EPISODE 997

[INTRODUCTION]

[0:00:00.3] JM: Animations can be used to create games, app tutorials and user interface components. Animations can be seen in messaging apps, where animated reactions can convey rich feelings over a text interface. Loading screens can become less boring through animation and voice assistant products can feel more alive through animation. We still don't see much animation in our everyday applications. This is partly because animation tooling is difficult to use.

To make an animation, the typical workflow is to go into a tool like After Effects, render your animation and then export that animation in a movie format. This format is not dynamic enough to be easily used on the wide variety of development platforms, such as web and mobile and React. The animation library Lottie did improve the animation tooling, by creating a system for exporting animations to JSON and allowing them to easily scale up and down as vectors. The animations still were simple and unidirectional. The developer did not have much freedom for how to move an animation in response to user input.

Rive is a system for creating dynamic, animated movable objects. Rive allows for the creation of animated elements that respond to user input. Rive has a tool that runs in the browser and allows the user to define the animation. The animations in Rive use a bone system that allows animators and designers to define the points of the animated sprite that the developer can then manipulate with code. This improves the painful handoff process that exists between animators and developers and it gives the developer some programmatic control.

Guido and Luigi Rosso are the founders of Rive and they join the show to talk about the frictions of animation tooling and what they have built to improve the ecosystem. Rive is an impressive tool. If you're looking for something to build animations with, take a look at Rive.

Software Engineering Daily has partnered with SafeGraph for the SafeGraph fast-food data hackathon challenge. We're giving away $4,000 in cash prizes, as well as swag from Software Engineering Daily and SafeGraph. SafeGraph a geospatial data company, which curates a data

set of more than 6 million points of interest. SafeGraph has a high volume of location data. You can build apps and data science projects with that location data.

This is an open-ended interesting hackathon and if you've been looking for a creative opportunity to explore large data sets with the potential to win some cash prizes, this is a great opportunity. The hackathon is hosted on FindCollabs. There's a lot of freedom for what you can build with SafeGraph. SafeGraph is a data as a service company. To enter, you can go to findcollabs.com and sign up.

[SPONSOR MESSAGE]

[0:03:08.0] JM: Being on-call is hard, but having the right tools for the job can make it easier. When you wake up in the middle of the night to troubleshoot the database, you should be able to have the database monitoring information right in front of you. When you're out to dinner and your phone buzzes because your entire application is down, you should be able to easily find out who pushed code most recently, so that you can contact them and find out how to troubleshoot the issue.

VictorOps is a collaborative incident response tool. VictorOps brings your monitoring data and your collaboration tools into one place, so that you can fix issues more quickly and reduce the pain of on-call. Go to victorops/sedaily and get a free t-shirt when you try out VictorOps. It's not just any t-shirt, it's an on-call shirt. When you're on-call, your tools should make the experience as good as possible. These tools include a comfortable t-shirt. If you visit victorops.com/sedaily and try out VictorOps, you can get that comfortable t-shirt.

VictorOps integrates with all of your services; Slack, Splunk, CloudWatch, Datadog, New Relic. Over time, VictorOps improves and delivers more value to you through machine learning. If you want to hear about how VictorOps works, you can listen to our episode with Chris Riley. VictorOps is a collaborative incident response tool. You can learn more about it, as well as get a free t-shirt when you check it out at victorops.com/sedaily.

Thanks for listening and thanks to VictorOps for being a sponsor.

[INTERVIEW]

[0:04:59.5] JM: Guido and Luigi Rosso, welcome to Software Engineering Daily.

[0:05:03.3] GR: Oh, thanks for having us.

[0:05:04.0] LR: Yeah, thanks for having us.

[0:05:05.8] JM: The subject of today's show is going to be animation and particularly the technologies you guys have built around animation. I like to start by just asking a simple question, which is why don't I see very much animation in the applications that I use?

[0:05:22.2] GR: This is Guido. I guess, I'll tackle that first. I think it's a combination of things. There's a lot of initial resistance to animation like, “Oh, it's fluff. It doesn't really add value.” I think some of that initial reaction comes from, “I don't want to do the extra work, or it's hard to do that extra work.” I think a lot of companies nowadays with probably Apple and setting the bar with the iPhone and other Windows Phone 8 that came out a little later and had all these great animations that actually proved that they can improve UI and actually create a better UX with good animation, of course Google with their material design has done a lot of work there.

I think nowadays, there's a few things; tools aren't great. Probably what got us into animating UI was Flash and that was a really good tool that allowed designers to really drive the vision of the product from a design and an animation standpoint and actually build as you’re concepting the tool, build it around or whatever you're building the product that you're building, build it around design and animation as you're creating the concepts.

That never really happened again since Flash. There hasn't been a great tool that allows designers to really work on the assets that are actually going to go into production with high-quality design and high-quality animation. Flash had to go away for a lot of the reasons that it did. There was a lot of valid critiques to it. What has really missed since then in the UI field, I think is this ability for designers to really drive the vision of a animated product and an animated UI with the right tools to do that. It's just really hard to do that right now, right?

Now you have a designer will build something in one tool, then they'll maybe animate it in another tool that's not made for real-time animation, something like maybe After Effects that's made more for video animation. Then they hand that off to an engineer and say, “Well, here. Here are my key frames, here's a video describing how it was done. I might create some red lines and descriptions and stuff like that,” but it's still something that need to be completely recreated by the engineer.

That makes iteration really hard too, because if an engineer suddenly builds it out and now we're trying it, the designer might say, “Well, this actually doesn't work how I thought it would in the video when I was prototyping it. We got to go back and change that.” That whole process starts again. There isn't really a tool that's really great for sophisticated motion graphics at the level that we're talking about. There's little prototype tools that you can do transition from page A to page B. That's my belief as to why we don't see a lot of it. It's hard. It's hard to do and there aren't great tools to do it.

I think it's also been deprioritized for a long time and I think it's been recently that people really want these delightful experiences and there is the need to provide that. It is hard. It's hard to do that. It's hard to build that, either manually or build it from an engineering mindset. It really does take a lot of design and development effort in tandem to produce something that is delightful like that.

[0:08:33.1] JM: It sounds like the workflows are quite insufficient today, partly because the tooling is insufficient. You have it sounds like multiple handoffs, perhaps a handoff from an animator to a designer and from a designer to a developer. Perhaps the animator and the designer are the same person. What is the stack of technologies that people are using? You mentioned After Effects. I know there's also Blender. Give me the description of the stack of technology used for building animations and how the handoff occurs to the engineering department.

[0:09:14.0] GR: Sure. I can talk about it from the design side and then Luigi can probably cover the engineering side. We actually used to run before this, a design and development agency. We used to this work all the time for clients. I headed up the design side. By the way, I'm Guido and I'm the designer of the two of us.

On the design side, we'd often do a lot of mock-ups and designs and what you expect; Photoshop, Illustrator, Sketch. Figma wasn't around when we were running our agency, but now Figma has become a pretty big player in that field. We'd probably start there. We'd start doing some sketches and designs in that.

Then animation, like you mentioned After Effects. We'd probably export some of that stuff out, we'd cut up assets. We rarely do design work in After Effects. Sometimes it depends on the different designer that's working on it. That's really the bulk of it to create the level of animation, like when we were working with Windows Phone 8, Microsoft was one of our clients and we did a lot of metro UI concepts early on and animations for that. The work that we were doing there was all After Effects, to create that level of sophisticated animation that the UI required. You couldn't really build it in in another tool.

There's other tools for design. You mentioned Blender. Blender's a lot more 3D, but they do have a little bit of 2D now, but it's really more 2D for games, like it's game oriented and not a lot of UI work in there. Some companies are experimenting with Unity now, mostly because of virtual reality. Unity allows you to do stuff in the 3D environment. Really, yeah, from the design side, the stuff that we did from a motion prototyping perspective was mostly that.

Now before that, before we had died, Flash was probably our biggest tool, both on the prototyping side and on actually the implementation side. I do think that that was the biggest positive that Flash had was that you could do a lot of your prototyping and implementation and ultimately, your prototype could become your final product directly in the same environment.

You didn't have all this multiple handoff that had to happen. It was really great to be able to work on the assets that were actually your final product. That's something that we aspire to do with our tool, because we think it's the one thing that Flash got right, one of them many things, but probably one of the things, the best things was lost when Flash went away. I don't know if Luigi, you want to talk a bit about the tech stack. Oh actually, you had asked about how the handoff happens.

[0:11:45.9] JM: Sure. Yeah.

[0:11:47.2] GR: We did a lot of work with clients. One of the big ones we actually end up working for a long time at was Intel. We worked on a TV product with them. We actually have – we did a lot of After Effects videos there for that, to showcase how we wanted this TV product to look and feel. We knew it wanted to have a lot of motion and to feel just really sophisticated, like a level above your typical TV experience.

One of the things we found was super challenging was when the handoff had to happen and Luigi's team needed to rebuild it and I think they were doing it in WebGL at the time, or it might have even been in C++ and OpenGL. We tried a few different things. We did it in HTML5, then we went, just did the whole layout ourselves, because we couldn't find solutions to certain animation requirements in just pure CSS. We started exploring doing the layout in WebGL. We eventually went just a pure OpenGL with C++.

To help him and his team know how to recreate that stuff, we actually took all the videos that we had built in After Effects and we created redline videos, which basically are videos that it's hard to describe without seeing them, but they're basically – it takes all the curves and all the animation curves tweening and keyframing and all that and it shows you the animation, but it blocks out the basic timing and it shows you the animation curves. It basically dumps all this data on the screen while things aren't animating, so that the developer can look at the – on one side, the final animation and on the other side, the redline animation, which has all this data and all this information that is describing what's happening. They can pause it at any time and see like okay, this box is at this X and Y coordinate at this time and it's using this easing function.

It was just so much work to get to that. The design team not only had to come up with a whole concept first, but then they had to create these extra videos to show how the thing needed to be implemented. Going back to what we we're saying before, then Luigi's team would implement it. If for some reason something had to change, it was a total restart. We had to go back to recreating the animation concept, tweaking what we thought was going to look right and updating the videos. Sure, you can do a few of these things in the actual code and sit side by side and designer/developer just hash it out real quick. For bigger concepts like that, you end up losing that ability to do a lot of fast iteration.

[0:14:12.9] JM: At the end of this workflow, what's the artifact that is coming out of the animation and design tooling and getting handed off to the developer, what's the actual file format?

[0:14:25.2] GR: In this case, it's a movie. It's an MP4, or a .MOV. That's what After Effects spits out.

[0:14:32.1] LR: This was one very specific use case, where we really needed something super highly tailored. The design vision was really non-traditional for this product. It required also due to the constraints that we had on the hardware, some really clever engineering to virtualize lists and such that we're animating and elements within those lists were animating. We really wanted to call them, not draw them when they weren't on the screen as aggressively as we could. That required a custom layout engine.

That's why we went down this route with this project. I think that's fairly non-traditional. It does go to show that there really aren't tools to do things like that these days. The only environments that have those concepts are really video games. That said, nowadays there are tools that are coming out that do have a lot greater visibility into their layout engines and give you even access to do your own layout at a much lower level that are starting to come out and are starting to expose the ability to create tools that can tailor those experiences. That's something that we are striving to build.

That said, there are also today other solutions and other things that people use. There is the ability to export an actual JSON file from After Effects and a lot of things get dropped along the way. Basic shapes will get exported, animation keyframes will get exported. Then with a clever runtime Lottie, you can play that back. What ends up happening is that you're basically playing back a glorified movie clip. It's in real-time. It'll be much lighter to download than a movie and it'll scale properly. It's vector graphics, but you don't get the ability to control things at runtime, like mixing states, mixing animations together. You don't get that with a tool that doesn't have that concept built in. You've got to do that all after the fact in your code.

That is something that we're also striving for is having the ability for designers to be empowered and see those states being mixed in the tool that they're using. What we think is really lacking is

a is a runtime animation tool and the supporting pipeline for getting those assets out of it into an engineering-friendly framework that can then manipulate anything that comes out of that.

[0:16:42.9] JM: You mentioned Lottie there. Lottie if I recall is a React set of tools. It animates React components, or something like that?

[0:16:57.9] GR: We actually we actually worked with one of creators of –

[0:17:00.5] JM: Did with Airbnb. Right.

[0:17:03.4] GR: The story is actually, it started with a guy called Hernan Torrisi that we've worked with, who's not related to Airbnb. He created the Bodymovin plugin for After Effects. He also created a web player for it. Basically, you could use the Bodymovin plugin to export a lot of the animations from After Effects, like Luigi was mentioning into a JSON file, but you lose a lot of stuff along the way. Obviously, it converts to vectors so not all imagery comes across, not all of the After Effects plugins are usable. You have to know what you can and can't use.

It gives you this real-time movie clip, basically. That's not a movie though, it's data. That can be played back by a web player that he created. Hernan created this this runtime for the web. Airbnb as far as I know the story goes, he could probably tell it to you better, then approached him and said, “Hey, we want to take this and productize it, make it into something called Lottie. We'll build the other runtimes. We’ll build an Android runtime for it. We'll build an iOS runtime for it, so that basically essentially players for iOS and players that will play back this JSON file on iOS, on Android and all that.”

I think, actually you had someone on your podcast, Gabriel Peal recently, who I think he worked on the Android one, if I remember correctly. I believe they also have a JS runtime. Yeah, that's the one –

[0:18:25.6] LR: The JS runtime with components for React and I'm sure all the other major UI libraries. Yeah.

[0:18:31.6] JM: Basically, the idea of the Lottie system is you could make something in After Effects, which is the most commonly used video editing tool, animation production tool and you could use this Bodymovin tool to export it to a JSON file. The JSON file needs to be interpreted by whatever platform is running the animation, so whether we're talking about an Android, or iOS, or web, or React in the web, you need a runtime to actually execute, or process and render that big JavaScript file that you've generated from After Effects.

[0:19:13.0] LR: Yup, exactly.

[0:19:15.0] JM: That is one model of bringing animations to the web, or bringing animations to your mobile applications. I've seen the Lottie animations. It looked really nice. Having a refresher just now of that tool chain, that's a kludgy tool chain. It's still using After Effects. It's still producing this artifact that sounds very difficult to build a graceful handoff process with. It also sounds – I'm not sure how interactive those animations are if they're just unidirectional.

[0:19:48.7] GR: Yup, that's right. That's exactly it. After Effects itself isn't built for interactivity. It's built for video. Well first of all, you're working in three different tools. The N10 pipline is not owned by one vision. It's all these different third-party tools that are added on to create the final output. Things definitely get lost along the way. Not everything is perfect; your animation. There's some trial and error you have to do in After Effects to see what's actually going to export correctly and work as expected.

One of the biggest issues we see with that pipeline is that like I was saying, After Effects is made for videos. Traditionally, it's for After Effects, effects for video editing and post-production. The tool itself is not interactive. First of all, everything has to pre-render. When you hit the spacebar to watch something, it's not actually playing in real-time. That also means that while it's playing, you can't dynamically manipulate anything. You can't for example create a video game character that has a run animation, a swim animation, a jump animation and a shoot animation and then mix those things together, so that while the character is running, it can also aim and shoot, but maybe in another instance using the same run animation, it's jumping, or it's looking around.

In After Effects to do that, you basically have to create all those different states and then create all the different permutations of those mix together. You can't tell it to mix those things at runtime, in After Effects you just can't do that. Already for a lot of video game developers that are require a lot of animation, it's not an ideal tool. Then Lottie itself, I believe if I remember correctly, it only supports one animation so you export a single animation. If you want to add any interactivity to that, you as the engineer now need to go and look into that file, the JSON file and manipulate that all with code, which you can do, but the tool isn't ideal for that.

As an example, well, so the animation mixing is a obvious one that just because you the designer can actually create these multiple animations and preview them in the file themselves and After Effects themselves and the editor, you're not going to be able to – okay, what am I trying to say here? Let me take a step back. Lottie plays back really well a single animation. That's just a single video is great; a single one-time animation. Then if you're trying to say, grab a player's hand or something like that in the game and make them aim up and down, you have to do that all with code.

Whereas, we think in our tool we've built it to be built for interactivity, so you can actually create a bone chain in for your character's arm and then put a controller at the end of that arm and the designer in the tool can actually manipulate that and see what it’s going to behave like and then expose that controller to the engineer that just says, “Okay, I'm going to grab this at runtime and move it.” Because of how the designer set up that arm to work, the inverse kinematics and the bone system is going to respect all of that and not break.

Anyway, that's just one example of something that with After Effects, you can find add-ons and plugins that do bones and stuff like that, but it's just not made for runtime and that stuff isn't going to translate to lobbying.

[0:23:10.6] LR: Well, I think from the from the UI perspective, there's an even better example that maybe can clarify the difference between the ability to mix states. If you imagine and we actually have a blog and a couple different posts about this, so if you want to see it in detail you can look for it. It's called Liquid Downloader. We have this example which is a download process that basically has an indeterminate and a determinate state at the same time. If you imagine

something that's spinning on your screen indeterminately, but then it also shows progress by maybe if it's a circle, it could fill up as the download proceeds.

If you had to design that in After Effects, you'd have to make two separate animations and then you couldn't preview mixing them together. With Lottie, there are clever developers can get around to make that work by for example, looking at one portion of the timeline, applying that and then maybe applying the second part of the timeline at a different time. It only works if you guarantee that those keyframes don't touch the same things.

While with a tool that's made for runtime animation and the ability to blend the animations together, it should actually based on input from the designer, blend those keyframes on top of existing keyframes before applying them to the final result, which means that you can see a result that is a combination of multiple states of multiple animations. That's something that's very hard to do with something that's just not built for real-time.

[SPONSOR MESSAGE]

[0:24:41.0] JM: Apache Cassandra is an open source distributed database that was first created to meet the scalability and availability needs of Facebook, Amazon and Google. In previous episodes of Software Engineering Daily, we have covered Cassandra's architecture and its benefits. We're happy to have DataStax, the largest contributor to the Cassandra project since day one as a sponsor of Software Engineering Daily.

DataStax provides DataStax Enterprise, a powerful distribution of Cassandra created by the team that has contributed the most to Cassandra. DataStax Enterprise enables teams to develop faster, scale further, achieve operational simplicity, ensure enterprise security and run mixed workloads that work with the latest graph, search and analytics technology all running across hybrid and multi-cloud infrastructure.

More than 400 companies, including Cisco, Capital One and eBay run DataStax to modernize their database infrastructure, improve scalability and security and deliver on projects such as customer analytics, IoT and e-commerce. To learn more about Apache Cassandra and DataStax

Enterprise, go to datastax.com/sedaily. That's DataStax with an X. D-A-T-A-S-T-A-X, at datastax.com/sedaily.

Thank you to DataStax for being a sponsor of Software Engineering Daily. It's a great honor to have DataStax as a sponsor and you can go to datastax.com/sedaily to learn more.

[INTERVIEW CONTINUED]

[0:26:20.5] JM: Just to cut to the chase a little bit, your system that you've built, it's basically a fully – it's a full stack animation tool. Rive is the name now, right? I mean, formerly called Flare or Two Dimensions. Now it's called Rive. Rive is as you described, a system that produces interactive artifacts. The bone system you describe, where you essentially define how – so you could take a sprite, a cartoon of a character in a game and you can define bones, meaning these are going to be the pivot points that the character can bend its arms around, or be controlled around.

The designer or the animator can actually in your case, in the Rive case, create control points that define a file that once it's exported, it actually exposes an interface to the developer, so that the developer can actually manipulate this exported artifact, right? Is that is that a big differentiator of this file, this animation file that actually gets exported from your product?

[0:27:42.3] GR: Yeah, I think that's absolutely right. The designer sets up these control points and sets up the animation, so that when they're working in the tool, they know how it's going to behave when the engineer touches those points that they've exposed those interfaces. That's to say though that having said that, the engineer can still has access to the entire file and can do whatever they want in code. You don't have to just use the API that we expose to you, although that is like you mentioned, our differentiator. That's what makes it easy for designers and developers to make that handoff simpler.

[0:28:18.7] LR: Yeah. I think the big point is the fidelity. There's the same fidelity in the runtime, as compared to what the artist sees on the screen when they're working in the editor and the tool. Everything that they do in the tool, whether it's bones, animations, custom constraints, inverse kinematics, all that stuff is supported by the runtime. The runtime will play back exactly

what you see in the editor. It'll mix the animations exactly as you see them mixed in the editor. That means that there isn't the opportunity for things to get dropped along the way. The vision is cohesive from the start, the tool and the runtimes are built to work hand-in-hand.

[0:28:56.3] JM: Just to give a example of how this interactivity is invoked in practice, if you were to think of one of these unidirectional animations, these less dynamic animations, you could think of something like a progress bar. You land on a webpage and there's just a progress bar and the progress bar animates the same way no matter what is going on in the page. In contrast, if you were to use an animation from your tool, from Rive, you could have something like a character, like you have this bear, this teddy bear thing, where it loads on the page and as your mouse is moving around the page, the character's eyes are tracking where the mouse is moving. It's very clear that there's a link between my mouse movements and the bear, the bear animation.

I just use this as an example, because I'm really trying to drive home the point that creating an interactive interface with a sprite with an animation seems a pretty important breakthrough in animated UI development. If we want to actually make animations consumable by the average web development team, this is a pretty big step change in that direction.

[0:30:20.5] LR: Sure. Yeah. We definitely think so. To your example, we actually have a demo of this bear in a login screen. You can take it even further than just tracking your eyes. For example, the bear might track your cursor as you're typing your username. Then if you go on the password field, the bear might cover its eyes, just not to look at your cursor. It's tracking all these different states. If you hit the submit button and your password is wrong, the bear can react with a upset animation. If you get the right animation, or the right password credentials, the bear reacts with a happy animation.

You can add all this interactivity that also has meaning. It's also tied to the UX of the experience. We think that that, like you were mentioning, is something that traditionally with just creating a simple asset that's there, maybe it's just a bear looking at you, bobbing up and down, breathing and it's not reacting, that feels like the – what we started talking about at the beginning, “Oh, that's just fluff. That doesn't really add anything. Okay, I've got a mascot here that maybe it's part of my brand, but that's all it's really adding to this.” Instead, we add some of the interactive

elements we talked about and we actually make the experience delightful and superior to a typical login screen.

[0:31:41.5] JM: Well, it's worth pointing out one more time. I think our current user interfaces are pretty brittle. They're pretty brittle in the sense that – or maybe brittle is not the right word, but it's all boxes, rounded rectangles. It's all pretty static and it's just very functional-oriented. You hover over something and maybe it becomes slightly highlighted. To some extent, yeah, that's what we want out of our interfaces. We want it to be practical. We don't need it to be shaking and changing colors all the time.

That's not to say that animation doesn't have a place in UI development. We could certainly have more dynamic interfaces. Particularly if you think about tutorials, or showing people how particular applications work, I think animation has a really big place there. I really see the practicality in animation and I think exposing people, exposing the right interfaces to working with different animated components can make a big difference. I'd like to get into the engineering side of things. Just taking a little bit more of a top-down approach, again the system for creating these animations is you have a full browser editing environment.

You open your editing environment and it looks in some ways, like a Photoshop application, but it's for creating animations in the browser. I guess, the first thing I'd like to ask is is there some newer technological development that has allowed that product to exist in the browser?

Because I have not seen many things that are that dynamic and that fully featured in the browser. I mean, it does remind me of Figma in some ways. Figma, if people use Figma, it's oftentimes an amazing experience if you use it for the first time, because it's like you're looking at Photoshop in the browser. It's that level of computation and creative freedom in the browser, which for many people that can be shocking that that's not required out of a desktop application. Your editor looks like it's in that same neighborhood. Is it difficult to get that level of computational complexity in a single browser tab, in a single interface in the browser?

[0:34:14.6] LR: I think that technology has actually been there for this for quite a while. It's that a lot of people haven't really taken the chance on it. I think a lot of that does come from the fact

that traditionally, people haven't been used to using full-blown apps in the web. When they did, it was from a plug-in system, like a Flash, or an ActiveX, or a Silverlight plugin.

In the last 10 years or so, the web has really come a long way. There's the ability to do all kinds of things, have low-level access to graphics and have more sophisticated ways to manage UIs and the state of those UIs. Really tailor visual components to whatever you want them to look like, building your new layout systems.

On top of that, there's also been this push for stability the web is much more stable. There's the ability to build software that really is full-blown software with the web. JavaScript has evolved a lot. I think that that's definitely a big part of it. There's the ability now to create sophisticated, mature applications with a technology stack that's been tried and true and developed for a really long time and now has these players, your web browsers that give you the groundwork to build upon.

Obviously, WebAssembly is a big one. We make use of WebAssembly for a few different components in our system, just measuring the gap with lower-level code, having access to some of the – maybe some libraries that you couldn't have easily ported yourself to the web before, some of the more traditional low-level access to things like OpenGL that a lot – There's always been a bit of a stigma of using WebGL, just because it's not been as mature as OpenGL, or it doesn't have the same features. That's changed. There's WebGL too that you have access to a lot more.

I think that a lot of the tools that were built for OpenGL are now able to allow the libraries that were built for it, can now be written and developed in the web and furthermore, work really performantly with WebAssembly on the web. I think it's just all these things coming together and allowing for the building blocks to build these kinds of applications.

The truth is that I think, the biggest change has been that people are willing to use apps online now. I think that that's something that's actually been possible for quite a bit, but without someone like a Figma, or a tool like ours to prove the point, people were just scared. That's one of the biggest things that we heard. When we launched Nima, our first tool in 2016, it was also

an online editor for 2D characters, completely built with WebGL and we used React for the UI. All your data was saved in the cloud.

The biggest thing we heard again and again was, “I don't trust this. What happens if this crashes? Where's my data? Where's the save button? How do I know? How do I get backups for this?”

[0:37:04.6] GR: You get a lot of comments about things that are total edge cases, but that people really hang on to really strongly like, “What if I want to be animating at the beach, where I don't have an Internet connection, or on an airplane?” The truth is that while we have an answer for that, you actually can keep our tool open and it saves locally. Then when you're connected again, it reconnects to our servers and synchronizes.

The thought that there's this synchronization that needs to happen, or there's this risk of not being able to do that turns a lot of people off. They think of themselves as, “I've been doing this. This is a problem I don't have right now and it's not a problem I want to have.” It adds this extra thing that I don't need to worry about. I think there's probably a lot of that thinking that prevented a lot of people from building these types of tools. Like Luigi was saying, it wasn't until someone did it and people started using it and realized, “Okay, maybe I don't care about that so much.”

[0:38:01.8] LR: Or the flexibility of just opening a website on whatever machine I'm on and having my tool working there is worth the –

[0:38:08.2] GR: The trade-off.

[0:38:08.8] LR: The trade-off. Yeah.

[0:38:10.1] GR: Yeah. We think that there's so many advantages to that model. I mean, just not having to install anything and knowing that your software is always up to date, or not having to manage multiple licenses for multiple computers and just being able to open it up on your laptop, or wherever and you've got all your files. You don't need a tool like Dropbox or something else to sync them, because we take care of that for you.

I think the model now and Figma has dramatically helped us in this regard, where they've proven something that we don't need to prove, because they've grown so much so quickly over the last few years, that they've proven this is a better way to do things. We get to reap the benefits of that. I think users get to reap the benefits of that with new stuff that will be coming out.

[0:38:55.1] JM: The question of actually being able to edit offline, it has taken me a long time to trust Google Docs on an airplane. I think actually, I mean, you guys can correct me if I'm wrong, but I think your browser can write to disk just fine, right? If you open a Google Doc on a Chromebook, for example, and there's this little indicator you see. It's like a lightning bolt that is suggesting that you're offline. I think when you're writing your text document, you might as well be on Microsoft Word. You might as well be on an entirely client-side experience, because although you are in your browser and the browser historically is this thing that is just taking up ephemeral resources, unless you're connected to the cloud, actually if you're in an offline mode, you can write to disk just fine.

Is that what your tool does? Is that what Rive does? If you're animating in an offline circumstance, it's just writing to your local disk and it's just going to sync to the web next time you get connected?

[0:40:04.2] LR: Yup, that's exactly it. It's still is still fairly nuanced how you do that with the web. There is a variety of different ways to write to the disk and most of them are fairly abstracted. It's not your typical file IO. Even though there are a couple different specs that are being pushed for having something that's more akin to actual file IO on the web.

For example, you have local storage, that's a really simple key value that is synchronized to disk. It's a little limited. If you want to use more space and maybe save some bigger data, you can use a IndexedDB, which is actually what we do. We use a combination of both of those things to save your revisions. Whenever you work on something, the change that you make, the changes that you make get saved to disk. Then shortly after if there's a connection available, they'll be sent up to the server. Once they're acknowledged, we can clear out the local saves.

Until we get that acknowledgement back, we keep them on your local computer, so that the next time you open it up, it'll still be there. All your changes will still be there. We actually revisioned it as you go along, we have a couple different heuristics that we use to figure out whether it's time to create a new revision. For example, if enough time has passed, we'll just make sure that will create another little snapshot in your timeline of the file, so that you can go back to exactly as it was at that time period. That way if for some reason, you think that something didn't get saved, or you think that some else may have opened the same file and there are conflicting changes or something to that effect, you have all that history, all that backup. You can go at any point back and check.

The truth is that we actually do a lot of work to make sure that those conflicts don't exist, but it helps people to know that there is this history there and they can go back at any point and see what they were doing yesterday and what the file looked like yesterday. That's thanks to the fact that we do have the ability to save to the drive and just sync as quickly as we can.

[0:41:53.6] JM: That interface between the browser and the disk, this is getting a little bit off-topic, but they have to keep that pretty sparse, because of the security risks, right? Because you don't want any random website you go to to be able to basically take over your hard drive.

[0:42:11.1] LR: Yeah. It's abstracted and it's sandboxed. I think it's based on the domain, you'll get access to the sandbox and you can read and write to this key value store. One thing that you could do is that if you do clear your local data that will also clear a lot of that data. Through the dev tools, you have the ability to browse that data and you can see what's in there. It's not like I can create a tool that reads what some other tool wrote, unless it's hosted on the same domain, I'm not going to get access to that data. Likewise, you can't just take up a ton of space to our quotas there and there are ways to manage them to make sure that you can't just eat up all of someone's hard drive.

[0:42:50.2] JM: What do you guys use WebAssembly for?

[0:42:52.3] LR: We use WebAssembly for a few different things, but the biggest one and it was a change that we did early in 2019 was to replace our renderer. We were using before – Our first product used just WebGL written in JavaScript. Then for the second iteration when we

created Flare, which is now Rive, we just used context 2D for rendering vector shapes. Then we quickly realized that there were some very specific features that we had in our first tool that we couldn't do easily in context 2D and in the canvas 2D painting context.

We found from the Flutter team, another great Google team that is embarrassed to admit that we didn't know of them at the time, but the Skia team is creates a really amazing low-level graphics engine that works on the concept of vectors, shapes, image meshes, raster meshes with textured 2D images on them. All they need is an OpenGL context, or now they have support for Vulcan and Metal and a bunch of other stuff too.

Thanks to that, we were able to web assemble it and bind it to WebGL and have this really high-powered renderer that goes quite a bit further than what's traditionally available on the web for the canvas. That unlocked a lot of the newer features that you'll see that are on Rive, like the ability to do some of the shadows that we have, some of the blur effects, the masking effects that we have. Yeah, so that's a big one.

Then we use it for a few other smaller libraries, for things like managing the state of what's currently visible. We want to do that in a really efficient way. We'll build up our art tree for what's on the screen and then we'll call it aggressively and something that's lower level. Some of our mesh triangulation stuff is also done in a C library, so that we can do that really efficiently and know that. It's these things that are really nicely compartmentalized and need to be really performant and are tried-and-true and have a bit of history. It's nice to know that we can use them throughout WebAssembly and we do.

[0:44:55.4] JM: That's cool. I'm sure we could go deeper on that subject. I want to get to discussing Flutter. A couple years ago, we did a few shows on Flutter. My recollection of Flutter is essentially, a system for doing cross-platform UI development. If you write an app in Flutter, you can potentially have that app work in both iOS and Android. Can you guys could just explain what Flutter is and why you started working with it, why it's relevant to Rive?

[0:45:30.6] LR: Sure. I think that this is going to be a little – it's hard to find the right words, because Flutter is so much more than what I'm about to describe it as, but it's essentially a

framework or a UI toolkit that will then cross-compile to native machine code on Android, iOS, Mac OS nowadays and even the web today too.

[0:45:52.2] JM: Because it's arm code, right? It's lower level processor. I mean, this gets out of my realm of expertise, but I just remember arm code.

[0:45:58.8] LR: That's right.

[0:45:59.4] JM: The processor.

[0:46:00.6] LR: The dart SDK can compile a snapshot of your code directly all the way down to arm code. Likewise, it'll also compile the snapshot to byte code that you can run through a VM and that's how most of the hot reload works. Again, I'm not an expert in this. I'm a big fan of dart and this is what I've gleaned from working on it and with it. I think it's a really powerful language and the tooling is really spectacular. That's one of the best things about Flutter is really the tooling that goes into dart.

There's amazing static analysis. There's amazing compilers. Actually, one of the first animations that we showed on screen with Google was – it was in 2018, the history of everything app that we built. This one fish that was on the screen that it's back kept breaking was really funny. It looked like the back of the fish was getting broken in half. What we found is that something was going wrong in this feature we have, which is called jelly bones, which basically takes – it takes two bones and then it sub-divides them with Bezier curves and it lets you control how it curves. A really talented animator that works with us used that to create the swimming motion for a fish.

Every once in a while, we saw that it would look broken. In looking through it, we saw that there was one part of the code that only when it was AOT compiled, so when it got compiled to arm code, some of the instructions would go awry and one of the control points would end up in the wrong spot.

We broke it down frame by frame and we figured out exactly where it was happening. We sent it over to the dart team and they had a fix within days. It was something that was in their AOT compiler. They didn't have enough register – Certain code that required using a certain number

of registers and I guess that Flare at the time was the only runtime that put that much pressure requiring those extra registers. It would start not copying things properly.

The morale the story is that they own the whole pipeline there and the dart code is literally writing out the machine code that's going to be running your app. They were able to really quickly fix that and turn it around for us. For us having that access to a team that has that much passion and that much desire to make a product that's high-quality is unparalleled. I think that that's really one of the biggest selling points of Flutter for us is the fact that there is this super high-quality development environment that goes – is even lower level than the UI toolkit and everything else. It's dart that enables the hot reload features. It's dart that enables all the static analysis and this crazy performance that Flutter gets by compiling to a native machine code.

[0:48:28.9] JM: Sorry if this makes you repeat yourself, but why do you use Flutter and what do you use it for?

[0:48:36.0] LR: The point of using Flutter is that you can write code once and it'll run on these, on multiple operating systems, right? You can write your –

[0:48:43.9] JM: Specifically the animation code.

[0:48:46.5] LR: Oh, oh. Sorry. For Rive, we have a runtime. It was one of the first runtimes that we wrote for Flare at the time was in Flutter. The reason we did that was because it was a really easy way for us to get on both iOS and Android and also unlock an animation ability in Flutter that wasn't currently available. One of the things that we heard from our friends at Google was that they were – it was very hard for them to create certain animations that you can see if you open up any Google device.

[0:49:14.0] GR: Sorry Luigi, real quick to clarify what Luigi was saying before about using Flutter to build apps for that deploy immediately to iOS, or Android, or whatever, that's what an end user would see, a flutter end-user. That's why they would choose to use Flutter. A developer nowadays chooses to build their – say they're building a mobile app. They choose to build their mobile food delivery app once in Flutter and it automatically builds an iOS and an Android version.

Now flutter is a lot more than that, because as Luigi was saying, it's actually a great dev environment on its own, regardless of that feature. It also now deploys to multiple different platforms. Aside from iOS and Android, they now have desktop, they have web. Really, you can build an app that you build it once and provided you have resizing logic and UI logic that works across different devices, you don't need to rebuild all these different apps. That's to answer your question before about why someone – what is Flutter and why would someone use it.

[0:50:15.7] JM: I still understand what matters there for your team, because I get it. If I'm a Flutter developer, I'm building a food delivery app, right I build it in Flutter. That's fine. Maybe I'm using some animations, but I guess I don't understand what the depth of –

[0:50:34.1] GR: For us, there was a clear fit for an animation tool there. There was no tooling available at the time when we started talking to the Flutter team for creating animated graphics in Flutter. An old friend of ours actually heads up the products management team there. We actually worked together when he was at Microsoft. His name is Tim Smith. Luigi and I used to have a design and development agency that did some work for them back then. When he started working at Flutter, we caught up and said, or we saw he was at Google and we said, "Let's catch up. What are you up to?"

He showed us Flutter and he showed us – he mentioned when we were telling him what we were up to, we were building this animation software specifically for game characters, which was Nima, the one Luigi mentioned earlier. He mentioned, we have this pretty involved process right now to create animations with Flutter. For example, one of the things I think Luigi was starting to get to is I don't know if you've seen the Google Assistant, the G that appears and it spins around with four dots if it's listening to you. It morphs into a little microphone that's listening and all that. Those were all After Effects animations that were handed off to the Flutter team. There were I don't know how many, but a lot of different animations showing this is what happens if it's listening, this is what happens when it's processing, this is what happens as it's loading data.

They had to go in and just recreate that all in code. Tim was telling us how much of a burden that was. It took them – I don't remember the exact number now, but it was months to create everything, to recreate it exactly as the designer wanted it to look like in Flutter.

He mentioned to us, “Would your animation tool, this game character animation system you've built, would that help us in any way?” That sparked the light bulb for us and we were like, “Well, yeah. We're missing some key things that the designers would need for UI.” Mainly, we were missing vector graphics, so our initial tool was primarily for game engines, so it was all raster graphics, which is what game engines are optimized to be able to render quickly. Vectors are not great at that.

We added that in. That's part of why we went with Skia, what Luigi was talking about. We just realized that there was this great opportunity for something that we had already built to just improve and fill this hole that Flutter was missing. While we were finding this hole that Flutter was missing, we realized that it's a bigger problem not just with Flutter, but that all design and development teams need.

We've talked to people at all sorts of different companies that have a need for this type of software. That's when we started realizing, “Okay. Well, Flutter’s our first runtime, but we're going to keep doing what we did with Nima, where with Nima we had a Unity runtime, we had a JS runtime.” We're going to keep doing that with all other dev platforms, so now we have a swift runtime, we have an Android runtime that's about to be released. We are working on game engine runtimes as well. That's really Flutter gave us the inspiration for – it helped us realize that there was a much bigger product than the simple game animation system we were building. There was this bigger vision that we could go after.

[0:53:51.9] LR: On top of it, it's our first runtime. It was the first runtime that we tackled, because it gave us the ability to run our animations on iOS and Android right off the bat.

[0:54:03.2] JM: By runtime in this case, you mean there is an artifact that gets exported from the Rive editing tool, the user can use to the browser that we were discussing, it's going to produce this artifact, this this dynamic animation artifact that depending on what client device you're going to be using that animation in, you're going to need a runtime, whether it is iOS, or Flutter, or React. Whatever it is, you're going to need some way of consuming that artifact and manipulating it.

[0:54:39.0] LR: Exactly. Yeah. Think about it as an NPM package, a package on pub dev for Flutter, or just the library that you can go grab from github and install yourself if you want to.

[SPONSOR MESSAGE]

[0:54:58.6] JM: Looking for a job is painful. If you are in software and you have the skill set needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace that connects highly qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies. Access is exclusive and you can apply to find a job through Vettery by going to veterry.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters, so that you only get job opportunities that appeal to you. No more of those recruiters sending you blind messages that say they are looking for a java rock star with 35 years of experience, who's willing to relocate to Antarctica. We all know that there is a better way to find a job.

Check out veterry.com/sedaily and get a $300 signup bonus if you accept a job through Vettery. Vettery is changing the way people get hired and the way that people hire. Check out veterry.com/sedaily and get a $300 signup bonus if you accept a job through Vettery. That’s That's V-E-T-T-E-R-Y.com/sedaily. Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:56:48.1] JM: Can you tell me more about your interactions with the Flutter team and in the Flutter ecosystem, just as an example of how people are using your tooling.

[0:57:03.8] GR: We can't share specifics about the teams that are using it, but there are many teams in Google that have actually launched already their products using – I'll say that if they're

using Flutter, they're probably using Rive for their animations. You can go find out which of the Google teams are using Flutter for their products. One of the ones that has publicly shared the information is Google Assistant. Their new version that's coming out sometime this year, I'm not sure if they've launched yet. They might have not yet. That's using Rive for all their animations.

[0:57:37.8] JM: All the animations in Google Assistant?

[0:57:41.1] GR: Well, I don't know if it's everything. We're not involved in their day-to-day stuff, but they are using Rive for a lot of stuff, like tutorial modes, like swipe from the right to bring up this menu, swipe from the top to do that. Those are Rive animations that they're working with. There are other teams within Google that do that. That's one of the big benefits we have with interacting with the Flutter team. Flutter is they're trying to get adoption within Google of their own platform.

When they're convincing some big new product within Google, build your mobile apps, or your whatever app, you should be using Flutter for it. At the same time, they're pitching us too, because if they're using Flutter, their designers we hope, are going to want to use our tool to create their animations and their graphics. That's part of why our – we're super fond of the Flutter team and all the individuals there. There are also champions for our own product within Google, which is pretty special.

[0:58:44.4] LR: We know of some other external teams. I think it's no secret that the Hue app is all built in Flutter.

[0:58:50.2] GR: Yeah. Philips Hue.

[0:58:51.4] LR: All the animations in there are built with Rive. A lot of the Sonos ones are as well.

[0:58:56.3] LR: Yeah, there's a lot of apps nowadays.

[0:58:58.3] GR: There aren't others that come to my head, but I know that there's a few startups that we know of that I think we can't probably share, but there are quite a few startups that are pivoting towards using our technology too.

[0:59:09.6] JM: Do you guys have – a far-flung question, but do you guys have any perspective on the Flutter versus React Native for a developer that wants to build a cross-platform application?

[0:59:23.6] LR: We have our perspective, I think that at the time I was really fond of React Native. I found a lot of the flexibility and the concept behind it was really smart. It applied a lot of the concepts of state management for UI widgets that had been built by the React components to mobile was genius.

One of the problems that I ran into with it early on was the fact that just running a JavaScript context on an embedded system required – well inherently, it requires a certain amount of memory. Specifically when you start working on some of the not so powerful systems, that can become a constraint really quickly. I think that that was one of the big things that I immediately saw a huge benefit over with Flutter, was the fact that it is so much more efficient in that regard. The fact that the code is also compiled to machine code and doesn't have to be interpreted is an enormous win.

Then on top of all that, the fact that you literally have access to the entire rendering pipeline, you get to paint every pixel yourself. You're not calling into an abstraction of a list view that's generated by either Android or iOS and then it's painted by either Android or iOS. You're calling code that then paints within the same ecosystem in the same platform. There's that transparency that particularly when working with someone like Guido who is a designer who wants the vision to be from design to shipping to be driven by design, there are a lot of cases that you'll run into where you want to customize something on Android or iOS and it'll be a lot of work and it'll be an exceptionally higher amount of work to get it done in React Native, because you need to pipe it through the JavaScript context and down to those different rendering systems.

Instead in Flutter, of having direct access to that rendering system at any point in your codebase, makes it so much easier for us to build these tailored experiences, these tailored UIs, that go beyond what the native platform offers. I think that that's something that maybe a lot of people don't realize, but with Flutter, you essentially have something that's much more akin to a game engine. You really do have access to the entire stack, which you don't with React Native.

[1:01:39.8] GR: Yeah. If you care about performance really, if you're building anything where performance really matters, I think it's a no contest. You have to use Flutter, if the choice is just React Native versus Flutter. Now if you're trying to quickly do something that is more static, or you're already entrenched in a React world, then there's reasons why you might want to stay with –

[1:02:04.4] LR: You have a lot of JavaScript developers. There's a lot to be said for that, the fact that you can use the same techniques and skills that you've learned to build a very high-quality web app, a testable web app and now you can apply it to mobile, that's really impressive, that's really empowering.

[1:02:19.3] GR: Yeah, that's React Native.

[1:02:20.6] LR: That is something that React Native does. It's a big point in its favor. With Flutter, you do have to learn dart, you do have to learn a different language. From my perspective, I welcomed that. I got so much more out of Flutter that learning this new language, which was by the way, very familiar, it's very similar to a C-sharp, a Java, a JavaScript, a typescript. They're all very, very similar. If you're fluent in any one of those, you'll pick up dart really quickly. Like I said, the benefits far outweighed the technological challenge of jumping into this new ecosystem. Yeah, the React Native, the fact that you can just use the same tech that you're using on your web is pretty – it's really empowering.

[1:03:03.2] JM: It has historically been unwise to bet against JavaScript.

[1:03:07.4] LR: Absolutely. Yeah. Our site, our platform is currently entirely built in JavaScript. We web packed Rive right now. We use modern ES6 JavaScript and we're always updating it and seeing when new features are coming out. It's a force to be reckoned with. It's so flexible.

It's so it's so easy to get going and build something and deliver it to millions of people quickly, that's undeniable.

[1:03:36.5] JM: How close did you guys watched the developments in the React ecosystem? Because I did a show a while ago about some of the newer developments in React, and I think they're doing a lot of engineering around trying to make the bridge, that JavaScript bridge more efficient.

[1:03:50.9] LR: For React Native specifically?

[1:03:51.9] JM: For React Native.

[1:03:54.3] LR: I don't follow it enough. One of the things that we are doing right now, like Guido mentioned earlier is that we are releasing an Android runtime soon; should be the end of this month. Once we have that, we will have the ability to offer Rive in React Native as well. We'll be exploring that a little more deeply soon. Last time I used React Native was a couple years ago that I built a full application with it. I haven't kept up as abreast of it as I'd like, but we do keep up to date with React. We use React on our site every day. There are builds running right now that are web packing React for us. Yeah, that's something that we'll tackle soon, the React Native side.

[1:04:34.6] JM: There was a listener that wrote in with this theory of how Google is going to remake the full stack of their operating system and they were just describing to me how Flutter and Skia and Fuchsia and all these different things are going to fit together to be the future of all of our runtimes. Have you guys had any conversations with anybody about any of this this stuff? Because Flutter is futuristic, along with Skia and theoretically, Fuchsia. Although nobody's talking about Fuchsia really.

[1:05:09.5] LR: Yeah. We don't get to hear a lot of that. Unfortunately, we're outside that circle. I mean, we've heard the same rumors. People are wondering about that. It does seem like that's the direction things are going in. One of the things that we really strongly believe and maybe even further thinking than that, it could be that the systems you just described end up adopting

that, but one big thing that we really strongly believe is going to be the future is WebAssembly. We really think that that –

[1:05:37.7] JM: Right. That one's clear.

[1:05:39.0] LR: - that will be just a common runtime for everything and everything. I mean, wazi will be at the system level, you'll have – hopefully we'll have operating systems that run across all different devices that basically just have that that small wasm bridge.

[1:05:54.6] JM: What does that look like in more detail?

[1:05:57.0] LR: I'm not as fluent in this as I should be, although we are working on it, because it is something that we think is going to be a big part of our future too. It's the ability to basically compile your code to WebAssembly and then have low-level system access to things like file.io, the graphics, input. It's this bridge, from my understanding wazi is the bridge between WebAssembly and your native system.

What we can see will be the future is this this common language runtime that's WebAssembly, that will allow all kinds of applications. It doesn't matter what they're written in, as long as they compile to that, to run across a variety of different operating systems that just need to expose these low-level hooks. Then all of a sudden, you have a variety of different frameworks and maybe one like Flutter will become the de facto standard on that.

What ends up happening is that there's no longer this need to be so close to the metal on the specific hardware that you're targeting. You'll be able to be just a few inches off, but with something that's abstract enough that can run anywhere. I think that's what's going to be really, really empowering for us as developers, but also I think it'll end up in there being way more tools and way more features for designers to use to support these kinds of systems, just because the flexibility will be – it'll just be so much more flexible.

[1:07:16.8] JM: There's no fundamental bottlenecks to that becoming a reality, right? It wouldn't necessarily break any security best practice or anything. We can engineer around all those things, right?

[1:07:30.5] LR: That's the hope. Yeah. There's tons of conferences and discussions about that subject specifically. I am definitely not qualified to comment on it. I do think that – I believe that we can and I think that it's worth exploring that to get to that.

[1:07:45.9] JM: Did you guys have any particular interest in animation, or was this just a random engineering problem that you stumbled on to?

[1:07:54.9] LR: Guido and I have been doing animation since I think we put our hands first on a computer. Then when we were I think 16, we had an opportunity to – we were living in Malaysia at the time and we had an opportunity to – so we were working with this company called Homeland. They called Diamond Multimedia.

[1:08:12.3] GR: DMM. They made the first Rio MP3 player. I don't know if you remember that, one of the very first MP3 players you could buy. It was called the Rio Diamond Multimedia.

[1:08:20.2] JM: Yeah. The hardware device.

[1:08:21.3] LR: Or they had the Jukebox after that. Anyway, we were playing a lot of video games at the time and that's actually how we started in this world. We were modding games like Quake and Tribes and Counter-Strike and they needed some people that could do that for them and build things. One of the things they wanted to do was advertising within these systems, so they wanted to be able to have the ability to sell spots within a Counter-Strike game and that stuff. This was back in 1998, or 99, or something. It was a long time ago.

Guido and I – well, they were looking for people to build things like this for them. They had a challenge that was design our new logo. I worked on it for weeks at a time. I came up with something, submitted it.

[1:09:02.8] GR: You could you could – Yeah, I think they said that they were going to – they had prizes that you can win one of these Rio MP3 players, which was super rare at the time.

[1:09:11.1] LR: Or the first prize, the grand prize was a job. Guido did his submission for the design for the logo, I think the night before he's like, "Ah, whatever. You did this and maybe I'll do it too." He won first place and he got the job. I got the consolation prize, the Rio MP3 player, which I was happy with, but I also have made the very distinct decision that you know what? I'm going to stick to programming.

It ended up being a good decision, because Guido and I then, I got brought onto that company a little after too and we started working together hand-in-hand on building user interfaces and experiences for them at the time. Then that led to our first agency. We've always been straddling the world of design and development and specifically with animation. This was during the big boom of Flash.

[1:09:55.6] JM: Yeah. Our first company, so after this short stint with Diamond Multimedia, which ended with the dot-com bust, they had to cut back a lot on the remote contractors, which is what we were. Obviously, starting out in Malaysia and then we had moved to Rome in the meantime, started going to college there. Rome in Italy. Since then though, since that moment where we have made that conscious decision, I'm doing design from now on, you're sticking to programming, we found Flash somewhere along the way there.

When the Diamond Multimedia job went away, because I believe if I remember right, it transitioned to a month-to-month contract, instead of a yearly contract. We decided, we know enough of this now that we want to try to run our own company. I believe we were 17 or 18 in Rome and we started this small Flash design and dev agency and we were just building rich Internet applications. I don't even think they were called that then. That was a word that came later.

We started really early in our careers in tech building, really animated experiences from everything for music production companies, EMI was one of the first projects we worked on. Red Bull and Rai, which is the Italian national TV. Along the way, we were getting a lot of contracts. Well, we were not getting a lot of contracts, but we started getting a few contracts in the United States and we realized that the appreciation for the work that we were doing and also they paid a lot better, was a lot higher here in the states. That's when we decided that we needed to move here and start our company here.

[1:11:31.1] JM: As we begin to wrap up, just a few more questions on your trajectory. Right now, it's only 2D animations, right? What would be required to get to 3D animations?

[1:11:41.2] GR: Well, Luigi can talk about the engineering there, but we definitely don't see that as a blocker. We see that as something that we do want to get to.

[1:11:52.2] LR: It's a challenge. Yeah. I think that the bigger question is how do we do that from a UX perspective, in a way that doesn't break the simplicity and the elegance of what we have now? Inherently, by adding that third dimension, there's the ability to really complicate things. We want to make sure that we still have the same UI concepts and paradigms that are available in our tool now that are just as easy to use in 3D. I think that that's something that we absolutely want to tackle, but it's something that we also want to hear from our users if that's something that they really, really want. There's plenty for us to chew in the 2D world right now and we have some exciting things that bridge the gap between 2D and 3D.

Eventually, that's a huge passion of ours. We didn't get to mention it earlier, but yeah, because we started in games, we also did do a lot of – we've built a few different games and that was another company that we did along the way. That is something that is near and dear to us, so that –

[1:12:46.7] GR: Luigi has built full-blown game engines, which is where a lot of the concepts for this tool came from.

[1:12:52.4] LR: Yup. One of the big concepts for Rive is absolutely that the line between game engine and app engine shouldn't be so distinct. An app engine should be able to make a game and a game engine should be able to have beautiful UI. That's something that's still challenging today.

[1:13:13.9] GR: Well, and you can think of one quick example of where a tool like ours in the near future might need to, or could provide value is virtual reality. There's definitely a need for user interfaces that work in 3D space. 2D UI that may be needs to wrap to a 3D environment, or

appear in a 3D environment. There's definitely lots of exciting stuff there. Yeah, we're definitely looking at it and thinking about it.

[1:13:41.9] JM: We didn't really explore much about the business, but basically you've got public and private settings for how people create their animations. The default is if you create an animation, it's in public and you can pay for private access, private editing and animation. There's this whole gallery of animations that people have created on your website on rive.apps. If anybody is just remotely interested in adding animations to their application, or building games, or anything, this is a pretty cool – your site's really cool to check out. It's just fun to explore.

Just to wrap up, I want your perspective on how animation could be used to make the average user interface better. Just most of my mobile apps, most of my desktop apps do not have animation in them. For the most part, I don't know the difference. I'm like, "Okay, that's fine. I don't need animation, I don't think. I don't really want it." There are these little moments in certain applications, like Facebook Messenger, or Assistant, absolutely with Assistant, where the animation just adds a little bit of bounce and a little bit of liveness and a little bit of artistry to these applications. Where are we going to see animation in our day-to-day applications as this technology becomes more accessible?

[1:15:12.3] GR: I really think that there's no limit there. I mean, I think you can use it – We are building tools that will allow you to add animation to any part of your UI. I think that what that enables and what gets me excited about that, which is something was probably the biggest feedback we get from our clients when we were working with clients to build products for them was that they really want a personality. They want their brand to shine through.

When you talk about how forms nowadays are all just boxes, or simple lines and stuff like that, there are companies like Red Bull that would specifically come and say, "We want something that's not cookie-cutter. We want something that screams Red Bull and that's different." Something like this allows you to add a lot of personality and a lot of differentiation. I do think that for the average developer, that's also going to be a way to really differentiate your app from the thousands, if not millions of other apps that are on the various app stores. It gets harder and harder to stand out. Something that creates a delightful experience and a reaction more than

just hitting a login button and seeing a page go blank will make your apps stand out in a big way.

[1:16:32.1] LR: Yeah. I think that a big part of animation that a lot of people don't realize is that it's basically just managing state. What animations are good at are transitioning those states in a way that makes it really clear that the thing is – or not makes it really clear, but gives the illusion that the thing is moving and it's got a soul. Animation comes from that anima, is I think Latin for soul.

[1:16:55.2] GR: Italian.

[1:16:57.0] LR: To give something animation is to give it life. I think that there's also this distinction of having a tool that enables you to give something life, doesn't just mean that it'll have a character, or that things are going to be moving on the screen. It also just means how do you move those things on the screen effectively?

There's a lot of people that have tried to do this and are trying to do this today. I think that it's a shame that there isn't a tool that does do both, manager states. How do I go from a login screen to the next screen? Do that really elegantly and it doesn't have to take up a lot of time, but do it in a way that guides my eye and makes it clear that okay, I've now changed. This thing didn't break. I'm not just popping into something else. I'm transitioning into something else into the next view.

The way that that works is controllable by a designer, who's building the whole vision and then easily implementable by the developer that's working with the tooling. I think that's a big thing that that maybe isn't quite clear yet, but a tool like ours enables not just characters that sit there and stare you while you're typing and can give you feedback of whether you're doing something right or wrong, which is valuable, but it can also help you animate things off the screen, or align things on the screen.

I think that that's something that we're really excited to explore in more depth and we think that is why a tool like this that is really reminiscent of something that we did used to have in the early 2000s with Flash is going to have a resurgence in this Renaissance of animation and design,

where it's the designer driving the actual end application and it's not a compromise of we'll ship something that's good enough and looks okay.

I think that when you look back in the future at a lot of the apps that are running on our desktops and our mobile apps nowadays, we'll think back, we'll have the same effect that we have now when we look at apps from the 90s. We'll be like, "Wow. I can't believe that used to look like that."

[1:18:58.0] GR: Yeah. To add just a little bit to what we were just saying there earlier too about the – There's no reason for there to be this big distinction between an app engine and a game engine. If you start thinking about what experiences look like in games and how different they are from one game to the next, you start to see where we think apps could go, because there doesn't need to be this complete disconnect between a game engine and an app engine.

[1:19:24.6] JM: Guys, thanks for coming on Software Engineering Daily. It's been really fun talking.

[1:19:27.6] LR: Yeah. Thanks, Jeff.

[1:19:28.2] GR: Thanks for having us.

[END OF INTERVIEW]

You can get started on DigitalOcean for free at do.co/sedaily. One thing that makes DigitalOcean special is they're really interested in long-term developer productivity. I remember one particular example of this when I found a tutorial on DigitalOcean about how to get started on a different cloud provider. I thought that really stood for a sense of confidence and an attention to just getting developers off the ground faster. They've continued to do that with DigitalOcean today. All their services are easy to use and have simple interfaces.

Try it out at do.co/sedaily. That’s do.co/sedaily. You will get started for free, with some free credits. Thanks to DigitalOcean for being a sponsor of Software Engineering Daily.

[END]