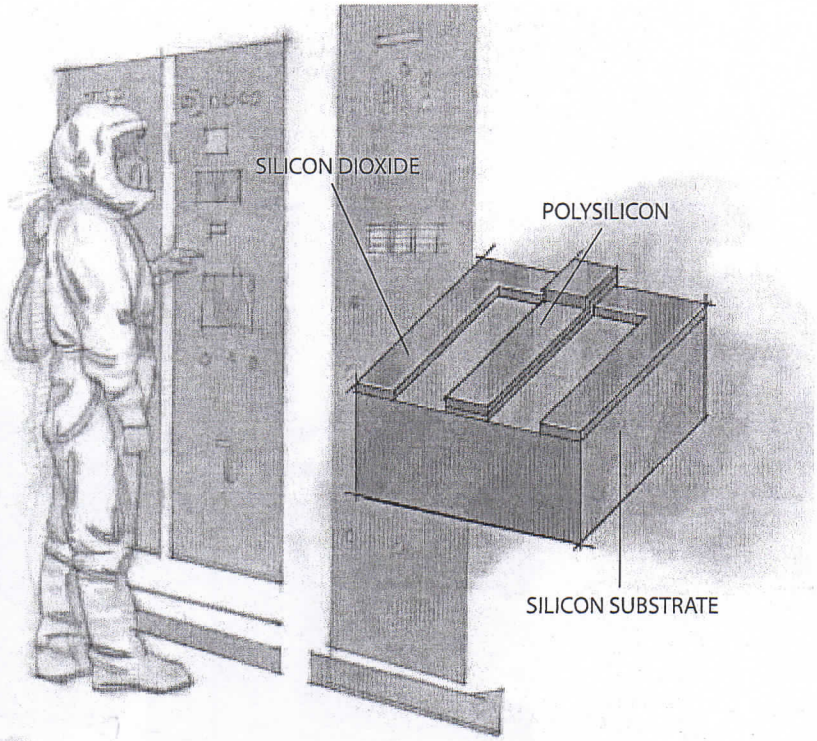
Etching

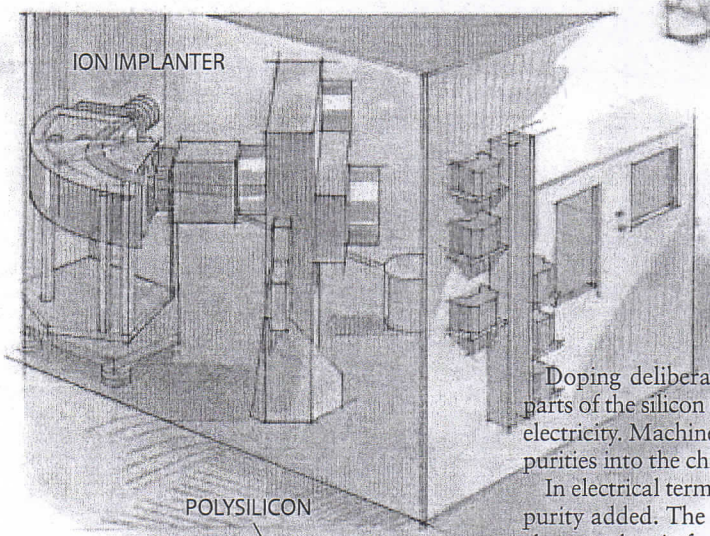
During this step, photoresist remaining on the surface protects parts of the underlying layer from being removed by the acids or reactive gases used to etch the pattern on the surface of the wafer. After etching is complete, the protective layer of photoresist is removed to reveal electrically conducting or electrically insulating segments in the pattern determined by the mask (*left*). Each additional layer put on the chip has a distinctive pattern of this kind.

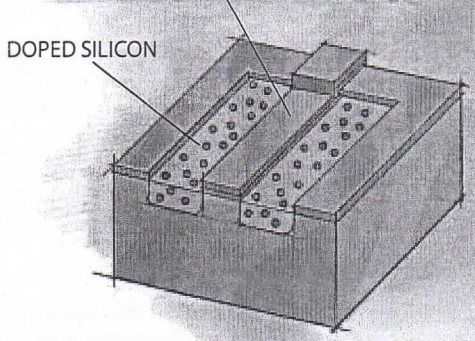
Copyright 1997 Scientific American, Inc.

Adding Layers

Further masking and etching steps deposit patterns of additional materials on the chip. These materials include polysilicon as well as various oxides and metal conductors such as aluminum and tungsten. To prevent the formation of undesired compounds during subsequent steps, other materials known as diffusion barriers can also be added. On each layer of material, masking and etching create a unique pattern of conducting and nonconducting areas (*right*). Together these patterns aligned on top of one another form the chip's circuitry in a three-dimensional structure. But the circuitry needs fine-tuning to work properly. The tuning is provided by doping.







Doping

Doping deliberately adds chemical impurities, such as boron or arsenic, to parts of the silicon wafer to alter the way the silicon in each doped area conducts electricity. Machines called ion implanters (*left*) are often used to inject these impurities into the chip.

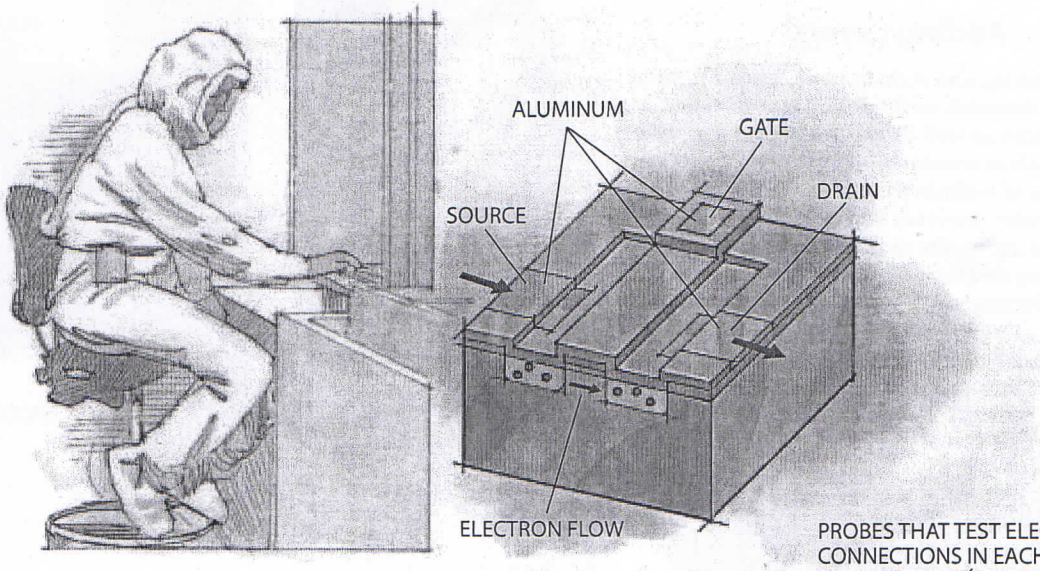
In electrical terms, silicon can be either *n*-type or *p*-type, depending on the impurity added. The atoms in the doping material in *n*-type silicon have an extra electron that is free to move. Some of the doping atoms in *p*-type silicon are short an electron and so constitute what is called a hole. Where the two types adjoin, the extra electrons can flow from the *n*-type to the *p*-type to fill the holes.

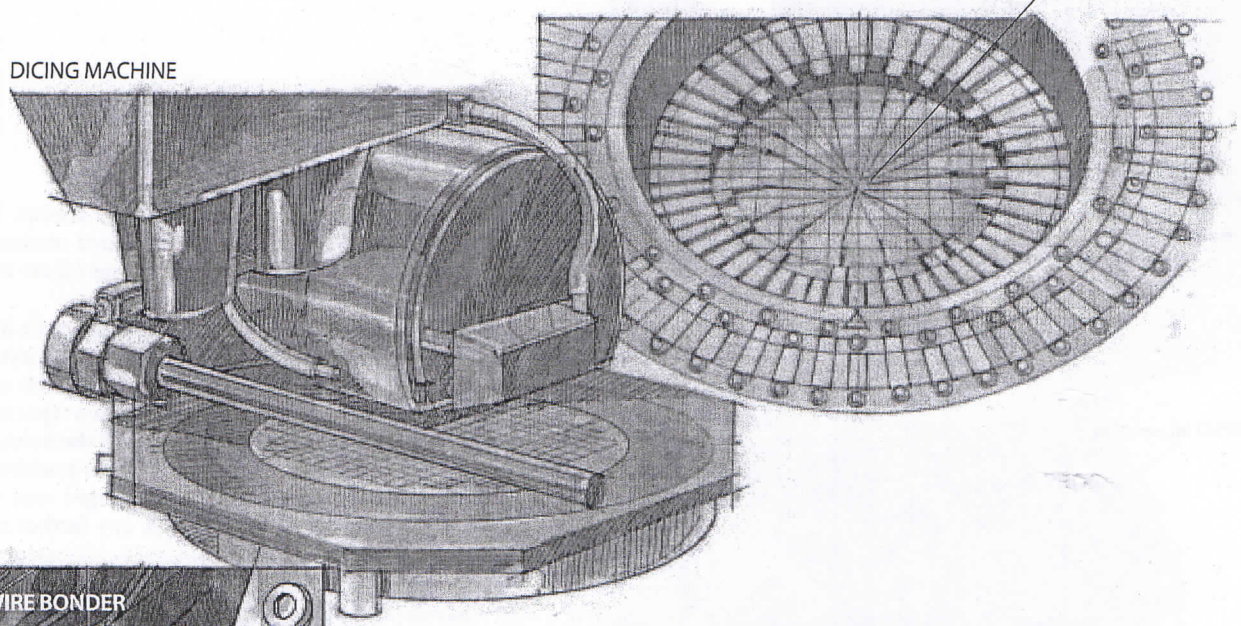
This flow of electrons does not continue indefinitely. Eventually the positively charged ions left behind on the *n*-type side and the negatively charged ions on the *p*-type side together create an electrical force that prevents any further net flow of electrons from the *n*-type to the *p*-type region.

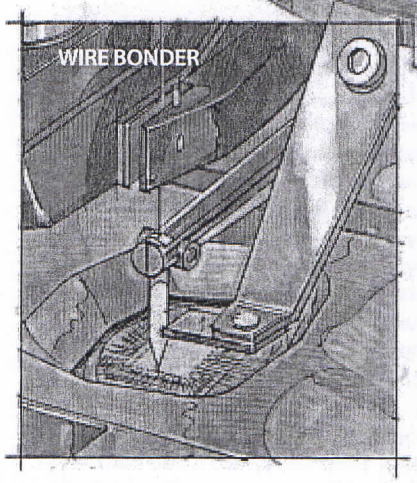
The material at the base of the chip is *p*-type silicon. One of the etching steps in the manufacture of a chip removes parts of the polysilicon and silicon dioxide layers put on the pure silicon base earlier, thus laying bare two strips of *p*-type silicon. Separating them is a strip that still bears its layer of conducting polysilicon; it is the transistor's "gate." The doping material now applied to the two strips of *p*-type silicon transforms them into *n*-type silicon. A positive charge applied to the gate attracts electrons below the gate in the transistor's silicon base. These electrons create a channel between one *n*-type strip (the source) and the other (the drain). If a positive voltage is applied to the drain, current will flow from source to drain. In this mode, the transistor is "on." A negative charge at the gate depletes the channel of electrons, thereby preventing the flow of current between source and drain. Now the transistor is "off." It is by means of switching on and off that a transistor represents the arrays of 1 and 0 that constitute the binary code, the language of computers.

Done many times in many layers, these operations provide the chip with its multitude of transistors. But just as provision must be made to run electrical wires and plumbing pipes between floors of a building, provision must be made in chips for interconnecting the transistors so they form an integrated circuit.

Copyright 1997 Scientific American, Inc.







Interconnections

This final step begins with further masking and etching operations that open a thin layer of electrical contacts between layers of the chip. Then aluminum is deposited and patterned using photolithography to create a form of wiring that links all the chip's transistors (*top*). Aluminum is chosen for this application because it makes good electrical contact with silicon and also bonds well to silicon dioxide.

This step completes the processing of the wafer. Now the individual chips are tested to ensure that all their electrical connections work using tiny electrical probes (*above right*). Next, a machine called a dicer cuts up the wafer into individual chips (*above left*), and the good chips are separated from the bad. The good chips—usually most of the wafer's crop—are mounted onto packaging units with metal leads. Wire bonders (*left*) then attach these metal leads to the chips. The electrical contacts between the chip's surface and the leads are made with tiny gold or aluminum wires about 0.025 millimeter (0.001 inch) in diameter. Once the packaging process is complete, the finished chips are sent to do their digital work. SA

Copyright 1997 Scientific American, Inc.

Android Ads Could Attack, Study Warns

Ad libraries, bundled with free apps, could sniff data and even install malicious software.

- Monday, March 19, 2012
- By David Talbot

Trap Door: This permission screen—typical for a free Android app—also grants the same permissions to the app's ad libraries.
Technology Review

To guard against malicious behavior, apps that run on Android smart phones must ask politely for permission to do things like access your personal information, track your location, or transmit data via the Internet. But once they have approval, these apps can share the permissions with the advertising libraries they use to serve up ads, creating serious potential privacy and security vulnerabilities, researchers have found.

Combing through 100,000 apps selected randomly from Google Play (the erstwhile Android Market), a team of researchers at North Carolina State University found that 48,139 of the ad libraries used by these apps tracked the user's GPS location; 18,575 tracked the identity of the phone (its IMEI number); 4,190 let advertisers track the user via GPS; and 4,047 accessed the device's phone number.

Dozens of such "ad libraries" exist, generating onscreen ads with the associated apps. When you click on an ad, the app maker gets a fee. One of these libraries, called energysource, uses an insecure method of loading code from the Internet, says Xuxian Jiang, the North Carolina professor who led the study, released as a paper to be presented at a conference in Tucson next month. Though the researchers did not detect malicious behavior from the app, they did say it poses a security threat simply by allowing code to be downloaded and run.

Of the 100,000 apps, 297 contained ad code that allowed the phone to run code downloaded from the Internet, providing a potential path for malicious software to get inside the device. "If your app has permission to access personal information, the ad library also has permission to access your information," Jiang says.

The North Carolina research is only the latest evidence of gaping security and privacy holes in smart phones. In April of last year, iPhones and Android devices were found to track users' locations automatically. Then, in December, these and other smart-phone devices were found to carry diagnostic software that also tracks a wide range of user information. More recently,

it was discovered that both iPhones and Android devices share users' address books and other information with apps. And instances of mobile malware have been rising.

points de recherche des défauts — derrière

The new findings point to a flaw in the business model behind apps, Jiang says. Developers rely on revenue from ad libraries to support free apps, but they have no control over what those libraries do. "The current model of embedding ad libraries in mobile apps for monetization purposes poses security and privacy risks. These ad libraries will essentially have the same set of permissions granted to the apps that enclose them. And certain ad libraries may abuse them for other unwanted purposes."

Mobile device makers should provide ways to isolate the two, Jiang says, so that the ads run separately from the host apps—and require separate explicit permissions. "There are fundamental concerns in the way mobile apps are being monetized," he adds.

préoccupations

Adding insult to injury, other research has recently found that ads associated with free Android apps are also battery-drainers. Abhinav Pathak, a computer scientist at Purdue University, and colleagues at Microsoft Research found that as much as three-quarters of the juice used by such apps is spent to serve ads and transmit user data back to advertisers. 3/4

flaw = défaut / imperfection / faute / inconvénient / faille / tripoter / carr

business = entreprise / affaires support = soutenir

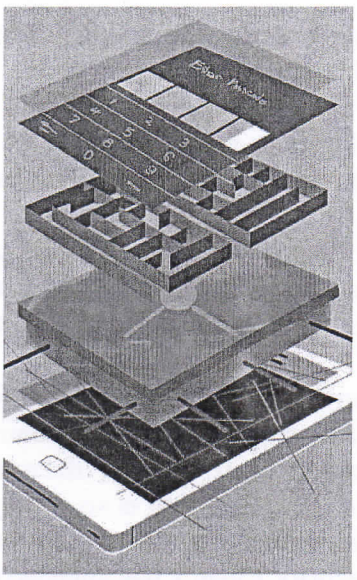
rely = compter

ad libraries = biblio publi embedding = intégrant

purpose = but / intention

enclose = joindre / inclure battery-drainers = batterie épuisoirs

The iPhone Has Passed a Key Security Threshold



before 2008
hack early

Does society really want extremely private mobile devices if they make life easier for criminals? Apple's newly toughened standards sharpen the focus on that question.

Simson L. Garfinkel

Monday, August 13, 2012

Less than a month after Apple first shipped the iPhone in June 2007, a group called Independent Security Evaluators documented deep security design flaws in the device. Apple's most embarrassing flub: every iPhone application that Apple had written ran with so-called root privileges, giving each one complete control over the entire phone. Hackers found bugs in those apps that could be used to take over the phone from the inside. Apple didn't fix the design flaw until January 2008.

default

But after that rocky launch, Apple invested heavily in iPhone security. It's still possible for a hacker to take over a phone, but it's increasingly difficult, largely because each app runs in its own isolated "sandbox." The phone even verifies its operating system when it boots. Today the Apple iPhone 4S and iPad 3 are trustworthy mobile computing systems that can be used for mobile payments, e-commerce, and the delivery of high-quality paid programming—all of which bring Apple significant revenue in the form of commissions.

digital
confer
field

In fact, in its efforts to make its devices more secure, Apple has crossed a significant threshold. Seuil
Technologies the company has adopted protect Apple customers' content so well that in many situations it's impossible for law enforcement to perform forensic examinations of devices seized from criminals. Most significant is the increasing use of encryption, which is beginning to cause problems for law enforcement agencies when they encounter systems with encrypted drives.

"I can tell you from the Department of Justice perspective, if that drive is encrypted, you're done," Ovie Carroll, director of the cyber-crime lab at the Computer Crime and Intellectual Property Section in the Department of Justice, said during his keynote address at the DFRWS computer forensics conference in Washington, D.C., last Monday. "When conducting criminal investigations, if you pull the power on a drive that is whole-disk encrypted you have lost any chance of recovering that data."

la commercialisation to mean

Mass-market cryptography hasn't been thought of as a potential threat to law enforcement since the "crypto wars" of the 1990s. Back then there was a very public battle against U.S. laws and regulations that limited the use and export of cryptographic technology. On one side, civil liberties groups and business interests said that the public needed strong cryptography to protect privacy and financial transactions. On the other side, law enforcement organizations warned that the same technology would empower drug dealers, kidnappers, money launderers, and terrorists.

blanchisseurs

Digital forensics Research Conference

Law enforcement lost the crypto wars: today there is essentially no restriction on mass-market cryptography. Fortunately, few of the predicted horribles came to pass. One reason is that the encryption systems developed and sold to consumers over the past 20 years have had an Achilles' heel: there has been no good way to let users securely manage encryption keys. Cryptography, for all its power, provides no security unless the keys used to lock the data remain secret.

Enter the iPhone. Apple's security architecture is so sturdy, and so tightly woven into its hardware and software, that it is both easy for consumers to use encryption on their phones and very difficult for someone else to steal the encrypted information.

At the heart of Apple's security architecture is the Advanced Encryption Standard algorithm (AES), a data-scrambling system published in 1998 and adopted as a U.S. government standard in 2001. After more than a decade of exhaustive analysis, AES is widely regarded as unbreakable. The algorithm is so strong that no computer imaginable for the foreseeable future—even a quantum computer—would be able to crack a truly random 256-bit AES key. The National Security Agency has approved AES-256 for storing top-secret data.

Apple did not respond to requests for comment on this story. But the AES key in each iPad or iPhone "is unique to each device and is not recorded by Apple or any of its suppliers," the company said in a security-related white paper. "Burning these keys into the silicon prevents them from being tampered with or bypassed, and guarantees that they can be access only by the AES engine."

What this means in practice is that when iOS devices are turned off, the copy of the encryption key in the computer's accessible memory is erased. That is why an investigator who gets a suspect's phone would have to try all possible keys—the task deemed impossible by the NSA.

The iPhone and iPad do keep a copy of the encryption key deeper in flash memory—otherwise there would be no way for the device to recover data when it was turned back on. But that encryption key is itself protected by the user's "PIN lock," a code that must be entered before the device can be used.

The iPhone always supported a PIN lock, but the PIN wasn't a deterrent to a serious attacker until the iPhone 3GS. Because those early phones didn't use their hardware to perform encryption, a skilled investigator could hack into the phone, dump its flash memory, and directly access the phone's address book, e-mail messages, and other information. But now, with Apple's more sophisticated approach to encryption, investigators who want to examine data on a phone have to try every possible PIN. Examiners perform these so-called brute-force attacks with special software, because the iPhone can be programmed to wipe itself if the wrong PIN is provided more than 10 times in a row. This software must be run on the iPhone itself, limiting the guessing speed to 80 milliseconds per PIN. Trying all four-digit PINs therefore requires no more than 800 seconds, a little more than 13 minutes. However, if the user chooses a six-digit PIN, the maximum time required would be 22 hours; a nine-digit PIN would require 2.5 years, and a 10-digit pin would take 25 years. That's good enough for most corporate secrets—and probably good enough for most criminals as well.

"There are a lot of issues when it comes to extracting data from iOS devices," says Amber Schroader, CEO of Paraben, a supplier of forensic software, hardware, and services for cell phones. "We have had many civil cases we have not been able to process ... for discovery because of encryption blocking us."

Another iPhone innovation has to do with how and where data gets encrypted. Years ago encryption wasn't used very often because it was difficult to implement and computationally expensive—it took a lot of resources. Not so with the iPhone. Apple designed iOS devices so that the hardware that encrypts data is in the path the data travels when it moves from flash storage to the iPhone's main memory. This means that data can be automatically decrypted when read from flash into memory and reëncrypted when saved from memory back to flash. On the iPhone, encryption is essentially free.

That makes it possible to offer services like Foxygram, an iPhone app that allows users to share encrypted data in the knowledge that it cannot be intercepted and provided to law enforcement. Markus Kangas, cofounder of the app's creator, FoxyFone, says the goal is to "provide easy-to-use secure messaging for everyone and at the same time protect user privacy." He adds: "We are not there to police people."

Google's Android operating system also supports encrypted storage, but only for some of the data on the phone. More important, there is no key burned into the hardware, so even complex passwords can be broken by extracting them and using a network of a few hundred computers. BlackBerry phones, on the other hand, also have a strong encryption system that can be based on multiple factors in addition to the user's PIN.

But the BlackBerry system is designed for business customers and is harder to use than Apple's, which is made for the consumer market. Now that hardened, military-grade encryption is tough and easy for consumers to use—assuming the user has set a PIN lock that's both long and hard to guess—the nightmare scenario of the crypto wars may finally have come to pass.

Car-to-Car Communication System to Get a Massive Road Test

A trial involving thousands of cars could pave the way for technology aimed at cutting accidents and traffic jams.

- Wednesday, September 7, 2011
- By Julie Halpert

Fair warning: A car issues a warning to stop as it approaches an upcoming intersection.
Credit: Ford

Technology that would allow cars to talk to each other—to help prevent accidents and improve traffic flow—is about to get a real-world road test following new funding from the U.S. Department of Transportation.

Many high-end cars already come with sensors capable of spotting a vehicle in a driver's blind spot, or warning that the car is drifting out of lane. However, these technologies, which use radar, laser, or video sensors, have a limited view. Car-to-car communications could provide even more sophisticated earlier warnings—for example, when a car several vehicles ahead brakes suddenly.

Last month, the DOT awarded $14.9 million to the University of Michigan's Transportation Research Institute to test the technology, known as vehicle-to-vehicle and vehicle-to-infrastructure communication. The system to be tested relies on dedicated short-range radio communication to allow cars to signal one another and receive messages from traffic equipment.

The DOT estimates that 80 percent of serious crashes could be addressed by this technology. "This is the next major safety advancement, one that's comparable to seat belts, air bags, and electronic stability control," said Scott Belcher, president and CEO of the Intelligent Transportation Society of America, a nonprofit founded to promote advanced car technologies.

The technology will be tested in a variety of situations; it will alert the driver when it is unsafe to pass, and when someone is approaching an intersection at a speed that could cause a collision. Each car will be equipped with a radio that signals its speed and direction of travel, as determined by GPS, to other cars. It will also send this information to suitably equipped traffic equipment.

The University of Michigan is partnering with eight automakers, a number of which began working collaboratively to develop a uniform platform for implementing the technology in 1995. These carmakers will provide 64 cars equipped with the radios, while an additional group of ordinary cars will be fit with devices so they can transmit signals, making up a total of roughly 3,000 vehicles. Drivers will be recruited from among the 20,000 employees of the university's medical center.

Peter Sweatman, director of the Transportation Research Institute, says Ann Arbor is an ideal test bed, since it's a concentrated area with only three central thoroughfares out of the city, making it likely that the equipped cars will regularly encounter each other. The driving portion will run for a year, and data will be collected and may be used by the DOT's National Highway Safety Traffic Administration to decide, by 2013, if the technology has enough benefits to be approved. If approved, the technology would be rolled out over 10 years, Sweatman says.

"We believe this will happen in the near future," says Nady Boules, director of the electric and controls integration research lab at General Motors.

Jim Keller, senior manager and engineer at Honda Research and Development, adds, "We see this technology as having huge potential in the future to affect safety."

The DOT's Research and Innovative Technology Administration, which is overseeing the program, released the following statement on the project: "This technology has the potential to be a game changer for safety. Research from NHTSA found that combined vehicle-to-vehicle and vehicle-to-infrastructure technologies have the potential to address about 80 percent of all unimpaired car crash scenarios."

Joe Stinnett, a research engineer in active safety for Ford, is similarly enthusiastic. He says that, in addition to preventing common accidents, the technology could prevent traffic backups by keeping cars in step with one another. But he says one key area that needs to be addressed is security. "People could hack into the system, sitting on a bridge with their laptop transmitting false information," he warns. So a major challenge will be ensuring that the network is secure and that misbehaviors can be identified, he says.

Europe is on a similar track. In January 2011, the European Commission launched a three-year pan-European field test in seven sites across Europe to ensure the interoperability of the system. The effort includes 40 carmakers as well as suppliers, electronics manufacturers, and research institutes.

As vehicle-to-vehicle communication goes mainstream, it could even pave the way for fully autonomous driving. Google has been testing its own self-driving cars in California. So far those cars have logged 160,000 miles, but they rely on costly sensors. Vehicle-to-vehicle communication could allow for autonomous driving that's far less expensive, Belcher says. He expects that some autonomous driving features could appear in commercial fleets within five years. But he doubts that fully autonomous driving will take hold in the foreseeable future for one key reason: "Americans like to control their own cars," he says.

Chris Gerdes: The future race car -- 150mph, and no driver

Filmed May 2012 • **Posted** Jul 2012 • TEDxStanford

So, how many of you have ever gotten behind the wheel of a car when you really shouldn't have been driving? Maybe you're out on the road for a long day, and you just wanted to get home. You were tired, but you felt you could drive a few more miles. Maybe you thought, I've had less to drink than everybody else, I should be the one to go home. Or maybe your mind was just entirely elsewhere.

Does this sound familiar to you? Now, in those situations, wouldn't it be great if there was a button on your dashboard that you could push, and the car would get you home safely? Now, that's been the promise of the self-driving car, the autonomous vehicle, and it's been the dream since at least 1939, when General Motors showcased this idea at their Futurama booth at the World's Fair.

Now, it's been one of those dreams that's always seemed about 20 years in the future. Now, two weeks ago, that dream took a step forward, when the state of Nevada granted Google's self-driving car the very first license for an autonomous vehicle, clearly establishing that it's legal for them to test it on the roads in Nevada. Now, California's considering similar legislation, and this would make sure that the autonomous car is not one of those things that has to stay in Vegas.

(Laughter)

Now, in my lab at Stanford, we've been working on autonomous cars too, but with a slightly different spin on things. You see, we've been developing robotic race cars, cars that can actually push themselves to the very limits of physical performance.

Now, why would we want to do such a thing? Well, there's two really good reasons for this. First, we believe that before people turn over control to an autonomous car, that autonomous car should be at least as good as the very best human drivers. Now, if you're like me, and the other 70 percent of the population who know that we are above-average drivers, you understand that's a very high bar. There's another reason as well. Just like race car drivers can use all of the friction between the tire and the road, all of the car's capabilities to go as fast as possible, we want to use all of those capabilities to avoid any accident we can.

Now, you may push the car to the limits not because you're driving too fast, but because you've hit an icy patch of road, conditions have changed. In those situations, we want a car that is capable enough to avoid any accident that can physically be avoided.

I must confess, there's kind of a third motivation as well. You see, I have a passion for racing. In the past, I've been a race car owner, a crew chief and a driving coach, although maybe not at the level that you're

currently expecting. One of the things that we've developed in the lab -- we've developed several vehicles -- is what we believe is the world's first autonomously drifting car. It's another one of those categories where maybe there's not a lot of competition.

(Laughter)

But this is P1. It's an entirely student-built electric vehicle, which through using its rear-wheel drive and front-wheel steer-by-wire can drift around corners. It can get sideways like a rally car driver, always able to take the tightest curve, even on slippery, changing surfaces, never spinning out.

We've also worked with Volkswagen Oracle, on Shelley, an autonomous race car that has raced at 150 miles an hour through the Bonneville Salt Flats, gone around Thunderhill Raceway Park in the sun, the wind and the rain, and navigated the 153 turns and 12.4 miles of the Pikes Peak Hill Climb route in Colorado with nobody at the wheel.

(Laughter)

(Applause)

I guess it goes without saying that we've had a lot of fun doing this. But in fact, there's something else that we've developed in the process of developing these autonomous cars. We have developed a tremendous appreciation for the capabilities of human race car drivers. As we've looked at the question of how well do these cars perform, we wanted to compare them to our human counterparts. And we discovered their human counterparts are amazing. Now, we can take a map of a race track, we can take a mathematical model of a car, and with some iteration, we can actually find the fastest way around that track. We line that up with data that we record from a professional driver, and the resemblance is absolutely remarkable.

Yes, there are subtle differences here, but the human race car driver is able to go out and drive an amazingly fast line, without the benefit of an algorithm that compares the trade-off between going as fast as possible in this corner, and shaving a little bit of time off of the straight over here. Not only that, they're able to do it lap after lap after lap. They're able to go out and consistently do this, pushing the car to the limits every single time. It's extraordinary to watch. You put them in a new car, and after a few laps, they've found the fastest line in that car, and they're off to the races.

It really makes you think, we'd love to know what's going on inside their brain. So as researchers, that's what we decided to find out. We decided to instrument not only the car, but also the race car driver, to try to get a glimpse into what was going on in their head as they were doing this. Now, this is Dr. Lene Harbott applying electrodes to the head of John Morton. John Morton is a former Can-Am and IMSA driver, who's also a class champion at Le Mans. Fantastic driver, and very willing to put up with graduate students and this sort of research. She's putting electrodes on his head so that we can monitor the electrical activity in John's brain as he races around the track.

Now, clearly we're not going to put a couple of electrodes on his head and understand exactly what all of his thoughts are on the track. However, neuroscientists have identified certain patterns that let us tease

out some very important aspects of this. For instance, the resting brain tends to generate a lot of alpha waves. In contrast, theta waves are associated with a lot of cognitive activity, like visual processing, things where the driver is thinking quite a bit. Now, we can measure this, and we can look at the relative power between the theta waves and the alpha waves. This gives us a measure of mental workload, how much the driver is actually challenged cognitively at any point along the track.

Now, we wanted to see if we could actually record this on the track, so we headed down south to Laguna Seca. Laguna Seca is a legendary raceway about halfway between Salinas and Monterey. It has a curve there called the Corkscrew. Now, the Corkscrew is a chicane, followed by a quick right-handed turn as the road drops three stories. Now, the strategy for driving this as explained to me was, you aim for the bush in the distance, and as the road falls away, you realize it was actually the top of a tree.

All right, so thanks to the Revs Program at Stanford, we were able to take John there and put him behind the wheel of a 1960 Porsche Abarth Carrera. Life is way too short for boring cars. So, here you see John on the track, he's going up the hill -- Oh! Somebody liked that -- and you can see, actually, his mental workload -- measuring here in the red bar -- you can see his actions as he approaches. Now watch, he has to downshift. And then he has to turn left. Look for the tree, and down.

Not surprisingly, you can see this is a pretty challenging task. You can see his mental workload spike as he goes through this, as you would expect with something that requires this level of complexity. But what's really interesting is to look at areas of the track where his mental workload doesn't increase. I'm going to take you around now to the other side of the track. Turn three. And John's going to go into that corner and the rear end of the car is going to begin to slide out. He's going to have to correct for that with steering. So watch as John does this here. Watch the mental workload, and watch the steering. The car begins to slide out, dramatic maneuver to correct it, and no change whatsoever in the mental workload. Not a challenging task. In fact, entirely reflexive.

Now, our data processing on this is still preliminary, but it really seems that these phenomenal feats that the race car drivers are performing are instinctive. They are things that they have simply learned to do. It requires very little mental workload for them to perform these amazing feats. And their actions are fantastic. This is exactly what you want to do on the steering wheel to catch the car in this situation.

Now, this has given us tremendous insight and inspiration for our own autonomous vehicles. We've started to ask the question: Can we make them a little less algorithmic and a little more intuitive? Can we take this reflexive action that we see from the very best race car drivers, introduce it to our cars, and maybe even into a system that could get onto your car in the future? That would take us a long step along the road to autonomous vehicles that drive as well as the best humans.

But it's made us think a little bit more deeply as well. Do we want something more from our car than to simply be a chauffeur? Do we want our car to perhaps be a partner, a coach, someone that can use their understanding of the situation to help us reach our potential? Can, in fact, the technology not simply replace humans, but allow us to reach the level of reflex and intuition that we're all capable of?

So, as we move forward into this technological future, I want you to just pause and think of that for a moment. What is the ideal balance of human and machine? And as we think about that, let's take inspiration from the absolutely amazing capabilities of the human body and the human mind.

Thank you.

(Applause)

A Smarter, Stealthier Botnet

The "most technologically sophisticated" malware uses clever communications tricks and encryption to avoid disruption.

- Friday, July 1, 2011, By David Talbot

A new kind of botnet—a network of malware-infected PCs—behaves less like an army and more like a decentralized terrorist network, experts say. It can survive decapitation strikes, evade conventional defenses, and even wipe out competing criminal networks.

The botnet's resilience is due to a super-sophisticated piece of malicious software known as TDL-4, which in the first three months of 2011 infected more than 4.5 million computers around the world, about a third of them in the United States.

The emergence of TDL-4 shows that the business of installing malicious code on PCs is thriving. Such code is used to conduct spam campaigns and various forms of theft and fraud, such as siphoning off passwords and other sensitive data. It's also been used in the billion-dollar epidemic of fake anti-virus scams.

"Ultimately TDL-4 is simply a tool for maintaining and protecting a compromised platform for fraud," says Eric Howes, malware analyst for GFI Software, a security company. "It's part of the black service economy for malware, which has matured considerably over the past five years and which really needs a lot more light shed on it."

Unlike other botnets, the TDL-4 network doesn't rely on a few central "command-and-control" servers to pass along instructions and updates to all the infected computers. Instead, computers infected with TDL-4 pass along instructions to one another using public peer-to-peer networks. This makes it a "decentralized, server-less botnet," wrote Sergey Golovanov, a malware researcher at the Moscow-based security company Kaspersky Lab, on this blog describing the new threat.

"The owners of TDL are essentially trying to create an 'indestructible' botnet that is protected against attacks, competitors, and antivirus companies," Golovanov wrote. He added that it "is one of the most technologically sophisticated, and most complex-to-analyze malware."

The TDL-4 botnet also breaks new ground by using an encryption algorithm that hides its communications from traffic-analysis tools. This is an apparent response to efforts by researchers to discover infected machines and disable botnets by monitoring their communication patterns, rather than simply identifying the presence of the malicious code.

Demonstrating that there is no honor among malicious software writers, TDL-4 scans for and deletes 20 of the most common forms of competing malware, so it can keep infected machines all to itself. "It's interesting to mention that the features are generally oriented toward achieving perfect stealth, resilience, and getting rid of 'competitor' malware," says Costin Raiu, another malware researcher at Kaspersky.

Distributed by criminal freelancers called affiliates, who get paid between $20 and $200 for every 1,000 infected machines, TDL-4 lurks on porn sites and some video and file-storage services, among other places, where it can be automatically installed using vulnerabilities in a victim's browser or operating system.

Once TDL-4 infects a computer, it downloads and installs as many as 30 pieces of other malicious software—including spam-sending bots and password-stealing programs. "There are other malware-writing groups out there, but the gang behind [this one] is specifically targeted on delivering high-tech malware for profit," says Raiu.

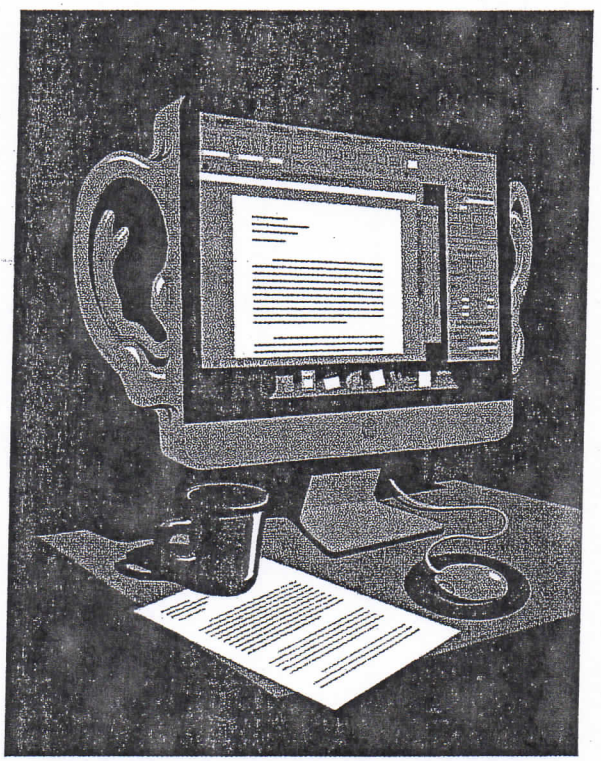
David Pogue is the personal-technology columnist for the *New York Times* and an Emmy Award-winning correspondent for *CBS News*.

Talk to the Machine

Speech-recognition programs are no longer clumsy exercises in futility

In the past couple of years speech-recognition software has quietly grown tendrils into every corner of our lives. It's at the other end of customer-support hotlines and airline reservation systems. It's built into Microsoft Windows. It's an alternative text-input method for touch-screen phones such as the iPhone and the Android. But let's face it: most people who use this software wish they didn't have to.

That's because speech recognition is usually plan B: a least terrible alternative to typing or actual human conversation. Corporations use it for their phone systems because it's cheaper than hiring real people. Many people who dictate into their computers do it because they must, perhaps because of a disability. And speech recognition is cropping up on touch-screen phones because typing on an on-screen keyboard is slow and fussy.



So what would it take to make speech recognition more than a work-around? How close are we to the *Star Trek* ideal of conversational computers that never get it wrong?

Well, we're getting there. It turns out that after a decade of buyouts, mergers and embezzlement scandals, there is only one major speech-recognition company left: Nuance Communications. It sells the only commercial dictation software for Windows, for Macintosh and for iPhone. Its technology drives the voice-command systems in cars from Audi, BMW, Ford and Mercedes and cell phones from Motorola, Nokia, Samsung, Verizon and T-Mobile. It powers voice-activated toys, GPS units and cash machines, and it answers the phone at AT&T, Bank of America, CVS and many others.

Every year Nuance releases another new version of its consumer dictation programs, such as Dragon NaturallySpeaking. Usually it doesn't add many new features. Instead it devotes most of its resources to a single goal: improving accuracy.

In the beginning, you had to train these programs by reading a 45-minute script into your microphone so that the program could learn your voice. As the technology improved over the years, that training session fell to 20 minutes, to 10, to five—and now you don't have to train the software at all. You just start dictating, and you get (by my testing) 99.9 percent accuracy. That's still one word wrong every couple of pages, but it's impressive.

Speech engineers use all kinds of tricks to boost accuracy. The earliest dictation programs required you to pause after each word; the software had no clue how to distinguish "their" from "there" and "they're." But in time, ever more powerful PC processors made continuous-speech analysis possible. Today you are encouraged to speak in longer phrases, so the software has more context to analyze for accuracy.

Another trick: Last year Nuance offered a free dictation app for the iPhone, called Dragon Dictation. What you say is transmitted to the company's servers, where it is analyzed, converted to text and zapped back to your screen within seconds.

What nobody knew, though, is that the company *stored* all those millions of speech samples, in effect creating an immense storehouse of different voices, ages, inflections and accents against which to test different recognition algorithms.

So, yes, the technology is improving. But readers often ask me: "If dictation software is so good, can I use it to transcribe phone calls and interviews?"

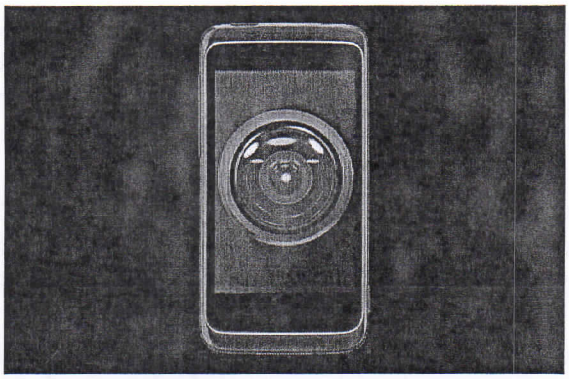
The answer is still no. The software isn't much good unless you are speaking into a microphone, without background noise, preferably without an accent. You still have to speak all punctuation ("comma"), like this ("period"). And goodness knows, we humans have enough trouble understanding each other; it's a bit much to ask for a computer to get it all right. No wonder today's dictation apps still make mistakes such as "mode import" for "modem port," "move eclipse" for "movie clips," and "oak wrap" for—well, you get it.

So, no, the keyboard isn't going away in our lifetime. Conversational-style *Star Trek* computing is still decades away. Sure, 99.9 percent accuracy is darned good—but until it reaches 100, speech-recognition technology is still plan B. SA

FOUR VOICE-RECOGNITION TIPS www.ScientificAmerican.com/dec2010/pogue

Illustration by Chris Whetzel

Where Speech Recognition Is Going



Voice-controlled interfaces are showing up in mobile phones, TVs, and automobiles. One company believes it can give just about everything a voice.

- Tuesday, May 29, 2012
- By Will Knight

Until recently, the idea of holding a conversation with a computer seemed pure science fiction. If you asked a computer to "open the pod bay doors"—well, that was only in movies.

But things are changing, and quickly. A growing number of people now talk to their mobile smart phones, asking them to send e-mail and text messages, search for directions, or find information on the Web.

"We're at a transition point where voice and natural-language understanding are suddenly at the forefront," says Vlad Sejnoha, chief technology officer of Nuance Communications, a company based in Burlington, Massachusetts, that dominates the market for speech recognition with its Dragon software and other products. "I think speech recognition is really going to upend the current [computer] interface."

Progress has come about thanks in part to steady progress in the technologies needed to help machines understand human speech, including machine learning and statistical data-mining techniques. Sophisticated voice technology is already commonplace in call centers, where it lets users navigate through menus and helps identify irate customers who should be handed off to a real customer service rep.

Now the rapid rise of powerful mobile devices is making voice interfaces even more useful and pervasive.

Jim Glass, a senior research scientist at MIT who has been working on speech interfaces since the 1980s, says today's smart phones pack as much processing power as the laboratory machines he worked with in the '90s. Smart phones also have high-bandwidth data connections to the cloud, where servers can do the heavy lifting involved with both voice recognition and understanding spoken queries. "The combination of more data and more computing power means you can do things today that you just couldn't do before," says Glass. "You can use more sophisticated statistical models."

The most prominent example of a mobile voice interface is, of course, Siri, the voice-activated personal assistant that comes built into the latest iPhone. But voice functionality is built into Android, the Windows Phone platform, and most other mobile systems, as well as many apps. While these interfaces still have considerable limitations (see Social Intelligence), we are inching closer to machine interfaces we can actually talk to.

Nuance is at the heart of the boom in voice technology. The company was founded in 1992 as Visioneer and has acquired dozens of other voice technology businesses. It now has more than 6,000 staff members at 35 locations around the world, and its revenues in the second quarter of 2012 were $390.3 million, a 22.4 percent increase over the same period in 2011.

In recent years, Nuance has deftly applied its expertise in voice recognition to the emerging market for speech interfaces. The company supplies voice recognition technology to many other companies and is widely believed to provide the speech component of Siri.

Speech is ideally suited to mobile computing, says Nuance's CTO, partly because users have their hands and eyes otherwise occupied—but also because a single spoken command can accomplish tasks that would normally require a multitude of swipes and presses. "Suddenly you have this new building block, this new dimension that you can bring to the problem," says Sejnoha. "And I think we're going to be designing the basic modern device UI with that in mind."

Inspired by the success of voice recognition software on mobile phones, Nuance hope to put its speech interfaces in many more places, most notably the television and the automobile. Both are popular and ripe for innovation.

To find a show on TV, or to schedule a DVR recording, viewers currently have to navigate awkward menus using a remote that was never designed for keying in text queries. Products that were supposed to make finding a show easier, such as Google TV, have proved too complex for people who just want to relax for an evening's entertainment.

At Nuance's research labs, Sejnoha demonstrated software called Dragon TV running on a television in a mocked-up living room. When a colleague said, "Dragon TV, find movies starring Meryl Streep," the interface instantly scanned through channel listings to select several appropriate movies. A version of this technology is already in some televisions sold by Samsung.

Apple is widely rumored to be developing its own television, and it's speculated that Siri will be its controller. The idea has been fueled by Walter Isaacson's biography of Steve Jobs, in which the late CEO is said to have claimed that he'd "finally solved" the TV interface.

Meanwhile, the Sync entertainment system in Ford automobiles already uses Nuance's technology to let drivers pull up directions, weather information, and songs. About four million Ford cars on the road have Sync with voice recognition. Last week, Nuance introduced software called Dragon Drive that will let other car manufacturers add voice-control features to vehicles.

Both these new contexts are challenging. One reason voice interfaces have become popular on smart phones is that users speak directly into the device's microphone. To ensure that the system works well in televisions and cars, where there is more background noise, the company is experimenting with array microphones and noise-canceling technology.

Nuance makes a number of software development kits available to anyone who wants to include voice recognition technology in an application. Montrue Technologies, a company based in Ashland, Oregon, used Nuance's mobile medical SDK to develop an iPad app that lets physicians dictate notes.

"It's astonishingly accurate," says Brian Phelps, CEO and cofounder of Montrue and himself an ER doctor. "Speech has turned a corner; it's gotten to a point where we're getting incredible accuracy right out of the box."

In turn, the kits shore up Nuance's position, helping the company improve its voice recognition and language processing algorithms by sending ever more voice data through its servers. As MIT's Glass says, "there has been a long-time saying in the speech-recognition community: 'There's no data like more data'." Nuance says it stores the data in an anonymous format to protect privacy.

Sejnoha believes that within a few years, mobile voice interfaces will be much more pervasive and powerful. "I should just be able to talk to it without touching it," he says. "It will constantly be listening for trigger words, and will just do it—pop up a calendar, or ready a text message, or a browser that's navigated to where you want to go."

Perhaps people will even speak to computers they wear, like the photo-snapping eyeglasses in development at Google. Sources at Nuance say they are actively planning how speech technology would have to be architected to run on wearable computers.

Gaming can make a better world

I'm Jane McGonigal. I'm a game designer. I've been making games online now for 10 years, and my goal for the next decade is to try to make it as easy to save the world in real life as it is to save the world in online games. Now, I have a plan for this, and it entails convincing more people, including all of you, to spend more time playing bigger and better games.

Right now we spend three billion hours a week playing online games. Some of you might be thinking, "That's a lot of time to spend playing games. Maybe too much time, considering how many urgent problems we have to solve in the real world." But actually, according to my research at The Institute For The Future, it's actually the opposite is true. Three billion hours a week is not nearly enough game play to solve the world's most urgent problems.

In fact, I believe that if we want to survive the next century on this planet, we need to increase that total dramatically. I've calculated the total we need at 21 billion hours of game play every week. So, that's probably a bit of a counterintuitive idea, so I'll say it again, let it sink in: If we want to solve problems like hunger, poverty, climate change, global conflict, obesity, I believe that we need to aspire to play games online for at least 21 billion hours a week, by the end of the next decade. (Laughter) No. I'm serious. I am.

Here's why. This picture pretty much sums up why I think games are so essential to the future survival of the human species. (Laughter) Truly. This is a portrait by a photographer named Phil Toledano. He wanted to capture the emotion of gaming, so he set up a camera in front of gamers while they were playing. And this is a classic gaming emotion. Now, if you're not a gamer, you might miss some of the nuance in this photo. You probably see the sense of urgency, a little bit of fear, but intense concentration, deep, deep focus on tackling a really difficult problem.

If you are a gamer, you will notice a few nuances here: the crinkle of the eyes up, and around the mouth is a sign of optimism, and the eyebrows up is surprise. This is a gamer who is on the verge of something called an epic win. (Laughter) Oh, you've heard of that. OK, good, so we have some gamers among us. An epic win is an outcome that is so extraordinarily positive you had no idea it was even possible until you achieved it. It was almost beyond the threshold of imagination. And when you get there you are shocked to discover what you are truly capable of. That is an epic win. This is a gamer on the verge of an epic win. And this is the face that we need to see on millions of problem-solvers all over the world as we try to tackle the obstacles of the next century -- the face of someone who, against all odds is on the verge of an epic win.

Now, unfortunately this is more of the face that we see in everyday life now as we try to tackle urgent problems. This is what I call the "I'm Not Good At Life" face, and this is actually me making it. Can you see? Yes. Good. This is actually me making the "I'm Not Good At Life" face. This is a piece of graffiti in my old neighborhood in Berkeley, California, where I did my PhD on why we're better in games than we are in real life. And this is a problem that a lot of gamers have. We feel that we are not as good in reality as we are in games.

And I don't mean just good as in successful, although that's part of it. We do achieve more in game worlds. But I also mean good as in motivated to do something that matters, inspired to collaborate and to cooperate. And when we're in game worlds I believe that many of us become the best version of ourselves, the most likely to help at a moment's notice, the most likely to stick with a problem as long at it takes, to get up after failure and try again. And in real life, when we face failure, when we confront obstacles, we often don't feel that way. We feel overcome, we feel overwhelmed, we feel anxious, maybe depressed, frustrated or cynical. We never have those feelings when we're playing games, they just don't exist in games. So, that's what I wanted to study when I was a graduate student.

What about games makes it impossible to feel that we can't achieve everything? How can we take those feelings from games and apply them to real-world work? So, I looked at games like World of Warcraft, which is really the ideal collaborative problem-solving environment. And I started to notice a few things that make epic wins so possible in online worlds.

So, the first thing is whenever you show up in one of these online games, especially in World of Warcraft, there are lots and lots of different characters who are willing to trust you with a world-saving mission, right away. But not just any mission, it's a mission that is perfectly matched with your current level in the game. Right? So, you can do it. They never give you a challenge that you can't achieve. But it is on the verge of what you're capable of. So, you have to try hard, but there's no unemployment in World of Warcraft. There is no sitting around wringing your hands, there's always something specific and important to be done. And there are also tons of collaborators. Everywhere you go, hundreds of thousands of people ready to work with you to achieve your epic mission.

That's not something that we have in real life that easily, this sense that at our fingertips are tons of collaborators. And also there is this epic story, this inspiring story of why we're there, and what we're doing. And then we get all this positive feedback. You guys have heard of leveling up and plus-one strength, and plus-one intelligence. We don't get that kind of constant feedback in real life. When I get off this stage I'm not going to have plus-one speaking, and plus-one crazy idea, plus-20 crazy idea. I don't get that feedback in real life.

Now, the problem with collaborative online environments like World of Warcraft is that it's so satisfying to be on the verge of an epic win all the time that we decide to spend all our time in these game worlds. It's just better than reality. So, so far, collectively all the World of Warcraft gamers have spent 5.93 million years solving the virtual problems of Azeroth. Now, that's not necessarily a bad thing. It might sound like it's a bad thing. But to put that in context: 5.93 million years ago was when our earliest primate human ancestors stood up. That was the first upright primate.

Okay, so when we talk about how much time we're currently investing in playing games, the only way it makes sense to even think about it is to talk about time at the magnitude of human evolution, which is an extraordinary thing. But it's also apt. Because it turns out that by spending all this time playing games, we're actually changing what we are capable of as human beings. We are evolving to be a more collaborative and hearty species. This is true. I believe this.

So, consider this really interesting statistic; it was recently published by a researcher at Carnegie Mellon University: The average young person today in a country with a strong gamer culture will have spent 10,000 hours playing online games by the age of 21. Now 10,000 hours is a really interesting number for

two reasons. First of all, for children in the United States 10,080 hours is the exact amount of time you will spend in school from fifth grade to high school graduation if you have perfect attendance.

So, we have an entire parallel track of education going on where young people are learning as much about what it takes to be a good gamer as they are learning about everything else in school. And some of you have probably read Malcolm Gladwell's new book "Outliers." So, you would have heard of his theory of success, the 10,000 hour theory of success. It's based on this great cognitive science research that if we can master 10,000 hours of effortful study at anything by the age of 21, we will be virtuosos at it. We will be as good at whatever we do as the greatest people in the world. And so, now what we're looking at is an entire generation of young people who are virtuoso gamers.

So, the big question is, "What exactly are gamers getting so good at?" Because if we could figure that out, we would have a virtually unprecedented human resource on our hands. This is how many people we now have in the world who spend at least an hour a day playing online games. These are our virtuoso gamers, 500 million people who are extraordinarily good at something. And in the next decade we're going to have another billion gamers who are extraordinarily good at whatever that is. If you don't know it already, this is coming. The game industry is developing consoles that are low energy and that work with the wireless phone networks instead of broadband Internet so that gamers all over the world, particularly in India, China, Brazil, can get online. They expect one billion more gamers in the next decade. It will bring us up to 1.5 billion gamers.

So, I've started to think about what these games are making us virtuosos at. Here are the four things I came up with. The first is urgent optimism. OK, think of this as extreme self-motivation. Urgent optimism is the desire to act immediately to tackle an obstacle, combined with the belief that we have a reasonable hope of success. Gamers always believe that an epic win is possible, and that it is always worth trying, and trying now. Gamers don't sit around. Gamers are virtuosos at weaving a tight social fabric. There's a lot of interesting research that shows that we like people better after we play a game with them, even if they've beaten us badly. And the reason is, it takes a lot of trust to play a game with someone. We trust that they will spend their time with us, that they will play by the same rules, value the same goal, they'll stay with the game until it's over.

And so, playing a game together actually builds up bonds and trust and cooperation. And we actually build stronger social relationships as a result. Blissful productivity. I love it. You know there's a reason why the average World of Warcraft gamer plays for 22 hours a week, kind of a half-time job. It's because we know, when we're playing a game, that we're actually happier working hard than we are relaxing, or hanging out. We know that we are optimized, as human beings, to do hard meaningful work. And gamers are willing to work hard all the time, if they're given the right work.

Finally: epic meaning. Gamers love to be attached to awe-inspiring missions to human planetary-scale stories. So, just one bit of trivia that helps put that into perspective: So, you all know Wikipedia, biggest wiki in the world. Second biggest wiki in the world, with nearly 80,000 articles, is the World of Warcraft wiki. Five million people use it every month. They have compiled more information about World of Warcraft on the Internet than any other topic covered on any other wiki in the world. They are building an epic story. They are building an epic knowledge resource about the World of Warcraft.

Okay, so these are four superpowers that add up to one thing: Gamers are super-empowered, hopeful individuals. These are people who believe that they are individually capable of changing the world. And the only problem is that they believe that they are capable of changing virtual worlds and not the real world. That's the problem that I'm trying to solve.

There's an economist named Edward Castronova. His work is brilliant. He looks at why people are investing so much time and energy and money in online worlds. And he says, "We're witnessing what amounts to no less than a mass exodus to virtual worlds and online game environments." And he's an economist. So, he's rational. And he says ... (Laughter) Not like me -- I'm a game designer; I'm exuberant. But he says that this makes perfect sense, because gamers can achieve more in online worlds than they can in real life. They can have stronger social relationships in games than they can have in real life; they get better feedback and feel more rewarded in games than they do in real life. So, he says for now it makes perfect sense for gamers to spend more time in virtual worlds than the real world. Now, I also agree that that is rational, for now. But it is not, by any means, an optimal situation. We have to start making the real world more like a game.

So, I take my inspiration from something that happened 2,500 years ago. These are ancient dice, made out of sheep's knuckles. Right? Before we had awesome game controllers, we had sheep's knuckles. And these represent the first game equipment designed by human beings. And if you're familiar with the work of the ancient Greek historian Herodotus, you might know this history, which is the history of who invented games and why. Herodotus says that games, particularly dice games, were invented in the kingdom of Lydia during a time of famine.

Apparently, there was such a severe famine that the king of Lydia decided that they had to do something crazy. People were suffering. People were fighting. It was an extreme situation, they needed an extreme solution. So, according to Herodotus, they invented dice games and they set up a kingdom-wide policy: On one day, everybody would eat, and on the next day, everybody would play games. And they would be so immersed in playing the dice games because games are so engaging, and immerse us in such satisfying blissful productivity, they would ignore the fact that they had no food to eat. And then on the next day, they would play games; and on the next day, they would eat.

And according to Herodotus, they passed 18 years this way, surviving through a famine by eating on one day and playing games on the next. Now, this is exactly, I think, how we're using games today. We're using games to escape real-world suffering. We're using games to get away from everything that's broken in the real environment, everything that's not satisfying about real life, and we're getting what we need from games.

But it doesn't have to end there. This is really exciting. According to Herodotus, after 18 years the famine wasn't getting better, so the king decided they would play one final dice game. They divided the entire kingdom in half. They played one dice game, and the winners of that game got to go on an epic adventure. They would leave Lydia, and they would go out in search of a new place to live, leaving behind just enough people to survive on the resources that were available, and hopefully to take their civilization somewhere else where they could thrive.

Now, this sounds crazy, right? But recently, DNA evidence has shown that the Etruscans, who then led to the Roman Empire, actually share the same DNA as the ancient Lydians. And so, recently, scientists have

suggested that Herodotus' crazy story is actually true. And geologists have found evidence of a global cooling that lasted for nearly 20 years that could have explained the famine. So, this crazy story might be true. They might have actually saved their culture by playing games, escaping to games for 18 years, and then been so inspired, and knew so much about how to come together with games, that they actually saved the entire civilization that way.

Okay, we can do that. We've been playing Warcraft since 1994. That was the first real-time strategy game from the World of Warcraft series. That was 16 years ago. They played dice games for 18 years, we've been playing Warcraft for 16 years. I say we are ready for our own epic game. Now, they had half the civilization go off in search of a new world, so that's where I get my 21 billion hours a week of game-play from. Let's get half of us to agree to spend an hour a day playing games, until we solve real-world problems.

Now, I know you're asking, "How are we going to solve real world problems in games?" Well, that's what I have devoted my work to over the past few years, at The Institute For The Future. We have this banner in our offices in Palo Alto, and it expresses our view of how we should try to relate to the future. We do not want to try to predict the future. What we want to do is make the future. We want to imagine the best-case scenario outcome, and then we want to empower people to make that outcome a reality. We want to imagine epic wins, and then give people the means to achieve the epic win.

I'm just going to very briefly show you three games that I've made that are an attempt to give people the means to create epic wins in their own futures. So, this is World Without Oil. We made this game in 2007. This is an online game in which you try to survive an oil shortage. The oil shortage is fictional, but we put enough online content out there for you to believe that it's real, and to live your real life as if we've run out of oil. So when you come to the game, you sign up, you tell us where you live, and then we give you real-time news, videos, data feeds that show you exactly how much oil costs, what's not available, how food supply is being affected, how transportation is being affected, if schools are closed, if there is rioting, and you have to figure out how you would live your real life as if this were true. And then we ask you to blog about it, to post videos, to post photos.

We piloted this game with 1,700 players in 2007, and we've tracked them for the three years since. And I can tell you that this is a transformative experience. Nobody wants to change how they live just because it's good for the world, or because we're supposed to. But if you immerse them in an epic adventure and tell them, "We've run out of oil. This is an amazing story and adventure for you to go on. Challenge yourself to see how you would survive," most of our players have kept up the habits that they learned in this game.

So, for the next world-saving game, we decided to aim higher: bigger problem than just peak oil. We did a game called Superstruct at The Institute For The Future. And the premise was a supercomputer has calculated that humans have only 23 years left on the planet. This supercomputer was called the Global Extinction Awareness System, of course. We asked people to come online almost like a Jerry Bruckheimer movie. You know Jerry Bruckheimer movies, you form a dream team -- you've got the astronaut, the scientist, the ex-convict, and they all have something to do to save the world. (Laughter)

But in our game, instead of just having five people on the dream team, we said, "Everybody's on the dream team, and it's your job to invent the future of energy, the future of food, the future of health, the

future of security and the future of the social safety net." We had 8,000 people play that game for eight weeks. They came up with 500 insanely creative solutions that you can go online, if you Google "Superstruct," and see.

So, finally, the last game, we're launching it March 3rd. This is a game done with the World Bank Institute. If you complete the game you will be certified by the World Bank Institute, as a Social Innovator, class of 2010. Working with universities all over sub-Saharan Africa, and we are inviting them to learn social innovation skills. We've got a graphic novel, we've got leveling up in skills like local insight, knowledge networking, sustainability, vision and resourcefulness. I would like to invite all of you to please share this game with young people, anywhere in the world, particularly in developing areas, who might benefit from coming together to try to start to imagine their own social enterprises to save the world.

So, I'm going to wrap up now. I want to ask a question. What do you think happens next? We've got all these amazing gamers, we've got these games that are kind of pilots of what we might do, but none of them have saved the real world yet. Well I hope that you will agree with me that gamers are a human resource that we can use to do real-world work, that games are a powerful platform for change. We have all these amazing superpowers: blissful productivity, the ability to weave a tight social fabric, this feeling of urgent optimism and the desire for epic meaning.

I really hope that we can come together to play games that matter, to survive on this planet for another century. And that's my hope, that you will join me in making and playing games like this. When I look forward to the next decade, I know two things for sure: that we can make any future we can imagine, and we can play any games we want. So, I say: Let the world-changing games begin. Thank you. (Applause)

It Project
The Guidelines for the English Part.

General Info:

- There will be one grade (Oral and Written grade put together)
- It will be considered as a *note de contrôle continu* with an appropriate "coef"
- The jury will be an English teacher.

Written part:

A **400 words account** from each student recounting what happened during the first three presentations of the project. It should be no longer than **one page** and the deadly mistakes rules will be applied as usual.
The summary should be given to the jury on the day of the presentation and separate from the rest of the report.

Oral part:

The first 10 minutes of your 25-minute presentation must be in English. The time limit will be strictly respected by the jury. The time should be equally divided among the students present on the day.

Here are the criteria:

Oral part

- Good preparation is necessary in order to obtain a decent grade.
- Good use of notes
- Eye contact – look at your audience
- Clarity and good pronunciation.
- Good transitions between students
- Speak loudly and try to vary your tone
- Good energy; be enthusiastic

Written part

- Typed (handwritten work won't be accepted)
- Be careful with the deadly mistakes
- No plagiarism, use your own words
- Even if the facts are the same, give your own version
- Proper layout and a minimum font size of 11

Technical English Major Project

The Technical English major Project has two very different parts:

- The report – 30% of the total grade.
- The presentation – 70% of the total grade.

In a group of 3 to 4 students you will be ask to choose an IT related subject (to be cleared by the teacher in advance) and present it to the rest of the class.
This chosen subject should be divided into two distinctive parts, one for the report and the other for the presentation. **Under no circumstances should you have the same part for both the report and the presentation, they have the same general subject but deal with two different chapters**. A good idea would be to talk about past events in the report and future ones during the presentation, but this wouldn't apply for all subjects.

Both the report and the presentation should include technical details but it should be comprehensible for any reader or listener. If necessary take a little bit of time to explain it carefully.

The idea of this exercise is to train the students to present various types of subjects in the world of IT. Make sure to emphasize the vocabulary and don't hesitate to explain it to the rest of the class during the presentation.

prezi.com

Here is a list of subjects used in the past:

Talk about technical facts/details

Operating systems
Technical evolution of Apple
Technical evolution of Microsoft
Google and its modules
Mobile evolution
Cloud computing
Evolution of Computers
Nano technology
Imbedded systems
Augmented reality
Evolution of music in video games
Artificial Intelligence
Hadopi
3D Technologies
Hackers and Fishers
Cracking security in Video Games
Wireless Networks
OLED technologies
IT in medicine
Robots
Computer anti-virus
Peer to peer
Evolution of game consoles

These are just examples, feel free to choose others.

Technical English Major Project

The Technical English major Project has two very different parts:

- The report – 30% of the total grade.
- The presentation – 70% of the total grade.

In a group of 3 to 4 students you will be ask to choose an IT related subject (to be cleared by the teacher in advance) and present it to the rest of the class.
This chosen subject should be divided into two distinctive parts, one for the report and the other for the presentation. **Under no circumstances should you have the same part for both the report and the presentation, they have the same general subject but deal with two different chapters**. A good idea would be to talk about past events in the report and future ones during the presentation, but this wouldn't apply for all subjects.

Both the report and the presentation should include technical details but it should be comprehensible for any reader or listener. If necessary take a little bit of time to explain it carefully.

The idea of this exercise is to train the students to present various types of subjects in the world of IT. Make sure to emphasize the vocabulary and don't hesitate to explain it to the rest of the class during the presentation.

prezi.com

Here is a list of subjects used in the past:

Talk about technical facts/details

Operating systems
Technical evolution of Apple
Technical evolution of Microsoft
Google and its modules
Mobile evolution
Cloud computing
Evolution of Computers
Nano technology
Imbedded systems
Augmented reality
Evolution of music in video games
Artificial Intelligence
Hadopi
3D Technologies
Hackers and Fishers

Cracking security in Video Games
Wireless Networks
OLED technologies
IT in medicine
Robots
Computer anti-virus
Peer to peer
Evolution of game consoles

These are just examples, feel free to choose others.

Rules for the Major Project Report

Only one report per group needs to be handed in.

Late reports and soft copies will not be accepted. You have to print and hand them in to me on paper (hard copy) on time.

The report should be totally typed and bound.

Make sure to check spelling and grammar with the right spell corrector. Have other students in the group re-read your text.

The first page of the report should be the grading sheet included in your Tech English Package. That means that you need to detach it carefully from the package and use it for the report.

The second part of the report should be a short (funny?) personalized presentation of all members of the group and should include pictures. **100 words each**

The third part should be the table of contents followed by the rest of the report.

The next part is the report and it should be between **10 and 15 pages long including annexes (2 to 3 pages max).**

Annexes should be put at the end of the report.

The last part should be the bibliography.

Plagiarism will not be tolerated at any level. If you are caught cheating, you automatically receive a 0 for your grade, and the administration will be notified.

Rules for the Major Project Presentation

The presentation should last **one hour exactly**, no more no less. It will be followed by question from both the teacher and the students. Answering questions appropriately will improve your grade. Students from the audience asking questions may also improve their grade.

Each student should speak **an equal amount of time**. You may speak more than once. "A conversational presentation" is also allowed but needs to be very well prepared. Don't hesitate to get the class involved (debates, quick questions, etc.).

Good preparation is necessary in order to obtain a decent grade. Plan your work carefully and make sure to **rehearse in your free time**.

You may use notes during the presentation, but you are not allowed to read. **Reading will be severely penalized.**

It is totally recommended to have a Power Point ready (Prezi.com is your friend); it may include videos as long as they are not too long. **Anything appearing on the screen should be in English.** Power Point presentations need to be carefully prepared, it's not because you use Power Point that the presentation is well organized.

Deadly mistakes will apply orally as well as in writing (Power Point, etc.).

You may have hand out documents; just make sure that everyone in the class gets one.

The Major Project is a serious exercise, but you may use humor in the middle of it as long as it is subject related; just make sure we get it...

You need to check the classroom where your presentation will take place, YOU will book the video projector and/or YOU will get a room change if needed (just make sure to let me know in advance).

Key points to keep in mind for your presentation:

- Good use of notes.
- Eye contact – look at your audience.
- Clarity and good pronunciation.
- Good transitions between students.
- Speak loudly and try to vary your tone.
- Good energy; be enthusiastic.
- Present yourself and the group.

Start: _____

Finish: _____

Technical English Sup Presentation Promo 2017

Date: _____

Group Name: _____

Subject: _____

Name		
	Oral	
	Writ	
	Total	

Name		
	Oral	
	Writ	
	Total	

Name		
	Oral	
	Writ	
	Total	

Name		
	Oral	
	Writ	
	Total	

Group