Episode 9: Pushing the Boundaries of What's Possible
Host: Nicole Huesman, Intel
Guests: Alex Wells, Intel; Max Liani, Pixar Animation Studios

---

Nicole Huesman: Welcome to *Code Together*, an interview series exploring the possibilities of cross-architecture development with those who live it. I'm your host, Nicole Huesman.

To deliver increasingly captivating stories in the world of animated movies takes passion and determination. The passion to continually push the boundaries of what's possible, and the determination to continually seek ways to make technology easily approachable by artists so that the artists can focus on what they're best at—telling stories that capture our imaginations and move us.

I'm excited to welcome two guests to today's program who live at this intersection of technology and art.

Max Liani. With over 15 years as a talented, charismatic CG lighting supervisor and software engineer spanning Pixar Animation Studios and Animal Logic, Max has developed and optimized technologies used in movies like *Cars 3*, *Coco* and *The Lego Movie*. Thanks for joining us, Max!

Max Liani: It's great to be here. Thank you!

Nicole Huesman: And Alex Wells. He leads Intel enabling of data parallelism in digital content creation for movie rendering, character animation and special effects tools. Along with Martin Watt, Alex Powell, and Jason Reisig, Alex was recognized in a 2018 Technical Achievement Award from the Academy of Motion Pictures Arts and Sciences. Welcome, Alex.

Alex Wells: Hi, how are you doing?

Nicole Huesman: Great, thank you.

The canvas that animators use is different than that of painters or sculptors. Max, can you talk about the importance of working in real time and of the live feedback of tools?

Max Liani: As an industry, we've been always pushing the boundaries of what is possible to achieve visually. And pushing the boundary means from a storytelling perspective or like creating something that has never been seen before, but also the richness of the imagery and the complexity of the environment in characters we need to create to make them believable. You know, in animation, we share a lot of technology with visual effects where the words need to be to the real world, so that others can immerse themselves, but it doesn't go that far away from that truth in animation where the words are equally more complicated.

And so pushing those boundaries means pushing also what computers can really do. Working with toolsets and technology will allow us to visualize interactively, working directly for an artist, that means for an artist to be able to create better art because artistry ultimately is an exploration process. You start with an idea and you refine the idea as you go, as you see it appearing on your screen. When it takes too long to see your image, that

interaction of exploration of your art becomes partially impaired. You cannot really find those happy accidents or really narrow down to what you really want to show. And those compromises are when we're trying not to deal with. So the process of rendering ultimately is a problem of high performance computing.

Nicole Huesman: An animator's visions are limitless, yet the capabilities of a tool are finite. Due to these limitations, Max, what challenges do artists and animators face today?

Max Liani: The challenge is often, what can we do with the finite time and finite budget? How easy, or how quickly, can we develop the technique to really reach both the creative process, but also fulfill the creative intent? How can we create those pictures, those animations, those effects is often running against the clock of when the movie is going to be delivered.

Technology needs to be developed first. Sometimes technology needs to be simply adapted from what we already have, and sometimes developed ad hoc to create what makes unique that particular style or visualization in the movie. And we need to create efficient solutions. Sometimes we are one cycle behind, you know. Sometimes some technologies are developed, it is used for a movie, and then it gets optimized and refined for the next movie, so that that one will be more efficient and getting better production value, reusing a better version of what was used before.

So it's always this interesting dialogue between your intent and what can we do technically, what techniques need to be developed, pure research …

Nicole Huesman: Absolutely. So there's this tension and between the artistic and the technical. As technologists, you and Alex have been collaborating on ways to give artists what they need to be truly expressive without the burden of these technical complexities.

I'd love for you to both share your experiences and Max, why don't we start with you? Can you share your insights?

Max Liani: Absolutely. It's kind of like, it's never enough. We need to give to the artists tools that are as interactive as possible, and that means taking best advantage of the computational resources that are in our computers. Computers are made of CPU and GPUs, which have very different performance characteristics and programming models and perhaps even choice of algorithms that works best in one or another.

Pixar RenderMan embarked on this challenge and this path of completely rewriting our production technology. The central piece of this design was, if you're a genius compute, we really want something where we can take a piece of technology that we can really easily adapt to the evolving nature of computing.

And something that was interesting in this process was moving away from more of the traditional object-oriented design. There was an extensive utilization of the technology and move toward a more of a data-oriented designing, kernel launch-based approach to

rendering that, based on our past experience, we thought we could optimize more powerfully.

And as part of the collaboration within Pixar and Intel, we matured a set of techniques and we applied what we learned over the years of our collaboration. See how much we can both develop our programming models and develop our techniques that some are specific to the renderer, to our specific niche sector, but some are kind of pushing the boundaries of what compilers can do and what really can be the amount of complexity in the solution.

Alex Wells: Well, certainly with trying to take advantage of a new architecture and new hardware, it's a great opportunity, as Max mentioned, to change the way that the compute is happening and move away from a traditional object-oriented to more of this data-oriented approach. What we've discovered along the way is we're able to apply a lot more optimizations, we're able to get much better code generation, and we're able to more easily take advantage of new instruction sets and upcoming future hardware. And, you know, as far as some of these things that are happening, one of the trends we're seeing is Just-In-Time (JIT) compilation. This is very effective with things like Open Shading Language. But there's always this balance of when to use these new features, which scene is it going to be good for, and so I know RenderMan has done some development of auto-tuning to control when to use these features and when not to, and deliver good results that scale forward.

Nicole Huesman: Yeah. And I know that we talked earlier about, if we can make something fast enough, then we can reduce the technical burden on the artists. Can you talk a little bit about that and maybe what that means in the context of heterogeneous computing?

Max Liani: Part of the tools that we create, they're more like commercial airliner in terms of the amount of control in the cockpit than, you know, a consumer car. They have a lot of knobs, they have a lot of tuning parameters, because they need to be used in a variety of circumstances that weren't foreseen yet.

You know, like the filmmaker pushing the boundaries by definition means doing something that was not expected, and that often ends up in the technicality of it to fulfill the artistic vision. And so these complex machines, they have a lot of tuning parameters for trying to optimize the performance of the machine itself.

Sometimes there are technology breakthroughs coming from research or the engineering side where we managed to create more robust solutions where we don't need a tuning parameter, but sometimes these parameters are required, and they are often required because the amount of compute is never enough.

Typical production uses large clusters. I don't have any numbers that I can quote but they use a really, really large amount of compute during the movie. And so there are expert users that know how to dial and configure at best one scene for fast throughput.

If something is fast enough, you're not going to spend time tuning. The amount of technical talents can be spent developing techniques rather than babysitting the renderer, trying to

maximize the amount of images that we produce overnight. And so by either developing better technique or implementing more thorough optimization throughout the system, we can get to a point where a renderer, in fact, it becomes easier to use.

Because there is less chance that you are going to manage it the wrong way, you can start from a solid baseline. And if you are happy with that result, you know, you might have one configuration that applies to an entire sequence or to an entire production, and you're happy with the result. So that is a big opportunity for simplification.

At the same time, when something is simpler, when you have a simple machine, you can focus your optimization resources from an engineering perspective. You know, smaller, subset of different code paths and corner cases, so there is a point in time when things get fast enough that you get in a positive … cycle where you create a better technology that can be optimized better and that will result in a better yet speed of rendering, therefore, better interactivity. The artists will be more interactive. Therefore, you will work to fix problems and make changes visually rather than perhaps adding more complexity because of the lack of be able to figure it out from artistic perspective, or without asking feature to do very specific, fine-tuning of the image using numbers.

I make a silly example. If you can position a light source in space and see the illumination feedback live, if the image doesn't look quite right, you might move the light until it looks right. And if it still doesn't, you might put some screen in the middle that will cast a shadow in the region of the image that you want to control in terms of exposure, in terms of driving the user, the audience's attention, from a composition perspective.

But if you cannot do that interactively, then you will start asking features, can I do this thing? Can I add a control that allows me to expose down that image based on distance from the light? And that adds complexity. Complexity turns into more computation and often lower rendering. So you kind of have this, you're pushing a rock up the hill.

Or you might, you know, look at rolling the rock down the hill. And so if we make things fast enough, then we might enjoy seeing this rock tumbling down, enjoying that rather than pushing all the way up, adding more complexity and then having slower compute.

Nicole Huesman: So, let me ask, are there any other things that you'd like to talk about either of you, about the way that you are working together, the way that you're collaborating to help artists to give them what they need for life feedback, from their tools to make things easier for them so that they can focus on what they do best rather than the slogging through the complexity of their tools?

Max Liani: Well, Alex is awesome working with, so, he really knows what it's doing and his experience in optimizing, vectorizing, execution is really, really deep. So the one way that we often work is by feeding back to each other findings. Sometimes we start with something that is quite not there and not well-defined, and we try to give it a go, and we figure that that's not the best way, we find limitations. And that feeds back into a design. Let's redesign

this part. It doesn't really change anything—or much—at the front facing to the user. So most of what we do needs to work with a level of backward compatibility. We tend to re-engineer the machine so we can take better advantage of it.

Alex Wells: Well, I think that's a good description. The ability to come into the code base and make suggestions, to go do experiments and see what techniques are working, what aren't, actually have real production test data, which is key for making these design decisions.

You really don't want to optimize around the fastest case. You want to optimize around the worst case. 'Cause that's the one that artists are really gonna notice. They're not going to notice that something took two milliseconds instead of three milliseconds. They're gonna notice something that now is working at 15 frames a second that used to only be one frame a second. So as far as working on things, definitely being able to make suggestions and then have them eventually be incorporated once we proved out that they're effective. And a lot of this goes towards trying to leverage all the hardware that's on the system and, you know, in a heterogeneous system, there's CPUs, maybe a server farm, GPUs. But in the CPUs case, there's a lot of code out there that isn't taking full advantage of the hardware that's there, you know, all the SIMD units that are actually already sitting on somebody's computer and working to best take advantage of those, you know, is really exciting. And it can really start speeding things up for artists.

Max Liani: And often these things are not like changing the nature of the computation. Sometimes it is, but most of the time, the type of optimization is, hey, how do we access memory better? How would we have to restructure the memory model or the data representation so that it can be accessed better.

And most of the stuff that we learn is not really a two-way thing between different compute devices. It turns out that, in most of the cases, optimizations that apply well to a CPU will translate well on a GPU and vice versa. So when we started years ago on this journey of creating a new renderer that is entirely based on heterogeneous computing, we were concerned with how much, with a small team such as ours at Pixar, how much technology we could maintain and thoroughly optimize for different devices. The more we progressed, there was an inflection point where we thought, okay, there's going to be some percentage of execution that specialized kernels for different devices. Now we're kind of converging back into, well, actually it feels that a lot of the code can be actually shared. Which is great because the last thing that we can afford is for example, through thorough optimization of two different code paths, they start behaving differently or the one would produce different results, or perhaps one has a … that doesn't show up in the other, and then it becomes really time consuming. And we want that time to be better spent into creating a faster renderer. It's that.

Nicole Huesman: Max, what has the move to heterogeneous computing meant to RenderMan?

Episode 9: Pushing the Boundaries of What's Possible
Host: Nicole Huesman, Intel
Guests: Alex Wells, Intel; Max Liani, Pixar Animation Studios

---

Max Liani: So, RenderMan has always been, even more so in the past, sort of like a toolkit to create a production pipeline. And so it has been always very extensible through a vast plug-in architecture. And your rendering is not going to be any less, so it is really important that what we give in the hands of our customer doesn't increase the technical level to extend the renderer. So if our customer needs to write the plug-in for their own specific production needs, they shouldn't have to worry too much about, you know, you need to vectorize your code, or you might have to write intrinsics to take advantage of SIMD instructions, or you need a separate code path for your GPUs. We don't want any of that because that will make it impractical for any of our customers to really spend that time, spend that effort, to create their own solutions.

And so it was really, really important from Day One that, for the majority of the code base and especially on the plug-in side, that the code that our customer writes looks exactly like scalar code. So it's intuitive to write. It looks like it is a very sensible sequence of statements, and all the magic happens behind the scenes through modern compiler technology. So, so far, you know, we have been experimenting on the CPU side with the explicit vectorization that the Intel compiler provides as a trampoline toward Data Parallel C++, which provides this more Just-In-Time (JIT) compilation and even device abstractions, where you write a kernel and that kernel will run in an optimal way on a number of different devices or different CPUs with different instructions. So that is really in line with what we want to deliver. You write this a little bit of code, you have to figure out the math and what you need to do to create a piece of solution that you want, but that will run. You don't have to worry too much. And so, we see this as one-step continuation toward a line of simplification, even from a coding perspective. And so, the work that we did with the explicit vectorizer allows us to identify a large number of performance bottlenecks, especially for memory access point of view and all that portion that then will benefit everything else.

Nicole Huesman: So let's shift a bit and talk about what's next. Alex, as we do that, what are you most looking forward to?

Alex Wells: I'm really looking forward to Intel's upcoming discrete GPU offerings. And as Max said, transitioning our work with explicit vectorization to Data Parallel C++, and being able to prove out that we can take these kernels that we've written and transfer them over to oneAPI and look forward to, you know, experimenting with the single code base and reusing it for multiple platforms. And we also really hope to enhance oneAPI's capabilities. You know, if we run into an issue, Intel is very active in the development of the standard and we can feed those right back in to help overcome any obstacles we run into.

Max Liani: And I think there is a very good case study because some of our compute kernels are incredibly complex. It has been fundamental engaging early on so that the whole technology stack evolves in our direction. To some extent we've been pushing the boundaries, not just from a filmmaking perspective because of the filmmaking is a very, very niche industry. So what, film, you know, satisfy a very wide audience, a large percentage of the world's population, but the people and the technicians and the engineers that actually

work on movies are not that many, but the type of compute that ultimately is a physical simulation of light. It's very, very complex. And I have nothing to envy, from that point of view, to a variety of all the high-performance computing type of challenges.

Alex Wells: As a guy who has to get this stuff to work on the compilers, I'll run out of digits on how many times I will bring something to the compiling team, and 'Oh, well, we haven't run into that before.' And it's like, well, this is being used currently in HPC for years and nobody's tried to do this yet? I'm amazed at how many times it seems like these are the first kernels that are exercising certain language features or certain data access patterns.

Max Liani: And so, the fact that our type of current compute are considered, you know, incredibly complex, it's kind of like, we are doing a service to other industries, like getting there with this level of complexity first so that when the others will get there, it's going to be ready.

Nicole Huesman: I think that's such a great point that what you're doing in terms of leading the way here, won't just benefit the media and entertainment industry, or the animation industry, but the industry as a whole. Fantastic point. As we close the discussion today, where can our listeners go to learn more?

Alex Wells: Well, we talked a little bit about oneAPI and Data Parallel C++. You can just go to software.intel.com/oneAPI, and that'll be a great starter page, lots of information, even some places online where you could try out oneAPI.

Max Liani: We also have a session for SIGGRAPH coming up where we are going to tell about our most recent development.

Nicole Huesman: And that's exciting. I cannot wait to see that. So thank you, Max. Thank you so much. It's been so great to have you on today's program.

Max Liani: My pleasure. Thank you.

Nicole Huesman: And Alex, thanks so much for sharing your insights with us today.

Alex Wells: Thank you.

Nicole Huesman: And for all of you listening, thanks so much for joining us. Let's continue the conversation at oneapi.com. Until next time.