Between 1959 and 1967, the Massachusetts Institute of Technology (MIT) hosted the pioneering Computer-Aided Design Project, funded by the US Air Force. Engineers, researchers and students were brought together to re-imagine 'design in the language of the machine', and one of the most renowned students on this programme was Ivan Sutherland, the inventor and programmer of Sketchpad – the first interactive graphics system. Here, **Daniel Cardoso Llach,** an architect and researcher based at MIT, traces the influence of this Cold War project on the culture of building design and production today.

Within the vibrant culture of research and development taking place at the Massachusetts Institute of Technology (MIT) during the Cold War era, a group of engineers re-imagined design in the language of the machine. Such redefinition rests on a series of technical and epistemological supports: on one hand, the encoding of perspective geometry in matrix form, the development of the first programming languages, and the indexical combination – in the computer – of data and graphics, enabled these engineers to make distinctions between traditional and 'computerised' design practices. On the other, a philosophical view of design as a kind of generalised problem-solving led them to articulate a cybernetic discourse of design as the iterative performance of a 'man–machine' problem-solving engine. Creative design was, under their view, a repetitive cycle of representation, analysis and materialisation, where a 'creative' moment was always followed by a 'mechanical' one: a symbiosis where 'a designer and a computer can work together as a team on design problems requiring creative solutions'.[1]

The collective of engineers, researchers and students behind these technical and discursive innovations worked under the umbrella of the Computer-Aided Design Project, a research operation active at MIT between 1959 and 1967 and funded by the US Air Force. Do their innovations underpin the contemporary ethos of building design and production?

DEPARTING FROM DRAUGHTSMANSHIP: A NEW CRAFT

Despite the Computer-Aided Design Project's military intent, the engineers followed what they thought was the most general approach possible to design. Instead of developing specialised applications to solve particular problems (such as systems for designing missiles or aircraft), they sought to develop abstract languages able to describe 'any' design problem.[2] In contrast to paper drawings, the engineers realised, the symbolic world of the computer allowed for different forms of description to coexist and interrelate. For example, in a computerised description of a house, shapes could be used as indexes for information about materials, prices, structural calculations and other attributes. Steven A Coons, one of the MIT project's leaders, nicely captured this idea of design as a multivariable endeavour in a 1963 paper:

> In the design process the designer is concerned with a large set of variables [some of which are] continuous, like the weight of a certain part [and some are] discrete 'point sets' (like material: steel, brass, lead, plastic).[3]

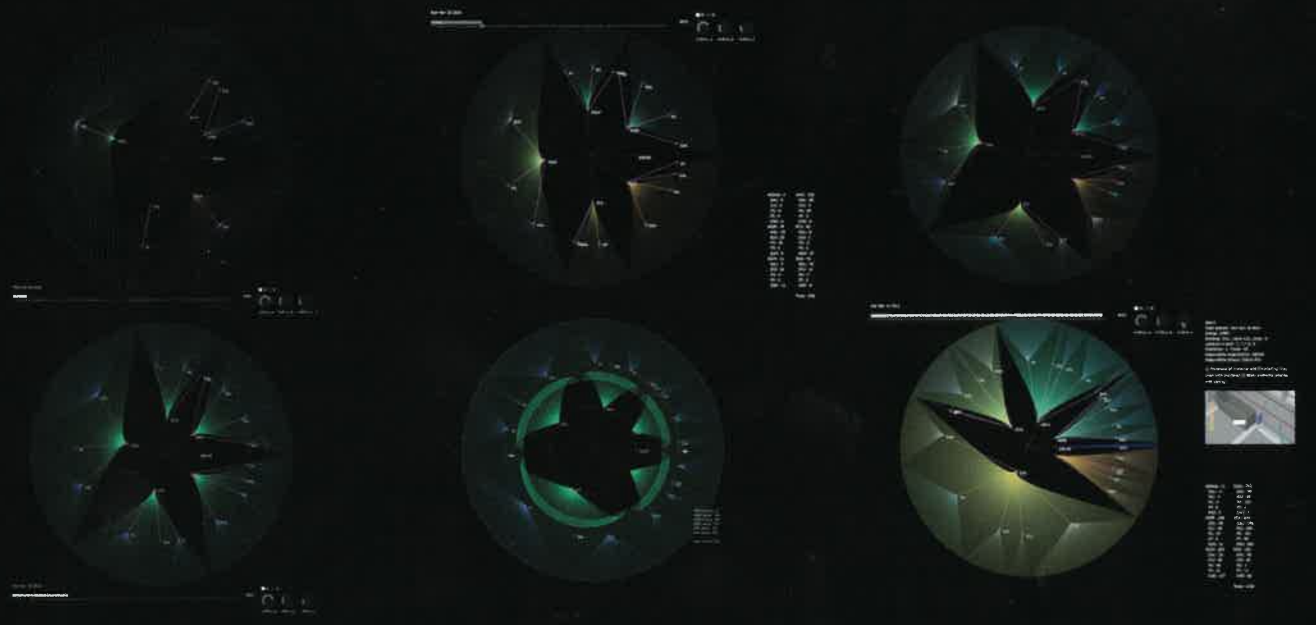
One of Coons's students, Ivan Sutherland – who famously programmed the first interactive graphics system, the Sketchpad[4] – thought that the production of computer-generated representations was essentially different from (and perhaps superior to) traditional drafting: 'An ordinary draftsman is unconcerned with the structure of his drawing material. Pen and ink or pencil and paper have no inherent structure. They only make dirty marks on paper,' he wrote.[5] Notably, Sutherland's claim was premised on the projection of tectonic qualities onto computational representations. The capacity of computers to index multiple layers of information[6] suggested a topological resemblance between representations and physical artefacts. Thus, the key challenge for the new practitioners of computational design, Sutherland thought, was to not think of computer-generated graphics as drawings of a design, but instead as computerised descriptions: things that are built instead of drawn. By identifying a new craft of building representations, Sutherland anticipated a contemporary culture of integrated approaches to project description and delivery:

> One should think of computer-aided design as producing not only graphical outputs but also material lists; labor estimates; floor area computations; heating, lighting, and ventilation simulations (to demonstrate the adequacy of the design); as well as many other auxiliary outputs. Only when the computerized version of the design is the master document from which all auxiliary information is derived, preferably with computer assistance, will a complete computer-aided design system have been created.[7]

ENCODING THE THIRD DIMENSION: ROBERTS'S 'PERSPECTIVE HACK'

It was Sutherland's colleague, Lawrence Roberts, who developed the technology to display objects in perspective on the computer screen. As a student and summer research assistant at MIT, Roberts shared with Sutherland the TX-2 machine, the computer where the two would develop some of the earliest computer graphics applications. While developing his doctoral thesis – a program to generate a 3-D object from digital descriptions of photographs of planar solids[8] – Roberts found that there was no known way of representing perspective on the computer. He thus hacked it by studying the mathematical methods for perspective geometry from German textbooks from the 1800s and then 'translating' them into the matrix form used to program computers.[9] His method made it possible for the computer to display a 3-D object from any given 'camera' point, and his algorithm continues to run in most modern 3-D software. Modestly, Roberts recalls that during this period of discovery they 'were picking up old things that people had done … and applying it to the real world'.[10] Modesty notwithstanding, Roberts's 'hacking' of perspective is a crucial moment in the establishment of our age's distinctive representational tradition: an indexical combination of Albertian perspectivalism, and data processing.

Daniel Cardoso Llach, Data visualisation of a contemporary building's design and construction coordination, 2012
above and previous spread: In line with the aspirations of the early CAD pioneers, contemporary design and construction projects rest on an intricate network of digital flows. This data visualisation traces the coordination process of a large project during four months of its development.



One should think of computer-aided design as producing not only graphical outputs but also material lists; labor estimates; floor area computations; heating, lighting, and ventilation simulations (to demonstrate the adequacy of the design); as well as many other auxiliary outputs.

Lawrence Roberts, Compound object construction, 1963
top: Roberts's method for creating 3-D objects was based on the aggregation of simple polyhedra.

Lawrence Roberts, Camera transformation, 1963
bottom: Roberts's 1963 PhD thesis at MIT invented the notion of a camera moving in a virtual 3-D space.

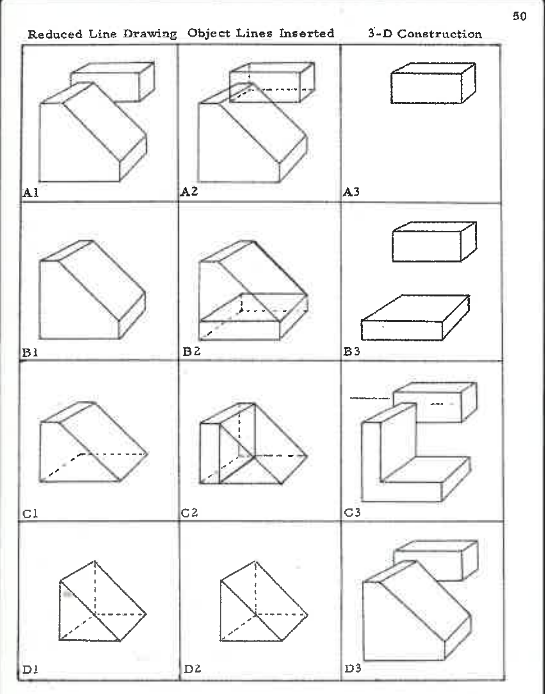


Figure 4: Compound Object Construction: Original Line drawing in A1 is processed to obtain 3-D figure in D3 by sequential recognition and deletion of four models in steps A, B, C, and D.

IMAGINING MECHANICAL ARCHITECTS: NEGROPONTE AND THE CAD PROJECT

In the spring of 1966, Nicholas Negroponte, then a Masters of Architecture student, attended Coons's Computer-Aided Design course. The course introduced Coons's pioneering method for the computational description and manipulation of parametric 3-D surfaces,[11] as well as Roberts's methods for perspective representation.[12] The young Negroponte asked Professor Coons to become one of his academic advisors, and went on to write a thesis on the computer's capacity for 'simulating' perception.[13] In his early career as a faculty member at MIT, he further sought to use the theoretical and technical framework developed during the Computer-Aided Design Project years to re-imagine architectural and urban planning practices as technologically enabled participatory endeavours, bypassing what he construed as the elitist sensibility of architects and planners.[14]

To the disappointment of these pioneers, during the decades that followed the MIT Computer-Aided Design Project, the CAD software industry was dominated by systems seeking to 'simply' automate traditional drafting procedures. However, the technical and discursive innovations of the project anticipate the infrastructural role of software in building design today. While Negroponte's techno-idealistic conceptualisation of human–machine design interaction has offspring across many different fields, the discourses of design and representation engineered by Sutherland, Coons, Roberts and others underpin the current trends towards increasingly synchronous, networked and collective 'building information models'.

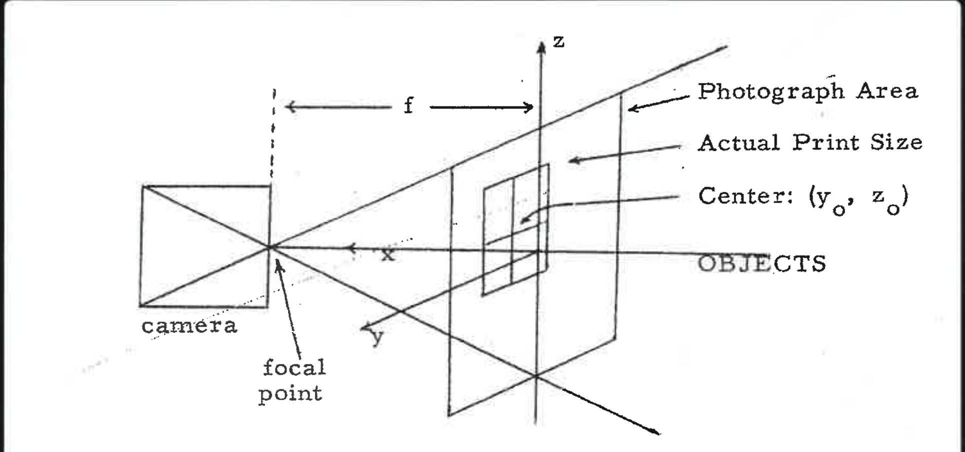


Figure 1: Camera Transformation

POSTSCRIPT

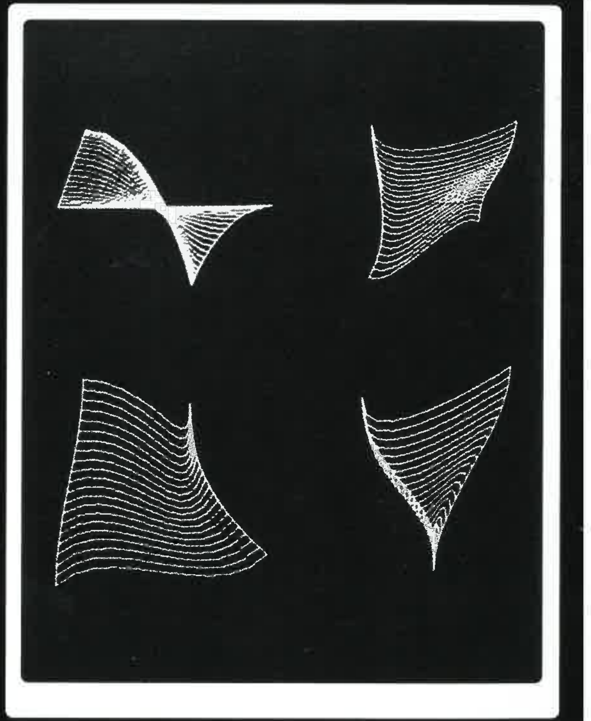
Reminding us of the inevitably material dimension of architecture, historian and architectural theorist Kenneth Frampton conjures the Greek voice 'Tekton' to redeem the structural articulation of architecture as the discipline's fundamental vector.[15] A similarly constructionist sensibility can be used to understand how, within the vibrant cultures of technology production that evolved at MIT during the Cold War era, engineering students and researchers formulated technological discourses of design premised on the structured character of computational abstractions. This brief overview has highlighted how, by construing computational representations as a form of building, these engineers gave new meanings to the words 'design' and 'representation', inaugurating a technological imaginary of design and creativity. Through its influence on nearby researchers like Negroponte, and on popular discourses about design, this imaginary had important effects on architecture and urban studies, illustrating a crucial moment in the ongoing building of algorithmic thought.

Notes

1. Douglas Taylor Ross, Steven A Coons and John E Ward, *Investigations in Computer-Aided Design for Numerically Controlled Production: Combined Interim Engineering Progress Report, 1 June 1965–31 May 1966*, MIT Report ESL-IR 2. Douglas Taylor Ross, *Computer-Aided Design: A Statement of Objectives*, MIT Electronic Systems Laboratory (Cambridge, MA) 1960.
3. Steven A Coons, *An Outline of the Requirements for a Computer-Aided Design System*, MIT Technical Memorandum ESL-TM-169, MIT Electronic Systems Laboratory, (Cambridge, MA), 1963, p 300.
4. Ivan Edward Sutherland, 'Sketchpad, a Man-Machine Graphical Communication System', PhD thesis, MIT (Cambridge, MA), 1963.
5. Ivan Sutherland, 'Structure in Drawing and the Hidden-Surface Problem', in Nicholas Negroponte (ed), *Reflections on Computer Aids to Design and Architecture*, Petrocelli/Charter (New York), 1975, p 75.
6. Ibid.
7. Ibid, p 76.
8. Lawrence G Roberts and Peter Elias, 'Machine Perception of Three-Dimensional Solids', PhD thesis, MIT, 1963.
9. Ibid.
10. J Hurst, JT Gilmore, LG Roberts and R Forrest, 'Retrospectives II: The Early Years in Computer Graphics at MIT, Lincoln Lab, and Harvard', *ACM SIGGRAPH '89 Panel Proceedings*, ACM (New York), 1989, p 56: http://doi.acm.org/10.1145/77276.77280.
11. Steven Anson Coons, 'Surfaces for Computer-Aided Design of Space Forms, Mac-tr -41', MIT Project MAC (Cambridge, MA) 1967.
12. Roberts and Elias, op cit.
13. Nicholas Peter Negroponte, 'The Computer Simulation of Perception During Motion in the Urban Environment', Masters thesis, MIT (Cambridge, MA), 1966, p 99.
14. I critically discuss Negroponte's technological reconfiguration of design elsewhere. See Daniel Cardoso, 'Inertia of an Automated Utopia: Design Commodities and Authorial Agency 40 Years After the Architecture Machine', *Thresholds*, No 39, July 2011, pp 39–44.
15. Kenneth Frampton, *Studies in Tectonic Culture: The Poetics of Construction in Nineteenth and Twentieth Century Architecture*, ed John Cava, MIT Press (Cambridge, MA), 2001.

top: Ivan Sutherland working on Sketchpad, 1963.

Steven A Coons, Computer-generated surfaces, 1965
bottom: Coons's method for surface representation, publishe
in 1965, is the origin of parametric surface descriptions.



Design environments are undergoing a perceptible shift in authorship. Advances in scripting interfaces are empowering architects to create parts of their own design environments. The boundaries between end user and developer are falling down around a network of designers sharing their creations as part of an emerging design ecosystem.

While computer-aided design (CAD) software has included scripting interfaces for many years, and there have been select individuals who have used this functionality to develop digital tools, the number of designers now using these scripting interfaces is increasing. What they are using it for, as well as how they are using it, are also changing. It is no longer only the CAD specialist writing scripts to increase efficiency or manage construction data, but also the designer creating geometry and finding form through sketching with code.[1]

Through the introduction of better-performing scripting capabilities in CAD software, designers have been able to quickly generate large amounts of geometry using relatively simple scripts. For example, the Visual Basic for Applications (VBA) scripting interface included with Bentley Systems' MicroStation v8 in 2001 was vastly superior to the Basic scripting interface included with v7 (1998). These generative or analytical algorithms can be shared, either through computer code or through packaging the script with an easy-to-understand interface. Through a combination of using common geometric elements, and outputting data to simple formats such as text files or spreadsheets, these computational techniques allow a diverse range of design software, analysis techniques and fabrication methods to be linked.

Beyond the scripting interface, some designers are finding further potential to shape the design environment itself. A number of designers have been developing 'plug-ins' that function as core parts of existing CAD platforms. The process of development is similar to the computer-programming techniques of scripting, but unlike scripts these plug-ins are packaged as small pieces of software and themselves become part of the design environment. Each plug-in emerges to address a specific problem or opportunity that an architect has identified in their work, thus widening the digital design environment around the desires of the individual designer.

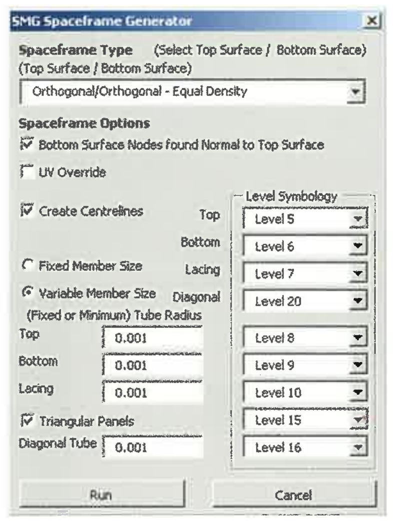
Not only are more and more architects computer programming, writing scripts and creating plug-ins, but these are increasingly being shared via the Internet, conferences and workshops.[2] This marks the formation of a new design ecosystem, one under constant evolution and catalysed by sharing at a scale never before seen, that is simultaneously a community of architects and a collection of related algorithmic concepts. This design ecosystem is probably growing in a CAD environment near you.

THE CATHEDRAL AND THE BAZAAR

One way of structuring a CAD environment is to have it do everything: a single self-contained application that can take a design from massing to construction drawings, from positioning doors to producing schedules, from creating geometry to composing visualisations. By integrating all of these components, each part can make assumptions and guarantees about how the rest operate, saving the designer the indignity of converting CAD files into different data formats (if this is even possible) and from having to purchase many different standalone softwares. This monolithic process is the holy grail of project lifecycle management (PLM) and, arguably, its brethren building information modelling (BIM).

The self-contained design environments set up to support these monolithic processes belong to a software category that Eric Raymond calls 'cathedrals' – large applications crafted by a highly talented group working together in isolation.[3] Raymond contrasts the cathedral with the 'bazaar' – a marketplace in which the collective action of individuals contributes to the larger community. For architects, the dichotomy exposes the fact that for many decades design environments have almost exclusively consisted of cathedrals.

Breaking the norm, Robert McNeel & Associates' Grasshopper® is a bustling bazaar-type environment. Grasshopper is a graphical programming environment that runs within Rhinoceros® CAD software, where architects visually link together components that are conceived of, and created by, other architects rather than by a team of software engineers. Even McNeel's sole developer on the project, David Rutten, has an architectural background rather than a formal computer science education. While Rutten controls the core, he is joined by a community of architects who freely share the plug-ins that make up Grasshopper.



SMG Spaceframe Generator

Spaceframe Type (Select Top Surface / Bottom Surface)
(Top Surface / Bottom Surface)
Orthogonal/Orthogonal - Equal Density

Spaceframe Options
[✓] Bottom Surface Nodes found Normal to Top Surface
[ ] UV Override
[✓] Create Centrelines

( ) Fixed Member Size
(•) Variable Member Size
(Fixed or Minimum) Tube Radius

Top	0.001
Bottom	0.001
Lacing	0.001

[✓] Triangular Panels

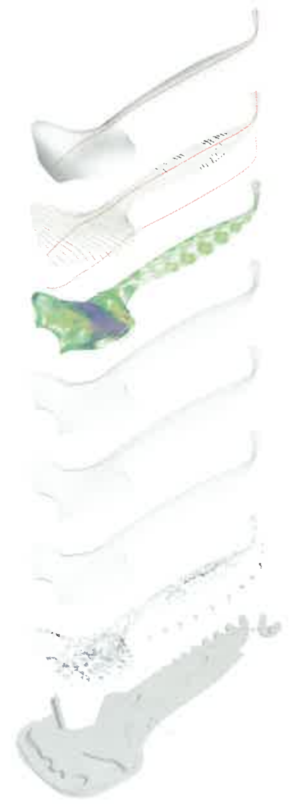
Diagonal Tube	0.001

Level Symbology

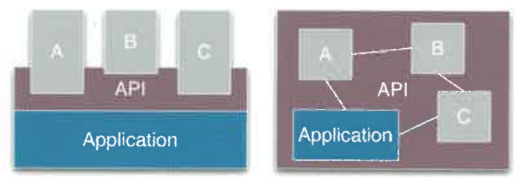
Top	Level 5
Bottom	Level 6
Lacing	Level 7
Diagonal	Level 20
	Level 8
	Level 9
	Level 10
	Level 15
	Level 16

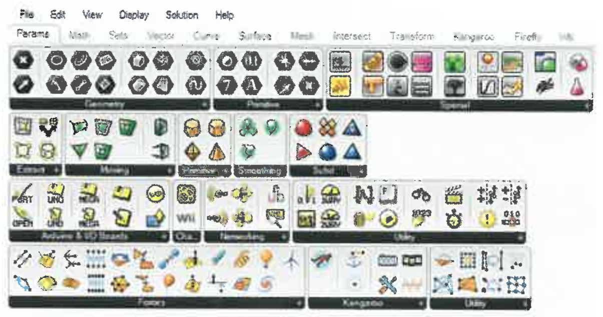
Run Cancel

Foster + Partners, West Kowloon Canopy Masterplan, Kowloon, Hong Kong, 2003
right and previous spread: Using a script written in VBA in Bentley's MicroStation, the designers were able to quickly draw in 3-D a 1,500-metre (4,921-foot) long space-frame structure. The use of this technique made the design of this complex structure simple and intuitive. The script was written by Brady Peters and modified throughout the design process.



Foster + Partners, West Kowloon Canopy Masterplan, Kowloon, Hong Kong, 2003
Scripts produced different types of data that could be used by various software packages to analyse the design or generate additional geometry. The design simultaneously existed in multiple representations, for example: design surface, space-frame structure, centreline model, environmental simulation model, 2-D panel layout diagram, and full 3-D model for visualisation.





ANIMALS IN THE JUNGLE

This section of the issue features the plug-ins: Galapagos, Kangaroo, Firefly, WeaverBird, GECO™ and Pachyderm Accoustical Simulation. Unlike the teams of specialist developers working on monolithic CAD applications, the creators of these Grasshopper plug-ins work alone or in pairs, and they are all end users. Significantly, their plug-ins are motivated by specific problems they have encountered in their own architectural practice. They also share their work, for free, through the sizeable online community. This 'bazaar-esque' community serves to shape the design ecosystem by implicitly encouraging or discouraging particular plug-ins. A traditional hierarchical structure does not apply since popularity is based on usability and functionality rather than on limitations of access, cost or compatibility. There is such a market for these plug-ins that one wonders whether, in future, they will be sold more formally as apps.

The focus of each plug-in on a particular niche problem follows what Doug McIlroy has termed the Unix philosophy of programming: to 'write programs that do one thing and do it well. Write programs to work together.'[4] It is the 'working together' that distinguishes the Grasshopper environment, for all the plug-ins within it can freely exchange data with one another. For instance, WeaverBird can panel a structure designed in Kangaroo without the designer manually converting the data, and without the authors of either plug-in needing to coordinate with each other. This is in large part due to the Grasshopper application programming interface (API) developed by David Rutten, which formalises the exchange of data around simple collections of basic geometric primitives. This 'geometric-content-based' data exchange is in opposition to BIM's 'assigned-attribute-based' data structures,[5] and is a simplification that enables plug-ins to easily work together.

There is a diverse range in what the plug-ins do, but each facilitates a translation of some kind. GECO and Firefly serve as translators between Grasshopper and other sources of data; GECO connects to the Autodesk® Ecotect® environmental analysis package, while Firefly sends and receives data from the Arduino microcontroller platform. The remaining plug-ins all translate research into accessible components. For instance, WeaverBird takes work done by computer scientist Edwin Catmull and others, and packages it for designers unfamiliar with the mathematics. As such, knowledge normally locked away in esoteric research is made accessible to the point where a designer can use it without needing to necessarily understand what he or she is designing with. Sherry Turkle describes this as 'Macintosh Transparency' (transparent because nothing gets in the way), which she places in opposition to 'Modernist Transparency' (transparent because you can see how everything works).[6] In Turkle's view, Macintosh Transparency risks producing designers who are 'drunk with code', designers so enamoured with the results that they fail to see that the plug-in translates a far more nuanced concept.[7]

The cathedral and the bazaar/plug-in structures
centre: Plug-ins are typically isolated from the core application – and each other – by an API, but in Grasshopper the API lets the plug-ins and application freely intermix and communicate with one another.

Iconic plug-ins
bottom: Once installed, the Grasshopper plug-ins appear seamlessly alongside the core Grasshopper components.