arq (2017), **21.1**, 53–64. © Cambridge University Press 2017

Computer-Aided Design pioneer Robin Forrest and Daniel Cardoso Llach discuss the early years of CAD research at MIT and Cambridge, and its architectural repercussions.

Of algorithms, buildings and fighter jets: a conversation with Robin Forrest

Daniel Cardoso Llach and Robin Forrest

A founding member of the Computer-Aided Design Group at the University of Cambridge, UK, and a student and collaborator of CAD pioneer Steven A. Coons at MIT, Robin Forrest occupies an important place in the history of computational design. Along with important contributions to the mathematics of shape representation, his coining of the term 'computational geometry' in 1971 offered a handle on design techniques that started to emerge – somewhat uncomfortably at first – in the interstices of engineering, mathematics, and the fledgling field of computer science. Initially fostered by government-sponsored research into Computer-Aided Design for aircraft and car manufacturing, the methods he helped develop have since been encoded in countless commercial software systems for 3D modelling and simulation, helping structure the intellectual work – and the professional identity – of architects, engineers, and other practitioners of design.

In this interview Forrest reflects on a career that spans different fields and institutions, and offers clarifying insights on the international and interdisciplinary network of researchers that during the 1960s and 1970s helped to define the seminal ideas and technologies for computational design – particularly those developing at the University of Cambridge and at MIT. He talks at length about the application of Coons's and Pierre Bézier's techniques for surface representation to aircraft and car design, and about the transition these ushered from manual to computational design methods across entire industries – crisply presaging future architectural developments towards parametrically controlled geometry. Finally, he shares candidly details about his collaborations with artists and architects, including Cybernetic Serendipity curator Jasia Reihardt and centre for Land Use and Built Form Studies director Lionel March, offering an unusual perspective on the early (and ongoing) fascination of architects and artists with both computation and computer scientists.

The conversation offers profound insight into the technical and conceptual rearrangement of design practices around the processing, display, and manufacturing capacities of digital computers, as well as into the infrastructural changes that undergird the last three decades of architectural production. It hints further at new interpretive directions that open up when accounting for the agency of technological systems, their designers, and the institutions that host them in the collaborative production of our artificial environments.

Daniel Cardoso Llach (DC): Can you tell me about the context that led you to Cambridge as a student, and then to MIT?

Robin Forrest (RF): One summer in Canada I worked on control systems for nuclear reactors and I got interested in control theory. So I decided I wanted to do a PhD in control theory. There were only two places that did it in the UK. One was Imperial College and the other was Cambridge. So I applied to the Engineering Department at the University of Cambridge after obtaining my mechanical engineering degree in the University of Edinburgh. I intended to study control systems but I didn't really like the topics that I was offered. I wanted something more mathematical, I thought. Basically I suppose I was a mathematician rather than an engineer. I started in October of 1965, and in November my then supervisor Donald Welbourn said, 'We are thinking of setting up a CAD group.' Both Welbourn and [Mathematical Laboratory director] Maurice Wilkes had gone to MIT and seen Sketchpad and the Timesharing system,[1] so they were going to set up a group at Cambridge.[2] They employed Charles Lang in the Mathematical Laboratory because he had worked with Doug Ross and Steve Coons at MIT.[3] His job was to link computers together: the graphics display to the mainframe. The Engineering Department hired Crispin Grey, and I had a research studentship from Trinity College, which didn't cost academic departments anything, so I sort of came 'free'. I had to find the topic, and the most pressing problem that was seen at that stage was the representation of curved surfaces. That's where a lot of the money for Computer-Aided Design (CAD) in the States came in,

doi: 10.1017/S1359135517000173

and that's why Steve Coons got involved, and that's why he was so important.

DC: Before becoming a faculty member at MIT in 1948 Steve Coons spent several years working in industry designing aircraft shapes. Starting on 1959 he and Douglas Ross co-directed the Computer-Aided Design Project at MIT, a joint research project between Mechanical Engineering and the Servomechanisms Laboratory, which laid the technical and theoretical foundations of Computer-Aided Design.[4] In what ways do you think Coons's experience in the aircraft industry shaped his research on CAD?

RF: Before computational descriptions were available, aircraft used to be *lofted* and engineers would draw full-scale plane sections. For example, the definition of the *Spitfire* [airplane] was a set of lines scribed on sheets of aluminum on plane tables somewhere in Hampshire. Everything was taken from that. You would take drawings off of the scribed sheets. These were full-scale drawings, and the problem was that if a bomb dropped there, of course, you would essentially lose the definition of the *Spitfire*. But of course you would send drawings – line drawings – down to the manufacturing department, but there were always problems, as the aircraft was only defined at the sections and interpolation between sections was not precisely defined. These were the problems Coons probably dealt with in industry.

At some stage North American Aviation designed a system called AUTOLOFT, which instead of drawing conic sections, which is what was normally used, actually defined the conic sections by their coefficients. So they had a numerical definition of all the curves, which of course you could put on a sheet of paper. So it's more precise and you can transcribe it exactly. They started having mathematically described shapes, but just the sections. But of course, with the development of numerically controlled machinery, you could machine those and, as you wrote, the Servomechanisms Laboratory at MIT had an important role to play in that area.[5] [With CNC] you got the precision. But the aircraft got more complicated and it became more important to actually see how you go from one section to another. I mean, you can't just describe a fully three-dimensional surface with a set of line drawings. So, that was where Steve got involved in trying to get the definition of curved surfaces. His surface methods basically interpolated surfaces through sets of two curves, and the curves could be anything you like. There are particular cases, such as the bi-cubic surfaces.

So, the Servomechanisms Laboratory team came in because you could have machines that could actually cut three-dimensional surfaces accurately. So there was a need, and the equipment – the manufacturing – was there. It was a question of what were the equations you were going to *cut*. And that's where Steve got going, and that's where I got going *[1]*.

DC: Yes, as you explain in your 1972 article 'On Coons and Other Methods for the Representation of Curved Surfaces', Steve's mathematical techniques are significant because they are the first parametric methods for interpolating surfaces from a set of bounding curves.[6] Can you tell me more about how these 'Coons patches' were used and why they were influential in practice?

RF: The point about Steve's work was that designers liked to design curves and sketch them in paper and that kind of thing. They liked a surface fitted to these curves but they didn't want to be constrained to the

1

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

kind of curve that they designed. So you had to find a surfacing method that would interpolate between any kind of curves provided the curves actually meshed – that they fit, that they intersected at the corners. And that's what Coons surfaces really were. Steve used to say that you could have any curve you like, that you could even have handwriting as a curve. And it would create a surface through the handwriting. You know, so you could go from a straight line to handwriting, to a circular arc, and you could put a meshed surface in between them all. But in practice very few people actually ever used Coons surfaces with arbitrary curves. They used what was called the tensor product version of them, where you have curves roughly of the same kind.

General Motors [GM] used them a little bit. But they had problems. But well, GM was ruled by the stylists. I don't know if they still do, but the stylists used to produce a full-sized clay model of the car and paint it up. There was a big styling room in the Tech Centre, and they would light the model from all directions, and you would walk around it and admire it, and they would say, 'That's it!' Then engineers would have to go and digitise this at incredible tolerances because, as far as the stylists are concerned, it was the perfect surface. To digitise the model they used to capture a point every few inches, and when the surface got a bit tricky, they captured around ten points more along and across the clay – a hundredfold increase locally. So you got a profusion of data from the curves across the car, and this required very sophisticated curve fitting and curve smoothing-fairing programs. It got really quite elaborate. [Pierre] Bézier used to say that it took GM a megabyte to do a trunk lid, but he got a full car in 64 K[ilo]bytes!

When these clay models of cars were digitised you would find that they weren't symmetrical. But the stylists wouldn't believe that. There are all sorts of horror stories of cars with one side five inches longer than the other. For example, when they were making headlights they couldn't take the left one and do the other in its image, they had to design it again because the other side was different. Stylists were bent on using manual methods and couldn't be persuaded to let the engineers clean them up. They ruled the company, really. They changed the models every year, and style was all – they were the gods. You know, the engineers had to fit the mechanics inside what was given to them by the stylists. It was not a question of wrapping the car body around the works. It was the other way around. I think to a certain extent this is still the case.

The idea of Coons's methods was then to try and cover all the shapes that designers would want, and also to hide the mathematics so that they wouldn't need to be mathematicians to use the system *[2]*.

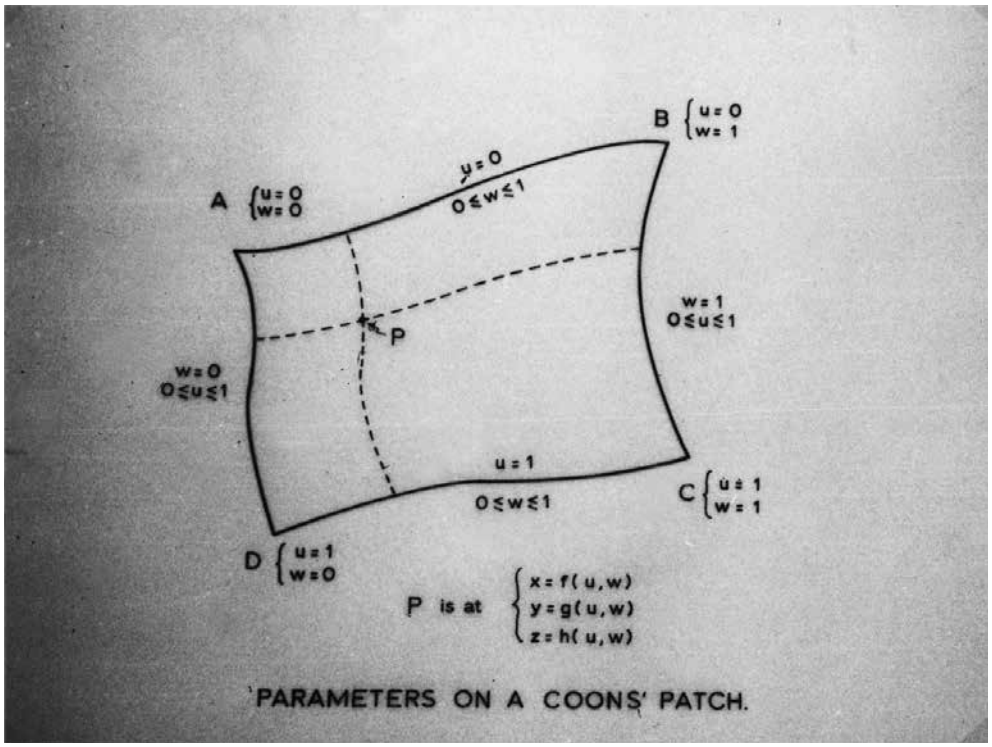
DC: So it was not only about being able to define and encode geometry but also about allowing designers to manipulate shapes easily.

RF: Yes, and also to do all the things that you had to do with the model: to be able to stress it, analyse it, to be able to calculate voids and weights —and eventually to calculate the aerodynamics, I suppose. It very much was building the geometric model from which all sorts of developments could be derived – including line drawings if you wanted them – but also instructions for machine tools and different sorts of analysis.

DC: Gauss defined mathematical methods for parametric surface representation in the 1830s, and Schoenberg published work on splines in the 1940s. Do you see a technical lineage between these

1 Prior to computers, mathematical definition of shapes made the design of artefacts such as the RAF's Spitfire warplane safer and more mobile.

2 Steve Coons pioneered parametric methods for interpolating surfaces between sets of bounding curves, circa 1967.



2

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

mathematicians and Coons? Did they have an influence on his work?

RF: I think they probably did not. I can say two things to that. One is that, at one stage, halfway through my thesis, Charles Lang said that I should go and talk to some mathematicians to get some help, so I went to talk to a couple different people. One was an old numerical analyst from the days of the numerical tables (which you probably don't remember at all). He referred me to a mechanical desk calculator manual from 1936, which was not really relevant. Those were the days when computers were people. The other one was Peter Swinnerton-Dyer, to whom I explained what I was doing. He said, 'Well some of this work on cubics was very fashionable at the turn of the century, but it's no longer fashionable now, and what you are trying to do is rather different from the interests of mathematicians. So I can see you've got problems!' That was the extent of my mathematical help at Cambridge. The mathematicians were, say in the cubic case, interested in what happens at infinity, and in classifying curves according to this. This wasn't really relevant to engineers – who wants to know about what happens at infinity?

The other thing is that Schoenberg had basically developed his techniques for B-Splines for smoothing actuary tables: data for insurance companies. And the mathematics he used enabled him to analyse a lot of their properties, but it was very unstable numerically. The algorithms were not something you would really want to use. So they were not really known except in theoretical numerical analysis until the correlation between Bézier and Bernstein came out and Bill Gordon at GM Research Labs recognised that Bézier is a special case of B-Splines, I would say. And then we had two independent developments of stable numerical algorithms for computing B-Splines, and that's when all became possible. But all that was in the early '70s, long after Steve was really innovating.[7]

DC: It sounds as if there was a sort of disconnect between mathematics and engineering, and that Steve Coons's methods for parametric surface representation were more closely linked to practical questions emerging from design and manufacturing than to established mathematical theories.

RF: Yes, he was an intuitive mathematician. He really … He just thought that way. I don't think he had deep mathematical knowledge. [Pierre] Bézier did, but not Coons. I recall that Steve and I thought that if you had a closed polygon you probably could have a closed B-Spline curve, and that if you increased the degree of the B-Spline, you could start using some of the points more than once – so the curve would shrink. We recognised that by increasing the degree to infinity the curve would shrink to a *point*. And Rich Riesenfeld and Elaine [Cohen] asked: 'Would it be a square point or a circular point?' And Steve and I both said: 'It would be a circular point!' And Steve said: 'What else could it be?!' That wasn't enough proof for Elaine. 'You can't have that!' But I'm sure Steve was right!

DC: Tell me more about your time at MIT and your collaboration with Coons and Douglas Ross and the CAD Project during your time at Project MAC.

RF: Project MAC was in this building separate from the MIT campus in Tech Square. It had the Artificial Intelligence lab, with Marvin Minsky at the top floor, I think, and then there was Doug's group on floor five. And the people there were basically doing various software engineering projects. I went to MIT to work with Coons but he had forgotten to tell me he was going to be at Harvard working with Ivan Sutherland. So, there was really nobody there who was interacting directly with what I was doing. However, I could always talk to Doug there and bounce ideas off him and others. Steve at Harvard was interesting.[8] I would go up to Harvard once a week and I, Danny Cohen, and Ted Lee would talk with Coons and Ivan Sutherland. Ivan had quite a short attention span and he would get bored and go out. Steve would also get bored and disappear, and then the rest of us would talk all day about surfaces and graphics. Ted Lee did a thesis with Steve and Ivan as advisors on rational bi-cubic surfaces. I don't think Dan was just working with Steve. They were developing the head mounted display at that stage. I think there was more interaction between Steve and Nicholas Negroponte than there was between Steve and his students in mechanical engineering. I don't know at what stage did Steve get involved with Nick. He took me to meet Nick in 1969. He and Steve Gregory cooked a meal for us in Salt Lake City on one occasion. Good chap.[9]

DC: Both the CAD Project at MIT and the Cambridge CAD Group were interdisciplinary and interdepartmental efforts combining computers, engineering and mathematics. How did these collaborations between different disciplines work in practice?

RF: At Cambridge this was a strength, but also a source for all sorts of problems later on. It was a rich man's game to begin with – I think that was one of the reasons. We were lucky in Cambridge, because there were not many university departments that did Computer-Aided Design in the 1960s. Most simply they couldn't get the grants, and couldn't get the money for equipment, whereas if you ran an Air Force contract [like the MIT CAD Project] you could get the funding. At the Cambridge CAD Group we were not funded by defence. Our funding came from the Science Research Council. And it was rather tricky because we were a joint group between the Engineering Department and the Maths Lab, and there was always a question of who would fund us. There was always an argument, which got quite difficult sometimes, whether we should be funded for the kind of work that we did. They didn't think it was mechanical engineering, or they didn't really see where things were going.

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

Things got difficult because none of us had tenured posts, and we were all on contract. At that stage the Maths Lab had only one tenured post, and didn't have any lecturers at all. The Engineering Department eventually appointed people as lecturers, but they came in from outside, and they were assigned to our group, but I think they did not really understand what we were doing.

For example, at one point Welbourn was frustrated at the rate of transfer of the research to practical systems – he got fed up waiting for us to do all the software and the tools – and he set up a side group that produced a program that specialised on ducts, because there was a market for people who would do ducts design. But you couldn't do anything else but ducts. So it was very narrow. We regarded this kind of system as the province of the CAD Centre, but I think Welbourn ignored this.[10]

A source of these problems was that 'graphics' was not part of computer science in those days. It was a part of engineering. At the same time, I don't think that engineers understood what needed to be done as far as computer software was concerned. You know? They didn't understand why we needed data structures, and why we needed to develop this and that. I don't think they really understood, to a certain extent as Steve Coons did, that you could build a model from which everything else flowed. It was not only to produce drawings. It was something more.

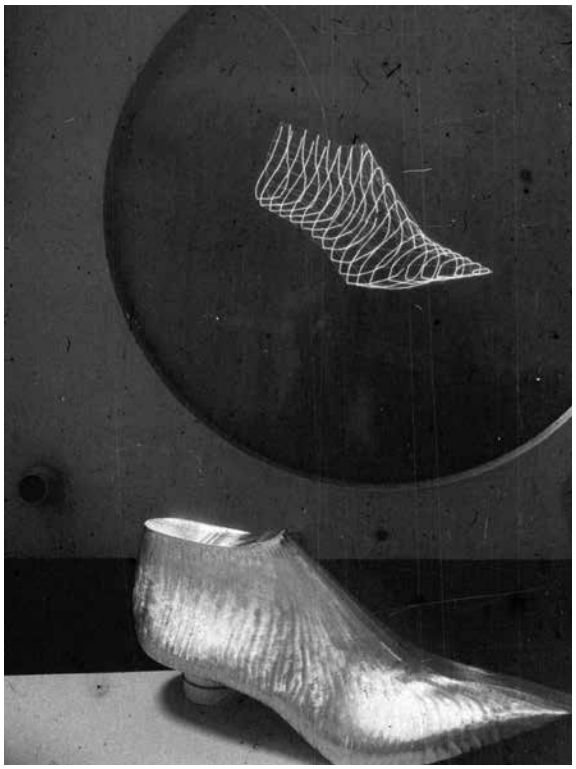
DC: It sounds like you had to convince both sides. On the one hand the engineers about the importance of computing, and on the other the mathematicians and computer scientists of the importance of graphics.

RF: Yes. After I finished my PhD in 1968 I was employed by the Engineering Department and at one point I got tackled by a faculty member who said that I could do nothing for him, that my work would be of no use to him. He actually said that!

We did a couple of PhD theses in our group in the Engineering Department. One was finite element stress analysis, that is the automatic generation of meshes for stress analysis. The other one was 3D and numerically controlled cutting. The first thing we cut was a shoe last, which was not exactly the easiest thing to start with. It caused lots of grief [3]!

DC: I would like to return to your previous mention of data structures. You mentioned that the Cambridge CAD Group built 'tools to build systems', and that part of this was to design new data structures. At the time you had LISP and FORTRAN, and therefore you had lists, arrays, and linked lists, but you were probably referring to higher-level data structures.

RF: Higher and much more complicated. We needed a framework from which you could hang all those things. You needed something which was not two-dimensional, so a [data] tree would have three, four, or five-dimensional links. This is what Doug Ross called a 'Plex'. But yes. It wasn't straightforward. You couldn't do things with lists and big searches. You had to reflect the three-dimensional nature of things.



3 Early uses of computational geometry at the Cambridge CAD Group included the geometric modelling and numerically-controlled manufacturing of a metal shoe last, October 1968.

Looking back, the '60s were very much the time when computer science came of age and many of the basics, which we now tend to take for granted, were being developed. Engineers did not understand the difference between being able to generate a drawing that looked like a recognisable artefact, and being able to construct a model that accurately represented the artefact and from which drawings could be produced but which fed into other aspects of design such as stress analysis and manufacture. Early commercial systems such as Computervision, in which Coons and Negroponte had a stake, were in practice systems for generating conventional mechanical and architectural drawings.

Coons was not a computer user but rather a visionary, and he could see how things had to develop. I don't think he understood the amount of computer science which was necessary and which had to be invented. Doug Ross did, but he probably became a bit detached from real applications. The same scenario played out at Cambridge. I started out as a Mechanical Engineer in the Engineering Department and ended up as Computer Scientist in the Maths. Lab. The point about interaction is that the interaction has to be with something and that something has to be a computer model.

DC: These ideas about interaction seem to resonate with cybernetics. Can you say something about the influences on the CAD Group during this period?

RF: [Benjamin Lee] Whorf's and [Joseph Carl Robert] Licklider's ideas were influential. I came across

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

Whorf's work through Licklider and others.[11] I think the point people took from Whorf was the idea that our own language to a certain extent dictates the ideas we can understand and express, and the introduction of interactive computing had the potential of creating a new form of language. Accordingly, Steve Coons used to emphasise how CAD would involve a designer 'talking to a computer' by which he meant using a teletype terminal or a light pen. Building practical computer models for real applications demanded new software tools.

In terms of geometric representation the one that really got going was the one that started off at Stanford on the AI lab – the winged edge polyhedron – which defined a data structure for polyhedra in which the basic element was the representation of an *edge*, which had pointers to edges in adjacent faces.[12] In this data structure you got links and pointers that go around the faces of the polyhedron, and there are ways of traversing all the faces of the polyhedron, so it was all three-dimensionally linked. And that was the basis from which Ian Braid started off for volume and solid modelling. He took it a little bit further than the original Stanford stuff.

DC: So this is the origin of Boundary Representations, or B-Reps, which were fundamental to many solid modelling software systems. It is tempting to see the Cambridge CAD Group as a point at which two distinct sensibilities about computation in design converged: on the one hand a concern with geometry and curved surfaces, shaped by work in aircraft and car design, in particular Coons's and Bézier's, and on the other hand, a concern with data structures and systems for describing and accounting for mechanical parts. The combination of geometric representations and attributes...

RF: Yes, well, I did the surfaces because we saw that was a problem at the time. But then Charles [Lang] would ask 'What about what we could call simple mechanical parts?' That's why we got into the volume modelling business, the solid modelling business. And there are all sorts of issues there, which we discovered, which we hadn't realised the problems before, such as combinatorial problems. The complexity of things gets actually quite difficult to handle. You got lots of numbers and lots of bits. Lots of pieces, edges, fillets, and holes. You are not dealing with a fuselage and a few bits, you have hundreds or thousands of pieces instead. And there were issues on how accurate the representation was. There were two conflicting systems. There was the B-Reps [system] and there was constructive solid geometry. And, there were turf wars between the two.

Basically constructive solid geometry says that you could assemble everything from half spaces, so you could create objects by union and intersections of half spaces. And you get this sort of tree of operations of intersection and union of these half spaces. The thing is that you can prove all sorts of things mathematically about this. But, you run into the real problems of real computational geometry, which are numerical problems. For example, I did lots of work later on just to talk people into working out how to intersect two line segments, which is a deceptively simple problem.[13] The mathematicians would say, 'Intersections of lines? That's easy: we just set up these two equations and invert the matrix.' That's the problem, sometimes you have to recognise that a matrix might not suit you, because lines are parallel or coincide. You see? If you read any book on numerical analysis it says that you can do all these wonderful operations on matrices, but accuracy depends on how orthogonal they are. What does that mean? Well that actually means that in geometric terms, the lines are at right angles. And then your errors are very small. But if they're not, if they're nearly parallel, where do these lines intersect? You have this big sloppy business there. You get all sorts of numerical problems.

DC: There is again a seeming contrast between mathematicians' and engineers' approach to geometric problems. The contrast stems from engineers' approach being closer to design and to material and manufacturing constraints, while mathematicians' approach leans more towards abstraction.

RF: Well, that's why I started talking about Computational Geometry. Basically the kinds of geometry we were dealing with were not particularly sophisticated as far as mathematicians were concerned – they couldn't see what the problem was. But we had these problems [in design and manufacturing], and in actually implementing them on a computer the problem was making them accessible to designers so that they could modify them and do the things they wanted. And that was something that mathematicians probably didn't quite understand. And then we also had the problems of just complexity – the number of pieces. You know. Things got more difficult once you moved into higher dimensions. So, going back to the numerical problems, when you designed using Ian Braid's system, you would insert a new face, split a face, put in a new edge, and this and that, and you actually had to evaluate the geometry to find out whether you intersected anything or created a new object, or chopped a bit off. This all involved numerical work. You might end with a data structure, which was in some sense invalid due to numerical error. There is an extension of Euler's equation for polyhedra, which Braid used, which guarantees that the data structure in a sense actually represents a legal object provided the numbers are accurate.

And the construction solid geometry people would say, 'We can prove ours is correct.' The only trouble is that you had these operations: you could slice things, cut them, and the rest of it. I would ask them the questions, 'Well, I've done this. But how many bits does it have? Is it one solid piece, or is it more than one bit?' B-Rep people could tell you that. CSG people couldn't.

DC: How would you characterise the differences between the work at Cambridge and what was going on at MIT? To what extent were there overlaps?

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

RF: The work of the Cambridge CAD Group was slightly divergent from MIT's. At a given point there wasn't, I think, a lot of interest from the MIT mechanical engineering department in what Coons was doing. They suddenly didn't teach any more courses on it. There was Ivan Sutherland, who was a PhD student, and there were a few students —Tim Johnson, Robert Parmalee, and others— who did Master's, but there was really nothing much after that. Doug [Ross] was really obsessed with software engineering.

When we started in Cambridge we were very lucky because Maurice Wilkes had £50,000 that he could spend any way he liked, and he went and bought a DEC PDP computer and display. Nobody else had that kind of money and the freedom to do that. I think we were one of the first academic CAD groups, in an academic department, rather than existing under military contracts. So, when I was doing my thesis, the people I talked to were in the industry. There was nobody doing anything remotely like it in a university —which made my life slightly difficult. There were people in the aircraft and car industries, and to some extent in the ship industry, so I started by writing to many of them, asking about what they were doing. Also, for some reason, we got a lot of the MIT reports sent to us by the air attaché in Washington, and from Doug Ross's group. So we knew what was going on. We also knew what was going on in GM, we knew what was going on in Ford. We were very privileged. We had to be careful about what we said to whom, but we did indeed have a good knowledge of what was going on.

When we started off at Cambridge we were one office at the Maths Lab, and in the next-door office there were three people who were writing the time-sharing operating system. And whenever Charles [Lang] needed some special instructions to link the PDP satellite to the big machine he would shout through the door 'Can you do this?' and they would shout back 'Yes, we'll do that!' We were a very small team.

I think it's difficult to understand nowadays, but when we started it really was the primitive days of computing and a lot of things were not around. The big mainframe computer at Cambridge, which was built by the lab, had an operating system that, when I got there, had been written by *one man*. It was a batch operating system that he wrote entirely in machine code. Allegedly, if anything went wrong you would phone him up and he would have the whole operating system in his head! You can imagine a patch...

DC: Sounds like he *was* the operating system.

RF: Yes, there were some amazing people around. This was Peter Swinnerton-Dyer, an eminent mathematician. He thought computer science was

4 The Cambridge CAD Group's model-making machine was an early 5-axis CNC machine inspired by Bezier's UNISURF system, February 1971.

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

easy. Another example of our work from this period is the FORTRAN compiler we eventually used. It was written as a PhD project because there just wasn't a FORTRAN compiler around. These were the days of people creating compilers and data structures: the infancy of computer science. You could knock off a system, which would do one particular thing. We thought that we were in the business of *building tools to build systems to do things*.

Another system we developed is the graphics package GINO. It was further developed by the CAD Centre [14] and became GINO for FORTRAN, or GINO-F, and Martin Newell went to Salt Lake City to work with Sutherland and with other people, worked with Jim Clark, and they built a system there that became GL. So Open-GL is a sort of grandson of GINO.

DC: To what extent was GINO also the result of connecting the Titan machine to the PDP display, and of the need to develop libraries or routines for graphical display?

RF: It was a device-independent graphics system. You had code-generators for different devices so that it would drive the PDP-7 display. But we also had the other systems such as the Tektronics storage tube display and a plotter, and two model-making machines – machines to cut foam – that we built. When we got the first machine working we realised that we didn't have anything to cut. So Peter Woodsford wrote a code-generator for the foam-cutting machine one evening, and the next morning we started cutting surfaces.

DC: Tell me more about these model-making machines. What motivated you and the group to build them?

RF: There was a series of conferences called ProLaMaT, Programming Languages for Machine Tools. They had one in Rome in 1969, and Charles Lang and I went there. Doug Ross went there, and Pierre Bézier went there. There were some hardcore numerical control people there but Charles [Lang] and I, and Bézier, and Doug Ross, we also knew what we were doing and we got to know each other very well. After the conference we went to visit Bézier and saw what he was doing. He had a system that could cut full-sized car panels, and we were very impressed by that. So by 1969/1970 we decided to build our own. Our machine was much smaller. I suppose it was an early 3D printer in a way. But it was a milling machine; it had a wood router on it. And then we built a second one, a five-axis machine, for which I designed the mechanical side. Like Bézier's system, ours had three linear motions, and then we could rotate the head's angle manually to get the best cutting angle.

Bézier was vice-president of production engineering at Renault, and was in a position to say to his staff: 'You can stop using clay models and now you will use my method.' But he told us he didn't do that. He had built his machine, and the workshop had a full-sized drafting system with which to draw a car full size, and could digitise shapes.[15] You could draw curves manually and then fit polygons round the curves using a ruler and digitise the polygon vertices. The machine then drew the corresponding Bézier curve enabling the fit to be examined and improved on by tweaking the vertices. I went to Renault once with people who were in the business of making car bodies in England. Bézier handed me a pencil and a straight edge and said: 'You produce a curve your way.' The curve I drew was about four feet long, and I got it within a quarter of an inch margin of error. And Bézier just got it down to about one sixtieth of an inch. That blew the minds of the people with me.

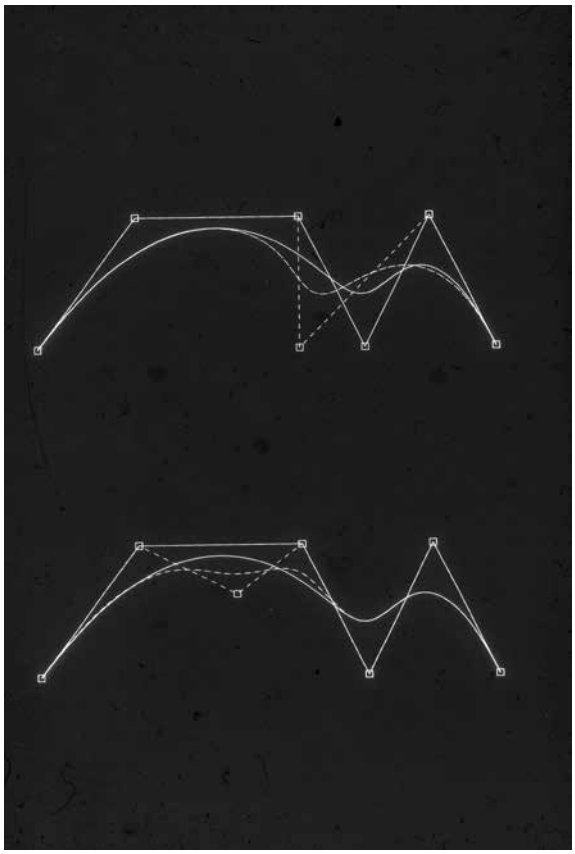
Bézier built this machine, with which you could build car panels, but he just left it. He got one or two star designers who were interested but wondered what the thing was. He told them how to use it. Some of them were convinced and played around with it. He said, 'One morning I came in, and this beautiful wooden sculpture was outside my door, and *that's* when I realised I had won the battle. Because this can do art.' And this is the way to introduce these technologies. You don't force it. You let people find it for themselves and do amazing things, and then they do things and you are left thinking, 'How the heck did they do that with my system? I don't know!' That's the nice thing about this kind of work.

But on the other hand, you saw these guys using these systems and ... I used to tell students that we shot ourselves in the foot because we made these things relatively easy to use. So nobody really saw the work that had gone into it. We should have made it very difficult, and then we could have charged vast amounts of money to do it properly for them *[4]*!

DC: Bézier is best known for developing an intuitive technique to design curves by moving external control points. You decoded Bézier's mathematics and revealed that it was based on Bernstein polynomials. Can you talk about what was important about that discovery and more generally about Bézier's methods?

RF: It explained why Bézier curves worked. People want to define curves by points and by passing curves through points. That's what I wanted. I wanted a curve to go *here*. The problem with interpolation is that the more points you have and the more you try to interpolate, the more opportunities the curve has to wiggle. If you are connecting the points with straight lines and you count the number of times you have to change from turning to the right to turning to the left, or vice versa, the inflections, the best you could get out of interpolation was the same number of inflections —but you generally got more, so it became wigglier. The point of Bézier is that you are designing basically with the points off the curve, but you are guaranteed to get at most as many wiggles as you had in your data, and then generally you could get fewer. So you could put noisy data and get smooth curves out. The other problem with interpolating points is that if you want to move the curve between two points and still go through all the others, it would wiggle between the other points,

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press



5

and it would change. With Bézier's methods if you pulled the curve then it would only move in one direction. So it's intrinsically a way of defining smooth curves and fair surfaces. He was a very smart man, Bézier *[5]*.

DC: It's interesting to consider the car design process you describe at GM and Renault, which was so much based on the digitisation of physical models, in relation with what architects like Frank Gehry do today. Work at Gehry's office often starts from physical models built with paper, cardboard, clay, which are scanned and then rationalised, optimized, and documented using software – notably a version of CATIA. So, while the designs themselves are not generated in the computer, they are recorded and developed digitally. Your description of the work by Bézier at Renault and the work of engineers and designers at GM suggests that architects have not only appropriated software tools from these industries, but also an approach to method – and perhaps an aesthetic sensibility – in order to develop new architectural languages.

RF: Right. So Gehry designs a little bit like the old way GM designed cars. Of course, there's a question of cost too, and of construction techniques. If you were an accountant you wouldn't do anything but straight lines and planes. Anything else is expensive – unless you do hyperbolic parabolas, where you can use straight materials. There are limits to what we can do.

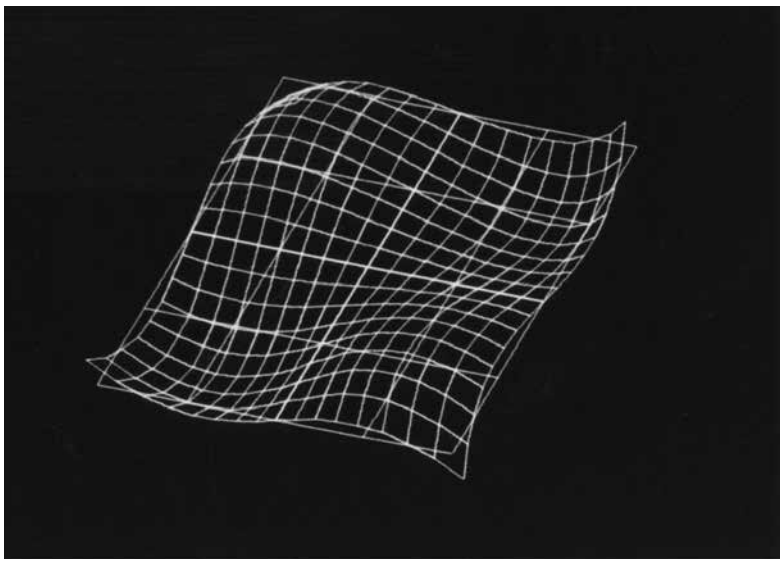
CATIA, on the other hand, is interesting. You know, when I was doing research everyone used their own systems, until that got too expensive. Eventually CATIA won out and then suddenly everyone started using CATIA. Until then, Boeing had their own system, McDonnell Douglas had their own system, Lockheed had their own system.

General Motors doesn't get enough credit for doing pioneering work. They had the DAC-1 computer and they did a lot of really interesting work, but they published about six months after Ivan Sutherland, so they didn't get all the credit for all of the techniques that Ivan describes. I think they were influenced by what was happening at MIT, but also developed their own stuff. These are very, very good people and of course they had the money in those days. They were very much in cahoots with IBM, and the displays they developed with them became the 2250, which was of course a very expensive machine – I mean, it cost a quarter of a million dollars, just the display.

Also, well, Bézier, was a very shrewd man. I think the other thing is that people got rather obsessed with graphics to begin with, and particularly shaded graphics, when raster graphics came out. You could look at things on the screen and that was it, and that was not right. We knew that there were things that

5 Sequence of images showing the effects of transforming a B-Spline curve's control polygon, July 1974.

6 Multipatch joining, an early technique to produce smooth curved surfaces by aggregating parametrically defined surface sections developed by the Cambridge CAD Group, circa 1968.



6

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

you couldn't really see on a big screen. You had to know what to look for. There was an obsession at one stage with something called 'twist vectors', and 'bi-cubic surfaces'. And nobody knew what to do with them, but you had to have them. And we made some models at Cambridge, of a bi-cubic patch, which was basically a bump and a 4x4 array of surface patches. And one we had properly computed twist vectors, and the other we set them all to zero, which is what we tended to do. If you looked at these two side by side, they looked identical. But if you ran your finger over them, you could feel that there was a flat area that didn't show up in the graphics. Well, you could probably see it if you read it in the screen in the right fancy way and if you got the orientation right. But, your finger is quite sensitive to things having bumps in them.

DC: Your 1971 paper 'Computational Geometry' is not only foundational to the field but also offers a remarkably clear case for the value of computational descriptions to design and manufacturing.[16] Architects in particular have had a fascination with geometry and, more recently, with numerical control. You knew and collaborated with members of the centre for Land Use and Built Form Studies here at Cambridge, and contributed a chapter to one of Lionel March's volumes. Can you tell me a bit about those collaborations and, more generally, about seeing Computational Geometry used across design fields?

RF: There was a lot of interest from architects in the work of the CAD Group. For example, one of the earliest users of the display system in the Mathematical Laboratory was William Newman, who had studied architecture at Cambridge and wrote a pioneering design system for building layout called NIBS – for Newman's Industrial Buildings System.[17] Crispin Grey left the CAD Group in the late 1960s to work for Land Use and Built Form Studies. Through him I met several of the members of LUBFS socially. The Maths Lab, being the site of the University's computing service, was a point of contact. Collaboration was informal but easy in the Cambridge environment. I had written a technical report on transformations and matrices, based on earlier notes by Bert Herzog of the University of Michigan, to explain some of the features of the GINO graphics package. Lionel March asked me for a version for one of his volumes and, being interested in modern architecture, I thought the Seagram Building would provide a recognisable model to illustrate the chapter. I cannot claim it is accurate, but it looks approximately correct! This loose collaboration had an unexpected result for me: a few months later I discovered that I had been listed as a member of the Architecture Faculty because I was the type of person they liked to associate with. There were later collaborations with architects after I left Cambridge. For example, a model of a hospital created using a modular design system was cut on one of the model-making machines we built.

Regarding the use of computational geometry across fields, we have a big building in the University here, designed by Norman Foster. And there was a retrospective exhibition of Foster's work with the British Museum atrium, and the Gerkin in London, and so on. Robert Aish was there. He worked for a company that helped Foster and Partners actually build the actual shapes.[18]

I hadn't realised how much the aerospace work had influenced architecture. Do you use Rhino? That came out of Boeing, I think. It's a NURBS-based system. I met some of the people from early on.[19] I was a consultant at Boeing at one stage, and I knew some of the people who worked with NURBS went to work with Boeing. I have met the Rhino people – I know of them and they probably know of me. But it's fascinating. Basically the students I lectured to at Syracuse in the 1970s, Rich [Riesenfeld], but also Lewis Knapp and Ken Versprille, they developed NURBS: their three theses led to NURBS.[20] So it's strange in a way, because now I row in a sea-going boat. And the boat comes in a kit, just like a model aircraft. You pop up the parts from plywood sheets and bend them to shape, and all the rest of it. The parts are usually cut using numerically controlled cutting tools, but I asked the guy who made it about the process. And he said, 'Well, we modelled these boats from the line drawings we get from the boat designer, and we turn them into 3D using a system called Rhino.' So, now I row on a boat designed in Rhino, which uses math that I helped to develop!

DC: I see that you have a poster for the 1968 exhibition Cybernetic Serendipity.[21] Can you share a few thoughts about your collaborations with artists or practitioners in other fields?

RF: I am actually the author of that image [*points at one of the images in the poster*]. Cybernetic Serendipity was the first conference in computer art, and Jasia Reichardt came around to Cambridge sort of 'trawling for data', and took a few polaroids of things in the screen, which she later superimposed. The image was … Well … We weren't really trying to design anything. We were just trying to see how flexible we could be. And she said, 'I'll take this!' So she picked up a bunch of our polaroids and said, 'That's it! That's art!' And off she went with it. This was early 1968, at Cambridge, when I was writing my thesis or shortly after.

DC: So she basically came to see what you were doing in the CAD group…

RF: Yes. And I ended up showing some work in that exhibition, and going on the poster – I think she signed it for me somewhere. So, whenever someone talks to me about art, I say, 'I exhibited in the ICA!' There were some interesting people there: Gordon Pask, John Lansdown, Lionel Penrose, Meredith Thring, and many others. As always, there's that question of what is art? I never considered my image to be art, but she did. So, it must be art! But why that image? I just … You know. Of all the stuff they had *[6]*!

DC: Robin, thank you.

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

Notes

1. 'Timesharing' enabled multiple researchers to access a computer remotely through terminals distributed across the MIT campus. Sketchpad, developed by Coons's student Ivan Sutherland as part of his PhD thesis at MIT, was the first interactive graphics system. Ivan Edward Sutherland, 'Sketchpad, a Man-Machine Graphical Communication System' (Massachusetts Institute of Technology, 1963).
2. Maurice Wilkes and Donald Welbourn are both important figures of postwar computing in Britain and, as Forrest recalls, played a founding role in the establishment of the CAD Group at Cambridge. Wilkes was the director of the Mathematical Laboratory (which is also referred to as Maths Lab throughout the interview) and Donald Welbourn, Forrest's first advisor, was a member of the Engineering Department faculty. The Mathematical Laboratory was renamed to Computer Laboratory in the late sixties. Haroon Ahmed, *Cambridge Computing: The First 75 Years, University of Cambridge: Computer Laboratory* (London: Third Millenium Publishing, 2013).
3. Steve Coons and Douglas Ross co-directed the Computer Aided Design (CAD) Project at MIT between 1959 and 1970. Sponsored by the US Air Force, the project explored the possibilities of computers in design, and laid the foundations of CAD systems. For an expanded discussion of the MIT CAD Project and its architectural repercussions, see: Daniel Cardoso Llach, *Builders of the Vision: Software and the Imagination of Design* (London, New York: Routledge, 2015).
4. For an expanded discussion of the Computer-Aided Design Project at MIT and its architectural repercussions, see: Ibid.
5. Daniel Cardoso Llach, 'Software Comes to Matter: Towards a Material History of Computational Design', *DesignIssues*, 31:3 (2015), 41–55, available online: < doi:10.1162/DESI_a_00337> [accessed 15 September 2016].
6. Robin Forrest, 'On Coons and Other Methods for the Representation of Surfaces', *Computer Graphics and Image Processing*, 1 (1972), 341–59.
7. The first publication on Coons's surface methods dates to 1964, but several accounts point to earlier developments. Steven Anson Coons, Institute of Technology Massachusetts, and MAC (Massachusetts Institute of Technology) Project, *Surfaces for Computer-Aided Design of Space Figures, ESL Memorandum 9442-, M-139* (Cambridge, MA: Massachusetts Institute of Technology, Electronic Systems Laboratory, 1964).
8. Forrest recalls that in the summer of 1967 Steve Coons was a visiting scholar at Harvard working with Ivan Sutherland.
9. Negroponte took a class with Steve in 1966 when he was still a student, and then it was because of Steve that he started teaching, filling in for Steve in the Mechanical Engineering Department before being hired in Architecture, where he and Leon Groissier started the Architecture Machine group in 1968. See more about this relationship: Daniel Cardoso Llach, *Builders of the Vision: Software and the Imagination of Design* (London, New York: Routledge, 2015).
10. Forrest recalls that the system, called Duct, 'ended up in a curious way involved with Rich Riesenfeld's work in Utah, and becoming part of a much larger system after both Riesenfeld's Alpha-1 system and Duct were bought by another company called Delcam.'
11. See J. C. R. Licklider, 'Man-Computer Symbiosis', *IRE Transactions on Human Factors in Electronics HFE*, 1 (March 1960), 4–11; and Benjamin Lee Whorf, *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*, ed. by John B. Carroll (The MIT Press: Cambridge Mass., 1964 [orig. pub. 1956]).
12. Bruce G. Baumgart, *Winged Edge Polyhedron Representation* (Stanford, CA: Stanford University, 1972).
13. See Robin Forrest, 'On the Intersection of Lines', *Computational Geometry Project* (Norwich: University of East Anglia, November 1984), Robin Forrest files.
14. The Computer-Aided Design Centre was funded in 1967 by the British government through the Ministry of Technology in Madingley Road, Cambridge, CB3 OHB. With time, it became a private organization, which, under the direction of Crispin Grey, evolved into Aveva PLC. According to Forrest, the Centre provided 'a direct route for the CAD Group's research to be developed and exploited by industry'. Aveva PLC still develops CAD systems for specialised applications such as oilrigs, refineries, and ships. Crispin Grey, one of the original CAD Group members, and later a member of the centre for Land Use and Built Form Studies at Cambridge, was the CEO in charge of taking the CAD Centre to the private sector.
15. P. E. Bézier, 'Example of an Existing System in the Motor Industry: The Unisurf System', *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 321:1545 (9 February 1971), 207–18, available online: < doi:10.1098/rspa.1971.0027> [accessed 15 September 2016].
16. Robin Forrest, 'Computational Geometry', *Proceedings of the Royal Society of London*, A. 321 (1971), 187–95.
17. Newman would later work in Xerox and co-author with Bob Sproull a book on interactive computer graphics. William M. Newman and Robert F. Sproull, eds, *Principles of Interactive Computer Graphics*, 2nd edn (New York, NY: McGraw-Hill, Inc., 1979).
18. At this time Robert Aish worked at Bentley Systems, where he developed a software system called Generative Components that made many computational geometric techniques available to architects. He was also at the Smart Geometry Group, which trained many practicing architects in parametric design. Aish now works at Autodesk, the makers of Autocad, developing a visual programming environment called Dynamo. He is interviewed in this volume, see 'The Evolution of Architectural Computing: From Building Modelling to Design Computation'.
19. Rhino is short for Rhinoceros, a 3D modelling software developed by the Seattle based company McNeel and Associates and used widely by architects and industrial designers. To develop the software the company's CEO Bob McNeel hired mathematicians from Boeing who had developed computational methods for surface modelling. For further detail, see: Cardoso Llach, *Builders of the Vision: Software and the Imagination of Design*, p. 93.
20. NURBS (Non-Uniform-Rational-Bezier Surfaces) are

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press

computational representations of complex surfaces. For a technical introduction and a history of NURBS, see: David F. Rogers, *An Introduction to NURBS: With Historical Perspective* (San Francisco, CA: Morgan Kaufmann, 2001).

21. The 1968 exhibition Cybernetic Serendipity was a pioneering event of computer art that brought together artists, architects, and computer scientists. Forrest's image in the poster, which hangs in his studio, is a series of superimposed wireframe drawings of curved surfaces. Forrest's contribution to the exhibition was titled 'Mathematically Defined Surfaces'. See Jasia Reichardt, ed., *Cybernetic Serendipity. The Computer and the Arts*, 1st edn (Praeger: New York 1969), p. 96.

Illustration credits
arq gratefully acknowledges:
Marc Evans (licensed under Creative Commons Attribution 2.0) 1
Robin Forrest, 2–6

Acknowledgements
This conversation is part of a larger project on the histories of computational design by Daniel Cardoso Llach. Thanks to the Martin Centre for Architectural and Urban Studies at the University of Cambridge, UK, and the School of Architecture at Carnegie Mellon, US, for providing the conditions and material support enabling it.

Authors' biographies
Robin Forrest worked with Steve Coons as a Visiting Research Fellow at MIT Project MAC in 1967 and as a Visiting Professor at Syracuse University in 1971–2. A founding member of the Cambridge CAD Group, he moved to the University of East Anglia in 1974 as a Reader and became a Professor in 1980.

Daniel Cardoso Llach is Assistant Professor in the School of Architecture at Carnegie Mellon University and the author of *Builders of the Vision: Software and the Imagination of Design* (Routledge, 2015). He holds a BArch from Universidad de los Andes and an MS and PhD in Design and Computation from MIT.

Authors' addresses
Robin Forrest
robin.forrest@gmail.com

Daniel Cardoso Llach
dcardoso@cmu.edu

https://doi.org/10.1017/S1359135517000173 Published online by Cambridge University Press