JAUES

Journal Of Al-Azhar University Engineering Sector

Vol. 14, No. 51, April, 2019, 868-874

DIGITAL DESIGN TECHNIQUES: TECHNICAL KNOWLEDGE AND SAUDI ARCHITECTURAL EDUCATION

WAJDY S. QATTAN
Umm Al Qura University, Makkah, Kingdom of Saudi Arabia

ABSTRACT

Digital design techniques became an inevitable trend, but to use these techniques architects need considerable higher computation skills. These skills encloud programing languages, mathematics, software and English. It is assumed that the relationship between Saudi architectural education and these techniques is still not strong, even though programing is being taught at Saudi computer science schools. Therefore, it is fundamental to know how these techniques have shifted the architectural design process and to know the way to start scripting.

Keywords. Architecture; Digital, Education, Techniques, Knowledge; Saudi.

1. INTRODUCTION

Despite the notion that Saudi architectural education is not new, its relationship with digital design techniques is still modest. As known digital design techniques usage has increased, and they are embraced by world-leading architecture schools and offices. To use these techniques, Saudi architects need to master some computational knowledge and skills.

The paper consists of two parts: part one will investigate the desirable digital design techniques technical knowledge from the Saudi architects' and computer science specialists' view. This will be based on a series of interviews done by the author with teaching staff and students at three Saudi universities. The second part will study the consequential shift of digital design techniques in the architectural design process and how this shift calls for computational knowledge especially scripting.

Part one is aiming to show the urgent need for high computation skills, as it is assumed that Saudi architects are missing them. At the same time, the computer science specialists emphasize collaboration between computer science and architecture departments, as they are a primary source of computational education.

Then the paper will highlight some essential aspects on how these techniques have shifted the conventional architectural design process. This is to inform the Saudi architectural educators and students. This shift contributes to integrating design, fabrication, and construction. In addition, this shift takes architects from exploitation of computers to exploration via computers.

Then the paper will discuss some vital aspects related to the increasing popularity of coding in architecture, such as understanding coding, access coding, techniques of coding, algorithms in coding, relationship between architects and scripting and using scripting.

2. DIGITAL DESIGN TECHNIQUES TECHNICAL KNOWLEDGE AND ARCHITECTURE

2.1. ARCHITECTS' VIEWS ON TECHNICAL KNOWLEDGE

As a technical notion, Saudi architects need to know that digital design techniques require high computation skills. The majority of the interviewees agree that programming skills, English and mathematics are the three main aspects to enter the world of digital design techniques. For example, AK (interviewed 2018) commented:

> Programming language is the main aspect. In addition, English language, physics and mathematics are very important factors. These four things are the most difficult things in the scripting world. I expect, these aspects will scare students. This is why pushing students to learn 3D modelling is easier.

Equally important is what MT (interviewed 2018) encounters when he teaches computer skills. He commented:

> I guess these techniques will potentially be difficult, especially with students. There will be language problems. Most of our students do not use English. For example, I am having a hard time introducing the software's interfaces. Students need to learn some terms, the commands' icons, and to understand the command functions. The students' English language capabilities do not allow them to use Building Information Modelling software. Thus, they will not be able to use programming languages, which require fluent English and mathematics and programming language knowledge. For example, if I introduced the term Extrude or Sweep, students will ask "where to find these commands on the AutoCAD or Rivet interface?" We are still sticking in this level and cannot move to programming now. Do you think a student in this level can write a script?

This comment shows how Saudi architecture students are currently using computers in architecture design and what knowledge they need.

OO (interviewed 2018) also points out that using scripting requires certain skills and experience, so being a typical architect is not enough to use Grasshopper or Processing. Unfortunately, Saudi architects are missing these skills, because they do not learn them at university. Moreover, English language competency differs between universities. For instance, King Fahad University of Petroleum and Minerals interviewees do not see English as a problem because they have their curricula in English. Whereas it is a major problem to King Saud University and Umm Al-Qura Uuniversity students because they have their curricula in Arabic. For other interviewees English is not a problem because it is used within the framework of architecture terminology. In other words, it does not require fluency, only a limited number of certain words. Therefore, English language is an individual matter: it varies from person to person and from university to university.

2.2. COMPUTER SCIENCE SPECIALISTS' VIEWS ON TECHNICAL KNOWLEDGE

According to the computer science students, there is some required knowledge to master programming, which are summarized in the following: architects need to know and understand how computers think, need to have enough information about programing language, need to have fluent English, need to understand the structure and logic of the programing language, need to solve problems using programing, and need mathematics as something fundamental.

Computer science students studying programing need to do at least five subjects which usually takes from two to three years to master this skill. Learning programing is not about the time spent and how many courses or languages, it is about assimilating programing logic.

In addition, programing is not a skill that any person can master without a teacher. According to SN (interviewed 2018), studying programing requires a guide, styles and elements not found in textbooks. Furthermore, computer science specialists agree that programing in architecture is different, even though it uses the same principles and logic. They also agree that programing will be difficult for architects especially at the beginning.

It is not an easy task for a computer science specialist to program for architecture. There is a misunderstanding or gap between them, thus it is better to be an architect with programing skills. Architects need to start programing in a separate subject to learn the basics and then they need to do another subject to link architectural requirements with programing.

The computer science interviewees provided noteworthy advice for architects. Now architects need to learn these technical skills to maximise their computational capability. It is interesting that programing as a technical skill exists at the Saudi university in the computer science departments, but not in the architecture schools.

3. Shifting the architectural design process

3.1. SHIFTING DESIGN PROCESS AND TECHNIQUES

Shifting architectural design processes and techniques happened with the introduction of computation in architecture. When architects add the possibilities of scripting a broad shift is defined, and appeared in some progressive schools of architecture, and in mainstream architectural culture (Leach 2009). In the 1990s, computers are significantly involved in the design process, from drafting and modelling to intelligent systems and processing architectural information (Terzidis & Vakalo 1992). The design process has changed from its traditional top-down forms of control towards bottom-up and behavioral form generation.

Architecture designers now have new roles that are dependent on their computational skills. According to Oxman (2006), architects now interact with generative and performative processes using information as a new material. The designer becomes a tool builder which means designers need to improve their computational skill sets to deal with this new architectural trend.

As a result of this shift, most of the digitally designed architectural projects belong to one of the following domains. First, projects with complex geometries are generated by algorithmic rules, digital sculpting processes or other computational tools. Second, architecture relies on computation and numerical processing to create significant buildings to achieve particular performance criteria (Marcus 2012).

3.2. INTEGRATION OF DESIGN, FABRICATION AND CONSTRUCTION

The architectural design process has also shifted toward the integration of design, fabrication and construction. With the use of computation, designers have a range of intricate surfaces available, but the challenge is how to determine fabrication techniques to construct these surfaces. Designers need to rethink their design process by developing new methodologies to address digital design fabrication requirements, which can happen by allowing the generation, integration and strategies of manufacturing to inform each other.

as a consequence of this shift, a new innovative, motivated, highly skilled generation of programmers and designers will engage in a discourse of material and fabrication processes with unprecedented results (Dunn 2012). The shift in architectural design is characterised by extensive knowledge sharing and

collaborative production, as well as a noticeable increase in digitally fabricated buildings.

Marble (2012) finds that there are three themes that shift the architectural design processes: designing design, designing assembly and designing industry. Designing design is a step to redefine the design process as integrated design systems to pose design itself as a design problem. Designing assembly is a material issue to address the influence of digital production and material properties on the design concepts. Designing industry is an organisational issue towards multidisciplinary practice. The range of information in a given architectural project is expanding faster, thus there is a demand to incorporate a range of expertise to link information with design, fabrication and construction.

3.3. FROM EXPLOITATION TO EXPLORATION

The computer exploitation phase has already passed, so architects are now focusing on using computation as an exploratory medium to reveal more possibilities and expand limitations. Benjamin (2012) argues that computation is a way to explore rather than to exploit, to creatively search within wide-ranging possibilities.

Computers are exploratory machines to uncover hidden ideas and solutions. To do that, it is better to perform all operations through computers independently without human intervention, starting from running the scripts until results are obtained. Thus, designers can negotiate the decision-making process via computers.

This indicates that computer techniques in architectural design range from representation and visualization to scripting, where custom algorithms are used as a design system to generate geometrical output from numerical input. This way of using computers accounts for the shift and expands the architect's ability and imagination.

4. CODING AND SCRIPTING

4.1 Understanding Coding Language

It is important for architects to understand coding languages and techniques, especially when commercial software does not fulfil the designer's desires and the design requirements. According to Shea (2004), it is not expected that architects become experts in programing, but to take advantage of the full capacity of computers, they need to understand and use coding languages. In doing so architects do not limit themselves to the use of commercial software, but they explore the possibilities of creating forms through algorithmic processes.

Architects need to use computers as a problem-solving tool. They need to use the generative capacity of computers, which means that design will be described with the use of algorithms. These algorithms will be translated to computers using coding languages, which make using such behaviors possible. But that does not mean that the computer will do the complete job. Architects cannot just set up the rules and let computers do the job for them. Indeed the process needs to be explored, played and mastered with care and cleverness (McCullough 2006).

4.2 ACCESS TO CODING

In most software, there are some modelling commands ready to use by clicking an icon. In the same software, there are other functions that are not accessible for architects through the interface. These functions are for designers with programming skills, and are extremely powerful (Aish 2004).

According to McCullough (2006), in 1986. Parametric design and other coding languages (e.g. Java processing) were introduced, in conjunction with some educational courses (as at MIT in the United States). As a result, most architecture circles set up digital research units to benefit from these technological techniques.
Computers are powerful tools, but they need human thinking to find solutions and describe them precisely. In architectural design, computers need a team of knowledgeable experts in all the design stages who are able to define a solution even if it is very complex (Scheurer 2012). At the same time, it offers an open source through the internet to all interested architects. The power of scripting is enhanced by the internet which provides a platform to share knowledge in a "dynamic reference hive", where the accumulation of information is far greater than the sum of individuals (Burry 2011).

4.3 TECHNIQUES OF CODING

Coding consists of a wide range of techniques, languages and interfaces in which each code is used to deal with a particular problem. These techniques are developed systematically. As an example of these techniques, the L-system is one of the common techniques of coding that often deal with biological forms. Technically, it is an algorithmic code to simulate branching in a way known as rewriting systems (Dollens 2005). This is when the code system writes and rewrites itself based on the information that has been given by the designer and then re-given by the system itself.
Usually architects start by planning the information to be fed to computers. Sometimes they add a simple interface, just a few buttons and sliders, to change the input variables fairly quickly.
Along with scripting, plug-ins are developed as part of the current computer-aided design software. The development of plug-ins is very similar to scripting, but they are packaged as small pieces of software and became part of the design environment (Davis & Peters 2013).

4.4 ALGORITHMS IN CODING

Coding techniques are linked strongly to the use of algorithms in scripting language to allow architects to access the calculations capacity of computers. According to Dunn (2012), algorithms are a very empowering method that work as a mediator between designer and computer. Dunn suggests two factors to consider. First, the process of algorithm must be specified gradually in order to build its logic effectively. Second, the accuracy of the algorithms, for if there is just one simple error such as a character, the script will not run properly or not at all. A missing semicolon will prevent the whole program from running or lead to unexpected wrong results. This highlights the significance of the architects' mindset and practice, as they work usually with flexibility, not precision.
Moreover, it is crucial to know how a set of algorithms could create something that the designer does not expect. Simon (2004) states that this could be possible through genetic algorithms where the interactions of random, unusual emergence and independent objects occur.

4.5 RELATIONSHIPS BETWEEN ARCHITECTS AND SCRIPTING

Scripting has changed the designer's role from tool user to toolmaker. Instead of producing digital models, designers need to write a program, which generates complex geometries. That means architects need to interpret algorithmic thinking to understand the results of the generating code and to know how to modify it in order to explore new possibilities (Peters 2013).

Architects can explore and generate architectural spaces and concepts via writing and modifying algorithmic codes. This places making these tools within the design itself. Where some software may lack some important features, it points to the need for scripting. For example, in the De L'Orme Pavilion in Barcelona, Bernard Cache (2003) states that because of the lack of projective geometry in the current software, the team need to implement this procedure using scripting.
The recommended way to start scripting is by writing small and simple scripts, as there are some challenges ahead. According to Scheurer (2010), scripting has three challenges. First, architects need to know how to program and how to deal with unambiguity. Second, abstracting a given problem. Third, knowledge about geometry.

4.6 USING SCRIPTING

Burry (2011) notes that "scripting is a driving force for 21st century architectural thinking". It is a road without clear signposts, so why architects want to join scripting? Are they joining mainstream alternative practice, a club, a movement, or counterculture? Coding is the process of describing all of the steps that a computer must perform to complete a task. Computers are not like people: they can do only one task at a time, and they cannot guess or interpret meanings if they are not described exactly. It is commonly known that computers are stupid. That means there is only one interpretation for every piece of code.
To take steps into coding, architects need to know some terms and what they mean, such as statements, sequences, conditions and loops. Shaw (2011), in his book Learn Python The Hard Way, presents 52 exercises to get architects to start coding, and describes coding in this phrase: "the hard way is easier".

5. CONCLUSION

It is desirable for Saudi architects to consider that using digital design techniques require high computation skills i.e. English, mathematics, programing languages and software. Overall, Saudi architects confirm their need to improve these skills. They see these needs as something normal that must happen, but it will be easy to achieve. It could be a minor problem and would be defeated easily, through providing the missing knowledge and the eagerness of students to learn. From the computer science students' perspective, there is some required knowledge to master programing, which are summarized in the six aforementioned points.
As a result of the new digital design techniques, architects need to change the design process and develop new methodologies to comply with the digital design and fabrication needs. This will be through allowing the influence between generation, integration, and manufacturing.
Saudi architectural educators and students need to benefit the full capacity of computers, and this might be through understanding coding languages. In this medium, architects can produce a huge number of iterations including forms, geometries, and materials in short time. To access this computational capacity, architects need to go beyond the commercial architecture software interfaces where only designers with coding skills can work. There is no standards or fixed measurements; the coding environment is an open source for sharing knowledge and experience. There is a broad range of coding techniques, languages, and interfaces, but they are linked strongly to the algorithms. Therefore, architects need to develop their relations with coding and algorithms to build custom tools and to produce a creative/innovative outcome.

6. ACKNOWLEDGEMENTS

The author would like to acknowledge Umm Al Qura University, King Suad University, King Fahda University of Petroleum and Minerals, and all teaching staff and students who participated in this research.

REFERENCES

1. Aish, R., 2004. Extensible Computational Design Tools for Exploratory Architecture. In: B. Kolarevic, ed. Architecture in the digital age: Design and manufacturing. Spon Press, New York, 244-247.
2. Benjamin, D., 2012. Beyond Efficiency. In: S. Marble, ed. Digital workflows in architecture: Designing design, designing assembly, designing industry. Birkhäuser, Basel, 14-27.
3. Burry, M., 2011. Scripting cultures: Architectural design and programming. Wiley, Chichester.
4. Davis, D. AND Peters, B., 2013. Design Ecosystems: Customising the Architectural Design Environment with Software Plug-ins. Architectural Design, 83 (2), 124-131.
5. Dollens, D., 2005. Digital-botanic architecture. Lumen Books.
6. Dunn, N., 2012. Digital fabrication in architecture. Laurence King.
7. Leach, N., 2009. Digital morphogenesis. Architectural Design, 79 (1), 32-37.
8. Marble, S., 2012. Digital workflows in architecture: Designing design, designing assembly, designing industry. Birkhäuser, Basel.
9. Marcus, A., 2012. Workflow Patterns: A Strategy for Designing Design. In: S. Marble, ed. Digital workflows in architecture: Designing design, designing assembly, designing industry. Birkhäuser, Basel, 46-49.
10. McCullough, M., 2006. 20 years of scripted space. In: M. Carpo, ed. The Digital Turn in Architecture 1992-2012. John Wiley & Sons, 182-187.
11. Oxman, R., 2006. Theory and design in the first digital age. Design Studies, 27 (3), 229-265.
12. Peters, B., 2013. Computation Works: The Building of Algorithmic Thought. Architectural Design 83 (2).
13. Scheurer, F., 2010. Materialising complexity. In: R. Oxman & R. Oxman, eds. Theories of the digital in architecture. Routledge, 287-291.
14. Scheurer, F., 2012. Digital Craftsmanship: From Thinking to Modeling to Building. In: S. Marble, ed. Digital workflows in architecture : Designing design, designing assembly, designing industry. Birkhäuser, Basel, 111-118.
15. Shaw, Z.A., 2011. Learn Python the hard way. Available to read on the web for free. Viewed 2015, <http://www.brookeweston.org/students/downloads/learn-python-the-hard-way.pdf>.
16. Shea, K., 2004. Directed randomness. In: N. Leach, D. Turnbull & C.J. Williams, eds. Digital tectonics. Wiley Academy, Britain, 89-101.
17. Simon, J., 2004. Authorship, Creativity, and Code. In: J. Maeda & R. Burns, eds. Creative code. Thames & Hudson, London, 46-47.
18. Terzidis, K. & Vakalo, E., 1992. The Role Of Computers In Architectural Desig. IAPS Proceedings.
19.