We hardly encounter anything that didn't really matter

Phil Langley in conversation with Possible Bodies

https://possiblebodies.constantvzw.org/book/index.php?title=Interview_with_Phil_Langley

As an architect and computational designer, Phil Langley develops critical approaches to technology and software for architectural practice and spatial design. Our first conversation started from a shared inquiry into MakeHuman, the Open Source software project for modeling 3-dimensional humanoid characters. In the margins of the yearly Libre Graphics meeting in Toronto, we spoke about the way that materiality gets encoded into software, about parametric versus generative approaches, and the symbiotic relationship between algorithms that run simulations and the structure of that algorithm itself. "I think there is a blindness in understanding that the nature of the algorithm effects the nature of the model … The model that you see on your screen is not the model that is actually analyzed."

Six years later, we ask him about his work for the London based architecture and engineering firm Bryden Woods where he is now responsible for a team that might handle computational design in quite a different way.

A very small ecosystem

Phil Langley: For the creative technologies team that I set up in my company, we hired twenty people doing computational design and they all come from very similar backgrounds: architectural engineering plus a postgraduate or a master's degree in 'computational design'. We all have similar skills and are from a narrow selection of academic institutions. It is a very small ecosystem.

[…]

When building the team, I was very conscious about not stepping straight into the use of generative design technologies, because we certainly haven't matured enough to start the conversation about how careful you have to be when using those techniques. We are working with quite complex situations and so we can't have a complex algorithm yet because we have too much to understand about the problem itself.

We started with a much more parametric and procedural design approach, that was much more... I wouldn't say basic... but lots of people in team got quite frustrated at the beginning because they said, we *can* use this technique, why don't we just use this? It's only this year that we started using any kind of generative design algorithms at all. It was forced on us actually, by some external pressures. Some clients demanded it because it becomes very

fashionable and they insisted that we did it. The challenges or the problems or the kind of slippage is how to try and build something that uses those techniques, but to do it consciously. And we are not always successful achieving that, by the way.

The biggest thing we were able to achieve is the transparency of the process because normally everything that you pile up to build one of those systems, gets lost. Because it is always about the performance of it, that is what everybody wants to show. They don't want to tell you how they built it up bit by bit. People just want to show a neural network doing something really cool, and they don't really want to tell you how they encoded all of the logic and how they selected the data. There are just thousands of decisions to make all the way through about what you include, what you don't include, how you privilege things and not privilege other things.

At some point, you carefully smooth all of the elements or you de-noise that process so much… You simplify the rules and you simplify the input context, you simplify everything to make it work, and then how can you say that it actually works? Just because it executes and doesn't crash, is that really the definition of functionality, what sort of truth does it tell you? What answers does it give you?

[...]

Sometimes it is not better than nothing

PB: When we speak to people that work with volumetric systems, whether on the level of large scale databases for plants, or for making biomedical systems … when we push back on their assumption that this is reality, they will say, “Of course the point-cloud is not a reality. Of course the algorithm cannot represent population or desire.” But then when the system needs to work, it is apparently easy to let go of what that means. The need to make it work, erases the possibility for critique.

PL: One of the common responses I see is something like, “Yeah, but it is better than nothing.” Or that is at least part of the story. They have a very Modernist idea that you run this linear trajectory towards complete know-how of knowledge or whatever and that these systems are incomplete rather than imperfect and that if you have a bit more time, you’ll get there. But where we are now, it’s still better than then. So why not use it?

Sometimes a point-cloud is not better than nothing because it gives you a whole other problem to deal with, another idea of reality to process. And by the time you get into something that’s usable, it has tricked you into thinking that it’s real. And that’s true about the algorithms as well. You’re wrestling with very complicated processes and by the time you think that you kind of control it, it just controlled you, it made you change your idea of the

problem. You simplify your own problem in order that you can have a process act on it. And if you're not conscious about how you're simplifying your problem in order to allow these things to act on it, if you're not transparent about that, if you don't acknowledge it, then you have a very difficult relationship with your work.

Supposed scientific reality

PL: We used a genetic algorithm on a couple of projects now and the client in one project was just not interested in what methods we were using. They did not want us to tell hem, they did not care. They wanted us to show what it does and then talk about that, which is kind of okay. It's anyway, not their job. The second client was absolutely not like that at all, they were looking for a full explanation of everything that we did. And our explanation did not satisfy them because it didn't fit with their dream of what a genetic process does.

We were fighting this perception that as soon as you use this technique, why doesn't it work out of the box? And then we're building this thing over a matter of weeks and it was super impressive how far we got, but he still told us, I don't understand why this isn't finished. It took the US military 50 years to make any of this. Give me a break!

[...]

It does not really matter that it is ultimately constrained

PL: I think with these generative algorithmic processes, people don't accept constraint either discursively or even scientifically. At most they would talk about the moment of constraint being beyond the horizon of usefulness. At some point, it doesn't create every possible combination. Lots of people think that it can create every option that you could ever think of. Other people would say that it is not infinite, but it goes beyond the boundary of what you would call, 'the useful extent of your solution space', which is the kind of terminology they use. I think that there's a myth that exists, that through a generative process, you can have whatever you want. And I have been in meetings where we showed clients something that we've done and they say, "Oh, so you just generated all possible options." But that's not quite what we did last week!

There's still that sort of myth-making around genetic algorithms, there's an illusion there. And I think there's a refusal to acknowledge that the boundary of that solution space is set not really by the process of generation. It's set at the beginning, by the way in which you define the stuff that you act on, through your algorithmic process. I think that's true of parametrics as well, it's just that it's more obviously to improve metrics. Like, here's a thing that affects this thing. And whether you complexify the

relationships between those parameters, it doesn't really matter, it's still kind of conceptually very easy to understand. No matter how complex you make the relations between those parameters, you can still get your head around it. Whereas the generative process is a black box to a certain extent, no one really knows, and the constraint is always going to be on the horizon of useful possibilities. So it doesn't really matter that it is ultimately constrained.

We're not behaving like trained software developers

PL: By now we have about twenty people on our team and they're almost all architects.

When I do a presentation in a professional context, I have a slide that says, "We're not software developers, but we do make software." And then I try to talk about how the fact that we're not trained as software developers, means that we think about things in different ways. We don't behave like them. We don't have these normative behaviors from software engineering either in terms of what we create or in the way in which we create things. And as we grow, we make more things that you could describe as software, rather than toolkits or workflows.

After one of these events, someone came up to me and said, “Thank you, that was a very interesting talk. And then she asked, “So who does your software development? To who do you outsource the development?” It is completely alien to this person that our industry could be responsible for the creation of software itself. We are merely the recipients of product satisfaction.

Architects are not learning enough about computation technology either practically or critically, because we’ve been kind of infantilized to be the recipient discipline.

[...]

There’s a myth-making around this, that makes you feel like you’re still engaged in the kind of practice of creating the technique. But you’re not, you’re just consuming it. It’s ready-made there for you. Because it sits on GitHub, you feel like a real coder, right? I think the recipient context becomes infantilized because you’re not encouraged to actually create it yourself.

You’re presented with something that will work, so why not use it? But this means you also consume all of their thinking all of their ways of looking at the world.