Hello there! Welcome to another episode of Q&A about history of science and technology. And I see a whole bunch of questions here.
See where to start.
Well, someone here is noting that they know that I spent some time recently in the UK, and they're asking, did I learn anything interesting about history of science and technology there?
Yeah, I did, actually. I grew up in the UK, but I haven't lived there for, what is it now?
Well, since, 1978.
so…
I… and I don't visit there very often. I was there briefly in 2019, and then before that in 2012, and before that, I don't even remember when. It's been, it's been a while.
So, I did a very intense trip, visiting lots of kinds of things, and giving lots of talks, and all those kinds of things, and had the pleasure of reconnecting with
all sorts of people that I have known in the past, or have known for a long time by email, but not in person. I think my record for reconnecting with someone was somebody that I hadn't seen in person for 53 years.
And it was, nice to see them again. But I did, have a chance to, sort of, explore some history kinds of things. Let's see, where to start?
Well… I,
I dropped in at the Bodleian Library in Oxford, one of the oldest libraries in the world, collects many, many archives. A person I know had arranged for us to take a look at the archive of a chap called Christopher Strachey, who was an early computation pioneer.
involved with the programming research group in Oxford, and so I spent a couple of hours going through a few boxes of Christopher Strachey documents. I can tell you a few things I found, which I thought were kind of interesting. So…
In… Strache was kind of involved in early computer languages. Probably most notably, he was involved with a language called CPL.
which either stood for Cambridge Programming Language or Christopher Programming Language.
And he was involved in writing the specifications for this around 1960, 1961-type time frame, and that language
was a very kind of… it had… it had a bunch of constructs that are very familiar from languages today. It was after Algol, which was around 1958 to 1960. That was… ALGOL was a kind of theoretically specified computer language. This is after Fortran. Fortran was earlier, was implemented earlier in the… in 1957-8 kind of
time frame, I think.
Then, and by the way, 1956 was when the Bacchus Nawa form, the kind of recursive specification of the syntax of languages, was introduced. So Fortran was earlier, COBOL was earlier.
the theoretical specification of Lisp was around 959, but wasn't implemented at that time. So this is kind of in the… and then ALGOL was kind of brought in as a kind of a, we'll have the… the theorists come and design the perfect language, a rather European,
Oriented language, and, had some interesting ideas, never was fully implemented.
kind of one of those stories of a very complicated specification that didn't really ever get completely worked through. I mean, the same thing happened in subsequent years with the ADA language, that was from the early 1980s. That was a language that the U.S. Department of Defense wanted to specify as the language to have all software built in.

And it became this unbelievably complicated committee design thing, which was never implemented fully, and people always sort of cheated in the end, and wrote their programs in whatever, Pascal or something at the time.
And, translated it into, into sort of a subset of Ada in some way, just to satisfy requirements. But anyway, so back 1961, 1962 type time frame, Christopher Strachie, involved in the creation of this language called CPL,
Now, what was perhaps interesting historically about CPL was that it turned into a language called BCPL,
And BCPL spawned a language called B,
And B spawned a language called C.
which is the C that we know about, that, you know, Ken Thompson, Dennis Ritchie, people at Bell Labs developed in the, in the 1970s, around, I think…
Late… yeah, around 1970.
So, this CPL language kind of eventually begat C.
what did CPL have in it? I looked through a bunch of the design documents for it. There was a lot of, kind of, committee discussion of how it should work and things like what character set it should use, because at that time.
there was a, ASCII, the standard character set that got used in the… there's a US-based character set.
hadn't really been standardized yet, it was beginning to be talked about. Perhaps more standard was Epsodyc, which was the IBM character set, and this was a character set for CPL, which, for example, had pound signs in the sense of British
pound sterling signs, not hash signs, included in it, and so on. So that was an example of one corner of the sort of design discussion. Another corner that I found interesting was the discussion of types. So one thing I've been curious about historically is where did types in computer languages come from? I've always thought they were not the best of ideas, but the question was, did types in computer languages come from the same place
that types and mathematical logic came from. So just to say something about that history.
Back around 1910, Bertrand Russell was thinking about, kind of, foundations of mathematics, and whether you could, kind of, construct mathematics
from Logic and so on, that's what led to the Whitehead Russell Principia Mathematica project.
But there were all kinds of, paradoxes that seemed to arise in things. For example, set theory, you know, is the set of all sets a set, and all these kinds of questions.
And, this led Russell to kind of introduce this sort of semi-kluge
of this idea of types, so that you could get out of this, is this thing a member of itself, and all those kinds of questions, because you say, well, they're different types of thing, and so that isn't a meaningful question to ask. Only things whose types match, you can discuss those questions for.
So it was sort of a theoretical kludge that was introduced.
Now, subsequently, the idea of types in computer languages has to do with, well, we're storing an integer, we're storing a floating point number, we're storing a character, a string, whatever.
We're storing different kinds of things, and the machine code instructions for dealing with those things are different, and so we should distinguish those as different types of data.
So I had sort of assumed that this notion of types of data had arisen from the sort of theoretical considerations of Russell and so on on the theory of types. The theory of types had kind of developed theoretically. It was not particularly prominent, but it developed theoretically. People like Godel certainly mentioned types. Godel has this famous quote.

in his original paper on Godel's theorem, where he basically says that there's this Godel's theorem thing which limits the way that you can kind of mechanically derive theorems of mathematics. But he imagines that,
There will be kind of a way that the human mind can access
the truths of mathematics in a way that sort of evades this mechanical restriction that Godel's theorem places, and he has this footnote that says that perhaps that will happen because somehow the operation of the mind has… involves an infinite hierarchy of types ramified into the transfinite.
A somewhat obscure footnote, but referring back to Russell's idea of types. By the way, this notion that,
That you can get, sort of, direct access from the human mind to the truths of mathematics is one of these notions that has a habit of never dying.
And, you know, people who do mathematics, and have done it their whole lives feel that they have sort of an intuition that allows them direct access to the truths of mathematics, not going through sort of the machinery of mathematics.
And I think that's a, they're not wrong that the description of mathematics as operating in practice through that machinery is not really the right description. It's a bad way of describing what mathematicians do, but the idea that there is sort of an evasion of sort of what is computational, a way to directly access mathematics that goes beyond what is sort of computationally possible.
I don't believe that that's possible, partly because I think that I now understand what… how the universe is constructed, and that the universe is fundamentally computational, and that therefore there's no sort of hyper-computational miracle
that's possible. But actually, when I was in the UK, I spent an afternoon with Roger Penrose, who, who certainly still is of the… is of the view that there are sort of access… that there is a notion of direct access to the truths of mathematics, not going through, kind of.
these computational steps that are subject to Godel's theorem and so on.
And he has various kinds of physics ideas about, sort of, quantum gravity in the brain and so on, which I don't think are right. But it's an interesting discussion to have. I mean, my own feeling is that these ideas that there's sort of more to brains than any known physics
It's partly true, in the sense that physics, in its reductionist form, can't describe… doesn't have a direct way to usefully describe how brains completely work.
But the idea that there's sort of the need for an extra sort of spark of underlying physics to explain brains, I don't believe. I think that the thing that people don't tend to have intuition about is kind of the fact that even if the rules are simple, even if you know all the rules, it still can be the case that the overall behavior of the system can be very complicated, very unknowable from those underlying rules. This is the whole sort of
story of computational irreducibility, and I have to say, whenever I sort of run across people whose intuition doesn't extend to those kinds of things, it's like, you've got to actually do some computer experiments. They will surprise you all the time. I've been doing them for 45 years, and they still surprise me every day.
when I do them. I was doing a bunch just last night, and a lot of things were like, I think it's gonna go this way, but no, sort of, what's actually out there in the computational universe is more surprising, more unexpected.
than even my long intuition about these things has led to. In any case, that was sort of a digression from talking about types and computer languages, because in looking at Christopher Stracci's stuff.

I was able to see his discussion of types in CPL, and types in CPL, he talks about it just like, well, we can store integers in this location in computer memory, we can store characters, and so on. It's really, he just means a type of data
And there's no reference to anything theoretical. What's interesting is, later on in his work.
in the 1970s, one starts seeing, kind of, reference to lambda calculus and other ideas from theoretical computer science, but they're simply not present when he's talking about the design of this computer language.
And, in fact, I think that the introduction of, kind of, these more theoretical ideas into what Christopher Strache worked on was the result of his interaction with Dana Scott, somebody I've known for a great many years.
who, came to be a philosophy professor in Oxford and interacted with that crowd.
And that led to a thing called the Scott-Straiche Theory of Denotational semantics, which is, and this was sort of in the later work of Christopher Strache, long after he'd worked on, kind of the, The, the, the, the, the construction of this, of this computer language.
By the way, I should say, so in other words, his… Christopher Strache was talking about types. just as type is a word you might use to describe what kind of data is sitting in that location in computer memory, that's, I think, how it got introduced. And the fact that it could be related to Russell's theory of types and the whole story of mathematical logic was
a later kind of thing, and maybe has never even really been connected. I think Dana Scott himself made a bunch of those connections.
I mean, I will say in the later theory of these things, the theory of types
has become something that is sort of an attempt to computerize, to have a computational kind of formalization of certain kinds of ideas in mathematics. It's become… it had a period of popularity, particularly around this thing called homotopy type theory, that was about 10 years ago now.
That, was kind of a… essentially, when one thinks about how mathematics is set up, and you're just describing mathematics, and you're talking about numbers, and you're talking about functions, you're talking about curves, talking about all these kinds of things, there's sort of a question, what are these different kinds of things you're talking about?
And back in the 1940s, category theory was developed as a way to sort of organize the kinds of things one's talking about. It arose first in algebraic topology, where there was sort of a giant zoo of different kinds of things you might be talking about, and it sort of gave a way to organize the connections between those things.
But then category theory has been applied to many other kinds of things, and it's sort of had this… this merger with type theory, because type theory is, again, talking about the kinds of things you can be dealing with.
Category theory is talking a bit about the connections between things. Type theory is more like the things, and then there are operations on those things.
In any case, that's… so I think my conclusion from looking at these papers is that types in computer languages just didn't come from the same place as this sort of theoretical idea of types. So something I've long believed to be the case about history of science or history of technology actually isn't the case. These probably came from different places.
Now, another thing that I learned looking at Christopher Straitree's documents was something from around 1960. There was a big push

to have… introduce school kids in England to computers. And in fact, there was a whole plan to have a truck that would drive around the country with a computer on it, kind of showing, kind of a… it was a computers in schools initiative.
Although it seemed to mostly be about one computer driving around to different schools and introducing
kids to computers, and there's a whole bunch of stuff that was written, and some whole government committee that was working on this, talking about, kind of, how… why it would be relevant to have
you know, introduce schoolchildren to computers, and how even the schoolchildren who weren't that technical might find computers useful. And it said, we want a plan
that will work in 1972, just as it works in 1962. They were sort of trying to plan for the future and imagining what computers would be like in the future, but actually, if you read the
What they wrote, it's really very similar to what people might write today to give, sort of, the minimal pitch for, kind of, why kids should learn about computers, even though most people had never seen a computer at that time.
Well, here's what I learned and learned. So, Christopher Stracci, before he was involved with all these things like programming research groups and all this kind of thing, he'd been a schoolteacher. He was a teacher at a school called Harrow. He had gone to Cambridge University.
for college, and he came from a distinguished British family that had produced various literary folk and so on.
But, he, after he finished in Cambridge, ended up becoming a teacher at a school called Harrow, in the
in the… in the ways of British… Sort of,
tradition or something, there are some particularly famous British so-called public schools. They're called public schools because back when they were founded in the 1400s and such, most, sort of, people who were getting educated were getting educated at home rather than in public, so to speak.
at an outside school. So, confusingly, schools which are by American… in the American nomenclature, what we call private schools, are called public schools in the UK, at least the sort of… the fancy ones that people pay a lot for are called public schools, even though they're not public in the sense that the government is paying for them. In fact, the government seems to be progressively, kind of attacking them, which is,
Different… a different thing. But in any case, the,
So the schools, there's a, sort of well-known schools are Eaton, which happens to be the school that I went to, which was founded around 1450, I think, and, had,
It had originally Henry VI,
was… had provided for 70 scholars of the college, as he called it, who were, you know, kids who would be sort of paid for, by some endowment, and, in the… many, many years later.
I was one such kid.
But,
In any case, so Eaton was one sort of notable public school, one that was actually, I think, earlier than it was Winchester. Winchester sort of has the reputation of being the school that lots of science types and so on go to. Eaton has kind of a reputation for being a place that sort of leaders, politicians, folks like that.
go to. And there's a third such school, which is Harrow.

which is more like a Eaton-like school, and like it's famous for being where Winston Churchill went, and so on. Harrell was the school that Christopher Strachie taught at.
Okay, while he was teaching at Harrow, one of his sort of big breaks was having heard
a… a program on the BBC, a radio program, by Alan Turing, and about,
And this must have been in the 1950s, because Turing died in 1954, so I think this might have been 1951,
And he wrote to Turing, and the letter that he wrote to Turing is in the archive, you know, Dear Turing, and it goes on for 3 or 4 pages, about how, his sort of thinking about, how computers might in particular be used to play games. And he was particularly interested in the game of drafts, otherwise known as
as, checkers in American drafts in British English.
And, so he was trying to sort of imagine how a computer might play… play checkers.
And, anyway, Turing ended up kind of bringing him in to, I guess, maybe first the Manchester Computer Project, perhaps? I'm not sure.
And that's how Straitsheet got involved with computers. But somehow his connection with being a teacher, led him to be sort of a central figure in this Computers for Schools initiative.
Okay, why am I yakking on about this? Because I discovered something quite fun about it.
So the end result of this Computers for Schools initiative was there was no truck, there were no large collection of computers delivered to schools, there was, in fact, in the end, just one computer delivered to one school.
Which was a computer that was bought from a company called Elliott Brothers, and it was a computer, an Elliott 903 computer. And, it was bought for the Royal Liberty School, about which I don't know very much, I will try and find out more. And it was delivered sometime in maybe 1963 or 4, something like that.
Okay, so at some point, that school fell up on hard times, I think, and that computer was sold. And the computer was sold to none other than Eaton. And it got sold there partly through the efforts of a chap called Norman Routledge.
Who's a teacher there.
who had also been involved in this Computers in Schools initiative. Norman Routledge had been sort of a friend of, and to some extent, student of Alan Turing's in Cambridge, and had subsequently worked at the National Physical Lab, doing computer kinds of things. So, in any case.
somehow this… this… the one computer that had been bought through this… this grand initiative, British government initiative about computers in schools, the one computer wound up at Eaton.
and wound up being the first computer that I had the chance to use. And I think I know a decent collection of the people who used that computer, and quite a few of them have ended up in the computer industry in one way or another.
But I think it's kind of interesting that this whole sort of government initiative with lots of committees and this and that and the other, in the end, that one computer ended up being the thing that has this thread that connects to things like my efforts in computers. And I don't know what that really says, other than
The fact that
these things, like, well, I happened to get exposed to a computer when I was 12 years old in 1972, and it was a little bit less… well, there were computers everywhere, and it was sort of obvious that that could happen. It was something where the sort of tower of coincidences was a

bit taller than I would have expected. So I was interested to discover that. I might try and do a little bit more historical research on exactly the story of that computer and so on. I did find a A couple of years ago from the archives at Eaton, I found the purchase documents for that computer when they bought it.
But,
That's, anyway, so that was one of my, sort of, historical adventures. I suppose, another one I was,
driving in England, and realized I was going near Bletchley Park. Bletchley Park is this place sort of roughly equidistant between Oxford and Cambridge, that was the place where the code-breaking effort in World War II was organized in England.
And, there turn out to be two museums on that site. One is the National Museum of Computing, another is the Bletchley Park Museum. The National Museum of Computing is a much scruffier museum, but has a lot of really interesting exhibits of computers.
And one of the things I had the chance to see there was none other than Elliott903.
There aren't very many of them out and about in the world.
That Elliott 903 was,
was there. You could even switch it on, and, it, it came up. But, it didn't seem to be working, and, the,
The docents there weren't keen on me trying to key in that the computer has a bunch of toggle switches, and I was… I'm pretty sure that the 152048 instruction will print a character on the tape punch, but they didn't seem keen on me actually trying that out. They claimed the tape punch was very loud.
But in any case, I, I, so I had a chance to meet,
my, at least a twin of my first computer there. I also learned a couple of other things there. They had the sort of a somewhat reconstructed Colossus machine.
was part of the, was one of the code-breaking machines in Bletchley Park. One of the things it featured was a tape reader.
a paper tape reader. So you have to understand that a lot of the technology for those machines was coming from the post office, and coming from electromechanical switching systems used for phones. But, and there had been a pre-existing technology of telex machines, which are Machines that, were sending, kind of, data
Through, just a way of, of, like, sending a telegram
I mean, I guess telegrams back in the day had been
based on people, you know, telegraph operators, tapping things out in Morse code, and somebody at the other end transcribing the Morse code, and that was the telegram. Later on, that became more electronic, and you could have a telegram that was just transmitted, and the way that it was often transmitted was using a telex machine, where you would have something that read, mechanically read, a 5-hole paper tape.
And went, you know, reading through each character and sending it over a phone line, which was the… to another such Telex machine. And part of the point was that you were paying for that phone line if it was, you know, an international phone call or something like this, and so being able to transmit it as quickly as possible by having you automatically be able to send through
the characters, rather than having to tap it out, was a good thing. So anyway, that technology had existed.
But what was apparently developed for the Colossus machine

was an optical tape reader that read… they had versions of the text of cryptograms on this tape, and they were using… they were correlating that
that text with things that they had tried to decode, and they wanted to just keep running that text to do this correlation over and over again. So they had a pretty fast optical tape reader. What I didn't know was that the optical tape reader of the Elliott 903, which was one of its notable features, was that optical tape reader.
Was directly derived from the Colossus tape reader through a couple of steps.
And the,
The one on the Colossus actually ran 10 times faster than the one on the Liott 903.
Well, actually, I think one of the steps that it went through was this very strange episode in the history of computing, which was, in the 19…
50s, early 50s, I guess.
The first kind of serious commercial computer
that was constructed or set up, which came from, you know, there had been a bunch of individual computers built, the ENIAC in the US, the Johniac, John von Neumann's computer, the, in, the… in the UK, the Manchester computer, the ACE computer in Manchester. You know, every computer was a bespoke thing that sort of had a separate name. There wasn't a production line of computers yet that would come with companies like IBM and Unibac and so on.
But,
the, the first sort of effort to really make a commercial computer in, was something in England that was a Lions Tea Company. So Lions always used to be this company, I don't know if it still exists, that,
was a catering company, or a food company, and you would see, kind of, these tea houses all over the place that were Lion's tea houses. Well, at some point, some forward-looking manager there realized, gosh, you know, we have this problem of distributing our foodstuffs and cakes and, I don't know what else, and trying to do that in an optimal way, trying to solve, sort of, the optimization problem.
I think this will have been a thing, sort of, in the aftermath of World War II, because World War II was the time when operations research
got developed by the US and the UK of using things like linear programming and so on to try and figure out how to optimize production and logistics of things. And this was probably a descendant of that. But anyway, the result was that
these lions tea houses, were responsible for investing in the creation of a serious business computer called Leo.
And, that Leo actually had a bunch of technology that was developed for it, including this optical tape reader. So, that was, the… that,
That was another part of the… of the story.
Let's see… Well, what else? I saw many other,
Old computers and, and, relics of, the,
Of the Bletchley Park effort in my visit there.
I do recommend this National Museum of Computing in the UK. If you're interested in the history of technology.
I also, in my,
wanderings in the UK. I was, driving past a town in Lincolnshire called Grantham, and I remembered, gosh, that's where Isaac Newton, kind of near where Isaac Newton was born and grew up.

And I knew that in that town, there was a church.
which happens to be the Church of St. Wolfram, usually spelt W-U-L-F-R-A-M.
And, it's one of only a couple of St. Wolfram's churches around the world. Now, obviously, given my name, I had to go visit St. Wolfram's Church.
Who was Saint Wolfram? He was a rather minor, saint in the 7th century AD, and, he, I think he was responsible, he lived in the area that's now France, and he was responsible for converting somebody,
to Christianity, we assume. In any case, somehow, I don't quite know how, there ends up being a church of St. Wolfram in Lincolnshire, in the UK, fairly near Cambridge. Well…
that was the place where Isaac Newton went to, to, well, grade school, elementary school, maybe high school also.
And, I was… so I was very curious to see this… this church. It's actually a pretty fancy church. It's a pretty big place. Something I didn't know is that the church had a library, and the library was actually a library of pretty serious books.
Which is something that I guess Isaac Newton must have been exposed to in his early life.
you know, going to school there, which is something that kind of… it's a… it's a slightly outlying area. This is not a kind of a… a center-of-the-world kind of… kind of town, at least it wasn't at that time, still isn't today.
And so, it was, like, sort of, there's a picture of Isaac Newton growing up, sort of, in the middle of nowhere, but at least this church was pretty fancy, and the library it had had some very serious, sort of, classical books in it.
The other thing I
is that, when… well, so then I thought, okay, I better go see the Newton's birthplace, and there's a little museum there, I think, a place called Woolsthorpe. So off I go, driving to Woolsthorpe from, St. Wolfram's Church, and I'm like, you know, I find it on my Apple Maps or something like this, and I'm driving there.
And I'm like, this is a very long way away from the church. How did old Isaac get to school every day?
Well, so I quickly found out he didn't go to school every day, he actually lived with an apothecary.
In, right near that church when he was in school.
That's interesting also, because Newton was very big on alchemy and doing various kinds of experiments, and my guess is he learned about that from hanging out with this apothecary when he was a kid. But I also, realized that this was a long way I was going to Wolsthorpe. Turns out.
Annoyingly enough, there's more than one Woolsthorpe in the area, and the one that's marked as being kind of Newton's birthplace, at least in Apple Maps, as of a few weeks ago, is the wrong one.
And so I got there, and there's nothing there, and I didn't have time to go see the other ones, so I failed to see that particular piece of history.
Let's see, in other,
gosh, I did have a chance to drop in a number of other historical places, but I think that's probably about as much as I can immediately,
come up with to describe from my… from my trip to the UK.
I'm… there's a question here.
About,

since I've spent so much of my life in the US now, has my perspective on… changed on what qualifies as history? You know, it's always funny, because it's like, things that happen in the UK, there's an awful lot of stuff which, like, oh yes, this is quite old.
It dates from the 1200s. You know, it was in the Doomsday book from 1068 AD. You know, that's quite old. Whereas in the US, if it dates from, you know, 1930, it's quite old.
And, I suppose I've… I've had a… I've, adapted to that change of what… what it means to be old, so to speak.
And certainly, I visited a friend of mine who's, has,
Has a large, large house, that his family has had for a long time, that on its grounds has the ruins of an abbey from the 1100s.
And it's like, yeah, well, you know, Henry VII dropped in at this abbey back in the day, but then when Henry VII, you know, had his falling out with the Pope.
And dissolved the monasteries while he came and, you know, his people came and tore the roof off the place and so on. And it's kind of, like, almost as if that happened. That was in, like, 1538 or something. It's almost like that happened just yesterday. And, you know, over the next century, people sort of,
took away a lot of the stone and so on, but it's, you know, that's the way it is in the UK, that these things that, these stretches of history are a bit longer than they tend to be in the US.
And I don't know whether it really makes that much difference at some level. I mean, if one's, you know, if things have been happening for a generation or two, it's kind of as long as people can remember. And the fact that it's been happening for 20 generations, is interesting, but I'm not sure that it has a direct,
immediate kind of import for, for, for things. I mean, it's always, yeah, it's a… Anyway…
Let's see…
Well, okay, somebody's asking about, Ham is asking, did you explain your physics project?
to Roger Penrose. I think he knows a decent amount about it. But,
I was actually asking him, and I can repeat some of that, about the history of studies of discrete space and his, his involvement with that.
So… Yeah, he told me more of that story than I knew.
So… Roger Penrose studied with a chap called Hodge, I think, in Cambridge.
Hodge, if you're… if you're into,
kind of math is that the Hodge dual is a Hodge-type thing, or the Hodge star operation. I think that's the Hodge. Anyway, apparently Hodge gave Roger Penrose a problem in algebraic geometry having to do with finding out when a series of equations would produce just a curve, when the solution to those equations isn't a point, or a few points, and isn't a surface, but it's just a curve.
And apparently, in doing that.
in figuring that out, it involved a whole bunch of tensor manipulation, whole bunch of, kind of, knitting together this tensor and that tensor, and so on. Apparently, Roger told me that that was when he came to invent what is now often called Penrose notation.
For, for tensors, this kind of these string diagrams where you have this sort of graphical representation of, of kind of how the indices of tensors knit together.
I mean, a little bit of backstory, I don't know the full backstory of Roger Penrose, but, I think… when he was a kid, I think he was quite involved with sort of geometrical, thinking about things geometrically. In particular, he invented these kind of impossible drawings, which subsequently

Escher kind of picked up on, and so on. A sort of, a weird footnote to history, Roger Penrose's father was a chap called Lionel Penrose.
who was a geneticist. I'm not sure what the whole family configuration was, I think. I'm not sure Roger grew up with him all the time, and so on. But,
in a curious, kind of, small world of history, my grandmother, a person called Kate Friedlander, who was a child psychoanalyst, was a friend of, of,
Lionel Penrose's, back, you know, in the, well, my… my grandmother died in 1948, I think? So this was sometime in the… in the 1930s and early 1940s. Just one of those weird footnotes to history. But in any case, the, so…
Roger Penrose started thinking about, kind of, these tensors and so on, and these diagrams, this diagrammatic representation of tensors. Okay, that was something he did for his thesis. Okay, he told me when he was a graduate student in Cambridge.
He took 3 graduate classes.
one… was… from… Gosh.
One was about,
pure mathematics, I'm now forgetting who that was from. One was about relativity from a person called Hermann Bondi.
And… One was about logic from a chap called Steen, mathematical logic.
What was the third one? My gosh.
Somebody very well known. Can't remember. But,
In any case, the… so… so that was… Roger told me that was his introduction to mathematical logic. Now, somehow he also…
knew Max Newman. I think maybe Max Newman
became his stepfather somehow, if I remember the story correctly. Max Newman was a person who'd been involved with, kind of Roger… with, with, Alan Turing and so on. Max Newman was also a big wheel at Bletchley Park. In fact, he was sort of the… I think he was… he was, I think he…
was perhaps Alan Turing's boss at Bletchley Park. In any case, he was a Cambridge topologist. Who, hadn't done so well in topology and had some missteps in topology, but ended up being quite a far-sighted person when it came to thinking about computation and so on.
In any case, so… that was sort of the early backstory for Roger Penrose. But then, somehow… N… Trying to remember now, the,
the story of this, I guess…
Oh, that's right. The third… the third class he took in Cambridge was taught by Dirac, by Paul Dirac, and was about quantum mechanics.
And, I had always heard that Paul Dirac was a very monosyllabic man.
At least when he was working in Cambridge. Late in his life, when he reached the mandatory retirement age of 65 in Cambridge, he moved to the University of Florida, and apparently there, he was a much more cheerful, sunny kind of fellow. Unfortunately, I never met him. I could easily have done, but I never did.
The, I had heard from my friend Dick Feynman that in his interactions with Paul Dirac, that Paul Dirac rarely said more than one word. Yes, no, whatever.
I'd heard from, actually from, Norman Routledge, the teacher I mentioned earlier, Norman Routledge had been a friend of, Paul Dirac's son.

Paul Dirac's son, I think, was a graph theorist in Cambridge, and Norman Routledge had quite often visited their house, and apparently the older Dirac seemed to be a very, kind of, aloof figure.
with lots of interest in mathematical puzzles and things like that. In any case, Roger Penrose's kind of third graduate class was by Paul Dirac, and Roger Penrose told me that he had actually managed to find it, find a way to actually have conversations with Paul Dirac.
He just said, he said to me, Roger said to me, you just had to know that you couldn't talk about the weather with Dirac. That if you were asking, you know, direct questions about, you know, about mathematics and science, then he would talk, but there was no small talk there, so to speak.
The, I think,
Anyway, the, the, so through his interaction with Dirac, I think.
Roger Penrose told me he'd gotten interested in quantum mechanics and in, questions about quantum mechanical spins and so on.
And sort of the… and that had sort of connected this whole business about tensors and tensor diagrams and so on was a story that connected with how you combine spins in quantum mechanics. Essentially, the question is, if you have
Something where you have a whole… you have a… particles in… have…
this attribute that's called spin that roughly can be thought of as kind of how fast they're spinning, but nothing is really actually spinning. It's just a feature that has certain common sort of resonances with actual spinning and angular momentum. In any case, the,
There's this notion of spin, and the question is, in quantum mechanics, spin is quantized. An electron, which has spin a half, if you define a direction, the electron either has a spin exactly pointing in that direction, or exactly anti-parallel to that direction.
And so, it has two possible spin states.
So the question is, if you have a collection of a whole bunch of electrons, how do you describe the combined spin state of all those electrons? And that's a story that relates to, well, representations of the rotation group.
and combinations of representation of the rotation group, things called Kludge-Gordon coefficients, things called 3J symbols, 6J symbols. In the end, it's a giant tensor festival.
And so, I think that's what got Roger Penrose interested in, kind of, how can you combine spins and think about that using the formalism that I think he'd already developed for thinking about these sort of diagrammatic formulasism for tensors.
And that got him interested in spin networks, and, the,
And sort of combinations of spins. And that was in the early 60s.
And then… In,
he told me, I think maybe 1962,
he had met a chap called David Finkelstein.
David Finkelstein was somebody I knew.
But David Finkelstein's history had been
That he had worked on,
things in general relativity, the mathematics of general relativity, in particular, in studying in the 1950s, studying kind of things about coordinate systems for describing black holes. Black holes were very confusing to people. There were all these kind of singularities associated with black holes, which turned out to be just singularities in the mathematical description, in the coordinate systems used.

Not physical singularities, but that was only clarified
by the work of, for example, in particular, Finkelstein and Eddington and Finkelstein, there's things called Eddington-Finkelstein coordinates for black holes. Anyway, so David Finkelstein. Had been, a sort of mathematical general relativity person.
And he and Roger Penrose had a bunch of interaction, I think around 1962, and the way Roger describes it is that he kind of swapped fields with David Finkelstein. That he got David Finkelstein interested in quantum mechanics.
And David Finkelstein got Roger Penrose interested in relativity.
Well, Roger Penrose then started applying mathematical methods to relativity. He was saying to me, actually, a couple of weeks ago, that he was sort of surprised that certain kinds of mathematical methods really hadn't been applied
in general relativity, in theory of gravity, and he sort of had the fun of doing that, and that led to, for example, his singularity theorems, and all sorts of other interesting things. But,
the,
the thing… so Finkelstein then went on to be very interested in, kind of, quantum logic and so on, the sort of generalization of logic to,
to include kind of quantum features, not probabilistic, but sort of quantum amplitudes, rather than ways of being ands and ors with ordinary variables. And quantum logic, in many ways, can be thought of as sort of a predecessor of quantum information theory and the whole qubit idea and so on.
Although I don't know whether that connection was very much historically made. I'm trying to remember. I actually had a long conversation with David Deutsch, the originator of qubits, also when I was in the UK, and we talked about a bunch of history. Maybe I should describe some of that, too. But I don't think we talked about the origin of qubits and so on then.
I think maybe I have another occasion. But in any case, I do have to say about David Finkelstein that he did many things which sort of
seem like they overlap with things that I've been very interested in. He wrote, for example, a book called Quantum Relativity, which is sort of an attempt to kind of bring together the ideas of quantum mechanics and the ideas of general relativity.
One of the things that David Finkelstein told me that greatly helped my understanding, or lack thereof, of his work.
was, you know, I had seen many papers he'd written, I'd found them very hard to understand, and he told me one day, you have to understand that, you know, I've had many different ideas in my life, and that, you know, sometimes I'll have a new idea that supersedes an old idea, but I won't mention that.
So it's a very difficult thing to read his successive works, because new things happen in new papers, but they don't say, oh, by the way, the things in the old papers just aren't right, or this is a rethinking of them. They just, you know, silently go and say a new thing.
I have to say, I, I, perhaps…
some part of, I don't know whether this is really one of those stories to be told, but why not? It's, David Finkelstein was a very colorful character in many ways, although he, had this long white beard, and he had a very white house, so I say a colorful character, but everything around him was, was, was, seemed to be perfectly white. In any case, he,
We were having dinner at a rather peculiar restaurant in Atlanta, Georgia, which is where he lived. He was at Georgia,
Georgia Tech.

for later years of his life, I think earlier he'd been in New York City for a long time. But, any case, he's… we're at this rather peculiar restaurant that happened to have… the servers happened to be, wearing monk costumes, which was a bit odd. But in any case, the,
There's David Finkelstein with this very long white beard, and we're talking about quantum mechanics.
And he sort of says to me, rather conspiratorially, he says, you know, to really understand the photon, you have to be Jewish.
And I'm like, what on earth do you mean? What does that possibly mean? Anyway, it led to an interesting conversation about, sort of, cultural, sort of the different kinds of cultural backgrounds or cultural ways of thinking about things. I think his main point was.
that, somehow in Jewish culture, you can kind of hold different views
contradictory views at the same time, and that was kind of a necessary thing for understanding quantum mechanics, that one have sort of a visceral relationship to that, but I thought that was a kind of an interesting… it a little bit summarizes the
The, the character of the person.
But in any case, the, just in terms of strange things said and so on. In any case, so what I learned from Roger Penrose is that Roger Penrose's interest in relativity had been sort of the result of swapping with, with, with David Finkelstein.
Now, subsequently, sort of spin networks as a foundation for spacetime, the idea that you might be able to sort of build space-time out of something discreet, something quantum mechanical, was a thing that Roger Penrose said he largely abandoned.
I guess Roger Peno has told me quite a bit about his interactions with John Wheeler. John Wheeler was a person who also thought a bit about things like discrete space-time, and John Wheeler also had a lot of… John Wheeler was at Princeton.
I, I met John Wheeler only once in person. We exchanged letters, quite a number of letters over the years. John Wheeler was a very,
I would say, given to quite, bombastic statements.
John Wheeler was a very creative scientist. He worked on the Manhattan Project and so on. One of his early students was, was Dick Feynman, and, the,
Wheeler had worked a lot on general relativity and so on. He was the person who named black holes black holes in the U.S. I mean, in Soviet Union, for example, black holes were called frozen stars. I think they were named that by Zoldovich.
But in any case, the, the, Wheeler…
was… Roger Penrose said he'd spent some time around Wheeler and hadn't really gotten much out of him. Wheeler…
Has this, sort of, had this kind of idea of, sort of, the participatory universe, that somehow was an explanation of quantum mechanics, that somehow the universe was looking at itself in some way.
I guess in Wheeler's early years, he'd worked with Bohr, perhaps, and Bohr had been much involved in the Copenhagen interpretation of quantum mechanics and so on, so he'd been exposed to, sort of, the early ideas about… about, kind of, the foundations of quantum mechanics. In any case, the,
The, the,
well, Roger Penrose said that… that Wheeler's work on… on, kind of, foundations of spacetime hadn't really influenced him much.
By the way, Wheeler's work showed up.

The place where it's most obviously visible is a big, thick book called Gravitation by Charles Miller, Kip Thorne, and John Wheeler. It was published in the 70s, I guess. It's sort of a classic kind of, I don't know whether to say bombastic textbook of quantum mechanics, but… but of general relativity, of, but it's definitely a book with character, let's say.
One section of that book, at the very end, talks about how space-time might be made of something lower level, in a little bit the way that chainmail, that, you know, a suit of armor could be made out of little rings and so on.
that that might be sort of the way that spacetime is constructed. Very interesting idea, something with quite a bit of resonance for things I've thought about. Well, that idea, in my own efforts to research that history, that idea in which I have not done completely fully, but that idea kind of was particularly related to a book that Wheeler wrote in the early 1960s called Geometrodynamics.
And that book…
has sort of an interesting description of the possibility of sort of… I don't know how far it gets into discrete spacetime, but that book is sort of a theory of
A little bit towards quantum gravity, the quantization of gravity, the idea that gravity could be a force like electromagnetism that was mediated not by photons, but by gravitons, and so on.
But the book has one horrible misstep.
Which is, it hypothesizes that gravitons, the carriers of gravitational force, are neutrinos.
That's just complete nonsense. And I think that's why I had never seen a copy of this book. I got myself a copy a few years ago, and you know, one doesn't hear about this book. And I think that's why. The book is otherwise a perfectly valid book, but it has this hypothesis that's just a complete misstep.
you know, I have to say, maybe I should tell the story, of the one time I met John Wheeler in person.
He was 95 years old. I dropped in on him in Princeton.
This is kind of a sad story, but,
you know, I'm telling him about things to do with computation and so on, and my work in new kind of science, and so on. This is long before our current physics project.
But, you know, I had some precursors of that, thinking about discrete space-time and so on. And eventually, you know, sort of talking about this a bit, and eventually he looks up and he says. You know who you should talk to about this? It's a chap over at the Institute for Advanced Study.
His name is John von Neumann.
So I'm, like… So I said, well, unfortunately, John von Neumann died before I was born. But, This is… this is what happens. I know the feeling, even at my not-so-ancient age, this question of, you know, when things happened in one's life. I mean, for John Wheeler, I'm sure he remembered many conversations he'd had with John von Neumann. The fact that John von Neumann had died
you know, you know, 60 years before that, the conversation that I had with him was something that's not… not so visible.
In any case, back to Roger Penrose and his work on discrete space-time, he said, well, actually, the main person who'd followed up on that was a person called John Mazuris.
who happens to be… have been a friend of mine for the last 40-something years. And John Mazuris was a student of Roger Penrose's in the 1970s.
And, I suppose I could tell you the, the, the very…

interesting tale of John Mazoris' life. I'm not sure I… I'm not sure I'm really signed up for telling the story of, the lives of friends of mine, but I think this one is interesting enough that perhaps it, deserves some mention. And anyway, John Mazoris is writing an autobiography right now, so… so, the story will be out there before, before too long anyway.
But what was interesting to me was that John Mazor… that Roger Penrose was like, John Mazoris is the one kind of link
to sort of discrete space-time that Roger Penrose has maintained.
And, I will say that, in,
Well, Roger Penrose was also telling me about the origin of twister theory, which is, kind of originates in this wonderful mathematical coincidence about four-dimensional space-time and the structure of complex numbers. Like, complex numbers, there's a… you know, complex numbers, there's the real part and the imaginary part. That's two things. Well, there's sort of a generalization of that that gives you sort of four things.
And it's not unrelated to a bunch of formalization that Dirac made about quantum mechanics. Dirac's, sort of, big break was the Dirac equation from 1930…
when was it? 1920s? The… in 1925 or 26, I think, Schrodinger had come up with the Schrodinger equation, which is
this description of, kind of, the matter waves, a wave equation for the matter waves that correspond to electrons in quantum mechanics. But the question was.
The Schrodinger equation was a non-relativistic equation. It described sort of slow-moving electrons, kind of the way they would be in typical chemical situations, and so on. The question was, how do you make that consistent with relativity?
Because the Schrodinger equation isn't consistent with relativity. The Schrodinger equation has a second derivative of space and a first derivative of time, which means that sort of space and time aren't acting in the same kind of way, like they do in relativity. So there was sort of a question, well, could you, could you get something which is like
the, the Schrodinger equation in the sense that it described quantum mechanical wave functions, but was doing that relativistically, and that was the thing that Dirac figured out. I mean, I should say that the, the,
Maxwell's equations for electromagnetism, which had originated in the 1870s, were sort of born relativistic. In fact, the thinking about Maxwell's equations was what led Einstein to invent relativity.
Because the Maxwell equations already have sort of relativistic invariance. The question was, how did that relate to the structure of space? And that's where Einstein started saying, well, we should modify the structure of space to make relativity work everywhere, so to speak.
But in any case, the, so the Maxwell equations can be thought of as equations for a photon field. And there's sort of a question, well, how do you… and they're relativistic. So what Dirac did was essentially take the square root of the differential operator that appears in, essentially, the Maxwell equations, and said, well, this might be the equation for an electron.
And that's what turned into the Dirac equation. I mean, famously, the Dirac equation involves saying, well, there are actually four components
to an electron wave function. Two of them, we understand what they are, because they're the spin-up and spin-down parts. What are the other two? Well, that was where Dirac said, well, maybe there's… you could figure out that the other two corresponded to an electron

that has the opposite charge from an ordinary electron, and that's where the idea of antimatter came from. And it was sort of at first confusing, because you had to kind of assume that there was this kind of,
filled-up sea of electrons that then there were… that… anyway, it was… it was kind of a… the interpretation was something kind of interesting, but anyway, that's that mathematical coincidence of sort of
The, taking the square root led to this four-component Dirac spinner, as it's called, that, had two components. What are these? Well, they're the anti-electron.
In any case, that was, the origin, so that 4 that sort of coincidentally occurs in the Rach equation, there's sort of a generalization of that that Roger Penrose
Worked on that leads to twisters.
Which are sort of generalizations of spinners, and there's a whole sort of idea of maybe that something fundamental to the way that things are constructed as spinners are sort of fundamental to the way things are constructed. It hasn't… it's led to some very interesting mathematics, hasn't really connected terribly much with sort of direct experimental kinds of things.
But, well,
let's see, the, where was I going with all of this? Oh, yes, I was, I mean, in terms of Roger Penrose's development, you know, I think he's returned now
to, the third of the classes that he took in graduate school. He's done… he's checked off quantum mechanics, he's checked off, general relativity, and now there's the logic class. And so he's been interested in, sort of, Godel's theorem and its interpretation and this question about whether minds can get further
than the mechanical exploration of mathematics. I have to say, I don't find his argument for that very satisfactory, and it's very much an experiential argument of
You know, as a mathematician, I can see things that don't feel like they're mechanical.
I must say that I've, in my own efforts and understanding the foundations of mathematics, I think there's a sense in which one can say that, in the sense that the mechanical operation of mathematics is really much more at the level of, I've got one axiom, I'm proving this thing based on this axiom, et cetera, et cetera, et cetera.
But, I'm,
I'm kind of going sort of theorem by theorem, axiom by axiom, at a very low-level mechanical level. That's not what mathematicians typically do. If you are working at that level, you are often going to be thrown into questions of undecidability, of, you know, this proof isn't of finite length, and things like this.
What I think is the case is that sort of mathematicians, as they think about mathematics in the sort of intuitive way that mathematicians typically think about it, are operating at a much higher level.
It's very much analogous to thinking about a fluid like water, where you could, in principle think about water in terms of the microscopic dynamics of every water molecule, but in practice, that's not how we think about the flow of water.
If we were different critters from the ones we are, we might be operating down at the scale of molecules and talking about what happens at these individual molecules, but in fact, we operate much more at this fluid dynamics level, and we think about, sort of, the overall flow of a fluid.
I think that same kind of thing is happening in mathematics, that actual working mathematicians like Roger Penrose.

think about mathematics as, at this sort of, the flow of mathematics level, rather than the mechanical, operation level. At this flow of mathematics level.
things like Godel's theorem are much less relevant.
But also, you feel like you're operating at a level where you're just… you're operating away from Godel's theorem. I think that's where one gets this feeling that's not entirely wrong, that, sort of, the human mathematician isn't engaged with Godel's theorem.
As opposed to the formal proof system, which is necessarily kind of engaged with being stuck with these infinite length proofs and Godel's theorem and so on.
But anyway, back to the story of discrete spacetime and the story of my friend John Mazurus. The,
John has had an interesting life. You know, grew up in New York City of Greek extraction, he, wound up going to fancy school in New York City, and then, on some scholarship, and then, wound up at Harvard as an undergraduate.
Where…
he ended up, among other things, interacting with another person I've known for a long time, or knew for a long time, quite separately, Ed Fredkin, who is a sort of exotic professor type at MIT, who, was… I wrote a very long obituary of Ed Fredkin, the longest obituary I've ever written. Ed Fredkin's life story is more complicated than any other I've tried to disentangle. And even though I had known Ed Fredkin for 40 years, there was a lot that I found out,
after he died, that I wish I'd known why he was still alive, that, sort of filled in pieces of a very exotic life story. But anyway, Ed Fredken had been… he always used to say, even… even, you know, a few years ago, he would still say… he would still
Characterize himself as the world's best programmer.
Which was a little bit hard to justify in, you know, 2020 or something. But when he'd learned about computers in 19…
50… in the mid-1950s, as a person in the Air Force, he was in the very first cohort of people who'd been trained on computers, when computers were being used for, missile tracking and so on.
And, so at that time, he was probably right. He was top of the class, he was the best programmer back in 1955, or whenever it was.
Probably, probably harder to justify in 2020. Anyway, Ed Fredkin had been sort of a computer entrepreneur, inventor-type person,
And, who had developed this idea that, he could sort of construct the universe out of computational kinds of things, and in particular, cellular automaton kinds of things, and just sort of make a universe where there would be things like electrons and so on, just like he could make a computer or make a software system.
And so, although I don't think he ever understood, kind of, my, sort of approach to our physics project and so on, he was very much set on the, we can construct a universe by engineering. But anyway, Ed Fredkin,
John Mazouris kind of, interacted with Ed Fredkin. Ed Fredken, at the time was involved in building
oh gosh, I'm… my threads are not completely connecting here, but there… I was involved in building a chess-playing computer that John Mazoris worked on, at, when he was a student at, at Harvard.
But then… John got a Rhodes Scholarship.

To study in Oxford, and originally, being interested in computer kinds of things, he wanted to go and study with somebody who I also mentioned in this livestream, Dana Scott.
Dana Scott was a mathematical logic person, and as I said, who I've known for many, many years.
John reported that… I mean, Dana is a… is a person who, can be a bit austere in some ways. The thing that's always interesting to me about Dana
is that he's worked on these topics of just unbelievable abstraction and obscurity, that, you know, whenever I've looked at things in mathematical logic, when I get to a corner that is just the most bizarre, abstract kind of thing, it's got a Dana Scott contribution to it.
Yet… in talking to Dana, Dana is a very clear explainer of things in a very direct.
and down-to-earth way, in many cases, and he's often told me, don't go and study Scott domains. They're not relevant to you, type thing. That's one of the more obscure areas which I don't understand. In any case, so Dana can be a bit austere.
And apparently John went to talk to Dana, and Dana was too austere, and John was like, I'm afraid of this person, I can't work with him. So instead, he went to work with Roger Penrose.
And… but he had been interested in this idea, John Mazoris had been interested in the idea of discrete space and so on from Ed Fredkin. And so, somehow, and I need to get the full story from both sides on this, the, somehow,
Roger Penrose suggested to John Mazoris, oh, you should work on spin networks, and in particular, work on generalizing them to the Poincare group.
Which is the group of rotations and relativistic transformations in spacetime, as opposed to the internal symmetry group that's associated with spins. So anyway, John Mazuris went and wrote his thesis about that. I think he had a good time doing that.
And then…
winded up… wound up, back in the US and working at IBM Research with a chap called John Cock.
who was a computer architecture designer. I mean, John Mazoris had had the previous experience of working on things like this chess machine, and working on computer system design, and had also been quite a devices and hardware person from his earlier years. In any case.
He, then, at that time, John Koch was trying to sort of redesign
kind of the, the whole sort of way computers were constructed, actually. Ed Fredkin had had some connection with this as well, as had Marvin Minsky, a whole gaggle of people that, that I've known. I knew John Cox somewhat also. But, in any case,
the,
And John Koch, by the way, had also been interested in cellular automata at some time, but…
John Cox's idea was, computers at the time, things like the IBM 360, had immensely complicated machine codes. They were, you know, they had all these different instructions that did all these very special things. His idea was, let's make it as simple as possible. Let's have a reduced instruction set.
And so, that's what, got developed, in part at IBM, and John Mazuris was part of that, along with doing a number of other things at IBM, which actually had quite long-term consequences. But, then the,
that… Then, at some point.
John was sort of, from IBM, was sort of lent out to visit Stanford University, where he interacted with a chap called John Hennessy.

And somehow that turned into a whole computer architecture story with risk computation, and these… sort of the idea was… of risk is that you have a very simple instruction set.
And when you make programs, you make the compiler… it's easier to opt to make an optimizing compiler if the instruction set is simple, so you end up coming out ahead, even though you haven't built in machine code these much more elaborate instructions, you come out ahead, because for your actual program, you can do better optimization down to this reduced instruction set. Any case, that turned into
the, the creation of a company called MIPS that, John Mazoris was, a co-founder of, and I guess CTO of.
And, MIPS, ended up being a very successful company in many ways, although it was a very venture capital funded company.
And, at some point, the, as so often happens, the… the kind of the preferences of the venture capitalists diverged with the actualities of the people working at the company, and John sort of ejected from, from MIPS.
But by that point, I guess soon thereafter, MIPS went public, and John made a decent amount of money from it.
But then he started another company called MicroUnity, and actually I've been a… was an advisor to MicroUnity for most of its lifespan. MicroUnity, the idea was to extend this notion of inventing architectures
And two, at the time, this was early 1990s, the big goal was make a microprocessor that could hit the 1GHz clock rate point, and do it
using, with, and particularly make a bunch of instructions that were things like vector instructions, instructions that would deal with arrays, rather than individual, things. And I guess, part of John's expertise is inventing
Kind of the right sort of primitives to be able to do computations from.
Well, in any case, my community to develop those kinds of things. John, in, perhaps you could argue a,
A little bit of, you know, Silicon Valley is a very cutthroat place, and there's a lot of paranoia to be had around Silicon Valley. In the words of Andy Grove, former CEO of Intel, you know, only the paranoid survive in Silicon Valley.
So, in any case, one of the consequences of that was that, John didn't want to take a design that he'd made of a microprocessor and ship it out to somebody else to go manufacture the microprocessor.
He wanted to have, sort of, the whole… the whole thing, sort of all vertically integrated, and so he built a factory.
In Silicon Valley, a microprocessor fabrication facility in Silicon Valley. I believe it was the last such factory to have been built in Silicon Valley.
And it was done very economically, by the standards of semiconductor fabs. But in any case, that ended up being a thing that pulled in… had to pull in a very large amount of investment money.
lots of terrible things happened after that, and I think there was sort of a mismatch between kind of the price profiles that people were looking for for various different applications, from what could be delivered and so on.
I have to say, there was a… you know, in order to make a factory work, you have to be putting billions of dollars worth of stuff through the factory every year, and unless you have customers to buy that, things don't work well.

And indeed, they didn't in this case, and MicroUnity, as a company, didn't make it, although it ended up with a very valuable patent portfolio, because basically the things that John and his employees had developed
were basically the original blueprint for GPUs, and the original kind of GPU instruction sets and so on. And so, for a number of years, it was kind of a company with… where its only existence was patents that led to a giant battle of patent infringement lawsuits and so on.
Which eventually led
to, John, getting, quite a lot of money that, is used for, for lots of different purposes. But, but, the, the, sort of the end of this story, I suppose, in, in, to these points is,
John had been telling me for years, decades, he said, one day I'll go back to my thesis and, sort of finish the stuff I started on spin networks.
And I've been encouraging him to do that, and I think he actually made some progress on that a couple of years ago. But I hadn't really realized that he was the kind of lost surviving thread
of Roger Penrose's 1962-ish sort of push on discrete spacetime.
So it's… it's kind of interesting to see how that's, how that's developed. I encourage people when… I'm going to give a shout-out for my friend's autobiography when it, when it comes out. It's a little bit of a wild, it has many pieces that I'm not describing here that,
And kind of a… a curious…
kind of description of, sort of, the, in a sense, the magic of technology and the magic of, sort of, spiritualism and so on, and the merger of those kinds of things. In any case…
well, let's see… I don't know, that was a very long answer.
Sylvia is commenting, you've already spoken more words in this… in this, livestream than Dirac did in his lifetime. I don't know, but, could be. I wish I'd met Dirac. I… I… I… you know, it's one of these things, when I was younger, I just didn't…
have…
the same kind of, I suppose, respect and interest in history that I've developed subsequently. Plus, I will… I will, you know, ascribe it to the web didn't exist. So, you know, I just didn't know as much about, sort of, Dirac.
than as… as, you know, the grapevine story of Dirac.
was only random fragments that I'd heard from people, not, oh, you can kind of read the gossip column about Dirac, so to speak, on the web, and get more of a sense of what… or see, you know, the video clips of Dirac, or whatever else it is.
Let's see… well… Let's see if there's anything else,
That is related to what I was just saying here.
The… Brian asks, how much did the L8903 cost? I believe that Eaton bought it for £20,000.
In 1970 or 1971.
So… I think that's… what would that be nowadays? Maybe… $60,000, something like that.
It's quite a lot of money. I mean, it, it, and I, I…
Yeah, I think they ponied up the money. I mean, I don't know. That's probably in the archive material that I actually got a few years ago from them.
Let's see… anymore.
that I can address about the things I was just talking about.
The, okay.
Ali, comments?
about, I think about my comments about computers in schools from the early 1960s, that they thought

Ali thought that in the 1980s initiative with the BBC Micro and Sinclair and so on, was the kind of educational push for computers. Nope, these things have happened many times. There have been many waves of, let's introduce computers to kids.
Many waves, both on the hardware side and on the software side.
I, I would say that…
It… it's never really taken as much as people might have hoped.
I think that on the software side, for example, there was BASIC in the 1960s, which was pretty good, actually.
Then there was Pascal as a teaching language, and then was developed at ETH in Zurich, by Nicholas Wirth.
And then, you know, there have been a series of other sort of initiatives of, you know, now it's time to teach kids computing.
I think what happens is, it's… it's like computing as a very…
with low-level languages, it's sort of inevitable that you're teaching the mechanics of writing programs, the mechanics of writing low-level programs, much as teaching mathematics might look like the teaching the mechanics of doing arithmetic and algebra and so on. And the set of people who are interested in that mechanics is actually not really even the same as the set of people who are interested in the ideas and so on.
So I think that, you know, I… one of the things that I think is important about our efforts with Waltham Language and the whole idea of computational language to describe things computationally, rather than a computer language to say how to move around the bits in a computer, so to speak.
What's important about computational language is it really talks about the world, which is something that sort of everybody could engage with. It's not talking about the mechanics of doing low-level programming. And I think the effort to repeatedly teach kids the mechanics of low-level programming is sort of doomed. And, you know, it's come up, there have been about 5 iterations since the 1960s of doing this.
And it's never really worked out.
And I think it's just not the thing one should be teaching. What one should be teaching is sort of computational thinking, kind of computational X for all X, not
the kind of, this is how to program at a low level, and make loops, and variables, and arrays, and conditionals, and all this kind of thing. That's the… that's the machinery of computers, which is very relevant.
to the small set of people who are going to grow up to be people like John Mazuris, who are computer architecture designers, but it's not relevant to the vast majority of people, and it's decreasingly relevant because of the automation that folks like me have built
in the construction of programs, and which we now also get some help from AIs and so on for.
As we see that, sort of, the low-level stuff is all getting automated away, what still matters is kind of the, how do you think about the world computationally? How do you formalize your thinking about the world in computational terms? And my goal as a language designer is to provide, sort of, a language in which that formalization can immediately be executed, so to speak. And I think that's a level that actually makes sense to teach kids, and certainly a fair amount has been done on that in the last few decades.
with our technology, a lot more could be done. I think that's really the thing that is worth teaching. You know, I was realizing something, I'll just make one last,
comment. The, about,

Oh, I was realizing that, you know, in economics, there's this sort of notion of comparative advantage, and the idea that everybody is better off if people specialize. I was realizing that maybe we're coming to an end of that time.
Because a lot of that specialization has to do with… you learn something, you learn how to do the mechanics of a thing. As you automate.
what becomes more important is kind of the big picture of what you should do, and less the kind of the specialization into these different areas, and that probably has some economic consequence, which I haven't figured out yet. I see one last question here from Brady. Does Penrose believe space-time will turn out to be discrete?
Did we talk directly about that question?
I think…
I think he's not thinking much about that these days. I think he thought about that in the early 1960s and 1950s, but hasn't really thought about that much now.
I think that Roger Penos is really a mathematician.
And as a mathematician, he kind of thinks of functions and spaces and these kinds of things, and I think for him, in some sense.
That's what's fundamental, is the mathematics, which has certain aspects of continuity, and certainly the things he's worked on in general activity
depend on that continuity. They do not… it's not like the singularity theorems say, well, actually, there won't really be a perfect, identical singularity, sort of a 1 over 0 singularity, because, by golly, you know, space-time is ultimately discrete. That's not…
the way of thinking that he has. He's much more thinking about things in really very mathematical terms, with sort of a mathematician's intuition about things. And I think that Arguably might lead him into trouble in thinking about, sort of, sort of the physical instantiation of things, about, you know, are there gravitons running around in the brain type thing?
And that's, you know, that's not… that's not something you get to from mathematical intuition. That's something you have to get to from sort of a… more of a physics intuition, which is a different kind of thing. Now, that sort of makes one ask the question, is the universe fundamentally mathematical?
In the sense that it is about the constructs of human mathematics, or is the universe fundamentally computational, in the sense that it's about the kinds of things that we can define with rules that we can imagine implementing by the kind of thing that we think of as the formalism of computation today?
And I think… I would say that if push came to shove, I suspect Roger would be down on the side of… it's sort of fundamentally mathematical. Its constructs are fundamentally the kinds of things humans have thought of in mathematics, whereas I'm pretty sure that the way to think about it is in terms of
of that the underlying constructs are these things that are sort of rules that we can describe in modern times as being computational. Now, are these really necessarily incompatible points of view? Probably not.
In fact, Roger's very own sort of project in graduate school of finding these curves that came out from solving… solving equations.
Excuse me, that, that, that question…
Of how do you take these kind of continuous equations and derive from that this sort of distinct curve

is this kind of way, and this happens all over the place, that you can say, well, I've got these continuous things, but there's something discrete that emerges from them. Like, for example, imagine you have a landscape, and it's got, the landscape is continuous, it's, you know, it's all sort of got hills and dales and so on.
But the place where there is a maximum, that's just a point. Or the place where there's a valley, that's a line. There's something discrete that sort of emerges from the continuous. And it's very possibly the case that there's a formulation of the kinds of things that I've worked on, that we've worked on in our physics project.
that…
where that project is stated in terms of discrete kinds of things, but where that could be interpreted as the skeleton, in some sense, of something that can be thought about in terms of continuous mathematics. So these things are not… are a bit closer than one might imagine. But, anyway, a few thoughts on that.
But, and I realized there's… there's more that I can tell you about, sort of, history I have… have learned,
In recent times, particularly from,
Doing a bit of oral history discovery with people.
And, well, I guess that's… that's it for today. I'm… I'm late for my day job. Better go and do that.
But thanks very much for joining me, and, talk to you another time.
Bye for now.