Significant new inventions in computing since 1980

Asked 15 years, 11 months ago Modified 2 years, 1 month ago

Viewed 60k times

561

votes

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

This question arose from comments about different kinds of progress in computing over the last 50 years or so.

I was asked by some of the other participants to raise it as a question to the whole forum.

The basic idea here is not to bash the current state of things but to try to understand something about the progress of coming up with fundamental new ideas and principles.

I claim that we need really new ideas in most areas of computing, and I would like to know of any important and powerful ones that have been done recently. If we can't really find them, then we should ask "Why?" and "What should we be doing?"

computer-science

Share

edited Nov 17, 2022 at 22:36

community wiki
21 revs, 15 users 28%
Alan Kay

77 Jeff Atwood confirmed, that the user "Alan Kay" is THE "Alan Kay". You know, the guy who worked for that copier machine company... ;-) en.wikipedia.org/wiki/Alan_Kay – splattne Jan 11, 2009 at 15:01

1 I watched this video: video.google.com/videoplay?docid=-533537336174204822 - A historical Video (1979) about the development of the Dynabook, Children and Computers and a lot more presented by Alan Kay. AMAZING things done before 1970 - especially the "Sketchpad" part in 1962. – splattne Jan 13, 2009 at 19:02

2 depending on your own definition the answer could be anything from "none" up to an enumeration of every possible technology. And all those answers would be either correct or incorrect depending on the definition of "a new idea" the reader/observer uses... – Emile Vrijdags Jan 21, 2009 at 15:07

3 After looking at all the answers here: Good grief! Have we done *nothing* in the past 30 years?? – Jeremy Powell Oct 2, 2009 at 23:44

2 @Will: Oddly enough I believe I have recently learned of a interesting answer to this question: *fast* clustering algorithms. DBSCAN is the state of the art for a lot of this (O(n log n) in

the number of points in the data set), and it dates to 1996. Alas, with the question closed I will not take the time to read the many answers to find out if someone beaten me to it.
– dmckee --- ex-moderator kitten Nov 12, 2011 at 2:43

Comments disabled on deleted / locked posts / reviews |

129 Answers

Sorted by: Highest score (default)

311 votes

The Internet itself pre-dates 1980, but the **World Wide Web** ("distributed hypertext via simple mechanisms") as proposed and implemented by Tim Berners-Lee started in 1989/90.

While the idea of hypertext had existed before (Nelson's Xanadu had tried to implement a distributed scheme), the WWW was a new approach for implementing a distributed hypertext system. Berners-Lee combined a simple client-server protocol, markup language, and addressing scheme in a way that was powerful and easy to implement.

I think most innovations are created in re-combining existing pieces in an original way. Each of the pieces of the WWW had existed in some form before, but the combination was obvious only in hindsight.

And I know for sure that you are using it right now.

Share

edited Jan 11, 2009 at 16:06

community wiki
2 revs
splattne

26 +1 for the most obvious but also the most easily forgotten because we all take it for granted :) – PolyThinker Jan 11, 2009 at 15:22

20 I'm not using the World Wide Web right now. I'm using a series of tubes known as the internets, achieved via the google. – Robert S. Jan 11, 2009 at 21:54

13 @bruceatk: Hypertext is an implementation of text. Text was invented in 3500 BC. – Portman Jan 12, 2009 at 20:46

1 @bruceatk: I don't believe he wrote about the WWW until 1989. w3.org/People/Berners-Lee – Portman Feb 4, 2009 at 18:53

2 @splattne: And think has become search – u0b34a0f6ae May 26, 2010 at 8:04

235

votes

Free Software Foundation (Established 1985)

Even if you aren't a wholehearted supporter of their philosophy, the ideas that they have been pushing, of free software, open-source has had an amazing influence on the software industry and content in general (e.g. Wikipedia).

Share

edited Apr 25, 2009 at 22:17

community wiki
3 revs, 3 users 44%
Oddthinking

9 Agree that FSF has been very influential, but there is a tendency among its advocates to espouse "group think". So many FSF cannot accept that Apple OSX and MS Windows are much better than any open source OS for the average user. No one wants to admit that. – RussellH Jan 12, 2009 at 20:32

32 The entire purpose of the FSF is to promote software that can be freely used, modified, and redistributed by all. OSX and Windows are not "better" at this by any definition.
– Adam Lassek Jan 20, 2009 at 14:27

5 @RussellH: you're confusing "Open Source" and "Free (as in Freedom) Software". Your comment, in fact, illustrates precisely why the distinction is important. But anyway, Firefox is better than Internet Explorer and Safari, and it's more important to users than Windows vs MacOS vs Linux. – niXar Feb 3, 2009 at 18:24

8 Janie, you don't have to be a supporter to see that the principles that they are pushing have had a major effect on the industry. I have no interested in getting dragged into a discussion as to whether the FSF is communistic, or whether you should embrace some communist principles.
– Oddthinking Jul 24, 2009 at 12:54

9 Legal invention, not computing invention. – Charles Stewart Jul 14, 2010 at 8:42

149

votes

I think it's fair to say that in 1980, if you were using a computer, you were either getting paid for it or you were a geek... so what's changed?

- Printers and consumer-level **desktop publishing**. Meant you didn't need a printing press to make high-volume, high-quality printed material. That was **big** - of course, nowadays we completely take it for granted, and mostly we don't even bother with the printing part because everyone's online anyway.
- **Colour**. Seriously. Colour screens made a huge difference to non-geeks' perception of games & applications. Suddenly games seemed less like hard work and more like watching TV, which opened the doors for Sega, Nintendo, Atari et al to bring consumer gaming into the home.
- Media compression (MP3s and video files). And a whole bunch of things - like TiVO and iPods - that we don't really think of as computers any more because they're so ubiquitous and so user-friendly. But they are.

The common thread here, I think, is stuff that was once impossible (making printed documents; reproducing colour images accurately; sending messages around the world in real time; distributing audio and video material), and was then expensive because of the equipment and logistics involved, and is now consumer-level. So - what are big corporates doing now that used to be impossible but might be cool if we can work out how to do it small & cheap?

Anything that still involves physical transportation is interesting to look at. Video conferencing hasn't replaced real meetings (yet) - but with the right technology, it still might. Some recreational travel could be eliminated by a full-sensory immersive environment - home cinema is a trivial example; another is the "virtual golf course" in an office building in Soho, where you play 18 holes of real golf on a simulated course.

For me, though, the next really big thing is going to be fabrication. Making things. Spoons and guitars and chairs and clothing and cars and tiles and stuff. Things that still rely on a manufacturing and distribution infrastructure. I don't have to go to a store to buy a movie or an album any more - how long until I don't have to go to the store for clothing and kitchenware?

Sure, there are interesting developments going on with OLED displays and GPS and mobile broadband and IoC containers and scripting and "the cloud" - but it's all still just new-fangled ways of putting pictures on a screen. I can print my own photos and write my own web pages, but I want to be able to fabricate a linen basket that fits exactly into that nook beside my desk, and a mounting bracket for sticking my guitar FX unit to my desk, and something for clipping my cellphone to my bike handlebars.

Not programming related? No... but in 1980, neither was sound production. Or video distribution. Or sending messages to your relatives in Zambia. Think big, people... :)

Share

edited Apr 25, 2009 at 22:31

community wiki
2 revs, 2 users 92%
Dylan Beattie

1 I think media compression is not a new concept (it goes back to Shannon's work in 50s), it's just become feasible with improved hardware (fast enough, able to play the media). – Kornel Jan 11, 2009 at 16:44

I would have to agree with fabrication being something I think may be one of the next big things. When object "printers" become mainstream (printers that can replicate simple physical items that are durable) I think we will be there. – Andy Webb Jan 11, 2009 at 17:43

It would also be great to scan existing items so replacements can be made. I have on many occasions had to shop for an odd screw or part to replace one that broke around the house or on my bike. With such a system I could scan the old part, repair it in software, and then create the replacement. – Andy Webb Jan 11, 2009 at 17:48

44 Desktop publishing and high quality printing was invented at Xerox PARC in the 70s, some of the Altos back then also had high quality color screens. The Internet predated 1980. Media compression predated 1980. The question is about what fundamental new technologies have been invented since 1980 – Alan Kay Jan 15, 2009 at 3:09

3 You sir, are a visionary. Do not let the man get you down. 'Printing' printers is the next big revolution. – Waylon Flinn Apr 16, 2009 at 23:48

137

votes

Package management and distributed revision control.

These patterns in the way software is developed and distributed are quite recent, and are still just beginning to make an impact.

Ian Murdock has called package management "the single biggest advancement Linux has brought to the industry". Well, he would, but he has a point. The way software is installed has changed significantly since 1980, but most computer users still haven't experienced this change.

Joel and Jeff have been talking about revision control (or version control, or source control) with Eric Sink in Podcast #36. It seems most developers haven't yet caught up with *centralized* systems, and DVCS is widely seen as mysterious and unnecessary.

From the Podcast 36 transcript:

> 0:06:37
>
> **Atwood:** ... If you assume -- and this is a big assumption -- that most developers have kinda sorta mastered fundamental source control -- which I find not to be true, frankly...
>
> **Spolsky:** No. Most of them, even if they have, it's the check-in, check-out that they understand, but branching and merging -- that confuses the heck out of them.

Share

edited Jan 18, 2021 at 12:38

community wiki
4 revs
merriam

1 If one should count as a significant new invention, it's git. – hasen May 7, 2009 at 16:57

8 hasen j: git is a fantastic DCMS, however there were several others implemented before git - git, is a significant new - implementation- of an idea. – Arafangion Jun 1, 2009 at 5:54

+1 for Package Management. Still one of the major things that Linux/BSD has to hold over everybody elses' heads, although the rest are getting there (just **really** slowly). – new123456 Mar 5, 2011 at 20:12

Even server-based revision control systems are largely post-1980 developments, and going from just having the current state to having the history of the state as well… that's a colossal and subtle change. – Donal Fellows Oct 1, 2011 at 9:13

Distributed revision control is the wrong name. Nobody cares if your system is centralized of not. What is important is whether you track change-sets or versions. But most of the time, they come together (GIT, Mercurial), which confuses everybody. Joel Spolsky said it himself in a blog post: `With distributed version control, the distributed part is actually not the most interesting part.`
– Benjamin Crouzier Feb 19, 2012 at 23:54

122

votes

BitTorrent. It completely turns what previously seemed like an obviously immutable rule on its head - the time it takes for a single person to download a file over the Internet grows in proportion to the number of people downloading it. It also addresses the flaws of previous peer-to-peer solutions, particularly around 'leeching', in a way that is organic to the solution itself.

BitTorrent elegantly turns what is normally a disadvantage - many users trying to download a single file simultaneously - into an advantage, distributing the file geographically as a natural part of the download process. Its strategy for optimizing the use of bandwidth between two peers discourages leeching as a side-effect - it is in the best interest of all participants to enforce throttling.

It is one of those ideas which, once someone else invents it, seems simple, if not obvious.

Share

edited Jul 19, 2010 at 10:26

community wiki
2 revs, 2 users 62%
Kief

True, ahtough while BitTorrent may be somewhat different/improved, the *significant new invention* really should be P2P-distribution, rather than any specific implementation like BitTorrent. – Ilari Kajaste Oct 14, 2009 at 8:36

10 I disagree. P2P is not at all new, it's older than USENET. Pre-bitTorrent "P2P" apps for the desktop (Kazaa and the like) are simply repacking of the client-server concept, adding a dynamic central directory of servers. Each "peer" client connects to a single other "peer" server to transfer a file. The fact that a single node does both is old hat (at least for pre-Windows systems). The bitTorrent protocol is (AFAIK) a completely new way to transfer files, which leverages multiple systems to transfer a file between one another in a truly distributed manner. – Kief Oct 14, 2009 at 12:59

7 @JL: In theory, direct download is faster, but not in practice. With one seeder and one leacher, there shouldn't be any difference. As soon as you add another leacher, that leacher can start taking pieces from whoever has a faster connection (even if the client with the faster connection doesn't have the complete file). With a direct download, to take advantage of the faster connection, you would first have to wait for the client to finish the download before you could start.
– Peter Di Cecco Mar 3, 2010 at 14:15

1 I think the better question becomes how much bandwidth do you save by hosting a torrent and seeding it with what would have been a direct download box. Only companies like Blizzard know that now, and I havent seen them talk numbers. Without a 'super seed' torrents will rely on users to seed, which just doesnt work with async connections and people not wanting to leave their computer on and upstream saturated. – semi Mar 17, 2010 at 4:20

6 @JL: torrents are slower than direct download? My "practical" experience says different; try going to download Eclipse both ways. – Dean J Aug 4, 2010 at 14:08

120 votes

Damas-Milner type inference (often called Hindley-Milner type inference) was published in 1983 and has been the

basis of every sophisticated static type system since. It was a genuinely new idea in programming languages (admitted based on ideas published in the 1970s, but not made practical until after 1980). In terms of importance I put it up with Self and the techniques used to implement Self; in terms of influence it has no peer. (The rest of the OO world is still doing variations on Smalltalk or Simula.)

Variations on type inference are still playing out; the variation I would single out the most is Wadler and Blott's *type class* mechanism for resolving overloading, which was later discovered to offer very powerful mechanisms for programming at the type level. The end to this story is still being written.

Share

answered Jan 12, 2009 at 3:04

community wiki
Norman Ramsey

3 +1 Static type systems are a huge *huge* step in software development. I couldn't agree with this answer more.
– Jeremy Powell Oct 2, 2009 at 23:41

104 votes

Here's a plug for **Google map-reduce**, not just for itself, but as a proxy for Google's achievement of running fast, reliable services on top of farms of unreliable, commodity machines. Definitely an important invention and totally

different from the big-iron mainframe approaches to heavyweight computation that ruled the roost in 1980.

Share

answered Jan 12, 2009 at 3:07

community wiki
Norman Ramsey

10 map-reduce isn't an invention of Google at all. – akappa Jun 29, 2009 at 16:17

20 I'm a functional programmer. My first language was APL. Your point, exactly? – Norman Ramsey Jun 30, 2009 at 0:49

15 So (mapcar f l) and (reduce f l) in Lisp automatically run on arbitrary numbers of commodity machines, handling all intercommunication, failures, and restarts? – Jared Updike Jul 27, 2009 at 1:36

16 The Google map-reduce doesn't have much at all to do with functional map-reduce. – aehlke Aug 24, 2009 at 5:32

91

votes

Tagging, the way information is categorized. Yes, the little boxes of text under each question.

It is amazing that it took about 30 years to invent tagging. We used lists and tables of contents; we used things which are optimized for printed books.

However 30 years is much shorter than the time people needed to realize that printed books can be in smaller format. People can keep books in hands.

I think that the tagging concept is underestimated among core CS guys. All research is focused on natural language processing (top-down approach). **But tagging is the first language in which computers and people can both understand well.** It is a bottom-up approach that makes computers use natural languages.

Share

edited Jan 17, 2011 at 22:07

community wiki
3 revs, 2 users 79%
Greg Dan

1 Agreed - this correlates with my submission that the only new thing I can think of is syntactic markup to query among many domains - but you stated it better. – dkretz Jan 14, 2009 at 0:39

40 Check out Engelbart ca 1962-72 – Alan Kay Jan 15, 2009 at 2:56

For me tagging is very much like early search engines that used meta=keywords tag (that's post-80's too, I'm just making argument that tagging isn't worth mentioning). – Kornel Jan 16, 2009 at 22:21

1 While tagging in computing is relatively new approach, tagging is also a concept inherited from books; in books, it's called indexing. – Domchi Nov 1, 2009 at 13:52

6 libraries have been using "tags" since... well I don't know but since a long time. Think about the *book cards* (sorry, I'm not sure how they're called in English) tagged "books about xxx". – nico May 25, 2010 at 17:58

80

votes

I think we are looking at this the wrong way and drawing the wrong conclusions. If I get this right, the cycle goes:

Idea -> first implementation -> minority adoption -> critical mass -> commodity product

From the very first idea to the commodity, you often have centuries, assuming the idea ever makes it to that stage. Da Vinci may have drawn some kind of helicopter in 1493 but it took about 400 years to get an actual machine capable of lifting itself off the ground.

From William Bourne's first description of a submarine in 1580 to the first implementation in 1800, you have 220 years and current submarines are still at an infancy stage: we almost know nothing of underwater traveling (with 2/3rdof the planet under sea, think of the potential real estate ;).

And there is no telling that there wasn't earlier, much earlier ideas that we just never heard of. Based on some legends, it looks like Alexander the Great used some kind of diving bell in 332 BC (which is the basic idea of a submarine: a device to carry people and air supply below the sea). Counting that, we are looking at 2000 years from idea (even with a basic prototype) to product.

What I am saying is that looking today for implementations, let alone products, that were not even ideas prior to 1980 is ... I betcha the "quick sort" algorithm was used by some no name file clerk in ancient China. So what?

There were networked computers 40 years ago, sure, but that didn't compare with today's Internet. The basic idea/technology was there, but regardless you couldn't play a game of Warcraft online.

> I claim that we need really new ideas in most areas of computing, and I would like to know of any important and powerful ones that have been done recently. If we can't really find them, then we should ask "Why?" and "What should we be doing?"

Historically, we have never been able to "find them" that close from the idea, that fast. I think the cycle is getting faster, but computing is still darn young.

Currently, I am trying to figure out how to make an hologram (the Star Wars kind, without any physical support). I think I know how to make it work. I haven't even gathered the tools, materials, funding and yet even if I was to succeed to any degree, the actual idea would already be several decades old, at the very least and related implementations/technologies have been used for just as long.

As soon as you start listing actual products, you can be pretty sure that concepts and first implementations existed a while ago. Doesn't matter.

You could argue with some reason that nothing is new, ever, or that everything is new, always. That's philosophy and both viewpoints can be defended.

From a practical viewpoint, truth lies somewhere in between. Truth is not a binary concept, boolean logic be damned.

The Chinese may have come up with the printing press a while back, but it's only been about 10 years that most people can print decent color photos at home for a reasonable price.

Invention is nowhere and everywhere, depending on your criteria and frame of reference.

Share

answered Mar 13, 2009 at 10:27

community wiki
Sylver

1 +1. Take a look for instance at the iPad ;) See stackoverflow.com/questions/432922/… – VonC Apr 11, 2010 at 21:00

4 If only there was a fav. answer tag... if only there was an option to give 2 upvotes... – tshepang May 26, 2010 at 11:25

Great answer. Maybe we should be asking then, what *new ideas* have there been in the past 30 years (not new products/inventions). And since it's too hard to say whether or not they'll be "significant" or revolutionary before they're even built.... maybe we can speculate and then decide where to spend more energy. – mpen Jul 14, 2010 at 8:05

3 There have been countless amazing new ideas in the last 30 years, but there hasn't necessarily been time to see which ones matter. Pick any field of computing and just flick through the research released in the last year, and you'll find no shortage of new ideas, from small improvements to complete overhauls. However, the 1980s and before seem so revolutionary and packed because those ideas have now come to fruition and are ubiquitous, so they seem significant. We'll be having this same discussion in 30 years, when the ideas from now have boiled down into wonderful inventions. – Perrako Aug 6, 2010 at 1:00

@Mark: What qualifies as a "new idea"? Every idea, piece of code, biological organism has a context, which in one view would make nothing truly new. The problem with Prof. Kay's question is that the philosophy behind the fire that he and his colleagues at Xerox Parc (and Engelbart 10 years before him) lit under the tech/computer industry has been burning like an

uncontrolled fire and changed the world, the context. Truly new ideas out there have no impact so none of us have heard of them -- OSes written with proofs of their correctness and kernel security, non-ARM, non-x86 architectures, etc. – Jared Updike Mar 9, 2011 at 18:38

68

votes

Google's Page Rank algorithm. While it could be seen as just a refinement of web crawling search engines, I would point out that they too were developed post-1980.

Share

answered Jan 12, 2009 at 15:27

community wiki
Bill the Lizard

"Just a refinement" is often an oxymoron. In this case, the refinement is the technology. The internet was a much scarier place before google brought ought that page rank algorithm (and delivered the results quickly and without page clutter, and all the other dredge that we use to have to suffer through to use other search engines in the past). – David Berger Apr 25, 2009 at 22:39

19 i don't think you know what an oxymoron is. – Jason Aug 18, 2009 at 6:47

1 Do you remember altavista and that little unknown company: yahoo? – Esteban Küber Sep 24, 2009 at 12:56

@voyager: Hotbot and Lycos weren't bad, either. – Dean J Aug 4, 2010 at 14:09

2 @martin it's a **non-oxymoron oxymoron**. contradiction is in the definition: ninjawords.com/oxymoron – Jason Aug 30, 2010

at 3:03

66

votes

DNS, 1983, and dependent advances like email host resolution via MX records instead of bang-paths. **shudder**

Zeroconf working on top of DNS, 2000. I plug my printer into the network and my laptop sees it. I start a web server on the network and my browser sees it. (Assuming they broadcast their availability.)

NTP (1985) based on Marzullo's algorithm (1984). Accurate time over jittery networks.

The mouse scroll wheel, 1995. Using mice without it feels so primitive. And no, it's not something that Engelbart's team thought of and forgot to mention. At least not when I asked someone who was on the team at the time. (It was at some Engelbart event in 1998 or so. I got to handle one of the first mice.)

Unicode, 1987, and its dependent advances for different types of encoding, normalization, bidirectional text, etc.

Yes, it's pretty common for people to use all 5 of these every day.

Are these "really new ideas?" After all, there were mice, there were character encodings, there was network timekeeping. Tell me how I can distinguish between "new" and "really new" and I'll answer that one for you. My intuition says that these are new enough.

In smaller domains there are easily more recent advances. In bioinformatics, for example, Smith-Waterman (1981) and more especially BLAST (1990) effectively make the field possible. But it sounds like you're asking for ideas which are very broad across the entire field of computing, and the low-hanging fruit gets picked first. Thus is it always with a new field.

Share

answered Mar 5, 2009 at 16:27

community wiki
Andrew Dalke

63

votes

What about digital cameras?

According to Wikipedia, the first true digital camera appeared in 1988, with mass market digital cameras becoming affordable in the late 1990s.

Share

edited Jul 25, 2011 at 18:11

community wiki
3 revs, 3 users 80%
Domchi

> But the idea, the invention and the patents were there in the early 70's (See the section on "Early Development")
> – saschabeaumont Jan 15, 2009 at 5:06

10 Digital camera? One wonders, judging from up votes, what people understand today by the term "computing". – MaD70 Oct 31, 2009 at 11:59

1 Pictures is what modern consumer computing is based around. Without a webcam, a point-and-shoot or expensive SLR (for newspapers), modern consumers wouldn't really need computers. – Marius Mar 6, 2010 at 13:00

14 @MaD70: I guess you're not so much into photography, are you? Just to name a few: automatic face recognition, autofocus, "panoramic mode", automatic white balance ... it definitely falls into computing. – nico May 25, 2010 at 17:55

6 Sorry, the first prototype digital camera was made by Kodak in 1975 apparently. pluggedin.kodak.com/post/?ID=687843 – Mark Ransom Aug 2, 2010 at 20:33

50 votes

Modern shading languages and the prevalence of modern GPUs.

The GPU is also a low cost parallel supercomputer with tools like CUDA and OpenCL for blazing fast **high level** parallel code. Thank you to all those gamers out there driving down the prices of these increasingly impressive hardware marvels. In the next five years I hope every new computer sold (and iPhones too) will have the ability to run massively parallel code as a basic assumption, much like 24 bit color or 32 bit protected mode.

Share

answered Jan 16, 2009 at 22:24

community wiki
Jared Updike

Try it. You won't like it. Multi-core systems are much faster for most real-world problems. YMMV. Good for graphics, and not much else. – xcramps Aug 26, 2009 at 16:08

There's a reason they're called GPUs and not PPUs... (Parallel processing units). Most people don't have the patience and/or skills to write good code for them. Though there is an increasing amount of research projects that are exploring using GPUS for non graphics purposes. – RCIX Sep 25, 2009 at 22:11

3 I tried it. I liked it. I can run all of my Matlab code on the GPU, with no source code modifications apart from a few typecast changes which you can do with a search'n'replace. Google "Matlab GPU computing". – Contango Jul 7, 2010 at 18:49

3 I agree with the OP. The programmable pipeline, while something we now might take for granted, completely changed the world of graphics, and it looks like it might continue changing other parts of the programming world. @xcramps: I think I'm missing something; last I checked, GPUs were multi-core systems. Just with a lot more cores. Kind of like... supercomputers. But I guess those aren't really being used for anything in the real-world... – Perrako Aug 6, 2010 at 0:50

Two years later (not 5 as I said) and mobile devices shipping with OpenCL are on the horizon: macrumors.com/2011/01/14/… – Jared Updike Apr 8, 2011 at 18:13

43 votes

JIT compilation was invented in the late 1980s.

Share

answered Jan 11, 2009 at 15:58

community wiki
Jasper Bekkers

Well, the whole work on the implementation of the Self language (which was completely JIT-compiled) was amazing, and its usefulness can be seen today for Javascript inside Google V8. And that's from the late '80s and early '90s. – Blaisorblade Jan 11, 2009 at 22:54

7 I first saw this idea in the last chapter of John Allen's book Anatomy of Lisp, published in the 70s. He gave a ref to a 70s PhD thesis as the originator. – Darius Bacon Jan 12, 2009 at 1:39

Maybe we should refine it to "profile based adaptive JIT compilation" such as the Self JIT or Suns' Java Hotspot – kohlerm Jan 12, 2009 at 14:54

34 One of the PhD theses in the early 1970s which had JIT was Jim Mitchell's at CMU -- he later went to PARC – Alan Kay Jan 15, 2009 at 2:48

2 Nori, K.V.; Ammann, U.; Jensen; Nageli, H. (1975). The Pascal P Compiler Implementation Notes. Zurich: Eidgen. Tech. Hochschule. (Thanks wikipedia) – Arafangion Jul 19, 2010 at 10:35

42 votes

To address the two questions about "Why the death of new ideas", and "what to do about it"?

I suspect a lot of the lack of progress is due to the massive influx of capital and entrenched wealth in the industry. Sounds counterintuitive, but I think it's become

conventional wisdom that any new idea gets one shot; if it doesn't make it at the first try, it can't come back. It gets bought by someone with entrenched interests, or just FAILs, and the energy is gone. A couple examples are tablet computers, and integrated office software. The Newton and several others had real potential, but ended up (through competitive attrition and bad judgment) squandering their birthrights, killing whole categories. (I was especially fond of Ashton Tate's Framework; but I'm still stuck with Word and Excel).

What to do? The first thing that comes to mind is Wm. Shakespeare's advice: "Let's kill all the lawyers." But now they're too well armed, I'm afraid. I actually think the best alternative is to find an Open Source initiative of some kind. They seem to maintain accessibility and incremental improvement better than the alternatives. But the industry has gotten big enough so that some kind of organic collaborative mechanism is necessary to get traction.

I also think that there's a dynamic that says that the entrenched interests (especially platforms) require a substantial amount of change - churn - to justify continuing revenue streams; and this absorbs a lot of creative energy that could have been spent in better ways. Look how much time we spend treading water with the newest iteration from Microsoft or Sun or Linux or Firefox, making changes to systems that for the most part work fine already. It's not because they are evil, it's just built into the industry. There's no such thing as Stable Equilibrium; all the feedback mechanisms are positive, favoring change over stability.

(Did you ever see a feature withdrawn, or a change retracted?)

The other clue that has been discussed on SO is the Skunkworks Syndrome (ref: Geoffrey Moore): real innovation in large organizations almost always (90%+) shows up in unauthorized projects that emerge spontaneously, fueled exclusively by individual or small group initiative (and more often than not opposed by formal management hierarchies). So: Question Authority, Buck the System.

Share

edited Oct 14, 2009 at 23:42

community wiki
4 revs, 2 users 94%
le dorfier

I loved Framework, and you can still buy it, but it's expensive. – Norman Ramsey Jan 12, 2009 at 3:05

7 It's always easier to have new ideas in a new area of knowledge, so a very large number of the important ideas came about in the 1950s and 1960s. We just can do most of them a whole lot better now. – David Thornley Jan 13, 2009 at 21:34

6 I think this reply and the comments are very well put. – Alan Kay Jan 18, 2009 at 3:02

5 @David: "whole lot better now". And cheaper. And smaller. Which enables new ways of doing *other* things better. E.g. 10 songs -> 1,000 songs -> 1,000 albums in my pocket, sure it is a matter of degree but it changes everything, even if someone

back before 1980 showed it could be done, in theory, on a giant mainframe. The pieces may have been there but some inventions, like the iPod, are more than the sum of the parts.
– Jared Updike Jul 24, 2009 at 0:05

@Alan Kay, @le dorfier : it seems to me that one partial counter-example with that entrenched attitude is Donald Knuth decision's to asymptotically increment TeX version number toward pi. But he is an institution, not a corporation. I am appalled by mozilla and google race for version number 100 of their browsers while intelligent and creative standardization as well as innovation in data access and transformation is lagging.
– ogerard May 4, 2011 at 14:57

36

votes

One thing that astounds me is the humble spreadsheet. Non-programmer folk build wild and wonderful solutions to real world problems with a simple grid of formula. Replicating their efforts in desktop application often takes 10 to 100 times longer than it took to write the spreadsheet and the resulting application is often harder to use and full of bugs!

I believe the key to the success of the spreadsheet is automatic dependency analysis. If the user of the spreadsheet was forced to use the observer pattern, they'd have no chance of getting it right.

So, the big advance is automatic dependency analysis. Now why hasn't any modern platform (Java, .Net, Web Services) built this into the core of the system? Especially in a day and age of scaling through parallelization - a graph of dependencies leads to parallel recomputation trivially.

Edit: Dang - just checked. VisiCalc was released in 1979 - let's pretend it's a post-1980 invention.

Edit2: Seems that the spreadsheet is already noted by Alan anyway - if the question that bought him to this forum is correct!

Share

edited May 23, 2017 at 11:53

community wiki
3 revs
Daniel Paull

5 I had thought of this answer, but Visicalc was released just a smidgin before the 1980 deadline. (en.wikipedia.org/wiki/VisiCalc) – Oddthinking Jan 11, 2009 at 14:02

but this reveals an interesting point: just presenting a simple way to display and manipulate data created a incredibly useful class of tools. is there some other 'enabling' idea like this? do we need one? i think so. – Javier Jan 11, 2009 at 15:27

See also: stackoverflow.com/questions/357813/… – splattne Jan 11, 2009 at 15:58

I agree wholeheartedly. Automatic dependency analysis could be and should be a part of modern programming languages. – Jesse Pepper Jan 12, 2009 at 4:14

1 @hasen j: Excel is a spreadsheet By the way there are modern platforms that keeps dependencies between calculations - for example Haskel (Excel and functional languages have much in common - for example pure functions and lazy evaluation).

Excel is just much more intuitive than Haskel :) – ajuc Oct 26, 2011 at 11:46

36

votes

Software:

- Virtualization and emulation
- P2P data transfers
- community-driven projects like Wikipedia, SETI@home ...
- web crawling and web search engines, i.e. indexing information that is spread out all over the world

Hardware:

- the modular PC
- E-paper

Share

answered Jan 11, 2009 at 16:58

community wiki
mjy

6 Virtualization was implemented on VM/CMS in 1972. What do you mean by "the modular PC"? – Hudson Jan 11, 2009 at 22:39

I think that by "the modular PC" he means that anyone can buy almost interchangeable components and build their own computer. – NGittlen Jan 12, 2009 at 21:40

14 P2P was invented at Xerox PARC in the 70s -- the Altos were all P2P and the file resources and printers and "routers" were all P2P Altos – Alan Kay Jan 15, 2009 at 2:49

1 I saw "E-paper" and thought, what? how does that effect me day to day. I'm glad it exists but e-Readers are not very important technologies on a widespread basis, compared to say, the cellphone or iPod. – Jared Updike Jan 16, 2009 at 22:05

3 I'd like to point out that about 40-50 years ago everyone was still doing math on paper mainly and saying the same about computers... – RCIX Oct 28, 2009 at 4:50

36

votes

The rediscovery of the monad by functional programming researchers. The monad was instrumental in allowing a pure, lazy language (Haskell) to become a practical tool; it has also influenced the design of combinator libraries (monadic parser combinators have even found their way into Python).

Moggi's "A category-theoretic account of program modules" (1989) is generally credited with bringing monads into view for effectful computation; Wadler's work (for example, "Imperative functional programming" (1993)) presented monads as practical tool.

Share

answered Jan 12, 2009 at 0:24

community wiki
solidsnack

35

votes

Shrinkwrap software

Before 1980, software was mostly specially written. If you ran a business, and wanted to computerize, you'd typically get a computer and compiler and database, and get your own stuff written. Business software was typically written to adapt to business practices. This is not to say there was no canned software (I worked with SPSS before 1980), but it wasn't the norm, and what I saw tended to be infrastructure and research software.

Nowadays, you can go to a computer store and find, on the shelf, everything you need to run a small business. It isn't designed to fit seamlessly into whatever practices you used to have, but it will work well once you learn to work more or less according to its workflow. Large businesses are a lot closer to shrinkwrap than they used to be, with things like SAP and PeopleSoft.

It isn't a clean break, but after 1980 there was a very definite shift from expensive custom software to low-cost off-the-shelf software, and flexibility shifted from software to business procedures.

It also affected the economics of software. Custom software solutions can be profitable, but it doesn't scale. You can only charge one client so much, and you can't sell the same thing to multiple clients. With shrinkwrap software, you can sell lots and lots of the same thing, amortizing development costs over a very large sales base. (You do have to provide support, but that scales. Just consider it a marginal cost of selling the software.)

Theoretically, where there are big winners from a change, there are going to be losers. So far, the business of software has kept expanding, so that as areas become commoditized other areas open up. This is likely to come to an end sometime, and moderately talented developers will find themselves in a real crunch, unable to work for the big boys and crowded out of the market. (This presumably happens for other fields; I suspect the demand for

accountants is much smaller than it would be without QuickBooks and the like.)

Share

answered Jan 12, 2009 at 15:21

community wiki
David Thornley

Turbo Pascal & C at $100 on a MS-DOS system provoked a $100 price tag on a C complier for a C/PM from others. – CW Holeman II Feb 5, 2011 at 15:29

Sorry, pretty sure Microsoft was selling shrink-wrap software before 1980. Not that they were the only ones. – Mark Ransom Aug 10, 2011 at 4:49

34

votes

Outside of hardware innovations, I tend to find that there is little or nothing new under the sun. Most of the really big ideas date back to people like von Neumann and Alan Turing.

A lot of things that are labelled 'technology' these days are really just a program or library somebody wrote, or a retread of an old idea with a new metaphor, acronym, or brand name.

Share

answered Jan 11, 2009 at 13:44

community wiki

frankodwyer

3 You can't see the forest since all the trees are in the way... The building blocks are much the same, but the result has changed/evolved. – Johan Mar 14, 2009 at 11:16

8 ...That's the definition of technology ;) "the practical application of knowledge..." – steamer25 Jul 22, 2009 at 19:25

1 I agree it's time for the next big thing. I'm tired of all the re-packing of things forgotten from the past as something new. Like Javascript = AJAX. – James Jan 1, 2010 at 21:33

32 votes

Computer Worms were researched in the early eighties of the last century in the Xerox Palo Alto Research Center.

From John Shoch's and Jon Hupp's **The "Worm" Programs - Early Experience with a Distributed Computation**" (Communications of the ACM, March 1982 Volume 25 Number 3, pp.172-180, march 1982):

> In The Shockwave Rider, J. Brunner developed the notion of an omnipotent "tapeworm" program running loose through a network of computers - an idea which may seem rather disturbing, but which is also quite beyond our current capabilities. The basic model, however, remains a very provocative one: a program or a computation that can move from machine to machine, harnessing resources as needed, and replicating itself when necessary.

> In a similar vein, we once described a computational model based upon the classic science-fiction film, *The Blob*: a program that started out running in one machine, but as its appetite for computing cycles grew, it could reach out, find unused machines, and grow to encompass those resources. In the middle of the night, such a program could mobilize hundreds of machines in one building; in the morning, as users reclaimed their machines, the "blob" would have to retreat in an orderly manner, gathering up the intermediate results of its computation. Holed up in one or two machines during the day, the program could emerge again later as resources became available, again expanding the computation. (This affinity for nighttime exploration led one researcher to describe these as "vampire programs.")

Quoting Alan Kay: "The best way to predict the future is to invent it."

Share

edited Jan 11, 2009 at 16:57

community wiki
2 revs
splattne

@Bobby: According to Computer security basics, 2006, Lehtinen, Russell & Gangemi, this work began "around 1980".

So if you disregard the sci-fi precursors, this counts.
– Charles Stewart Jul 14, 2010 at 8:38

31

votes

Better user interfaces.

Today’s user interfaces still suck. And I don't mean in small ways but in large, fundamental ways. I can't help but to notice that even the best programs still have interfaces that are either extremely complex or that require a lot of abstract thinking in other ways, and that just don't approach the ease of conventional, non-software tools.

Granted, this is due to the fact that software allows to do so much *more* than conventional tools. That's no reason to accept the status quo though. Additionally, most software is simply not well done.

In general, applications still lack a certain “just works” feeling are too much oriented by what can be done, rather than what *should* be done. One point that has been raised time and again, and that is still not solved, is the point of saving. Applications crash, destroying hours of work. I have the habit of pressing Ctrl+S every few seconds (of course, this no longer works in web applications). Why do I have to do this? It's mind-numbingly stupid. This is clearly a task for automation. Of course, the application also has to save a diff for *every* modification I make (basically an infinite undo list) in case I make an error.

Solving this probem isn't even actually hard. It would just be hard to implement it in *every* application since there is

no good API to do this. Programming tools and libraries have to improve significantly before allowing an effortless implementation of such effords across all platforms and programs, for all file formats with arbitrary backup storage and no required user interaction. But it is a necessary step before we finally start writing “good” applications instead of merely adequate ones.

I believe that Apple currently approximates the “just works” feeling best in some regards. Take for example their newest version of iPhoto which features a face recognition that automatically groups photos by people appearing in them. *That* is a classical task that the user does not *want* to do manually and doesn't *understand* why the computer doesn't do it automatically. And even iPhoto is still a very long way from a good UI, since said feature still requires ultimate confirmation by the user (for each photo!), since the face recognition engine isn't perfect.

Share

answered Jan 11, 2009 at 23:10

community wiki
Konrad Rudolph

3 Google's Picasa has had that for a while. In fact, picasa has so many other features that are slowly crawling into iPhoto.
– akshaykarthik Jun 7, 2010 at 16:25

30 HTM systems (Hiearchical Temporal Memory).

votes

A new approach to Artifical Intelligence, initiated by Jeff Hawkins through the book "On Intelligence".

Now active as a company called Numenta where these ideas are put to the test through development of "true" AI, with an invitation to the community to participate by using the system through SDKs.

It's more about building machine intelligence from the ground up, rather than trying to emulate human reasoning.

Share

edited Feb 20, 2012 at 0:20

community wiki
3 revs, 2 users 74%
sharkin

11 When they do something interesting, I will be the first and loudest leader of the applause – Alan Kay Jan 15, 2009 at 2:54

@AlanKay Well, it seems that HTM is already used in real products. For example: vitamind inc allows you to recognize objects or people in CCTV footage. Vitamindinc is entirely powered by HTM. In this paper, you can see that HTM actually beats the SVM approach for handwritten recognition on datasets such as USPS. The fact that it is at the same time biologically inspired and of high practical value blows my mind. I think you can start applauding right now. – Benjamin Crouzier Feb 20, 2012 at 0:13

26

votes

The use of **Physics in Human Computer interaction** to provide an alternative, understandable metaphor. This combined with gestures and haptics will likely result in a replacment for the current common GUI metaphor invented in the 70's and in common use since the mid to late 80's.

The computing power wasn't present in 1980 to make that possible. I believe Games likely led the way here. An example can easily be seen in the interaction of list scrolling in the iPod Touch/iPhone. The interaction mechanism relies on the intuition of how momentum and friction work in the real world to provide a simple way to scroll a list of items, and the usability relies on the physical gesture that cause the scroll.

Share

edited Jan 11, 2009 at 19:21

community wiki
2 revs
Steve Steiner

The earliest example I can think of was Randy Smith's Alternate Reality Kit, built in Smalltalk-80 at PARC in '86 or '87. You could implement new objects with a physical metaphor. Every object had location, mass, momentum, and a pop-up menu for interacting with it via its message interface. – PanCrit May 8, 2009 at 15:52

25

votes

I believe Unit Testing, TDD and Continuous Integration are significant inventions after 1980.

Share

edited Jan 11, 2009 at 13:56

community wiki
2 revs, 2 users 67%
krosenvold

2 Testing first was a very old method that has be ressurected i believe. – Johnno Nolan Jan 11, 2009 at 19:14

That's a software engineering thing, not a "computing" thing – SquareCog Jan 11, 2009 at 19:26

7 I'd agree with John, for instance Brooks describes a test-first approach in The Mythical Man-Month (1975). – Fabian Steeg Jan 12, 2009 at 10:38

28 Continuous integration was first done seriously in BBN Lisp 1.85 in the late 60s, which became Interlisp at PARC. Smalltalk at PARC in the 70s was also a continuous integration system. – Alan Kay Jan 15, 2009 at 2:51

3 TDD only became generally useful when computers got fast enough to run small tests so quickly that you are willing to run them over & over. – Jay Bazuzi Jan 16, 2009 at 18:46

25

votes

Mobile phones.

While the first "wireless phone" patent was in 1908, and they were cooking for a long time (0G in 1945, 1G launched

in Japan in 1979), modern 2G digital cell phones didn't appear until 1991. SMS didn't exist until 1993, and Internet access appeared in 1999.

Share

answered Jan 11, 2009 at 21:32

community wiki
Domchi

4 Japan in 1979, that's pre 1980. We're looking for new inventions - think research labs, universities, practical demonstrations of patent applications... all which will predate the mass-market availability by a number of years. – saschabeaumont Jan 15, 2009 at 5:14

1 The difference between 1G and 2G is about as big as difference between analog and digital computer. I think 2G (1991) deserves the status of "new" invention. – Domchi Jan 15, 2009 at 23:16

And is dependent on powersave technologies and good batteries. – Johan Mar 14, 2009 at 11:22

23 votes

I started programming Jan 2nd 1980. I've tried to think about significant new inventions over my career. I struggle to think of any. Most of what I consider significant were actually invented prior to 1980 but then weren't widely adopted or improved until after.

1. Graphical User Interface.
2. Fast processing.

3. Large memory (I paid $200.00 for 16k in 1980).
4. Small sizes - cell phones, pocket pc's, iPhones, Netbooks.
5. Large storage capacities. (I've gone from carrying a large 90k floppy to an 8 gig usb thumb drive.
6. Multiple processors. (Almost all my computers have more than one now, software struggles to keep them busy).
7. Standard interfaces (like USB) to easily attach hardware peripherals.
8. Multiple Touch displays.
9. Network connectivity - leading to the mid 90's internet explosion.
10. IDE's with Intellisense and incremental compiling.

While the hardware has improved tremendously the software industry has struggled to keep up. We are light years ahead of 1980, but most improvements have been refinements rather than inventions. Since 1980 we have been too busy applying what the advancements let us do rather than inventing. By themselves most of these incremental inventions are not important or powerful, but when you look back over the last 29 years they are quite powerful.

We probably need to embrace the incremental improvements and steer them. I believe that truly original ideas will probably come from people with little exposure to computers and they are becoming harder to find.

Share

edited Jan 11, 2009 at 19:36

community wiki
4 revs
bruceatk

"original ideas will probably come from people with little exposure to computers" so true. and even more sad since most of that 'numbing' exposure is windows/office. – Javier Jan 11, 2009 at 15:34

1 Some dates for earlier inventions: Engelbart's GUI was demoed in 1968 and the Xerox PARC Alto was developed in 1973. Multiple CPUs are new on the desktop, but not in the machine room -- the VAX cluster was first available in 1978. – Hudson Jan 11, 2009 at 22:45

You were programming before I was born. Dang I have a long way to go. – Kieran Senior Jan 12, 2009 at 12:58

Ouch. I didn't start until I was 26, now I really feel old. :) – bruceatk Jan 13, 2009 at 12:42

Did you factor in inflation for that $200 16k memory chip? – Tim Tonnesen Jan 19, 2009 at 18:16

22

votes

Nothing.

I think it's because people have changed their attitudes. People used to believe that if they would just find that "big idea", then they would strike it rich. Today, people believe that it is the execution and not the discovery that pays out the most. You have mantras such as "ideas are a dime a

dozen" and "the second mouse gets the cheese". So people are focused on exploiting existing ideas rather than coming up with new ones.

Share

answered Jan 11, 2009 at 19:08

community wiki
Edward Basena

3 So many of the existing ideas just haven't been implemented yet. – Breton Jan 12, 2009 at 22:56

3 There are always a few lunatics that will come up with new ideas, they just can't help it ;-) – Johan Mar 14, 2009 at 11:26

But they're lunatics, so they can't sell their ideas because nobody will listen to them. – Adam Jaskiewicz Apr 28, 2009 at 17:56

Ideas are more the province of artists. Practical implementation is what we guys do. Looking at engineers for brand new ideas is kind of fishing in the wrong pond. For bright new ideas, read Sf and figure out how this stuff could be done (I figure a lot of it could be done). However, implementing a wild idea can take years. Artists can get away selling ideas and dreams, but engineers are expected to come up with products... and they have to eat too. – Sylver Jul 9, 2010 at 10:12

16 votes

Open Source community development.

Share

answered Jan 11, 2009 at 14:13

community wiki
sharkin

2 Actually, the SIG/M user group disks kind of pre-date what we now call open source. It contained hundreds of disks (of the floppy variety) full of CP/M software, much of it open source (although the term "open source" didn't exist then).
– Mike Thompson Jan 12, 2009 at 11:26

2 In the sense of open cooperation and development among people who had access to a computer, it's much like the IBM user groups in the 1960s. It's just that more people can afford computers now. – David Thornley Jan 13, 2009 at 21:32

2 Agree with david, it's only become more prominent now as computers have moved from the education and scientific areas into the business world, this gave rise to "closed source" software, confusing licenses. It was always there, it just didn't need a name until the lawyers got involved. – saschabeaumont Jan 15, 2009 at 5:11

1 Yes, I must also agree with David here. Open Source is way earlier than 1980. Predates it by at least 20 years. I thought it was the 1950s not the 1960s though. – Brendan Enrick Jan 16, 2009 at 13:43

16
votes

The iPad (released April 2010): surely such a concept is *absolutely* revolutionary!

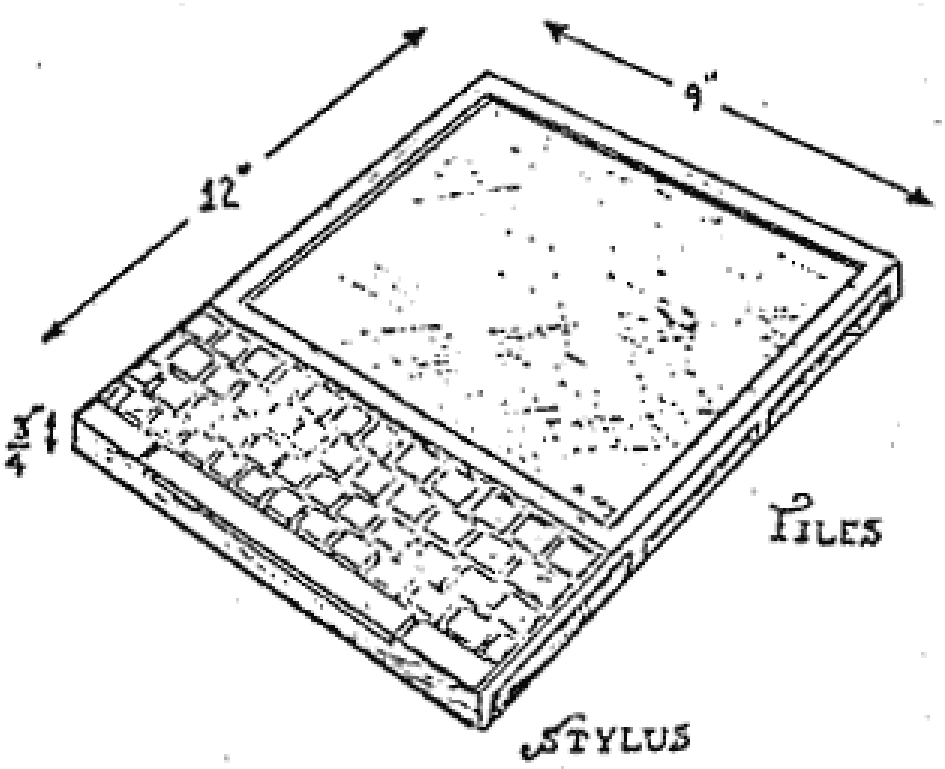
alt text http://www.ubergizmo.com/photos/2010/1/apple-ipad//apple-ipad-05.JPG

No way Alan Kay saw *that* coming from the 1970's! Imagine such a "personal, portable information

manipulator"...

...

Wait? What!? The Dynabook you say?



Thought out by Alan Kay as early as 1968, and described in great details in this 1972 paper??

NOOOooo$_{ooooo....}$

Oh well... never mind.

Share

edited Feb 8, 2017 at 14:23

community wiki
3 revs, 2 users 77%
VonC

See stackoverflow.com/questions/432922/… for a larger context illustrated by this answer. – VonC Apr 11, 2010 at 21:02

Well surely the idea was around before (for example apple newton); however the technology now proceeded so far that it's possible to build a cheap (and great) consumer device. – Nils Apr 29, 2010 at 11:04