EPISODE 1089

[INTRODUCTION]

[00:00:00] JM: Cruise is an autonomous car company with a development cycle that is highly dependent on testing its cars both in the wild and in simulation. The testing cycle typically requires cars to drive around gathering data, and that data is subsequently integrated into a simulated system called Matrix.

With COVID-19, the ability to run tests in the wild has been severely dampened. Cruise cannot put so many cars on the road and thus has had to shift many of its testing procedures to rely more heavily on the simulations. Therefore, these simulated environments must be made very accurate, including the autonomous agents such as pedestrians and cars.

Tom Boyd is the VP of simulation at Cruise. He joins the show to talk about the testing workflow at Cruise and how the company builds simulation-based infrastructure as well as his work managing simulation across the company.

If you want to reach 30,000 unique engineers every day, consider sponsoring Software Engineering Daily. Whether you are hiring engineers or selling a product to engineers, Software Engineering Daily is a great place to reach talented engineers, and you can send me an email, jeff@softwareengineeringdaily.com if you're curious about sponsoring the podcast, or forward it to your marketing team. We are also looking for writers and a videographer. If you're interested in working with us, you can also send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:01:38] JM: I've recently started working with X-Team. X-Team is a company that can help you scale your team with new engineers. X-Team has been helping me out with softwaredaily.com and they have thousands of proven developers in over 50 countries ready to join your team and they can provide an immediate positive impact and lets you get back to focusing on what's most important, which is moving your team forward.

X-Team is able to support a wide range of needs. If you need DevOps, or mobile engineers, or backend architecture, or ecommerce, or frontend development, X-Team can help you with what you need. They've got a full-range of technologists who can help with AWS, and Go lang, and Shopify, and JavaScript, and Java. Whatever your engineering team needs to get to the points of scale that you want to get to, X-Team can help you grow your team. They offer flexible options if you're looking to grow your team efficiently, and their model allows for seamless integration with companies and teams of all sizes. Whether you're a gigantic company like Riot Games, or Coinbase, or Google, or if you're a tiny company like Software Daily. You can get help with the technologies that you need. If you're interested, you can go to x-team.com/sedaily. That's x-team.com/sedaily to learn about getting some help with your engineering projects from X-Team.

Thank you to X-Team for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:03:23] JM: Tom Boyd, welcome to the show.

[00:03:24] TB: Thank you. I'm glad to be here.

[00:03:26] JM: Cruise is an autonomous car company. You work there. Your development cycle at Cruise is highly dependent on testing the cars both in the wild and in simulation. You are in an interesting position now because the ability to test in the wild has been somewhat restricted. But let's just start by talking about the testing cycle of cruise.

[00:03:53] TB: Yeah, at Cruise, we want to make sure that the software that drives our car is safe before we put it on the road. Cruise's first and foremost value is stay safe. So we want to test it thoroughly in simulation to make sure the car works and then we test it on the car itself to make sure the simulation testing was good. Our main loop is we send our cars out into San Francisco and some in Phoenix to gather data by driving our fleet throughout the city. The cars have over 20 sensors on them. They all create feeds that we capture of data and the cars interpret that data so that they can realize the world around them, figure out what the other vehicles and pedestrians and other objects are doing. Differentiate bicyclists from trees and all

of these things, and they capture that data and bring it back home after every drive, autonomous or otherwise.

When they bring it back, when the drive was over, we analyze the data for areas where we can improve. We look for events where safety drivers tag things that say, “This could have been better,” or moments when they took over. We always have a couple safety drivers in our car, a spotter and someone with their hands on the wheel and foot on the break. If anything looks fishy at all, they take over the car and log it and then we have people post-process that to look and see, “Is this an event where we need to change our software to do something better?”

We bring that home, look at these events, and then the AV team figures out which areas need the most improvement first. They basically rank everything that they want to improve about the car and they tackle the top issues. Sometimes it’s like we need to make left turns better. Other times we’re too chicken about merging. We need to get better at merging. They figure that out and then the AV engineers – AV is autonomous vehicle. The AV engineers work on improving the code that makes the car perform these maneuvers. Once the code has been changed, there’s always the chance that we’ve introduced a bug or actually made things worst. Like we may have made that one spot better, but now the car does a lot of other things worst. This happens in all software.

So we test that code in a huge gamut of simulation tests, and our simulation tests that they run are arranged around a lot of different things we call product areas. They aren’t all parallel. Some are like make right turns. Some are make left, unprotected left. We also have ones that are like make a safe merge. But then there are the ones that cross everything, which is like always be careful of what we call vulnerable road users, pedestrians, cyclists, motorcyclists and anything else that we’re sharing the road with.

Each one of those has a battery of some over hundred, sometimes in the thousands of tests that are these short simulated snippets where we’ll play the scenarios we didn’t like back to the car. Kind of like the Matrix. We have the car, we’d feed it all the sensor inputs and it doesn’t know. It’s not on the streets of San Francisco. We check to see if the car makes better choices, a lot like raising teenagers. It’s like, “Oh! We let him into the wild and it made a better choice in all of these simulations,” and we look at all the data across these tens of thousands of tests that

we run every week, sometimes every night and we make sure that we've improved the car overall and have it reduces the things that are important to us. That's the next step of the loop, is testing it.

Then we carefully deploy that code to the new cars in our test fleet, and the fleet will then drive around San Francisco and find out if not only did simulation think the code was better. Not only did the engineers think the code was better, but that the car performed better in the wild. That last step is the one that we've been impacted hard on with the new COVID crisis. We can't feel there're many cars. I think we're down to about a tenth the number of cars we normally have on the road. So we're starting to rely more heavily on simulation.

[00:08:07] JM: If I understand correctly, first, you can take the car, put it with its current model on the road. It drives around. You've got the human spotters, but the car is driving autonomously and it's gathering information on what's going on, what it's doing, and let's say you have some kind of situation where it makes a mistake, like a clear mistake, like passes a stop sign. It's obviously an egregious error that you probably wouldn't make. But if it did drive past the stop sign, then that would be problematic and you would want to make some fix to the code or you could even have that occur in the simulation with the simulated data that the car would be bringing back.

In either case, if you had some mistake that the model ran through, you would want to alter the code. You would want to fix it and then you would want to redeploy it and see if the car actually – Once it was deployed back in the wild, if that fix was durable enough and flexible enough to actually satisfy the real world use case once it goes back into the wild. Then that process of actually being able to validate in the wild, that has been restricted now is what you're saying, I think.

[00:09:18] TB: That's very accurate. That's well-stated.

[00:09:19] JM: Okay. Can you tell me more about how much can you gather from the simulated iterations? Is that satisfactory for building models that you know are making progress?

[00:09:35] TB: Yes. Simulation has its pros and cons, and I'll talk about the strengths first and then I can talk about some of the difficulties if they're interesting. But one of the strongest parts of simulation is that we can have a lot of different frameworks. We call them to test different problem areas. When the car goes on the road, it has – The first thing it does is gather data in the car driving loop through sensors. We call that perception, which is it takes cameras, radar and LiDAR and it takes the input and builds a model of what that world is doing. That's actually a bus, not just a picture or a billboard. It builds that model. That's perception. Then the next thing it does is it feeds it into what we call planning, which is where it decides that maneuver it's going to execute next. Where it's going to execute it? And it looks and figures out the timing and when to go.

At the very end, we have something we call controls, which when the car goes, "I'm turning left now. Here it goes," and starts driving and pays attention if something goes wrong so that it can do something to ensure safety, usually putting on the breaks.

As the car is doing that, we can capture that perception stack like I mentioned before. All the data coming in and we can recreate that to test the car in the exact same situation as far as the car knows again. That's replay. That's one of our most important frameworks. I think 90% of our tests that we run to it validate new code. We run by replaying old scenarios and making sure they're better.

We also have somewhere – Since replay is expensive. It takes a lot of cloud compute time and sometimes noisy and we'll miss the actual thing we're looking at, we can script scenarios like you would in a mission editor for a video game, and we can say, "Okay. There are three cars here. There are four pedestrians here. Our car wants to turn left. Here are all the tracks. Everything is on go," and we can now drop our vehicle into a highly scripted environment to test it against very exacting scenarios that we've invented.

We have a few of those to test planning and another one to test the controls, to make sure the maneuver is executed right and it makes a good choices about the maneuvers. Lastly, we have one that's an entire 3D world. You'll see in most of our press that we release about simulation, this video game style realistic 3D world that we can use to test the car and feed it into the

sensors and have the car react to a world that doesn't even exist but it's similar to what we want to test.

There are lots of different pieces of the car we can simulate and we can target different pieces of it. I'm going to stop and make sure I'm going the right direction with your question though.

[00:12:25] JM: Yeah. No. This is great, and it almost sounds like these scripted scenarios are sort of like unit tests.

[00:12:34] TB: Yeah. That is very much what they're like. They're a little bit bigger and we just call them product area test suites. We have an entire department of systems engineers that make sure we have full coverage of everything the car is going to do in these big suites that they are a lot like unit tests. I used to call them that when I got here, but the definition is slightly different if you're on the inside. That's accurate.

[00:13:04] JM: Tell me more about the software stack that goes into the simulation platform. I think the simulation platform is called Matrix, right?

[00:13:12] TB: Yeah, that's one of them. The one where we do the 3D world where we can actually render entire visual sets and feed them into the car as if it was the sensor feeds. That one is Matrix. We build that in a common high-quality game engine that people used to make HD games. Because about 90% of the code, you need to make a realistic-looking world that's already basked into that.

That one is a big C++ engine. We can run lots of tests on that in the cloud. We can use a GPU farm to generate all the visuals, because if you think about it, people talked about VR as being more graphically intensive, because it had to project the screen twice, one for each eye. Our car has something like 23 or 28 eyes when you think about it, because we have all these different cameras and things. We have to make sure that we have a very efficient system to be able to run this, and we run it at a slower frame rate than reality to make sure that we can be very accurate. That one is Matrix, and that's one that we're always evolving, because we believe that we'll end up heavily dependent on it towards the end for some of the final stages of AV

development, which is generating data we can use the train the cars without ever entering reality.

[00:14:30] JM: Wow! When you say the final stages, what do you mean exactly? Is that like in terms of the roadmap to complete autonomy?

[00:14:38] TB: Yeah. For me, I think of it as the final stages of building a simulation where you can test everything you would want to test and to be able to let the cars run for extended durations. We call that exposure testing, and to be able to feed it all sorts of new 3D situations.

The other frameworks we use are generally two-dimensional. The ones we used to test plannings and control sim, we usually have a top-down view of those, because you really only care about the things at the street-level; the cars, the cyclists, the buildings, the pedestrians. Where the lane markers are? Those are a lot cheaper to run. Every time we can run the tests in these two dimensional simulations, we can run a lot more of them because they use less cloud compute.

Finally, we have replay, which is kind of in the middle. We feed a lot of these old data to the car and we just run the car stack. We don't have to create a fake world, because we're feeding it the stream that once was its world. A lot of the replay testing framework is some Python and some C++. Just to get that feed to a car and to spoof the real world for the car so it think it's in one.

The Matrix framework, lots of C++, lots of code to create an entirely fake world and then build sensor imagery, whereas the replay framework is a bunch of scripts, code and backend framework pieces just to feed all data to the car and see what it does. The tech is all over that spectrum.

[00:16:16] JM: The simulations, as you mentioned, they need to mimic these sensor inputs. The simulations actually mimic the camera, the radar, the LiDAR. It somewhat sounds like a virtualized hardware situation. Can you tell me more about what exactly do you have to program in the simulation? I mean, it doesn't surprise me that you would have to program a physics engine, for example, to know how the car is driving along the surface and it's bumping along a

gravelly Road, but what about the actual sensor inputs? To what extent do you have to mimic like the actual hardware inputs?

[00:17:00] TB: That's a great question, because some things are decently easy to begin mimicking. Videogame technology is used to create the 2D images the cameras would catch, and we've all played videogames or at least seen them played and you say, “It's pretty real.” There's a lot of effort to get the last 5% towards real, but you often don't need to do that in simulation. Yeah, that’s mainly for training data, which we can talk about later. But some of them are harder.

LiDAR is also decently easy to mimic. We have an entire simulation group called sensors, by the way. It's a very clear name. The sensors group looks at the sensors we have, makes our models as best as possible. They make the best camera models they can, the best Lidar models they can and the best radar models they can, and they also test new hardware before we think about adapting it for cars. People will say, “What about this new LiDAR model that's coming out that looks like it has double the resolution? What would that be like?” They also test our calibration rigs before we build these big warehouse calibration rigs for the cars. Sensors are very important. That's why I like this question a lot.

Coming from videogames, I first was like, “That's cool. We can perfectly model all the sensors and then we can perfectly build the world, right?” They’re like, “Not so fast. We can do a great job on video. We can do a great job on LiDAR, but the defense industries of the world are still trying to get radar right.” Radar is harder and we can use a lot of statistical models to build out a radar simulation, but it's never as perfect as the AV engineers really want it to be. So we can come close, but we always know there's a little Achilles' heel in there. Radar has a lot of like atmospheric lensing effects and a lot of things that cause distortion, and it’s very complex to model.

But we do a pretty good job. That's what matrix will do a lot, is figure out how to see what the world would look like with new sensors before we invest your tens or hundreds of thousands of dollars into actually mounting them and trying them out. That team, our sensors team, has helped us make some really good decisions over the last couple years. Yes and no decisions on will this make things better? Having the 3D matrix lets us do things like identify blind spots for

the different sensors. Help us choose placement and even help us model as we change from the Chevy-based platform we have on the road today to the one you've seen in the news with no driver at all. That's a little more like a rectangle than a car shape. It's helping us make all the right choices for placement and which sensors to use so that we don't overbuild and that we don't under build.

[00:19:47] JM: There are also interactive agents in this simulation. You got pedestrians and cars. How are these agents programmed?

[00:19:58] TB: They're programmed a little bit differently than I would've done them in videogames. That's another one of those moments where I came in, and in my early videogame career, I love the program AI. In videogames it's called NPC's, non-player characters. All the other people running around are just computer-run and all the cars that aren't run by players. That term came over to the autonomous vehicle industry. It's called NPC AI here too. But at videogames, we would have programmed the AI's to do things that were fun. A lot of times they're targets or they're sidekicks. They're designed to take you to some of the falls for you while you try and be the hero in the game. You program them to be such.

In the world of autonomous vehicles, we want them to be as accurate as possible. The first thing we do is we just – We'll use the simplest thing, a scripted path. You can imagine in replay, if you got somebody crossing the street, they're going to do the same thing every time, because it's a scripted sensor input. But then later on, we want it to react if the car does a different thing. If the car is going to take a little bit of the right-of-way out of a pedestrian, you would expect him to stop.

We started the program thinking in reactivity where you have the AI's doing things like would expect them to do, and sometimes they're scripted and sometimes they're playing back something, and AI actually did. But our scenarios are short. They're like 20 seconds long, sometimes at the max. No one would ever play a 20-second videogame, but that gives us the advantage of saying, "Here's what this pedestrian is going to do. Now, let's let it react."

What we've been doing lately is we're starting to team up with the AI prediction group that the car has. There's an entire prediction group in the AV stack, building a piece of the AV stack that

goes, “Okay. I see the bus. I see the pedestrian. I see the bike. What it’s going to do? It looks like it's turning left based on its lien and it looks like this, that and the other thing.”

We’re starting to team with them to make it so that we can have AIs that use the machine learning models that they're building on how do we predict with these cars and cyclists are doing and bring that back to our totally fake simulated worlds to say like, “Okay. Now, can we have our pedestrians do these predicted behaviors based on the reacting think car we have?” There's a lot of work going on there, and this is one of the areas where we’re far from perfect. We have some basic AI where they follow paths, like things on an it’s a small world ride. Everyone starts with that. Then we have ones where they have basic reactivity of get out of the way. And we’re working on things like how can we make AIs that will accelerate the problem for trying to detect and solve like distracted AI? The person crossing the street while checking their phone, and kind of late-night party or AI. Kind of oblivious to what's the sidewalk and what’s the road. We’re looking at how we can take those AIs and then concentrate them to give them different variables that we can use to parametrically see what different kinds of people would do in these situations, because there's no one average person. We have to be ready for every person on the road.

[SPONSOR MESSAGE]

[00:23:22] JM: When the New Yorker magazine asked Mark Zuckerberg how he gets his news. He said the one news source he definitively follows is Techmeme. For more than two years and nearly 700 episodes, the Techmeme Ride Home Podcast has been one of Silicon Valley's favorites. The Techmeme Ride Home Podcast is daily. It's only 15 to 20 minutes long and it's every day. By 5 PM Eastern, it has all the latest tech news, but it's more than just headlines. You could get a robot to read you the headlines.

The Techmeme Ride Home Podcast is all about the context around the latest news of the day. It's top stories, the top posts and tweets and conversations about those stories as well as behind-the-scenes analysis. Techmeme Ride Home is like TL DR as a service. The folks at Techmeme are online all day reading everything so they can catch you up. Search your podcast player today for Ride Home and subscribe to the Techmeme Ride Home Podcast.

[INTERVIEW CONTINUED]

[00:24:25] JM: The simulation strategy obviously involves being in the real world, and in the 3D simulation, you have to mimic the real world. Do you have any feedback mechanisms for measuring the diff between the real world and the 3D simulation, like ways of kind of understanding? How closely are we actually resembling the real world with our simulation?

[00:24:57] TB: I love that question. It describes about 50% of our effort right now, and we call it reproducibility, the ability for our car to mimic on road behavior and let loose in the simulation. There're a lot of gaps. Simulating the world is really harder than it sounds at first. Vehicle dynamics modeling, there is a range of what you can do. You can start with the car just driving along in a track based on old data. Then you could say like, “Oh! Well, I want to have a model where it has some power and it has the mass. You usually can whip up something. We call it a bicycle model, because it has only two wheels and you just don't let it roll like a motorcycle and you say, “That’ll handle kind of bulk mass dynamics.”

Then the you will eventually end up with more of a planner model where you the car as a sprung mass on top of the wheels, which was stuck to the road. That's what we’re used to with cars, and then we can start modeling comfort and are we giving our passengers a good ride or are we making them sick? That's one example right there, and at the extreme end, there are third-party companies where their entire job is to build car simulation models, and they’re expensive, they’re used by companies like General Motors to test their cars and see how they react in a lot of different simulated situations for real cars that humans drive. Those are so big and so expensive, and also so proprietary that we can only use one or two of those to test the cars out and their overkill. Those have crazy things, like torsion on axles and the stuff that can create minor vibration effects, but we don't care about yet. We have to choose our battles. Where will the sim be realistic?

Just to give you some perspective. Right now we’re going like we are in San Francisco, we should probably put the hills in. Build the roads to be actually 3D. A lot of our situations happen in 2D, like left turns, right turns and stuff, but we’re starting to bring in data and say like, “Okay, we actually need to model these roads better to actually match the elevation.”

The way we've picked our battles is that as we run all these tests, especially the replay tests, we get out the metrics that we care about. How close are we to the other cars? What was the personal space we gave pedestrians? What's our timing for the car behind us? Did we break too quickly? We have names for all those metrics internally. They're all three-letter acronyms. So I won't bother you with them.

We get those out of the real drive and we're like, "Okay, now let's play that back either in replay or a sim and see how much they vary." Then we can go and analyze those. Find out where the variance came from. What the root cause is? Questions like – Sometimes we'll go like, "Why is our car a meter out of position? That's crazy," and we're like, "Oh, it didn't merge. Okay." Sometimes that's good. It's like merging with what caused the scary event last time. Let's not merge again. Sometimes it's bad. It's like, "Dang it! It was supposed to merge." We start to do all of these to get the dynamics to match, to get the cars behavior to match, and any of these other thousand variables that can make the car in the world not behave the same way going forward. That's a huge push for us right now, because our cars are off the road. So we need to rely on sim even more.

[00:28:18] JM: By the way, are all the cars off the road? I feel like I've seen some Cruise cars around SF.

[00:28:23] TB: No, they aren't. That's been great for us. We have some Cruise cars around SF. What we did is we have drivers that were saying, "We're itching to do something," and we had cars just part. We had to spend a while making a safety protocol for it to work so that we'd make sure it was careful. But Cruise contacted various charities in San Francisco and said, "Can we use our fleet and our drivers in some way to deliver food to the most vulnerable people in San Francisco?"

You'll see the Cruise cars driving around, and we're gathering limited data, but they're actually being used as delivery. Because our ideal is like, "Let's do some good while we're her. We've had this program that we've been very proud of where the volunteers who want to drive, follow the safety protocols, use our cars, and they're basically delivering food to people who need it and can't get it themselves.

You see that San Francisco, and there's a side effect. It allows us to collect limited data. This is not our whole fleet. It's only a few of the cars, but it's been helpful for us, because we can get up-to-date data, because a lot of our old data is like three months old now. There're also a few cars driving around in Phoenix as well. Once again, following safety protocols. We get that, and I feel that my gut is we're getting about 10% of the data we used to out of these two different initiatives.

[00:29:51] JM: 10%. How do you know the simulations that you're running are useful if you can't validate them right now?

[00:30:01] TB: That's what we're using, like once we do it, we can use the cars that are running to check them out. One of the positive silver lining things to come out of this situation is my simulation group now meets with the AV engineering group about three times a week to make sure we're doing coordinating on exactly doing what they need so they can validate exactly what they're trying to build right now. The groups were a little asynchronous before where sim was like, "Okay, you give us a list of five things and we'll build them this month. By the way, we're working on this matrix thing," that you don't quite use as much because you don't need it all the way. Now it's like, every day, we go, we groove our backlog and they say, "We got three new metrics to test safety and four new metrics to test comfort, and we need to get the simulation on-road results to match." It's sort of like, "Oh great. They don't."

About three times a week now, we focus our efforts on using those few drives we have and the significantly attenuated mileage accumulation to make sure we validate that. It's interesting, because it's harder, but we're trying to make it so our development doesn't really miss a beat during this time, and the lessons we're learning will be very useful when were allowed to fully deploy a fleet again, because we're so coordinated with the other team. There's been a lot of good come out of.

[00:31:27] JM: If it wasn't apparent from this conversation already, you're the VP of simulation. What does your work entail?

[00:31:35] TB: I guess the best way to describe my job is one of our founders and our CTO, Kyle Vogt, asked me to do two things that he thought were important when I signed up beside

just run the group, go to meetings. I'm going to leave that off the table, because we know that happens. But he said, "I want you in a certain amount of time to get simulation to a point where it's so good that simulation validation replaces on-road validation of a car's code." Now that doesn't mean that we would have the car stop verifying that the code was correct, but the goal was to get them so accurate that the AV engineers almost always go to simulation first.

He gave me a certain amount of time to do that, and my teams trying to beat that time, because we're really excited about it. The other thing that I do is I spent a lot of time making sure that it gets easier and easier to author the tests that aren't just the replay tests where you capture them and press play. Those tests, it can take a long time. It can take like a week to author some of the complicated ones, because there are so many tiny adjustments and variables that we do. I've been working and finding out what things are blocking people from being able to build those tests. That's another important thing he had me do.

I think it turned out somewhat serendipitously that I ran into the experience that we talked about before, where I'm spending a lot of my time kind of refocusing my group to be a customer service group internally. It is interesting. A lot of us built video games before. Not all of us. I'd say like 30% of the team, but we're like, "Oh cool. We'll go build that for you." But that's not how it works if you're serving internal customers. You actually have to go out and talk to the people using your code. You have three or four times a day and say, "Did that work for you? How can we adjust it?" We have to deploy our engineers to work directly with them to make sure it does what they want, not just what the product requirement thought it wanted. That's the other thing I'm doing, is I'm integrating the simulation team really tightly with all of our AV creation efforts. That's really fun for me.

One of my previous jobs was at a company where one of their top values is just focusing on the customer, that was Amazon. We know them well. I really liked that and I've been trying to bring that into our internal group at Cruise. Those are the three things I work on. Making the sim more accurate, like we talked about before, and then integrating tightly with all the groups that rely on us within Cruise so that we can deliver faster and more accurately to what they need.

[00:34:07] JM: Tell me more about what your interaction is like with the other teams. Specifically, there's the road testing team. What's your interaction with the road testing team?

[00:34:21] TB: By road testing team, do you mean there are systems engineers who analyze the data when it comes back from a drive. And then there're the actual people who are driving the cars. I assume you're talking about the systems engineering team that analyzes the data post-drive, right?

[00:34:33] JM: Yes. Yes. I actually really like the interaction with them. They're very data-driven, statistically-minded group that is building test-driven development model for improving the car. There are three things that they're improving. They're improving safety. That's always number one. Then they're improving ride comfort, because no one wants a safe ride that makes them sick. That's no good. The last one is trip time. You want to make sure you get there in a reasonable amount of time or you're going to walk or call a cab. That's what they do. They design these entire test suites like we discussed before, to test all of that functionality. As they design those suites, they have systems test engineers build out the tests. They build up the replay test based out of snippets from previous drives. They script the test so that we can do the 2D simulation test to make sure the car is planning right and that it's going to get run through the intersection correctly and all of that. Then they run the simulations. We build the simulations, but they run them.

System tests builds out all these tests to hit the coverage area their systems engineering group has asked for. The simulations run. At the end of that, then the systems engineers get the results back as all the logs, but really what they're looking for is the metrics. If you run 50,000 tests, you can't look at every single test. It's just impossible. You want to make sure that those tests are self-validating to make sure they're okay and we're always working on that. Then to make sure that the data come is the metrics you're looking for for the different product areas you're testing to make sure things got better while not making other product areas worse.

We're in the middle, and that system's engineering group is running the show. We always need to know what metrics are you testing next? That's almost what it boils down to, because they don't really right now need us to run long SOAP tests and look for trouble. We have plenty of trouble. I think we've got over a petabyte of data. I don't know for certain, but what our retention policy has turned it into lately, but we have a lot of old trouble that we can look at and try and improve.

That's a relationship with them, is we're a non-black box. They know how we work. They write the tests. They get the information out, and then when on-road doesn't correlate to simulation, they ask us to make the simulation better. That kind of brings the four pieces of our conversation together hopefully. That's the relationship.

[00:37:04] JM: What about the other management structures around you and the information flows around you? Can you give me a picture for how the simulation team fits in organizationally with the rest of Cruise?

[00:37:19] TB: Yeah. I think, organizationally, there are verticals a Cruise a little bit. Simulation is one of them, the different simulation frameworks. I report straight to Kyle, our CTO. Then my peers are – There is robotics groups the report in to Rashed Haq, who is our VP of robotics. I think he also runs the systems test group. Then our systems test engineers actually work directly for the groups that need the tests written so they can have a tighter loop.

What we have to do is put together a lot of process to communicate between the groups. We have a lot of small teams of 8 to 10 people working on different problems, but we have a lot of process to make sure that I talk to Rashed and his directors to make sure we're acting on their highest priorities. That's really spend a lot of time meeting, putting our heads together and figuring out what we're going to solve next. So me, directors, engineering managers and even lead engineers, when we go to write a design doc for a new thing we're doing in simulation, we work directly with our partners on the other teams so that we know we're answering the questions that they're trying to ask even if there is already a product requirement doc and a tech design doc. We want to make sure that we collaborated and get it done together.

There's a heavy degree of necessary collaboration across these different verticals and we're always trying to improve that, because any organizational model has its problems. If you're heavily matrixed horizontally where you have these teams that have people from different disciplines, then you run into the things like, "Well, who's the career manager of those different disciplines? Then you end up with a pooled model. I never get into org design that much, because whatever you get, my job is to make it work.

I like how it's working at Cruise. Lots of collaboration. The saving grace is our mission. I think if we've probably talked about it with you before, but just to restate it, everyone at Cruise is doing the same thing. If how we're going to do the mission changes, everyone at Cruise changes, because we have Kyle and Dan, our CEO, hit in front of the company every week and tell us what's up, what's changing, what the hot news. The mission itself, I've memorized it. Let's see if I get it right here. Cruise is building the world's most advanced self-driving vehicles to safely connect people with the places, things and experiences they care about.

Now, below that is more detail, which is we believe self-driving vehicles when deployed at scale have the potential to save millions of lives, reshape the physical environment. There's a trillion acres that's just parking lots. That's what we mean when we say reshape the environment, and there are streets that you have to wait to cross because there are too many cars. We want to give back billions of hours of time and restore freedom of movement to people. There's just so much good that would come out of this that we're all excited about that mission.

When we change how we're going to approach the mission. When we try research initiative A and it doesn't work and we all go to research initiative B, we get together and we figure out how to make the next one work. That's the gift of having a singular focus company and that gets us past all the organizational issues that usually have to put a lot of process together at other companies to fix.

[00:40:38] JM: Beautiful.

[SPONSOR MESSAGE]

[00:40:48] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:42:36] JM: You worked for many years in video games and I'm really curious about how the simulation work that you're doing at Cruise compares to designing real world in video games.

[00:42:53] TB: Oh! This is a fun one for me, because I loved making video games. All of my college work within cars and robotics and systems engineering for control systems, but computers are crummy back then. So I went to go simulate things for video games. I finally was able to boil it down in what are their main differences where the passion for video games carries over to the cars. It's a simulated world. In video games, you're always trying to make a world feel real. Not every videogame, platforming, jumpers and stuff like that. They don't try and be real, but I'm thinking of the HD video games I used to work on, like football games and car games and stuff like that. We're always trying to have the world look really realistic, which we need in the world of AV. If we're going to build fake data to train our machine learning algorithms for these cars, it's great to be able to have the data be as real as possible.

We also – In video games, we were hypercompetitive about performance. We wanted to use every clock cycle. We wanted to make sure that we always ran at the best framework and we

always want to say we did that first. Music to my ears in video games when I worked on these high-end video games that sold millions and millions of copies was I love to hear people say we didn't think this was even possible.

For us, that's where we do a victory dance. To do that, we had to have hyper-efficient resource management. We had to be really good in three dimensions. We had to be really good at understanding why graphics do what they do. I member in the early 90s, everyone was like, “Whoa! What is lens flare? No. It's boring, right? Because it was overdone.” They were like, “How do we do like ambient occlusion? How do shadows work?” Working it out in video games is kind of came first, because billions and billions of dollars are dropped on video games to do accidental RND on how to make look graphics better. How to make 3D engines for worlds more efficient? How to stream data you don't need in and out when you do need it? Off of discs or off of the Internet.

These big engines came into being third-party commercial engines and then custom engines on the inside of companies that people could use. We started solving all those problems. I was there for the whole ride. I joined video games right when 2D went to 3D and I got to enjoy all of this with like 2000 of closest friends. We’ve built all these games together. That was great.

I think when people were building AVs, they started to realize, “We need to use these videogame engines, because that's 5 million lines of code closer to our target that is already been tested by tens, if not a hundred games,” and it became natural to hire people who are good at that.

[00:45:33] JM: Can you tell me more about like does the development actually have any resemblance? I mean, I guess the game is more of an entertainment scenario. But like if you're – In some sense, it’s like the bar for the game is higher because you have to make everything look really pretty. But in another sense, the bar is lower because in a simulation, you're building a product that is going to eventually have life-threatening consequences.

[00:46:02] TB: You know, I just am thinking while you’re asking this, that a good way to answer this is how does the shape of the team changes between the two? Because that answers these

questions. In video games, one-third of my team would be artists, because we had to make everything look good. We had to make everything explode correctly.

At Cruise, we don't make anything explode. We needed to just be realistic. We see the art team is smaller. We have like four people working on art at Cruise and then they'll hire contract houses to build the art instead of when I was working on bigger games, we'd have 100 artists. Then we also – We have less gameplay and UI engineering. You have to make a videogame easy to use. You end up with maybe 10 people just trying to figure out how to make it so that anyone can play and that the UI is crystal clear, and that everyone knows what to do next.

For us, we're running a lot of these in the cloud. Our UI is more like running Jenkins sometimes, like I want to use a batch build. Then there's what you're talking about, is I think I'll boil that down to precision. I need to spend a lot of time making our simulations match on road performance for the car and to make the NPCs, the pedestrians, the cars, the bicyclists, all of that begin to match what the real world is like. I need the visuals to be accurate, not gorgeous.

We're working on high noon in San Francisco may not look stunning. It's not an Ansel Adams photograph. It's also not black and white, but at the same time, we're like, "What does it really look like? Let's get this right so that the car sees what the car would see."

There're a lot of parallels, but we spend more time on accuracy. We get the car models accurate. We don't actually enter the cars. So we don't have to model all the pieces that don't ever happen, like getting in the backseat and seeing the back of a seat. The other thing is, there is no development cycle like a videogame. And video games, towards the end, everyone works really hard to get the game done by Christmas, or football season, or something like that, and then they take a month off to recover.

We're just always running. So we have a standing team. We can't search people on at the end to get something done and then I'll take a month off. We end up with a team that's smaller than you would have to build a videogame, because we need less gameplay. We need less UI. It's heavily engineering- focused. There's less design. A lot of times in video games, you'll be designing a game for one market and then that market will either die or another company will dominate it. I think we saw that happen with the last man standing genre. The one where like

everyone gets in an arena and then whoever's left wins. We saw that evolve through a lot of different games until one really good company kind of captured the market and started evolving it somewhere else. The other games are all still there, but while you're developing your game, if you see that happening, you have to be light on your feet to hit these new creative targets. Our target isn't as creative. It's driving. It's like we take everything about building videogames and we take the fun out of the game. We still enjoy building it though.

I hope that partially answers your questions. Is there more I can talk about there?

[00:49:22] JM: No. That's plenty, and we could begin to wrap up. I do want to ask you, when it comes to outstanding engineering problems at Cruise, what are the other problems that you're faced with right now? What are the engineering problems that you're most focused on?

[00:49:39] TB: I think they're all subcategories. Well, most are subcategories under making simulation match on-road performance. There're some difficult problems with determinism there, like the butterfly effect. The cars themselves have all these different sensors, all these different processes for the sensors running at different threads with different timing. We have to solve the problem of how do we re-create that scene, which we call determinism? It's like can we get the timing just so so the car does the same thing the same time with the same AV stack?

There're people that actually aren't on the simulation team who've been working on that a lot, and then we incorporate their simulations in the sim. That's one of them. Then the next problems we have are detecting when hard tests succeed or fail. The example I gave earlier where our car is a meter out of position. Was that good or bad? Because let's say we don't want the car to change lanes now. That means that if the car is a meter out of position, it didn't change lanes perhaps. Then being able to automatically look at that and write these tests so that we know that that's like, "Oh, that test passed," versus if the car is a meter out of position because there was an error, we're like, "That test failed." That the next one we have, is how do we detect the validity of the tests themselves and solve a garbage-in, garbage-out problem? There's a lot of that.

The other things we're working on are there this problem that might seem complex, related to video so that everyone in the audience knows this. If you're working on a game and you save

your state, you have a videogame save file. But let's say you changed the game and now that save file doesn't match. There's no level 5. The sword you thought you want is gone.

As we change the state of the AV stack, saving the state of what the car was thinking starts to corrode. So we can't enter one of these six-hour replays in [inaudible 00:51:35] to do a 20-second run to see if it would do the same thing. As the AV stack evolves over time, that replays starts to become invalid. The car would never be between two double parked delivery vans now. How is this test even valid? We're looking at ways to be able save state and modify that status we developed. As we change the AV stack, what does that mean to what the car was thinking when we dropped [inaudible 00:51:59]?

The other alternative is to start the car further back in time and let it build a similar kind of headspace, but doing that doesn't always guarantee the same headspace, and it's expensive, because that takes longer than running the test itself. They could like double or triple the cost of running the test. That the next thing we're solving, is we call it – This is a contentious term a little bit. It's called the hot start problem, because we've applied it to a lot of different things, but it's like how do we just drop in the car and have it be thinking the same thoughts? It's really hard. So we're looking, "Can we solve around that problem? Can we do it?" That's another big one we're solving.

Then the other ones we're solving is just like, "Well, now we've got to work with heavy rain. What about snow?" All these different weather conditions, and then we're working on localization. How can you just pick a spot in San Francisco and say, "I want that spot modeled in 3D for my matrix simulation so I can start scripting." We have some projects we've been working on there, like where we have very specific spots in San Francisco that we have modeled out so we can say we can grab that.

Then we have to be able to change it. When the public works, people go in, throw out a bunch of cones, dig up the street and put new lane markings on. We have a fleet that runs around San Francisco that I'm just mapping so that we can catch that and then change it. Then the next problem is what if it's not San Francisco? What if we need to model Houston?

There's a lot of forward R&D work. I focused on the right now problems because I thought that was most interesting for the interview, but forward, we'll have to deal with snow. It doesn't snow much in San Francisco. It's not while I've been alive. Then we have to also focus on different cities, different countries, different locales. There's a lot of work to go on there. The other front we're working on is we're starting to get more and more request for training data. Can you use your 3D matrix simulation to create scenes that I can use to train my machine learning algorithm before we go out on the road and start training it with real data? Those are the big problems we're working on right now.

[00:53:55] JM: All right. Well, just to wrap up. Do you a sense for how close we are to full autonomy?

[00:54:02] TB: I think I'm so close to it, and I'm thinking so fast about all of it that I don't trust my sense of that. I think that because I don't trust my sense, I fall back and look at what actually the combined journalism that's approaching the topic looks at and say like, "Okay. What is the scientist outside of me think?"

But I think it's near. I don't want to really talk about timing and dates that much, because I'd like to not make up my own answers that might overlap Cruise's different answers. But it's really close. Before COVID, I was riding around in these cars every day and we had some rides where the car would drive to Chinatown, which is mainly pedestrian. It's little tiny bit of room for cars to go through, then drive down Market Street and then go drive – And there are whole drives for the vehicle operators, the AVT didn't have to take over. The car just knew what to do. I just see the rapid amount of change Cruise is making as they just brute force bang their heads on all these problems and solve them. I think it's closer than people think, but I'm going to be careful. I don't really want to commit to what, but I can say is really important to Cruise. I want to say, if you don't mind, which is we will release our vehicles to drive themselves when they're safe. That's the number one thing. They have to be safe. The word we use internally is superhuman. We are working to statistically measure how to prove that our cars are a number of times safer than human drivers before we'll let them on the road. You'll see it's been driven by safety and that's going to be what lets us decide.

[00:55:41] JM: Okay. Will, Tom, it's been wonderful talking to you. I won't hold you to any dates of autonomy.

[00:55:47] TB: I didn't give any, so we're in good luck.

[00:55:49] JM: Exactly. Cool. Well, thanks for coming on the show. It's been great talking to you.

[00:55:53] TB: Yes, thanks.

[END Of INTERVIEW]

[00:56:03] JM: Scaling a SQL cluster has historically been a difficult task. CockroachDB makes scaling your relational database much easier. CockroachDB is a distributed SQL database that makes it simple to build resilient, scalable applications quickly. CockroachDB is Postgres compatible, giving the same familiar SQL interface that database developers have used for years.

But unlike older databases, scaling with CockroachDB is handled within the database itself so you don't need to manage shards from your client application. Because the data is distributed, you won't lose data if a machine or data center goes down. CockroachDB is resilient and adaptable to any environment. You can host it on-prem, you can run it in a hybrid cloud and you can even deploy it across multiple clouds.

Some of the world's largest banks and massive online retailers and popular gaming platforms and developers from companies of all sizes trust CockroachDB with their most critical data. Sign up for a free 30-day trial and get a free T-shirt at cockroachlabs.com/sedaily.

Thanks to Cockroach Labs for being a sponsor, and nice work with CockroachDB.

[END]