Design Notes

with Adrian Secord

Liam Spradlin: Adrian, welcome to Design Notes.

Adrian Secord: Hey, Liam, it's so nice to see you. Thank you so much for having me.

Liam: Yeah, I'm really excited to have you on. I wanna start the episode the same way that we always do, which is to talk a little bit about who you are, what you're working on, and the kind of journey that led you there.

Adrian: Yeah, for sure. So, I'm an engineer, uh, unlike a lot of people (laugh) I'm working with who are, you know, designers and design-oriented. Um, been hanging out on Material Design for about 10 years now. Um, in fact, it was really fun listening to you talk to Will Larche, uh, in the other episode because, uh, he and I overlapped heavily for a long time. So it was- it was sort of fun to hear him talk about the past. Um, before I came to Google, I, uh, was doing a lot of grad school. I have a PhD in Computer Graphics, so algorithms for, um, movies and video games and that kind of stuff. Never used any of that information, uh, never used any of that knowledge. Uh, instead joined a startup like many people did. And then like many startups, uh, it fell apart and was sort of purchased quickly by Microsoft.

And I didn't want to go to Microsoft. I didn't wanna leave New York. Uh, so ended up jumping over to Google and joined Google Wallet and I was an iOS engineer on Google Wallet for a few years. Um, and during that time, I started teaching an Intro to iOS course at Google. And so during that point, I met some people who are working on Material Design from the engineering side of things. And as I'm sure you know, the original Material didn't really have a lot of engineering, um, it was sort of a group effort across the company to build components and patterns and things, um, as sort of needed, ad hoc.

Um, a lot of people working on Google Maps were building stuff, Google Search, all those kinds of things. Um, but Material Design had just started a nascent engineering effort to centralize a lot of that work. And one of the people who I often co-taught the Intro to iOS class, Alastair Tse, um, he was, uh, working in Material Design and pulled me in. And, uh, Alastair is fantastic and, I miss him a lot even though it's been eight or nine years now. But, uh, he pulled me in and started working on iOS components for Material Design, and that was kind of the very beginning. Um, and then over time I ended up, um, actually taking over that team, uh, as tech lead and then as manager.

Managed that team for a little while, um, and then jumped to be a sort of overall tech lead of all the different platform teams. So at that point, Material had invested in engineering and cross web and iOS and Android and Flutter even. Um, so I was sort of an overall tech lead for all those teams, trying to coordinate things and make sure that we did the same

things at the same time and maybe we released the same kind of components at the same time, um, that kind of work. And, uh, eventually moved into a different piece of Material where we were building out design tokens. So design tokens are, you know, not invented at Google, but very common across, uh, the design world.

But at Google, as you well know (laugh), uh, it's its own giant ecosystem internally, and we needed to build some infrastructure, um, some, you know, database, some servers, some services, APIs to store and serve design tokens across all the different teams who need to use them. Uh, so I got to build that backend out, which was super fun. And I basically kind of went from more or less front end and front end thinking to more or less backend thinking, which I really appreciated. It was a great opportunity at Google, um, to be able to do that. And so built out the backend for design tokens and things. Um, still pretty involved in that world. Uh, I know a lot of the people working on it, and I occasionally get pulled in to help out with things.

Um, and more recently I describe myself as a roving engineer. I am (laugh) still within Material Design or our greater org. Um, but, uh, I essentially get occasionally grabbed and thrown into a problem area and help out for six months or 12 months and then go and do some more troubleshooting somewhere else. So I'm kind of- I kind of bounce around from hotspot to hotspot as required.

Liam: Okay. I know that near the beginning of the journey, you said that you, uh, did a PhD in Computer Graphics.

Adrian: Yeah.

Liam: And then you said that you didn't really do anything with that information.

Adrian: (laugh). True.

Liam: One of the things that I'm really interested in on this show is like the ways that, that the different parts of the journey that like brings us all into the design world, um, ends up showing up in our work.

Adrian: Yeah.

Liam: So when you say that you didn't use the information, my first instinct was like, *are you sure*?

Adrian: Yeah, no, I'm lying. It's not true. Uh-

Liam: (laugh).

Adrian: I mean, everyone brings with them their, their background and their experiences. So I mean, there are technical things. Um, for example, uh, I think, uh, Will Larche was talking about, um, color spaces a little bit, the HCT color space and um, uh, how we had to sort of invent a new color space to better represent what we were trying to do with, um, personalization, uh, and those sort of related technologies. And it was actually really fun watching that develop. I was chatting with some of the engineers at some point, but, uh, I

did not personally work on that work, but it was really great to see a new color space come to be born.

You know, in the world there are many color spaces. You know, everything from like spectral stuff that, uh, you know, astronomers would use all the way down to some very simple things like RGB, um, and everything in between, you know. And it's, it's been sort of slowly developing for, I don't know, 50 or 70 years by this point. Um, but it's always exciting to sort of see a new one get born.

I think the other thing that a PhD teaches you is, um, perseverance. no one survives a PhD program without a lot of just, uh, you know, get your- put your head down and kind of crank out a bunch of work and, um, keep it going year over year over year. Um, so that's something I feel like I, I did learn. And, um, I have been very lucky to be on Material, uh, working here for over 10 years now, um, but there's an element of that, of, of trying not to bounce off of problems. You know, when you find a problem, dig in and kind of sit with it and-

Liam: Yeah.

Adrian: You know, maybe sit on it, uh, and not just sort of like, oh, this is getting too hard, I gotta- I gotta go do something else. Um, 'cause you know, we all have our various, uh, demands on our time and, uh, sometimes it's easy to go pay attention to the other things instead of tackling that really thorny problem. so yes, I mean, although I am- I'm not generating algorithms for, you know, X, Y, Z or anything like that. It's, uh,

the kind of technical content of the PhD is now out of date. And, uh, I didn't actually use it in the sort of, um, academic sense, which is what most PhD training really is.

Liam: Right. I know if, um, colleague, friend and friend of the show Yasmine-

Adrian: Mm-hmm.

Liam: Is listening, she will know what you're talking about when you talk about committing to the problem over time, especially at Google, it's very, like, our plans are measured in centuries, kind of.

Adrian: Yes.

Liam: Sometimes.

Adrian: 100%. Also, hi Yasmine (laugh). Yeah.

Liam: Shout out to her.

Adrian: Exactly.

Liam: When you mentioned the color space specifically and like seeing a new color space get born, I think this is something... We've had a few, like really interesting articles that go into how the color space works and like, what it allows us to do and things like that. But I'm interested in like, why it's so exciting to see something like that come into the world.

Adrian: Um, that's a good question. I mean, from a design point of view, all of these color spaces and all of these abstractions, uh, change how you can look at things. Like, uh, you know, uh, color comes from physics and, you know, all the light that's shining down on us right now has all these wavelengths and all this sort of stuff. But that's almost like, that's way too complex, um, to handle in any kind of realistic sense. I mean, like I said, like if you're doing astronomy, sure. If you're doing, you know, chemistry, sure.

But, um, for people that are responsible for doing things like either color on screens, which is already a sort of limited slice of the universe, uh, or color in printing, so that's another huge traditional place where color spaces, um, thrive, um, or color matching or looking at a table right now, you know, matching a color to the tone of this table or something, um, the color space that you work in is kind of like the language that shapes, uh, your thinking, you know, about the problem. So it's exciting to see a new color space come out of, uh, a new set of requirements really. It's a... It's a really interesting design problem, not because it's not software.

I mean, I know there's software supporting it, but it's really a, a mathematical model, you know, of what are the important parts of this giant, uh, you know, nearly infinite space of things? What is the most important parts of them, and how can we collapse those important parts down into like a set of numbers or, you know, a set of smaller

concepts or something like that, um, that make it useful and fruitful and convenient to use?

So to, to see a new one pop up... First off, I mean, there are a lot of color spaces, and I would probably say 90% of them have, you know, gone by the wayside. I mean, they just didn't survive the- uh, in the landscape of competition, of, of ideas. Uh, and so to see a new baby one come out (laugh) and, uh, and see where it's coming from, like why- you know, why did we make the choices we did? Why did we need something new? You know, which is always a great question. Um, uh, that's an exciting moment. And I think it's probably too early to tell, but maybe, you know, in five or 10 years' time when we're looking back and maybe you're on the Wikipedia page for color spaces or something, and I really hope that, you know, that one pops up. It'd be nice.

Liam: Yeah. We can talk about it again, uh, during the M20 campaign in 2034

Adrian: Exactly. I can't wait. It's gonna be great (laugh).

Liam: Um, that, that, uh, is really interesting to me because it has so much to do with perception and, you know-

Adrian: Yeah.

Liam: You're saying like color is based in physics and stuff. We know that people perceive colors differently. It brought to mind how, uh, working in engineering on a system at the scale of something like Material, you are probably dealing a

lot with finding ways to abstract perception to, to essentially like bring this subjective experience down into something more apprehendable, which is code or a mathematical model or something like that.

Adrian: Yeah. And I think, um, a lot of my work involves design systems, you know, uh, Material Design being the sort of parental design system at Google. But, um, I don't know if everyone listening knows this, but there are many design systems at Google and, uh, they're children of Material Design, uh, in a very literal sense. Uh, but like, um, Maps will have their own design system, and Search has their own, and... They're all born from Material Design, but they have their own local work to specialize to their domain, like a map or something. Um, and design systems in themselves are an interesting abstraction of all the possible things we could do on a computer screen or on a, you know, a phone.

Uh, it's a really interesting and almost brutally hard problem to sort of think about. Like, really, when I'm looking at a laptop like I am right now, there's, you know, so many pixels by so many pixels in a grid. They all can take on so many different colors. it could display pure static, we could display, you know, rendered 3D shapes. We could do all sorts of stuff, but it's too complex to think about it that way. And, um, not only for the designer and the, uh, person building, uh, the experience on the screen, but also for anyone trying to understand it and use it to get, you know, real things done.

Um, so we boil things down, you know. We... We create these abstractions, we create these systems. And in part of creating those abstractions and systems, we have to explain them to people, right? So I think this is where the storytelling and the, um, visualization maybe part of abstractions, like design systems come in, which is very interesting. So how do you explain, like, I sat down for six months, I figured out this great system of boxes and things on the screen that I- you know, I think I can make word processors and podcast apps and whatever with, and it's, it's great? But then how do you get that outta your head, um, that six months of hard work and into the heads of people have to build stuff on it, people have to use it, all that kind of stuff?

And for me, that's one of the difficult and interesting problems of building these big abstractions. Um, and, uh, there's a, a shout-out to the original Material actually. Um, they had a beautiful... It's just this beautiful, um, way of describing it that everything was supposed to be essentially cut out from pieces of paper. They were literal pieces of Material, right? I don't know if you ever saw this, but, the original Material Design, uh, they actually took real paper, cut things out and saw how the light and the shadows worked. And they have these- we have these great photographs and some of the historical stuff of them like setting up lamps on a table and casting shadows.

And that's how they built the original, um, interface components, um, by analogy, but sometimes almost just by copying, uh, these physical things. And the, the wonderful

thing about that is that it's pretty easy to explain to people. You know, like, we're all used to picking up pieces of paper and working with things and shadows and light and all that kind of stuff. Uh, of course you can go too far. You can go into the skeuomorphism, you know, which is, you know, I don't know, maybe entertaining, but, but not the most useful thing. Um, and I think further iterations of Material, M2 and M3 and everything that's, you know, to come, uh, have definitely wandered away from any kind of physical metaphor.

But explaining that, um, geez, we just spent so much time trying to figure out this system, explaining it to other people via a physical metaphor, uh, that was brilliant. I thought that was- it was really great. And, uh, um, I got to join the team actually just, just... It was just like a year-ish after the original release of Material. And I remember being so impressed with this sort of like, oh, this is a very clear system, you know. It was also very small and (laugh) maybe, uh, not so tested at that time, you know, and not hit up against the harsh bumps of reality as much, but, um, it really was a very nice, uh, clean system.

Um, so there's something in there about having an abstraction. Not only having abstraction that is useful, um, and, uh, fits the problem space that you're trying to solve, but also is, you can explain it to other humans. If you can't explain your abstraction to other humans, it's- you know, it's gonna be really tough.

Liam: Yeah, it's interesting, this point of like consolidating all of these ideas and then needing to turn them so that they come back out and kind of like spark the same imagination in other people. A previous guest on the show, Judith Donath, um, who's done a lot of work on, um, sociable computing and topics of the interface and things like that, um, said that metaphor has kind of always been one of the fundamental ways that we explain things that people have not encountered before.

Adrian: That makes sense.

Liam: Obviously, we see that reflected, like in the original version of Material. It had like an extremely strong metaphorical basis to the point that, yeah, you're right, like, there's videos of like Christian in his garage with-

Adrian: That was it. Thank you.

Liam: Flood lights and, and these paper rigs. Um, but the system has really evolved since then, and I think that that, like, one to one metaphorical basis is no longer there. And I wonder, one, how important you think that metaphorical basis is today- in today's context, and two, if you think that the software interface is something that has become so established that we don't need to rely as much on metaphor to explain the fundamentals as we might have 10 years ago.

Adrian: Yeah, 100%. The... Um, the initial problem of explaining how to use something to people who did not grow up with it or are not familiar with it is, is like real, and probably

something that happens all the time. I don't know, like maybe self-driving cars are having this problem today or something. Um, certainly anything to do with AI, we're having that problem today. No one has any clue of how to like hold or handle an AI thing right now. It's, you know, all over the place. Um, and I think with, uh, uh, orig- original, um, you know, metaphors like the desktop metaphor, I think, uh, you've talked about before, um, uh, the sort of paper Material type thing for, for original Material, all these things are familiar, uh, make things comfortable.

Um, hopefully make it understandable enough that people can make leaps based on what they know about things and have it applied to the actual digital imaginary interface. Um, so they're, they're very useful that way. But I think as people get more and more used to this and are like literally born into, you know, uh, like the cell phone era and, and things like that, uh, we are given the freedom to wander away and it's okay. People don't get mad. They... You know, um, I, I think the skeuomorphism thing is, is really interesting. I'm sure there's people who study that, uh, you know, academically and, and, uh, in interesting ways. But we, we just don't design things with faux wood grain and things that look like, you know, a podcasting app with a little microphone and stuff like that.

Uh, because partially I think we don't need to anymore, you know. Uh, we don't really need to spoonfeed, um, users on these, these concepts, at least because they grew up with them, they know how to use them, they're pretty familiar with them. Um, we no longer have (laugh)... Uh, you know,

we no longer have in high school courses that teach people about how to use a mouse, you know, although I'm sure maybe there's still typing, I don't know. Uh, but, uh, there- we don't really need it anymore. So it allows for more freedom of expression, which to me is exciting but also you could wander too far afield. You could... You know, your, your, your, uh, design UI elements or whatever could get too abstract or too- maybe too inconsistent.

Um, so like, I, I think, as you know, working on a design system, uh, one of the important things is to make it as consistent as possible, to have some principles, whatever they are, and have those shine through everywhere in the design system, in the UI elements and, and how they interact. Um, so I, I think there's- you, you sort of run the risk of like, it gets too floofy, it gets too abstract, it's no longer holding together, and then, you know, what do we got? Uh, I don't know, sort of falls apart.

Liam: Yeah, that's an interesting point that actually so many possibilities are opened up that you'd need to create kind of a- your own pocket world that-

Adrian: Yeah.

Liam: Holds some internal consistency.

Adrian: I love that.

Liam: So that people can build a model of how this works.

Adrian: Yeah, I love that. One of my friends is, um, a faculty at- of, um, of game design. Actually. His name's Andy Neland. And, uh, we talk about... You know, we play games together. We've been playing games together for, for a long time, uh, before he was a professor of games. But, uh, we p- play games together and, and we often talk about, um, sort of video games are really interesting because people aren't surprised that they have to learn the interface every time they sit down with a new game, right? You sit down with a new game and honestly, if it's a console game, you probably know how all the buttons work.

If it's, you know, a point and click thing, you probably know how most of this works, but there's always this tutorial phase, this, this opening phase where you're like learning, oh, that's where the jump button is. Or I don't know, there's no jumping in this game. Fine, I have to figure other things out. Um, and people aren't like surprised or shocked or bothered by that. Um, so it's, it's a really interesting sort of, uh, you said pocket world, I think.

Liam: Mm-hmm.

Adrian: Um, every game is a little pocket world for UI design, um, but also, you know, physics and inter- you know, because in games they can do whatever they want. So physics and interactions and, um, how characters interact, uh, do you speak to people? Do you not? Um, who are the bad guys? Who are not the bad guys? All that kind of stuff. Um, I find that really interesting 'cause it feels like- sometimes it almost feels like I can- I can see the original like game

designer sitting there and going like, like, trying to talk to me telepathically through the design of the game, right? They're not there in the room to say anything. And sometimes you get frustrated with it, you're like, "Why did you put this thing that looks like I can jump on it, but I can't jump on it?"

Liam: (laugh).

Adrian: You know, like, what, what are you doing? Uh, and I sometimes wish I could have that conversation with, you know, the original game designer. But, um, yeah, I love that idea of a little pocket world of, um, of interface or, or, um, something that can explain how to interact with things. That's great.

Liam: So in... Like, in recognition of that kind of possibility, does that then make the, the task of kind of engineering or engineering leadership even more difficult when you have like such a broad range? Because I know on Material, I've seen folks draw things or animate things that, um, you know, are, are brilliantly far away from anything that we know to be possible in the physical world.

Adrian: Yeah.

Liam: So I can imagine that looking at something like that and, and bringing it back down into the system can, can be, uh, intimidating.

Adrian: Intimidating is one word for it. Yeah. Um-

Liam: (laugh).

Adrian: Uh, there is forever a tension between design and engineering. And, um, ideally it's a creative tension that great things come out of, but sometimes it's a little tricky. And... And motion, I think actually is- it's a good example, is one of those places where we have not done well enough in the world of, uh, UI design. Uh, there have been so many, um, great, like, sort of I'd say micro design systems of, of motion about, oh, like, well, when you interact with this, it does a certain thing, or, um, or maybe tunable. So we could have something that's more hyperactive, we have something that's more calm and enterprisey. Uh, and they rarely get kind of well-translated when you get down into the, the code and the implementations and things.

Um, and it's... Uh, it's a really tricky, uh, kind of balance to make because, uh, the, the designer is designing, um, in, well, ideally not in a vacuum, no. I mean, a designer's job is to not, you know, necessarily be in a- in a vacuum. They're bringing all their experience and all their, you know, uh, world knowledge in with them, but in the end, they're not designing typically by writing code on an Android device. You know, that's, that's just not kind of how the process works. So they design some, some motion, some awesome, whatever bouncy thing. Uh, sometimes an element in isolation, you know, will have motion, but maybe we're not thinking about, okay, that's one button on a crowded screen of all sorts of stuff.

How does that... You know, maybe that makes the button, I don't know, overlap other things and that's tricky. Um, I think one thing that's interesting about being an engineer is in the end, your job is to make it all work. Um, and so there are often- engineering is often left, um, figuring out, uh, a lot of the little details and the little problems that either were not thought of or maybe they were thought of, but not documented, so we're not dead sure of what we're supposed to do. Um, so it- it's an interesting... The transition from design to engineering is, um, a really interesting one. And, uh, ideally it's working very well, you know. So you've got a designer and they sit next to your engineer, the- you know, the dream.

You've got a great designer sitting next to a great engineer at the literal table, and they- you know, they're bouncing ideas back and forth, oh, this doesn't work, oh, this does work. Great, let's do this. But in reality, uh, in these large company settings, you know, uh, design is designing a thing, kind of packaging up and maybe putting a bow on it and then handing it off to engineering for feedback or maybe even just final implementation. And then the feedback starts coming, oh, this doesn't work and that doesn't work and this doesn't work. And, uh, you know, if there's enough time, that actually rounds off the sharp edges of the whole thing and we end up with something that's better, um-

Liam: Mm-hmm.

Adrian: But I don't know if that's always the case.

Liam: Yeah. we talked earlier about how like our entire team basically is collectively, uh, speaking to the outside world and perhaps many teams within the outside world. And it's our job to kind of consolidate all of these ideas into some form of communication that then comes back out and helps everyone understand what it is that they're using that we've produced. And I'm kind of seeing like a smaller version of that playing out, like, um, among teams who might have to do certain parts of the work asynchronously from design to engineering.

Adrian: Yeah.

Liam: How do you, uh, convey the rationale in a way that sparks the same understanding in other people-

Adrian: Yeah.

Liam: That you had when you made the thing? I think it's the same also, like when I work with another designer, when I work with an engineer, when I work with an editor, it's like, not only, you know, here's the thing that I want to do, but also, you know, they might say, "Here's how it breaks." And then I have to say, "Okay, here was my intention. Like, what might we do..." (laugh)

Adrian: Yeah.

Liam: "In order to resolve that?"

Adrian: And I, I think that... Well, I mean, uh, the hard problems are always the human problems. And I think one of the hardest things is communicating intent, like you said.

Liam: Mm-hmm.

Adrian: Um, and I think one of the good things... I mean, we're talking about Material 10 here, coming back to the- either the original Material Design or its current incarnations, I think one of the greatest things about it was that we had all of this write- written guidance about how it's intended to be used. It's not just a sticker sheet of all the button variations and slider variations or something. It's trying to explain, this is what we were thinking. This is sort of how we think these things go together. Um, all the way down to like dos and don'ts for using a button.

Liam: Mm-hmm.

Adrian: You know, that kind of stuff. Um, but communicating that intent, especially when you're not there or can't be there because of geography, because of time, um, because of time and the way that (laugh) maybe, maybe you, you, you did all this work in designing all this stuff, but we decided not to build it just yet. And then 18 months later, we're like, "Cool, now we have time to build it." And, you know, that was 18 months ago. So there's... You know, there's, there's barriers to communication sort of everywhere. Uh, and I think it's, it's really tough because the... It... It's... It's hard because your, your intention... And I mean, maybe you had a beautifully fleshed out model in your head of how all this

hangs together and stuff. Um, and that's great, but if it didn't get kind of like written down and recorded and-

Liam: Mm-hmm.

Adrian: Um, in some form that, say an engineer can understand, um, it's not... I don't wanna say it's lost, but it's, it's gonna make the job way harder. Um, it's a really interesting and I'd say permanent problem. Like, I don't think AI is gonna solve it for us. I don't think anything's gonna solve that for us as long as there's people, um, collaborating on something, especially people with different backgrounds. So, um, I would... In an ideal world, I would love every single like UI designer to have, uh, engineering experience. And I would love every single engineer out there to have to go through a design sprint and a design crit and all that kind of stuff. It... In reality, we know that's not really the case. So without that shared background and experience...

Um, I mean, you are an exception, sir. Uh, but without that kind of shared background and experience for both sides of the, of the, the process, it- it's really tough. Like, the language is different, the metaphors are different, your training was different, you know, all of that. Um, it's a- it's a hard problem.

Liam: Yeah. So Design Notes is going on record as saying designers should code (laugh).

Adrian: Designers should code and I mean, engineers should design. That's... I-

Liam: Yeah.

Adrian: I would be super excited to see that. Yeah.

Liam: Totally. Um, yeah, I think there was a really interesting point you made about the- um, again, about the original Material Design spec, which was, uh, that there was so much of the intent behind the system embedded in the way that it was described there. And that resonates with me because I feel like over the years I've heard over and over again at I/O or just online or wherever, that those specs were a lot of people's, including engineers', first introduction to design.

Adrian: 100%.

Liam: Period.

Adrian: Yep.

Liam: Um, and I wonder... Like, the system has made some major shifts over the past 10 years with personalization, tokens, like new technical complexities, more components. Um, we're at a point now where I feel the system is actually quite powerful in the way that it abstracts things. And I think we also face a challenge in conveying that same level of intent in a way that, that comes out to the overall audience out there, and I'm curious what you think about that. As we grow in complexity, how do we keep explaining or keep, uh, embedding that knowledge or building that model?

Adrian: Yeah, that's a great question. Um, I know, uh, for instance, Material Design, the sort of the, the public spec that we have up on the web is used, um, in a lot of, uh, design curriculum, you know, which is great. You know, it's, it's such an amazing thing that someone's like teaching off of this or using this as a- as a resource in the, the teaching. I think it's super cool. I think-

Liam: Shout out to, Austin and Barbara and Megan and Euphrates and the whole team (laugh).

Adrian: 100%. I am extremely lucky to sit with some of those folks. Uh, my, my desk, uh, is right next to all of them, or a bunch of them and, uh, I'm- yeah, I'm a lucky guy. I'm a lucky engineer (laugh). Um, uh, the complexity question is really interesting and really hard. Uh, so like in software, there's a very, I mean, well-understood, not solved, but well-understood problem of systems that grow over time, right? So if I'm creating... Um, let's say I'm creating a 1.0 of, um, uh, a mapping API. So I can... You know, you can- you can hit this API and ask for, I don't know, um, the coordinates of the closest, uh, coffee shop, and it will, you know, spit back something, whatever.

You can imagine, some kind of API. And at the beginning it starts small, it's not- starts tight. Um, every different API entry point, every different call you can make follows the same patterns and the same like error responses and all this kind of stuff. Um, and that's about as tight as it's going to stay at the beginning (laugh), right?

Liam: Mm-hmm.

Adrian: Because over time, say your API gets very popular, or maybe it's a product, maybe it's being sold, um, you get these feature requests, uh, certainly bugs will show up. You miss something, we're all human. Um, uh, but you get feature requests to build on it, you start adding more things. And we all try to stay consistent as much as possible. Uh, but, uh, again, being human, we're gonna fail. Um, and sometimes there's serious business concerns, you know. Like, if you're at a startup and if you don't add this the next three days, we might all go down, you know. Um, so over time, the complexity of the interface grows. Even if it's, um, with the best of intentions, the more and more and more we add to this thing, they get sort of- it gets heavier and heavier and kind of creakier and harder to understand.

And I, I think this is a very natural result of successful systems, any kind of system. Um, and, uh, a lot of the time, at least in software, they... You know, you can see this in open source projects, actually. You know, you've got some... This is, I mean co... It's so common. You've got some very well-known library, say it's a mapping library or something. It's had a 1.0, now it's up to 1.7, whatever. It's... It's had many iterations, but it's getting creaky, creaky, creaky. And somewhere at some point, someone is gonna say, "You know what, I'm gonna rewrite it. Like, we're gonna start the 2.0 project." And it- you know, open source, it causes the schism in the community. And there's people that, no, no,

no, keep the original. No, no, no, I wanna build this new thing.

Uh, and they try to start again, because basically the complexity gets too high to, to make progress. Um, uh, there's something similar in terms of... You've probably bumped into this, uh, in terms of teams who maintain... Sorry, software teams who maintain a product. So at the beginning-

Liam: Mm-hmm.

Adrian: You know, I've got my product. Maybe I've got 10 engineers, great. And we're adding features at like, you know, I don't know, one feature a week or something, great. But then the size of your project or product is growing. Um, so the number of feature requests and bugs and, and the maintenance involved grows and grows and grows until the point you get to some point where the product is so big that the maintenance cost is 10 engineers. And at that point, you will never add another feature because it has grown to the size of the team who supports it. And... You know, and then you have to- you gotta do something drastic to, to, to fix that. Um, I've gone far afield of your original question, (laugh), I think.

Liam: No. No, but it's good because I'm, I'm also wondering, like... you know, obviously maintenance is one thing that I've bumped into, uh, over the years, but also I'm curious about code as a constraint or like the complexity of the system becoming a constraint in the sense that, you know, uh, no

spoilers, but I've proposed features for existing components that, you know, the components are built in a certain way in a library that people are actively using. And, uh, certain new behaviors would make it really hard to kind of build that component in the way that it currently-

Adrian: Yep.

Liam: Exists. So where design runs up against what is possible with what is built, um, how do you think about resolving that? How do you know when the balance of changes that you would have to make in the way that the code is built, uh, is worth it?

Adrian: It's a good question. Um, so I would say software engineering- engineering in general is, um, a game of trade-offs, right? It's always, we can do anything, almost anything. Uh, software engineering is a kind of wild world these days of like, it's a- there's a lot of magic. It seems like magic probably from the outside, but... And of course there's a billion constraints on the inside, but within those two, you know, bounds, there's a ton of room to build things in many ways. So like you said, so you're- let's say to be concrete, you're proposing, um, a new animation style for an existing button that- but that button is used in a couple hundred places in every, say... Let's say it's internal, every Google app. Great. How many Google apps are there? I don't even know. But there's- (laugh) dozens is too small of a number, right?

So, uh, that button is being used everywhere, and you want to add this new motion constraint to it. But the original

authors of the button weren't even thinking about motion. They were just like, "Oh, well, we know what the right motion for the button. It goes..." I don't know, "Click, click." Or something like that. Um, uh, and they have no concept or no ability in there to add anything new in terms of the motion curves or, or whatever it has to do. Uh, at that point, it's a- that's a big trade off, right? So you have one designer who has worked very hard and has brought all of their knowledge and training to bear against some problem. Um, and their answer is, okay, cool, what we need to do is add this new motion curve to this button.

That's like 100% cool, and that is exactly why, you know, designers are awesome and that's, that's why they're there. And then we have all these engineers, uh, who are responsible for, you know, shipping this button and getting into the apps and all the bugs that come back and all this kind of stuff who are saying like, "Yeah, that's cool, but for us to change this, say several thousand instances of this button everywhere, it would take us..." I don't know, "Six months of work or something." Um, uh, especially at something the size of Google. And then someone, has to balance the, you know, the, the, the trade-offs between the, the benefit that new thing would bring against all that engineering time.

And one thing I will say is a lot of engineers, especially if they're not- um, haven't been design adjacent and haven't worked with designers tightly, uh, I think a lot of engineers sometimes have this sort of crusty response of like, no, I don't wanna change anything. Like, it's working, leave it

alone, you know (laugh). Why are you changing my buttons, basically? Uh, you know, we, we get this all the time in Material Design. And I, I, I respect the position, but I think it's also a little shortsighted sometimes. Um, and, and someone needs to, you know, bridge the gap between those two, um, and ultimately make a call, at least in an organization like Google. Um, in a startup, I'm, I'm sure you could do something a little bit more agile. Um, the trade-offs are there, but you know, the scope is smaller.

Um, I, I, I don't think there's... The... The tension between those two things is, is, is never really going away. Um, and I, I don't know, I, I mean, the, the answer is always like, shove everyone in a room and make them figure it out. You know, get the humans talking to, to actually-

Liam: Mm-hmm.

Adrian: Talk about it. Um, but yeah, uh, making those kinds of changes is, is very hard.

Liam: Yeah. I think, um, definitely at previous jobs I've worked with the software engineers who, like, the first reaction that I get from a proposal is like, that simply isn't possible (laugh).

Adrian: Yes.

Liam: And I think something that we, um, have mostly gotten the hang of on the design side is answering every question with, well, it depends (laugh).

Adrian: Yes. Well, it depends (laugh).

Liam: I think that's ultimately the answer to everything.

Adrian: Yeah. And... And to come back to what you were saying before that... I... I know these are imaginary people we're making up here, but that engineer that says, um, "That simply isn't possible," maybe with this kind of like slightly dumbfounded, irritated view look on their face, like, they're thinking of the abstraction, to go back again. The abstraction that has sort of been built into that button, say, you know-

Liam: Mm-hmm,

Adrian: It's... It's... I can change the... Well, you know, I can change the font, I can change the color, but no, there is no way for me to change the motion curves, or no, there's no way for me to change the corner radius or something. And of course, there is in the sense that it's all software. I mean, we could- you can rip down or rip, rip things out, uh, and get pretty low down in the stack if you have to. But in their heads, their day-to-day conception of that button, uh, which exists for a very good reason, they have that conception for a very good reason, that their day-to-day conception of that button doesn't include these affordances that the motion and the corner radiuses and stuff.

So for them in that second, like, it's a- it's a true response. You know, like it's a real, like, I don't know what you mean.

Like, this brick is a brick. I don't know what you want me to do with this brick (laugh). You know?

Liam: Yeah.

Adrian: It's, what's going on here? But the, the one thing that's exciting about software, um, development is that, uh, you know, you could- you can disassemble the brick, you can bring the brick down to its component atoms and start again to-

Liam: Right.

Adrian: You know, to kind of, you know, bust this analogy badly. Uh, but, uh, I, I like that kind of reaction of like, oh, that's, that's impossible. Because there- you can clearly see at what level does their level of abstraction sort of stop.

Liam: Yeah.

Adrian: Or what's their working concept, you know?

Liam: Yeah. There's a concept. Um, again, like I have a background in sociology and art, so I approach design from that perspective a lot. And I think one of the things that we are talking about here, which I think we've been talking about the whole time, is the concept of intersubjectivity, which is basically like person A has a perception of a phenomenon. In this case, it's like the button that is their like model-

Adrian: Yeah.

Liam: And informed the way that they built it. I also have my own model of the button. But intersubjectivity is like knowing how to have a perception of the other person's perception (laugh), how to orient yourself towards that person's mental model of the button, which I think is what we're talking about when we talk about how engineers need to have a design perspective and designers need to have an engineering perspective. We need to understand how each person is kind of modeling the same concept in their mind, because it's not the same, going all the way back to the beginning of the discussion with, with perception and color (laugh).

Adrian: Yeah. Um, yeah, I- it's- I honestly think it's like one of... Well, it's one of the biggest challenges in life, you know. Like, uh, I've got some... Or it's the biggest source of, or a big source of misunderstandings in life. You know, I've-

Liam: Mm-hmm.

Adrian: Uh, I don't know. Uh, I've (laugh)... I've got a certain sense of like how clean the apartment should be, and maybe your partner has a very different sense of how clean the apartment should be based on like, how they were raised and like-

Liam: Mm-hmm.

Adrian: I don't know, stuff that they see. Uh, but if you don't ever talk about that, uh, you know, you're gonna just be butting

heads continually. Um, it's similar. I mean, luckily for design and engineering, um, uh, one nice thing I think actually be- about this problem in a work context is that, um, as long as everyone is respectful and cool and trying to, you know, in general, do the right thing, um, uh, the way around this is, like I said, is we all sit down and ideally in the same room physically, but you know, in the same virtual room and kind of talk it out. And then I learn your model and you learn my model.

And I love, I love, I love being in some of these problem solving sessions where we're like, "You know, no, we can't add motion to this button. It's impossible." I absolutely love... You can see the light go off in someone's eyes when they're like, "Oh, wait, you wanna do what?" And then you can see them finally mapping the two models together.

Liam: Yeah.

Adrian: And... And... And figuring out what you wanted. And then, like... And then usually it goes fast and it's like, oh, cool. Yeah, we can do this. We can do this. Um-

Liam: Yeah.

Adrian: Goes from there.

Liam: I think it's... I think it's... Um, I feel like we've had a breakthrough on this episode (laugh).

Adrian: (laugh).

Liam: I feel like designer-developer collaboration is unlocked. I feel like this makes it... Like, this perspective on things makes it such a more apprehendable problem. It's not like... It's not something that's unique to our work environment or our roles within that environment. It's actually just what we're doing all the time every day.

Adrian: Yeah, 100%. It's just been professionalized. And, um, I often think about the training we all got. Like, I didn't... I get... I didn't get trained as a designer. Um, it sounds intense. I've never been through a design crit despite me claiming I should do that. Uh, sounds intense. Um, and, uh, designers who came up through sort of a traditional training also didn't- uh, you know, didn't stay up all night in the engineering lab, like getting whatever done. Um, and we... So we've just... There's... There's different cultures, you know, kind of bumping up against each other, and ideally that is rich and fertile, and everyone is excited to learn about their- you know, uh, their coworkers, how they're thinking, all this kind of stuff.

Sometimes time pressures and geographic differences and, uh, time zone differences and stuff leads us to butting heads a little bit more than it should be. I... I... I hate the... I hate the standard curmudgeonly engineer thing of like, oh, well, design just designs these wild things and then throws them over the wall, and now I gotta learn how to build them, you know, or figure out how to build this thing. I... I really dislike that because, uh, you know, design is amazing. They do amazing stuff. They're also humans who sometimes

mess things up or don't understand, say, like, um, you know, really common things like, uh, my phone needs to run at 60 frames per second. That is the standard set by the system.

And if it doesn't run at that, like, there are severe consequences, therefore there are real constraints on your motion curve or whatever. Uh, but designers can correct that and then like, you know, we can get better together. But, um-

Liam: Yeah.

Adrian: It's-

Liam: Also... Also like, get over here, you know?

Adrian: Yeah.

Liam: The wall is not endless (laugh).

Adrian: Exactly. Yeah.

Liam: Come over to this side and, and, and I'll do the same.

Adrian: Yeah. Yeah.

Liam: Um, I want to wrap up, you know. Um, ostensibly we're here to talk about the history of the design-

Adrian: Oh, [inaudible 00:49:03]

Liam: System in the past 10 years, all of that stuff. but I also want to take all of that information that we've just talked about and, and get your idea, your extrapolation into what's gonna happen next. What do you want to happen next?

Adrian: Sure. Uh, so I see a fork in the road. I'm sure you see a similar fork. Um, and the fork in the road is this. We have the juggernaut of AI, um, bounding towards us. Um, obviously Google and many other companies are working very hard to figure out how AI might solve real problems Uh, but in terms of user interfaces, uh, one branch of the road is... Well, one branch of the road is this sort of, I would say, I guess now traditional UI design paradigm, right? So we work with designers, they come up with design systems. We, um, uh, come up with color systems and motion systems, and we, um, codify them into design tokens, which then get shipped to engineering.

And engineering uses those things to build experiences that turn into apps. That's sort of... I actually don't see that changing too, too much. There's a different branch in the road, uh, that I've heard people talk about, which is my phone or my app or something will basically become an empty container. Um, and then AI will, on the fly based on my needs and my preferences and what I'm asking for and stuff, ship UI down the wire or down the radio waves into my phone, and the phone's UI will, you know, update spontaneously based on the current task and, and this kind of stuff. Um, so sort of a fully AI-driven real time UI. Uh, I am having a hard time imagining that second branch of the

road being sort of practical, um, but it is totally possible I'm, I'm wrong.

But the, the future as I'm seeing it right now is, uh, a little fuzzy, uh, as is many things with AI. Uh, I have a hard time imagining a fully AI-driven UI, but, um, I would be excited to be wrong. Um, and I think the interesting- the more interesting part will be how does, um, new technologies like AI or whatever's coming next, um, modify the traditional design and engineering, um, uh, workflows? Uh, so... Um, so for example, one thing right now is designers, at least at Google and in many other places, are the ones who have to go through their design systems or go through their designs and pull out the design tokens, right? They need to pull out, oh, the primary color is FFF, the something else is something, something something.

And then they kind of either input those into a program or write them down in spreadsheet or whatever the workflow is. And then some... You know, somehow those get transformed to code. You know, it's all sort of dependent on your environment. Um, maybe that's a place where something like an AI could help extract that step and basically make it transparent. Like, make it to the point where we, kind of engineers, consider like a compiler or something. It's just like, oh yeah, I don't even think about all the machinations that are happening to extract that stuff.

Liam: Mm-hmm.

Adrian: And it works almost perfectly or perfectly enough that I can forget about it. Um, and that would be... Like if, if we get... Those are the interesting bits. How do... How do we make these technologies actually help this sort of human process? Um-

Liam: Mm-hmm.

Adrian: I'm hoping we don't end up in some spot where all these new technologies are like eliminating the human process, uh, 'cause I think that would be a giant loss and kind of lame and boring. Um-

Liam: Yeah.

Adrian: But it's possible. I don't know.

Liam: The... The optimistic, perhaps naive part of me thinks that humans are always gonna be human processing.

Adrian: Yeah (laugh).

Liam: I don't know. I don't know how that could be replaced. I think if humanity has no creative motivation, then we're like living in a different dimension (laugh).

Adrian: Yeah. Uh, I guess... I think some humans will always be doing the creative stuff. I... I worry about, um, how many of them can pay rent doing so-

Liam: Mm-hmm.

Adrian: With some of these technologies coming down the pipe. I don't know. Uh, it's, uh, let's say uncomfortably exciting.

Liam: Mm-hmm.

Adrian: (laugh). Um, we'll, we'll see where... We'll see where it goes.

Liam: Yeah. For my part, I have to say, um, you know, another thing I did when I was in school was glass blowing and-

Adrian: Nice.

Liam: My professor was really good friends with an artist, um, who worked in Venice and he had a big studio and he had many people who came out of that studio, uh, to start their own st- studios. And one student in particular, um, ended up copying one of the teacher's most, most iconic works-

Adrian: Mm-hmm.

Liam: And... And kind of selling it.

Adrian: Mm-hmm.

Liam: And people always ask this guy in interviews, they said like, "Aren't you upset that one of your students is like, ripping off your, your coolest piece or whatever?" And... And the guy was like, "No, it doesn't bother me because I think people who, uh, encounter this work will always know the difference."

Adrian: Huh, interesting.

Liam: And I, I do think it's the same for the interface.

Adrian: Yes.

Liam: We will always feel the difference.

Adrian: Yes. These... These things that have high touch, literal touch, but high, um, we're, we're experiencing them pretty deeply, I would say, or at least, um, uniformly through the day and, and regularly and stuff. I agree, you, you definitely feel it. It's like, um, font selection in books or something. Sometimes you pick up a book and you're just like, "Oh, nope, not reading that 'cause-"

Liam: Mm-hmm.

Adrian: "I don't know what the heck that font is." Uh, I don't know if it registers like, uh, consciously, but um, yeah, I agree, we're very sensitive to these things and, uh, yeah.

Liam: Thank you again, Adrian, for joining me.

Adrian: Thank you so much, Liam.

Liam: This was a really cool episode.

Adrian: Yeah, thanks so much. It was really fun to be here.