
Made in Rhino
0.053<270° OnOrtho OnTan OnPerp

Made in Rhino

An introduction to computer-aided design through the lens of Rhino

Camilla Wright, Unit 21
Tutor: Roberto Bottazzi
MArchY5 Thesis 2017

Contents

	Abstract	3
	Introduction	4-7
	Part I *What is it? How does it work?*	
Chapter 1	User Interface	8-14
Chapter 2	Software Process	15-18
	Part II *What does it mean? What are the implications?*	
Chapter 3	Software Tools	19-26
Chapter 4	Software Mediums	27-30
	Part III *Where is it going? Where is it taking us?*	
Chapter 5	Editability and Extendibility	32-35
Chapter 6	Software Development	35-39
	Conclusion	40-42
	Glossary	44-49
	Bibliography	50-59

Abstract

The degree to which architectural practice and education depend on computer-aided design (CAD) is increasing. Although the influence of technological development on architecture is broadly recognized, the role of software is rarely discussed. More often, those who design buildings and those who write about them, know little about the software used to create them. In CAD, software is the vehicle for realizing ideas, as such, an understanding of software beyond interface level is valuable to architectural theory. How is software effecting the way architects design? In response, this thesis will focus on an individual software program; Rhinoceros. A technical examination specific to this program will serve the overall intent; to assess the impact of software on the way architects conceive of their designs, what methods they employ to do so and what techniques formulate in the process? This Thesis is amongst the first to unearth the workings of software behind the interface of CAD applications. The material provides critical insight into the nature of software and the implications it has for architectural design.

Introduction

A central tenet of molecular biology states that the flow of genetic information in a cell is from DNA through RNA to protein. An exon is any part of a gene that encodes the RNA produced by that gene. In other words; the process of encoding a gene is made visible by the gene it creates.[1]

Today, CAD software is deep-rooted within Architectural practice and education. Software is integral to the design process, representation and delivery of a building. It is a pervasive presence, always between designer and his/her creation. Computers do not merely 'aid' design, the use of software tools has changed the way in which architects design, and think about designing. Software has infiltrated all aspects of the design process, however its undercurrents are overlooked as an instrument for design or a lens for discussion. In this vein, architect and theorist, Neil Leach, argues that the software used to design the architecture should be made visible in the building.[2] Molecular biology concurs. Despite this,

"Most criticism of digital architecture is performed without a clear understanding of the logic of CAD software." (Lynn, 2013).

What are software tools? How do they do what they do? What are the implications for design? Attempts to answer these questions are almost non-existent within architectural theory. In the field of media, there has been some attempt; "Lev Manovich is the only media theorist around who is talking not just about what computers do but how they do it" [3] In *Software Takes Command,* Manovich probes the processes by which media is authored and edited in software applications, his approach and findings provide a framework for this essay.

"What as users we experience as properties of media content comes from software used to create, edit, present and access this content." (Manovich, 2014)

It follows that, properties of buildings express the properties of the software used to design and construct them. Software is not separate from architecture, it passes out of the digital, onto the crit

---

[1] Meyers, R. ed., (1995). *Molecular Biology and Biotechnology: a comprehensive desk reference*. New York: VCH, pp.310, 428.

[2] Leach, N. (2013). *IaaC Lecture Series 2013-2014 "Adaptation"*. https://iaac.net/fall-lecture-series-2013-neil-leach/

[3] Ellen Lupton, Curator of Contemporary Design at the Cooper Hewit National Design Museum, USA, on the back of Manovich, L. (2014). *Software takes command*. New York: Bloomsbury.

wall and has permeated the built environment. Therefore, a knowledge of software is relevant to architectural theory.

Rhino is one of many CAD applications. Since its release in 1998 it has become a popular 3D modeling tool amongst students and architects.[4] Rhino's use is widespread; more than 300,000 commercial users and 10,000 schools.[5] Architects including Zaha Hadid, Herzog and Demeuron, Will Alsop, Peter Cook, Frank Gehry, Renzo Piano and Coop Himmelblau are known to have used Rhino for their projects.[6]

Evidently, the logic of software (how it works) is understated in the activity of design using software. This notion is shared among artists such as Casey Reas, whose work explores the ability of software to enable innovative ways of generating form and translating ideas.[7] To show how software logic surfaces in the design process, this thesis will describe the way in which Rhino works both interactively (human using the program) and at a programmatic level. Attention will be paid to the procedural process behind a Rhino tool and the nature of digital mediums. With this platform of knowledge, the consequences for the architect using Rhino can be more clearly identified.

Design inside software can be understood as a translation of the real-world design environment into a software design environment. Like all software applications Rhino simulates real-world design tools; physical tools and mediums become software legible. For example, digital files are simulations of the paper equivalents. 'Cut', 'Copy' and 'Paste' are real-word concepts performed by physical tools prior to software (e.g scissors, the printing press and glue). Inside software, the medium is data and the tool is an algorithm (function/procedure/routine/subroutine). Therefore, tools and mediums take on new properties and functionality; software augments and extends them.[8] However, all tools become similar in nature; they require explicit values and follow software logical methods. Use of the tool is defined by the software process, deviation returns a

[4] Rhino is the most popular 3D modeling tool at the following schools of Architecture: Westminster, Oxford Brookes, The AA and The Bartlett.

[5] Visualarq.com. (2017). *What is Rhino?* Available at: http://www.visualarq.com/info/what-is-rhino/ [Accessed 01 Apr. 2017].

[6] Archinect. (2017). *firms that use Rhino*. Available at: http://uk.archinect.com/forum/thread/1051/firms-that-use-rhino?ukredirect [Accessed 01 Apr. 2017].

[7] Reas, C., Barendse, J. and McWilliams, C. (2011). *Form+code in design, art, and architecture*. New York, NY: Princeton Architectural Press.

[8] Manovich describes the degree to which software augments and extends real-world tools in part I of Manovich, L. (2014). *Software takes command*. New York: Bloomsbury.

'null' value. Users gain the ability to use more tools, on more elements of their design, but lose the ability to do so freely. Creativity is confined to the parameters of the software. Consequently, the process by which designers use tools is changed and the cognitive model for *how* they design is altered.

The phenomenon of exaptation consists in a trait taking on a different function than the one it originally served. (Bryant, 2014)

It is by this phenomenon that Rhino and its tools are subject to the evolutionary trajectory of the wider software industry. CAD software is a software species; it is a variation of all other software applications.[9] All programs are essentially a series of procedures which process data. This model enables tools which originate in one software application to move into new spheres of industry. Universal commands ('Cut', 'Copy', 'Paste', 'Undo', 'Redo', 'Import', 'Export') present in all kinds of program menus are evidence of this migration. Once released into the user community, a tool may be used differently from the original intent. If a particular tool becomes popular in one sphere, in order to stay competitive, another software application is likely to integrate it. As a result, users can 'Copy' text, images, audio, vectors, files etc. using the same command (Ctrl C), a real-world tool equivalent is inconceivable. Furthermore, techniques developed for writing software (programming) can pass through the interface and provide a concept for a tool. In this way, programming paradigms become design techniques.

In part, Rhino's success is due to the scope of tools it provides and its compatibility with multiple software applications. From scripting to rendering, jewelry design to animation, Rhino is an extensive resource, supporting a diverse community of professions.[10] Responding to the needs of a particular design field would introduce this tool to all design fields using the resource. Thereby, tools stray from their intended use (another case of exaptation). Standardized file formats allow users to import and export their work*. Rhino is effectively a software hybrid, it can work with multiple data types produced by a myriad of applications.* This enables a designer to move

[9] Manovich outlines the software species model. "all new qualities of "digital media" are not situated "inside" the media objects. Rather they all exist "outside" - as commands and techniques of media viewers, authoring software, animation, compositing and editing software, game engine software, wiki software and all other software "species."... While we are indeed *"being digital," the actual forms of this "being" come from software*." Ibid. p.149.

[10] Categories in Rhino's project gallery include marine, jewelry, footwear, spacecraft, vehicle, film & set, furniture, creatures, architecture and more Rhino3d.com. (2017). *Project Gallery Rhinoceros 3D*. Available at: https://www.rhino3d.com/gallery [Accessed 8 Apr. 2017].

between applications (whether it be, media editing, compositing or rendering) for which each has a different purpose and design orientation. A software 'craftmanship' emerges; designers exploit different aspects of each resource to best serve their creative requirements. Software techniques combine to create media compilations; "media hybrids".[11] This, and the manipulability of digital data, alters the way an architect conceives his/her design. The design is not concrete, it can always be edited, rendered, redrawn, animated, or 'photoshopped'.

This Thesis is structured as followed. Part I will evaluate Rhino itself, providing an explanation of what the program is, how one uses it and what it does. Part II will discuss how mediums and tools have become 'softwarized'; In what ways has software changed the properties of these design elements and what are the consequences for design? Taking Manovich's approach, Rhino tools will be assessed as follows: Is it an extension of a real-world tool or does it have no physical precedence prior to software, and to what extent is it a product of another software application? Part III will observe the development of software, the features that expose it to change and the direction in which it is heading. How does the evolution of Rhino and other software species effect design and what are the implications?

[11] For examples of computational media hybrids see Manovich, L. (2014). *Software takes command.* New York: Bloomsbury. p. 163.

Part I *What is it? How does it work?*

This section provides a context for discussion and outlines the functionality of Rhino; from both the user's perspective and the activity behind the interface. Like any other program, there are two 'sides' to Rhino; the human legible interface and the software legible program.[12] The software process is the procedure that occurs inside software when a user clicks on a tool or manipulates a model etc., it is what makes the tool do what it does. The material in this section is organized accordingly; the user experience and the software process. The ways in which Rhino's functionality effects a designer's methods and cognitive process will be discussed.

Chapter 1: User Interface

When opening Rhino, a user is presented with a window through which he/she can interact with Rhino software. This window is the Graphical User Interface (GUI), and defines the parameters for a designer using the stand-alone software. The GUI is the user legible map of the software; usability of the program depends on the user's ability to navigate this map. This serves as a useful metaphor; it illustrates the way software can be 'learnt'. A map aids the process of navigation. When 'learnt', the process of navigating the real-world is influenced by this representation i.e. a bird's eye view or landmarks (mapped information) will be used to navigate the space rather than elevations or visual cues (unmapped information) in the real-world. Map aided navigation and computer-aided design are comparable. Learning Rhino software influences a designer's mental model of the design process so that it resembles the process of using Rhino.[13]

Led by Alan Kay, the Learning Research Group at PARC in the mid-70s developed techniques to construct a Graphical User Interface (GUI) that would support learning, experimentation and artistic expression. Influenced by the work of cognitive psychologist, Bruner, these techniques enabled the use of different mentalities (inactive, iconic and symbolic) in combination with each other. The group's aim was to create a dynamic medium for learning and creativity, enabling users to think through symbols, actions and images. As a result, Rhino's GUI involves constant

[12] Commonly referred to as the 'front-end' and 'back-end' in the software industry.

[13] Manovich describes this mental model in reference to media authoring/editing. "In this way, the logic of programming is projected to the GUI level and becomes part of user's cognitive model of working with media inside applications." Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 222.

interplay between different mentalities. The *Mouse* activates enactive mentality; the user can locate themselves on the screen and point to screen objects. Icons and windows activate iconic mentality; users recognize icons, examine contents in windows, define options or configure lists. Short Keys and the Command Line activates symbolic mentality; the user can invoke a series of functions abstractly.

In an article entitled "Personal Dynamic Media" (Kay and Goldberg, 1977) the computer medium that emerged were defined as quantitatively different and historically unprecedented. Not only could it represent and augment the properties of media, but also generate new tools to augment media and create new types of media. They described the computer as a "metamedium" whose content is "a wide range of already existing and not yet invented media".[14]

Kay went on to say; "It is not a tool, though it can act like many tools. It is the first meta *medium*, and as such it has degrees of freedom for creativity representation and expression never before encountered and as yet barely investigated."[15]

The computer "metamedium" enabled the invention of unprecedented media types, such as Architecture Machine Group's "Aspen Movie Map" and, intrinsic to Rhino, navigable 3D spaces. Manovich defines this "new computational media that has no physical precedence" in contrast to "simulations of prior physical media extended with new properties"[16]. The techniques used for the manipulation of media can be defined as media specific (specific to particular types of data) and media independent (can work with various data types.) The elements of the navigable 3D space medium are data for 3D objects combined with techniques for visualising them in perspective, creating and editing them and simulating the effect of different lighting on their surfaces etc.[17]

[14] Kay, A. and Goldberg, A. (1977). Personal Dynamic Media. *New Media Reader*, p. 391.
[15] Kay, A. (1984). Computer Software. *Scientific American*, 251(3), p.52. Quoted in Gassee, J. The Evolution of Thinking Tools. *The Art of Human Computer Interface Design*, p.225.
[16] Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 110.
[17] Ibid. p. 212.

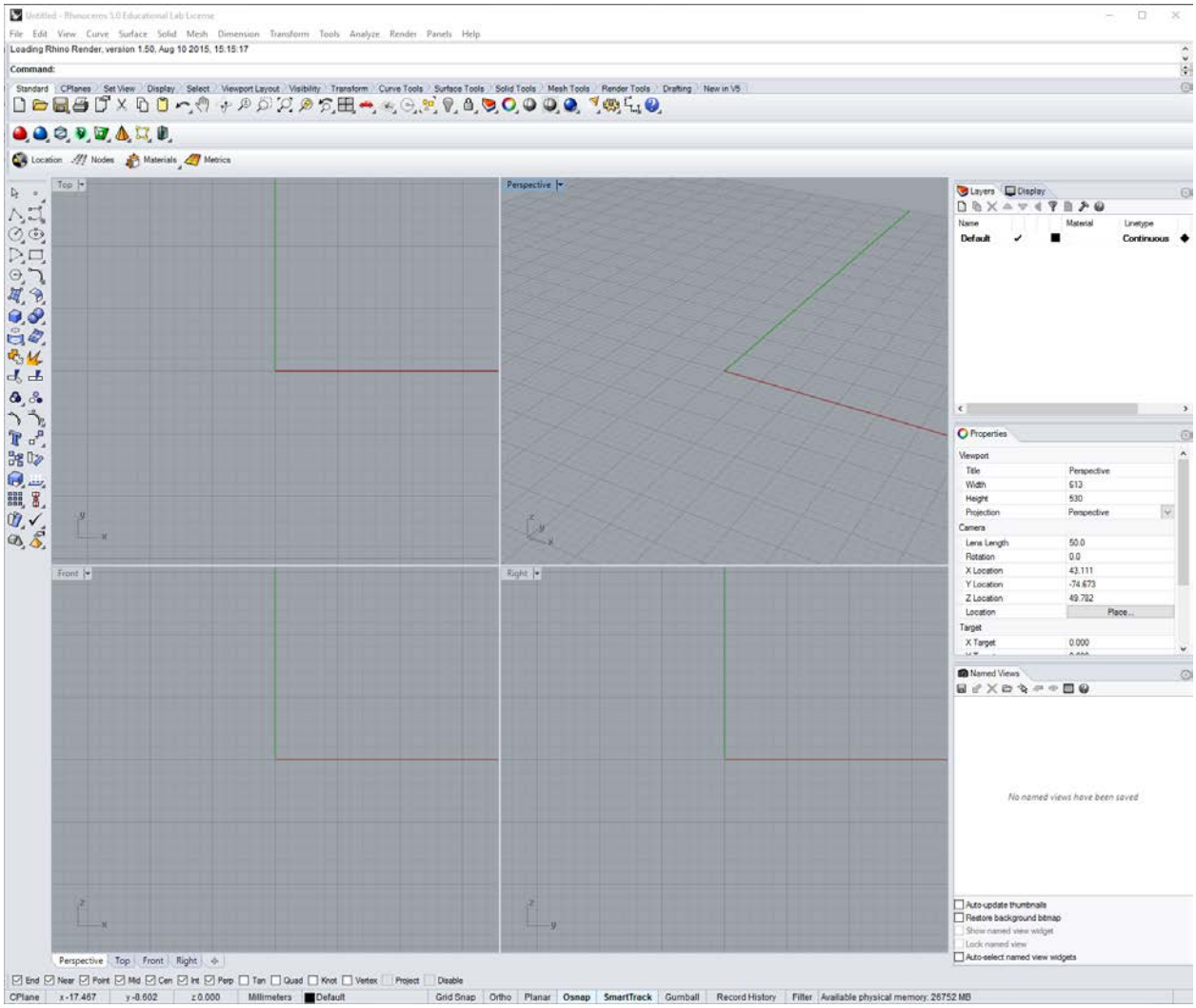

Figure1.0 Rhino V5 Screenshot: GUI

The tool related features of Rhino's GUI are the Sidebar, the Command-Line across the top and categorized tool groups for which the icons can be viewed via tabs. Panels on the right are for managing visibility, object properties and display settings. The centre of the window is divided into 4 viewports (top, front, right, and perspective) of the 3D workspace, a feature consistent with other CAD programs.[18] Along the bottom are a series of checkboxes and icons for drawing constraint options. And of course, the omnipresent menus 'File', 'Edit', 'View' etc. The presence of these menus (all in the locations one would expect) supports the notion that human legibility is paramount to the design of a GUI. A Rhino user is effectively interacting with a wider context;

[18] Tim Johnson (working for Ross on the Air Force Sponsored CAD project) extended Sutherland's Sketchpad from 2D to 3D. Sketchpad III was the first computer-based graphics system to provide three orthogonal views of a three-dimensional object together with a perspective view of that object. The perspective view could be at a different scale from the other views and any change made in one view would be visible in the others. See Johnson, T. (1963). Sketchpad III A Computer Program for Drawing in Three Dimensions. In: *Proceedings of the Spring Joint Computer Conference*. Spartan Books, p.348.

with software that has been before it and shaped the universal cognitive model for using a program. However, these words are general concepts; copying and pasting a curve in Rhino is different to copying and pasting some text in word, not just because of a difference in data but because of the process (what they do and how they do it) unique to different software applications. All modern software programs follow general principles regardless of use (writing text, drawing graphs, editing media etc.) and their interfaces use established conventions employed in all application software.[19] The concept of customizing settings, retrieving 'History' and numerically controlling techniques are shared by most software applications. All computer programs stem from the same evolutionary family; CAD software is a variation of a species.[20] However, although these programs are similar in principle, the programming procedures behind their interfaces are very different. The illusion of homogeneity at interface level masks the heterogeneous reality: every software package is different, down to the most primitive functions.

A Command-Line Interface (CLI) uses typed commands instead of a mouse to exchange information and instructions. The user types in a command and presses enter/return. The program may respond by displaying a prompt, requiring the user to type another command and so on, until the intended output is fulfilled.[21] Under Kay, development of the UI was geared towards a friendly programming environment, however, Apple's proverb, 'easy to learn, easy to use', eroded this vision, resulting in a more abstract computer-human interaction; the modern GUI.[22] Today, the

[19] "The unique properties and techniques of different media have become software elements that can be combined together in previously impossible ways." Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p.176.

[20] Ibid. p.140. On a side note: War and a desire to control were dominant influences on advances in early computer technology in the US. In turn, the first developed software was engineered for defence systems and to improve military resources. Its primary roles were simulation (flight conditions and nuclear weapons) and analysis (navy airplane control, enemy tracking and code deciphering). CAD software is an evolution of these.

[21] Commands must be typed correctly and in the right order, or else the command will not work. The CLI dates back to the earliest personal computers, enabling programmers to work on computers directly rather than through a team of personnel operating the machine. The first instance of the computer as a 'personal device', combined batch processing and time-sharing systems. This enabled a 'hands on' approach to programming, students at MIT 'write' programs individually, such as the first popular computer game, 'space wars'. Ceruzzi, P. (2003). *A history of modern computing*. 2nd ed. London, Eng.: MIT Press. pp. 73, 93, 283. And Weisenberg, D. (2008). *The Engineering Design Revolution*. [ebook] p.12. Available at: http://www.cadhistory.net/toc.htm [Accessed 7 Jan. 2017]. ch. 3. pp.4-5.

[22] Shift from CLI on IBM to GUI on Macintosh. See Parsons, J. and Oja, D. (n.d.). *New perspectives, computer concepts, 2014*. Prior to Kay and in the realm of CAD software; Douglas Ross (working on the Computer Aided Design Project at MIT) saw no need for users to understand the internal data structure of the CAD applications. His focus was on 'automated design' not 'computer-aided' design. See Feldman, C. (1968). Subsets and modular features of standard APT. In: *Proceedings of the Fall Joint Computer Conference*.

CLI is orientated towards high level users, typical to scripting applications such as Python or Forth.[23] Rhino uses the command-line within its GUI, a feature inherited from AutoCAD.[24] In this way, Rhino accommodates different user environments; icons and menus for the less experienced user and typed commands for the more experienced. The command line bypasses a layer of abstraction at interface level, and thus becomes more like the interface envisioned by Kay.

Using the command line to create a model offers a different way to search the tool archive. The command line integrates a search algorithm (a feature of web browsers) for 'related' commands which appear below the command line. If a user types 'Split', a list of 'related' commands appear. See fig 1.1.

Thompson Books, p.67. He went on to develop Automated Engineering Design (AED), a software which provided a springboard for Ivan Sutherland's creation of Sketchpad. Different ideas about the human-computer relationship have driven the direction of software development and degrees of computer 'aided' design. See Weisenberg, D. (2008). *The Engineering Design Revolution*. [ebook] p.12. Available at: http://www.cadhistory.net/toc.htm [Accessed 7 Jan. 2017]. ch. 3 p. 10. And Reintjes, J. (1991). *Numerical control: Making a New Technology*. New York: Oxford university press. p. 81.
Douglas Engelbart, who worked with Kay "envisioned users creating tools, sharing tools, and altering the tool of others" ENGELBART, D. and ENGLISH, W. (2003). A Research Center for Augmenting Human Intellect. In: N. Wardrip-Fruin and N. Montfort, ed., *The New Media Reader*, London: MIT Press. p. 232.

[23] Techopedia.com. (2017). *What is a Command Line Interface (CLI)?* Available at: https://www.techopedia.com/definition/3337/command-line-interface-cli [Accessed 02 Apr. 2017].

[24] Rhino is an offshoot from Autocad. It began as a plugin to integrate the NURBS library of an automobile design software, Applied Geometry. AutoCAD, its clones and Rhino are the only programs with a command line interface incorporated into the GUI According to McNeel Employee Adam Hollis on the McNeel Forum. (2017). *Command-line driven design software*. Available at: https://discourse.mcneel.com/t/command-line-driven-design-software/33081 [Accessed 10 Mar. 2017].

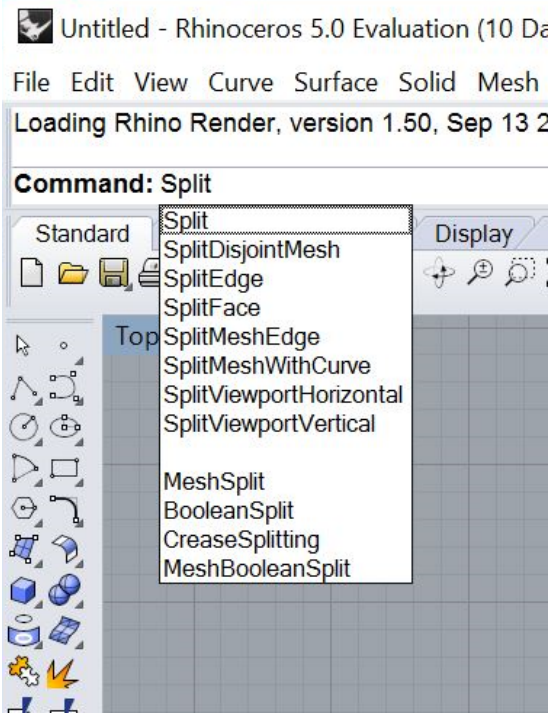

Figure 1.1 Rhino V5 Screenshot: Typing 'Split'

An algorithm identifies the tools with the word 'Split' in their name. This presents conceptually similar methods to the user; he/she can 'see' different ways of splitting something; a mesh, an edge or a viewport. The action of splitting becomes part of the user's mental model of design; geometry has the potential to be split, and splitting can be done using various types of geometry. Of course, 'Split' is a technique that exists outside of Rhino; a student working on a model might cut a piece of card or saw a piece of wood in two. Furthermore, a line (curve) or a template (surface) can be used as a cutting guide. While there are tools which split things in the real-world, the application of each tool is limited due to its specialization (e.g. scissors can cut card but not wire, pliers can cut wire but not card) yet the variety of tools or methods is unlimited (e.g. tearing paper, scoring and snapping plastic or driving wedges into wood). *In Rhino, the dynamic is reversed. The 'Split' tool can be used on any geometry in a model but can only be used in a way defined by Rhino.* A successful result requires the user to follow the tool method and explicitly define all geometric relationships. In this reversal, *a designer gains the ability to split more things yet loses the freedom to split them in any conceivable way.*

'Split' as a conceptual operation, is used on other kinds of media; on images (Photoshop can split channels), on music (Audacity can split audio tracks) and on film (Movie Maker can split video). The software 'Split' does not strictly apply to geometry, it is a variation of all other software splitting tools. In Rhino, editing a model is similar to editing other kinds of media. The user is

aware of this; he/she can ‘Cut’, ‘Copy’, ‘Paste’, ‘Move’ and ‘Split’ all kinds of digital media through different software applications. Contrary to their physical equivalents, *software mediums are material-less, there have no physical properties to limit their manipulation*. The digital model is no exception, it is processed both computationally and cognitively in a similar way to other digital data. Distinction between mediums and the application of tools is blurred; the design process requires navigating this software ‘soup’ where almost all techniques can be applied to all mediums.

There is another point to be made here. In the list of ‘related’ commands, ‘SplitMesh’ and ‘SplitViewport’ are related. It goes without saying, the process required to split a mesh (a piece of geometry), and the process required to split a viewport (part of the GUI), must be different. The word split (meaning divide into two or more) is used implicitly to aid human understanding of the operation the tool performs. Computers do not understand concepts; the set of instructions used to divide a mesh are entirely different to those used to divide a viewport. This reveals a degree of abstraction in the CLI, the word ‘Split’ is not legible at programming level, another layer of procedure for invoking the process specific to each command exists beyond the interface. However, the conceptual definition of tools and the prompting of ‘related’ tools enables access to the entire library of Rhino tools; one can type a word that describes conceptually what they want to do (e.g. divide) and in return, be informed of the different ways this concept can be performed.[25]

[25] The CLI makes it “easy to discover new functionality” and enables a “much faster workflow” according to “asbeastos’ on the McNeel Forum. (2017). *Command-line driven design software*. Available at: https://discourse.mcneel.com/t/command-line-driven-design-software/33081 [Accessed 18 Mar. 2017].

Chapter 2: Software Process

Behind the interface, the Rhino process is the concurrent execution of multiple functions. Rhino tools are a combination of functions, variables and data structures encapsulated by objects. The instructions (code) that express these tools are written in C++, a programming language that uses object-oriented paradigms. Objects are an instance of a user defined data type, or class.[26] The class provides initial values for member variables and implementation of member functions (all information needed for a tool to perform its defined action). Variables can be defined by a Rhino user entering values or selecting options in the GUI (typically the command line). Tools belong to different code libraries.[27] These are essentially collections of procedures and objects which are linked to or retrieved during the tool process. Libraries have no knowledge of Rhino; they are 'upstream' so to speak. Rhino itself is wrapped around this core, it both implements and extends the resource.

For a tool to 'work' Rhino follows a command pattern defined by a command object. The information encapsulated in the object includes a method name, the object that owns the method and values for the method parameters.[28] This pattern is synonymous with the 'behaviour' necessary to design in Rhino. A user builds a model by typing a method (command) by name into the command line and entering values for the specified parameter(s) at each step involved in the process.

The command pattern involves a command, a receiver (parameter), an invoker (argument) and a client (caller function). A command object knows about the receiver and invokes a method of the receiver. Values for parameters of this method are stored in the command.[29] The receiver then does the work. An invoker object has no knowledge of a concrete command, only how to execute a command. Both an invoker object and several command objects are held by a client object. The client triggers the execution of a command by passing the command object to the invoker object.[30]

---

[26] www.tutorialspoint.com. (2017). *C++ Classes and Objects*. Available at: https://www.tutorialspoint.com/cplusplus/cpp_classes_objects.htm [Accessed 2 Mar. 2017].

[27] See Glossary 'Library'. The largest library is the open NURBS library, another is Troutlake, McNeels own 3D geometry kernel.

[28] En.wikipedia.org. (2017). *Command pattern*. Available at: https://en.wikipedia.org/wiki/Command_pattern [Accessed 11 Feb. 2017].

[29] Learn C++. (2017). *7.1 — Function parameters and arguments*. Available at: http://www.learncpp.com/cpp-tutorial/71-function-parameters-and-arguments/ [Accessed 5 Mar. 2017].

[30] En.wikipedia.org. (2017). *Command pattern*. Available at: https://en.wikipedia.org/wiki/Command_pattern [Accessed 11 Feb. 2017].

Compare this to the user's process. He/she can access and edit parameter values and therefore knows about the receiver. Beyond this, the user is not aware of the process; he/she does not know how the tool does what it does only that he/she can 'trigger' these tools (by clicking on the tool icon, typing the name of the tool in the CLI or using a keypad shortcut). Therefore, in Rhino, the designers process is similar to the software process; call a function, enter a variable, and get a return value. Inside software, the diversity of methods used on physical tools are reduced to analogous methods, all of which adhere to programming logic.

A class is a data structure containing functions. Classes can be 'private', 'protected' or 'public'. Member functions enable programmers to organize data in packages rather than variables. A 'public' Class, allows access to parameter variables from any part of the program, whilst protecting the functions which do the work. For example, the source code for constructing a box on a plane, with x, y, z dimensions is shown below (Box Constructor);

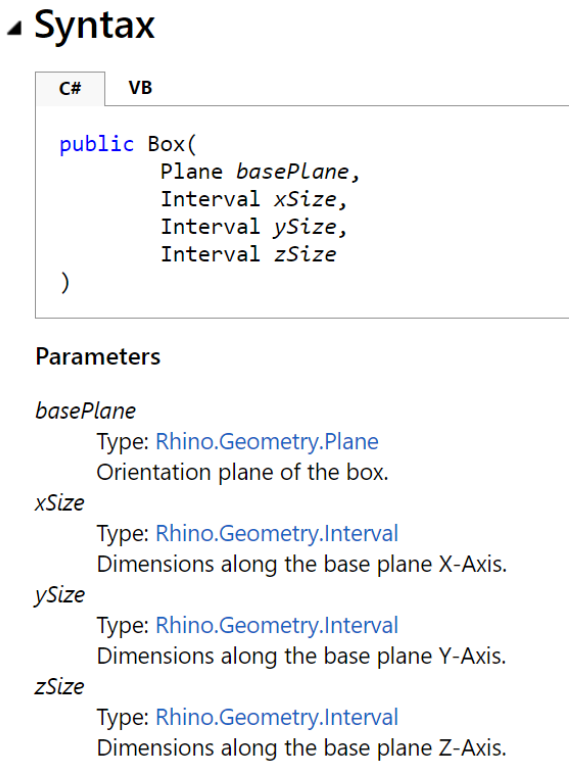

Figure 1.2 RhinoSDK, Box.Constructor, http://developer.rhino3d.com

A constructor function defines a class, in this case, Box is the constructor and the class is defined as public. 'Plane' and 'Interval' are member functions (inaccessible). The basePlane, xSize, ySize and zSize are public members, and thus their values can be set from outside of the class.[31]

[31] www.tutorialspoint.com. (2017). *C++ Classes and Objects*. Available at:

A tool which operates on another piece of geometry such as ‘ExtrudeCrv’, is different from a ‘Constructor’, it is a ‘Method’. See fig 1.3

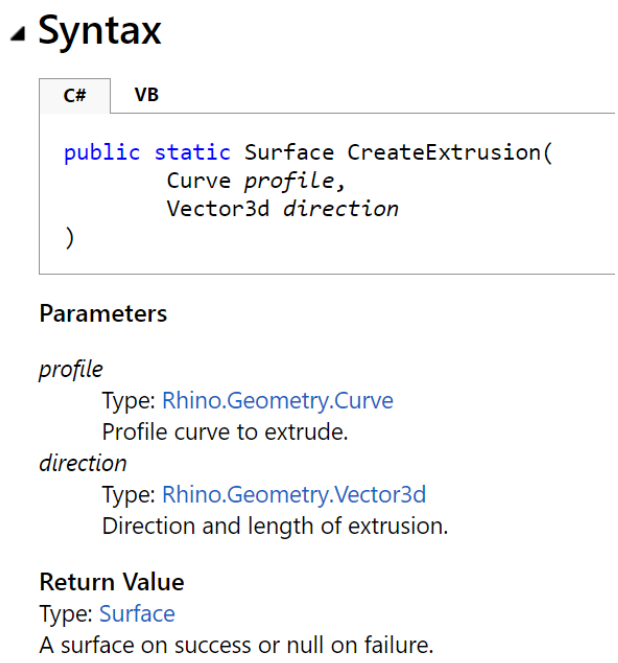

Figure 1.3 RhinoSDK, Surface.CreateExtrusionMethod, http://developer.rhino3d.com

Surface methods create surfaces and a Surface is a class. The use of static simply means that once the variable has been initialized (its value assigned), it remains in memory until the end of the program.[32]

At programming level, software is formulations of algorithms based on data structures, each inherent to the other.[33] Algorithms only work with the simultaneous existence of data structures.[34] The designer working in Rhino adopts the fundamental behaviour of software. Selecting a tool to operate on part of a model corresponds to using an algorithm to modify a data structure that stores

https://www.tutorialspoint.com/cplusplus/cpp_classes_objects.htm [Accessed 2 Mar. 2017].

[32] Cprogramming.com. (2017). *Tutorials - The Static Keyword in C++ - Cprogramming.com*. Available at: http://www.cprogramming.com/tutorial/statickeyword.html [Accessed 2 Mar. 2017].

[33] “decisions about structuring data cannot be made without the knowledge of the algorithms applied to that data and the structure and choice of algorithms often depend strongly on the structure of the underlying data.” See Preface of Wirth, N. (1978). *Algorithms + Data Structures = Programs*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

[34] “the one is pretty near useless without the other” Fuller, M. (2008). *Software Studies*. Cambridge: MIT Press. p.18.

the content of the part.[35] In this way, the properties of Rhino software are 'projected' to the User Interface (UI) level, influencing the way users design in Rhino practically, and how they understand this process cognitively.[36]

Tools require users to define variables; they must explicitly state (thus cognitively deliberate on) all geometric relationships in their model.[37] Design techniques are strictly controlled and depend on software logical procedure. This has implications for the methods and cognitive model for design.[38] Design in Rhino, is not dissimilar to what Fuller describes as the 'pragmatic dimension' of programming:

"the construction of algorithms as a precisely controlled series of steps in the accomplishment of a task is a clear indication of what might be called the pragmatic dimension of programming."[39]

[35] "a user's mental model of media creation and editing— both in the context of a particular application type, and when dealing with "digital media" in general— contains two fundamental elements, which, both conceptually and practically, correspond to two elements of computer programming: algorithms and data structures. When a user selects a particular tool from a menu and uses it on the part of the document s/he is working on, the algorithm behind this tool modifies the data structure that stores the content of the part." Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 219

[36] Manovich argues this 'projection' and the affect it has on those using the software in relation to working with media editing programs. Ibid. p. 222.

[37] Kolarevic, B. (2003). *Architecture in the digital age*. New York: Taylor & Francis. p. 12 ; Burry, J. and Burry, M. (2012). *The new mathematics of architecture*. London: Thames & Hudson, pp.38-39. ; Kilian, C. (2006). *Modern control technology*. Clifton Park: Delmar/Thomson Learning, pp.300-303.

[38] "When you make a parametric model, in fact, you are programming." see Kolarevic, B. (2003). *Architecture in the digital age*. New York: Taylor & Francis. p. 105.

[39] Fuller, M. (2008). *Software Studies*. Cambridge: MIT Press. p. 17.

Part II *What does it mean? What are the Implications?*

This section is a critical study of software as a design resource; tools and mediums. It seeks to explain the logic behind software tools and the properties of digital mediums. Individual Rhino tools will be assessed according to two criteria, the objective being to identify the ways in which Rhino simulates real-world design techniques that existed prior to software and, vice versa, the ways in which processes inside software have moved into the realms of design.[40] The degree to which a tool has strayed from its original intended purpose will also be considered. This analysis formulates the design environment inside Rhino and casts light on the properties of software that influence the nature of design.

Chapter 3: Software Tools

"With softwarization, the implicit possibilities and different ways of using physical tools are made fully explicit. "Artistic techniques" and "means of expression" ... are given explicit and detailed controls. Like all computer programs or functions, they now come with many parameters." (Manovich, 2014)

Parameters are integral to software programming; they enable programmers to break down a complex task into separate functions. A repetitive sequence can be described by a single function and then evoked within the program by its name as often as needed.[41] Functions that perform conceptually related tasks, such as visualizing geometry/objects, are collected in libraries. Rhino largest library is the (NURBS) Library.[42] It contains mathematical formulae for generating and

[40] Parametric modelling is "more similar to programming than to conventional design" Weisenberg, D. (2008). *The Engineering Design Revolution*. [ebook] ch.16, p.12. Available at: http://www.cadhistory.net/toc.htm [Accessed 7 Jan. 2017].

[41] This modulation of large programs is called Procedural Programming.

[42] Non-uniform Rational Basis Spline; In the 1960s, extensive work was performed in the aircraft, automotive, machine control and electronics industries for 3D (3-dimensional) construction, and for NC (Numerically Controlled) machine programming and design. The work performed relied on the development of mathematical expressions for polynomial curves and surfaces. Following development at Citroen automotive company, Steven Coons (MIT, Ford Motor Company) had a vision for interactive computer graphics in CAD. In pursuit of this, he expanded the B-Spline model technology to be both rational and non-uniform (this enabled sharp corners within the spline by placing multiple knots in one location.) Boeing combined rational Bezier curves and non-uniform B-splines into a new geometric model; Non-Uniform Rational Basis Splines (NURBS). Initial

representing curves and surfaces. These formulae contain variables for which values must be assigned, for this reason tools have parameters that control aspects of the output.[43] These are projected to the GUI in the form of 'Options'.

In Rhino, these options have a default values which can be changed by the user. Deviation from the default value ranges from a single specified alternative described by a word (i.e. flat) to any 4-digit number (i.e. tolerance = 0.001). For example, the algorithm behind a tool that translates a network of curves into a surface ('NetworkSrf') has a number of variables with defined parameters. The tool requires that the curves be sorted into two directions (to provide values for variables u and v), the user can opt for 'AutoSort' (evoking the default value for the parameter) or 'NoAutoSort' (allowing the user to manually control this value). Once the curves in each direction are defined, a window with more options (set to default) appears. 'Tolerances' for the 'Edge Curves', 'Interior Curves' and the 'Angle' (surface normal) require a numeric value. 'Edge Matching' where each edge (ABCD) can be specified to match the input curves with less accuracy (Loose) or with the continuity of the input geometry (Position/Tangency/Curvature). Use of the tool requires eight explicitly defined relationships; 'creativity' within the rules.

The return value of this instruction is a NURBS surface. If the user were to split this surface 'SplitSurface' he/she would be evoking a function from a different library.[44] The distinction of these libraries is not obvious at GUI level because they are always invoked through Rhino as an intermediary. Rhino uses wrapper classes to manage the process of invoking a function.[45] This provides a degree of abstraction, thereby removing the user from the implementation of the tool. Rhino is the intermediary facilitator, it authorizes parameter values for the functions it implements and locates the stored data on which they should operate. The software process is invisible to the user, their access to the tool procedure is limited to 'Options'.

Graphics Exchange Standard (IGES) adopted NURBS and it became the common language for digital graphics exchange between CAD software. See Bezier, P. (1998). A View of the CAD/CAM Development Period. *IEEE Annals of the History of Computing*, 20(2), pp.37-40. And Farin, G., Hoschek, J. and Kim, M. (2002). *Handbook of computer aided geometric design*. Amsterdam: Elsevier, p.10.
In 1991 AutoCad absorbed AG (CAD software for General Motors) and turned to McNeel to integrate their NURBS library into AutoCAD via plugin. The success of this plugin, encouraged Mcneel to create a standalone program; Rhino. Hence the use of NURBS.

[43] For example; A formula required to draw a graphical circle; $(x-h)^2 + (y-v)^2 = r^2$ where h (displacement from the x axis), v (displacement from the y axis) and r (radius or the circle) are variables with parameters.

[44] McNeel's own geometry library; Troutlake

A reminder of the research questions mentioned in the introduction: Is it an extension of a real-world tool or does it have no physical precedence prior to software, and to what extent is it a product of another software application?

An instance of computational method becoming a design technique are the Boolean Tools. Boolean logic is used to define binary states (true or false).[46] An AND gate (e.g. if this and that occur, do this) or an OR gate (eg if this or that occur, do that) is used to obtain a result for further processing.[47] In Rhino, Boolean operations identify the interaction between two closed objects, by testing whether or not a point on one object is inside the boundary of another object. The Boolean tools essentially pass data through these gates, and depending on the variation of the Tool; Difference (subtracting one volume from the other), Union (adding one volume to the other) Intersection (define the intersection of two volumes into new geometry) and Split (split and close solids at intersections), return a new closed surface/polysurface. See fig 2.0. There is no precedence for this technique outside of software, however, the use of Boolean methods in software applications is widespread (particularly search engines). A conceptually similar function exists in Photoshop; one can 'SubtractFromShapeArea' (difference), 'AddToShapeArea' (union), 'IntersectShapeArea' (intersection) using a defined shape. Unlike other Rhino tools such as 'Fillet', 'Join' or 'Explode' one cannot 'Boolean' an object in the real world; *the Boolean concept is inherent to computing* and thus only 'works' with digital mediums.

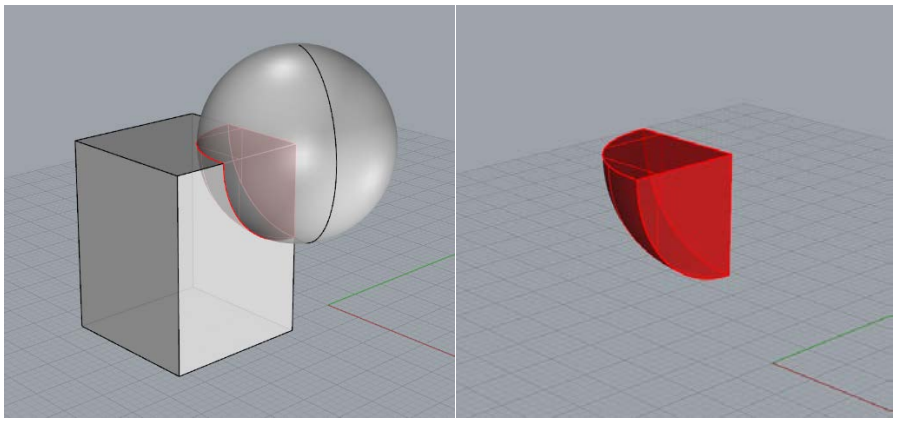

Figure 2.0 Rhino V5 Screen Shot: 'BooleanIntersection' a. Method b. Result

Software tools augment real-world tools, such that they take on new properties.[48] For example, the

[46] 'Boolean' is a primitive built in type of C++ programming.

[47] WhatIs.com. (2017). *What is Boolean?* Available at: http://whatis.techtarget.com/definition/Boolean [Accessed 12 Mar. 2017].

[48] Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 110.

polyline tool is the software equivalent of physically drawing a line between two points. A physical line, however, has a real-world scale and cannot sensibly be more than a few meters. Rhino's polyline is not limited by scale, it can be of any length, constrained by other geometry ('Osnap'), have absolute precision and, if desired, default to a closed circuit.[49] These properties reflect those of software; data is scale-less, it can 'converse' with other data and has explicit values. In truth, *only a fraction of a physical line-drawing device is replicated by the polyline tool*. This can be said of all Rhino tools that virtualize physical tools.

The output of the 'Polyline' tool is also an imitation. Lines have no thickness; they are paths between explicit points, and thus do not have 'something' along the entire length of the line. The visual representation of the software line has a thickness, but the representation is a process secondary to the process of drawing it. Rhino uses a different programming language to 'visualize' the line then it does to 'construct' the line, this means the two methods are alien to each other.[50] This is not the case in the physical world, where drawing a line is the same process as representing that line and puts something (lead/ink/chalk) at every point along it, which in turn gives it thickness. This absence of 'something' is true for all geometry in Rhino; surfaces have no thickness and solids are enclosed empty spaces. Geometry inside Rhino takes on the characteristics of data; it is essentially an instruction waiting to be processed by another software procedure.[51]

Most tools in Rhino require input geometry to operate on another piece of geometry.[52] Although a designer may use lines, stencils or guides in the real-world to assist with the operation of a tool,

---

[49] 'PersistentClose' is a tool option which can be enabled to make the polyline closed (connecting the last marker to the first).

[50] Rhino's GUI is written in .Net programming language whereas the Polyline Constructor tool is written in C++ programming language.

[51] "Fundamentally, computers follow a sequence of instructions they are given in the form of data. A set of instructions to perform a given task (or tasks) is called a "program". In the nominal case, the program, as executed by the computer, will consist of binary machine code. The elements of storage manipulated by the program, but not actually executed by the CPU, are also data. Program instructions, and the data that the program manipulates, are both stored in exactly the same way. Therefore, it is possible for computer programs to operate on other computer programs, by manipulating their programmatic data." En.wikipedia.org. (2017). *Data (computing)*. Available at: https://en.wikipedia.org/wiki/Data_(computing) [Accessed 10 Mar. 2017].

[52] In v5, 470 of the 512 tools that act on geometry cannot function without another piece of geometry. These include 'Split', 'Trim', 'Extrude', 'Divide', 'Offset', 'Boolean' and many more. See Docs.mcneel.com. (2017). *Command list | Rhino 3-D modeling*. Available at: https://docs.mcneel.com/rhino/5/help/en-us/commandlist/command_list.htm [Accessed 19 Apr. 2017]. And Developer.rhino3d.com. (2017). *Rhino.Geometry Namespace*. Available at: http://developer.rhino3d.com/api/RhinoCommonWin/html/N_Rhino_Geometry.htm [Accessed 19 Apr. 2017].

he/she can also use the tool 'freely'. For example, hands can fold paper without a line, scalpels can cut card without a stencil. It is not possible to use the software equivalents without these guides. In Rhino, the designer begins to think of geometry as integral to a tool, more so than he/she would in the physical world, where it is primarily for representation.[53] A piece of card might be a used as a stencil, but it cannot simply 'Extrude' itself or 'BooleanDifference' with another volume. In the real-world geometry has a physicality which, by default, represents something. In software, geometry can 'become', or be used for the creation of, something else. Geometry is required for the operation of a tool.

"Digital computers allow us to represent any phenomenon or structure as a set of variables. In the case of design and animation software, this means that all possible forms— visual, temporal, spatial, interactive— are similarly represented as sets of variables that can change continuously. This new logic of form is deeply encoded in the interfaces of software packages and the tools they provide."
(Manovich, 2014)

In Rhino, all geometry can easily change type (ie. point to line, line to surface, surface to volume). Furthermore, different types of geometry can do similar things (ie; a line, surface, and volume, can 'Split' another volume; points, lines and surfaces can be 'Patched' to make a new surface, a line or surface can be 'Extruded' to make a new volume and so on.) Such affinity does not apply to real-world geometry which have more distinct types, cannot easily switch form and require very different tools for conceptually similar operations. In Rhino, joining lines is the same as joining surfaces (physically, one would require a pencil, the other a fixative). This metamorphosis of type is enabled by the underlying software; all geometry is data, it is structured according to a format and can therefore be processed similarly. A surface in a Rhino model can define a plane to 'Project' onto, a 'Section' to cut or an area to 'Divide' as well as a representational element (wall, floor, ceiling etc). In summary, Rhino greatly extends the capabilities and functionality of real-world tools and geometry (both its existing and potential ability), yet it dissolves the distinction between them.[54] The activity of design is somewhat muddled and turns in on itself; using closely related geometry with closely related tools, and blurring the role between geometry as part of a tool process or as something representational.

[53] To clarify, this analysis applies to the design process and should not be confused with an architectural blueprint describing a building for which the design has already been 'set'.

[54] "Within the computer metamedium, all previously existing and newly invented mediums share some common properties - ie. the rely on a set of common software techniques for data management, authoring and communication." L. (2014). *Software takes command*. New York: Bloomsbury.p. 123.

Like any other tool, the ‘Polyline’ tool is an algorithm that takes some inputs and generates some outputs by applying a number of computations to these inputs. Particular input values do not affect the functionality of the algorithm nor the number of steps required for its execution. Data has no scale and algorithms aren’t particular about input values.[55] When a real-world concept is simulated by a Rhino tool, it takes on a similar nature to all other rhino tools. Consider a tool with more ambiguous origins; ‘Array’. An array has an architectural manifestation; an array of columns, windows or railings are all repetitions of the same module separated by a specified distance. In computer programming, an array is a data structure in which each module stores an integer or variable of the same type.[56] Although an array is a real-world concept, the computer extends and augments it.

[55] “Software is a codification of a huge set of behaviors: if this occurs, then that should happen, and so on…” George Stepanek. (2012). *Software Project Secrets: Why Software Projects Fail.* Apress, pp.10-11.

[56] www.tutorialspoint.com. (2017). *Computer Programming Arrays* Available at: https://www.tutorialspoint.com/computer_programming/computer_programming_arrays.htm [Accessed 8 Mar. 2017].

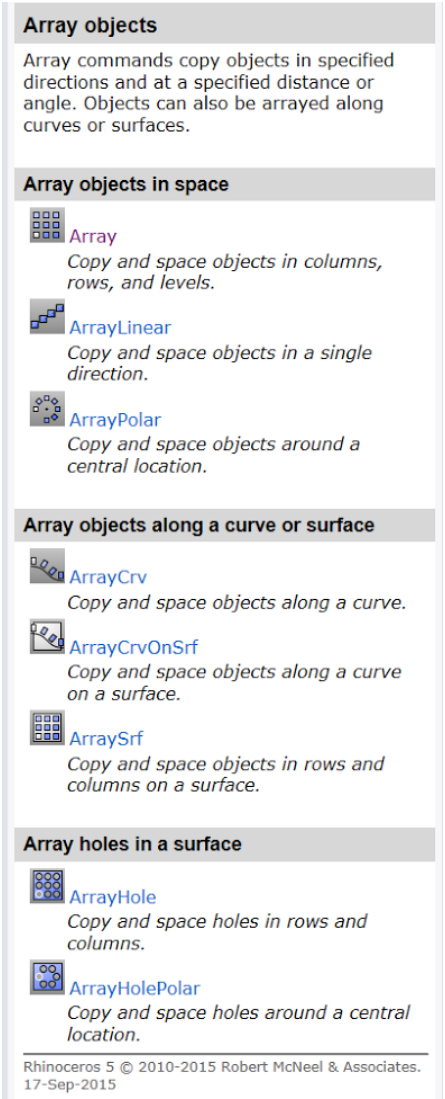

Fig 2.1 Rhino V5 Help Panel 'Array Objects'

The Rhino 'Array' tool can distribute objects at precise distances along a curve or array a grid of objects *on* and perpendicular *to* an uneven surface, furthermore, before setting the spatial separation of these objects (defined by a click of the mouse) the infinite possible outputs can be previewed continuously on the screen by moving the mouse closer to or further away from the reference point. This is a visual representation of the software process; inside Rhino, real-world constants are exchanged for variables with continuous variability, a common trait to all tools. The designer can quickly 'test' the outcome to his/her liking. Interaction with this continuously changing spatial form is likely to influence the user's way of thinking about form, and arguably why dynamic, complex and fluid forms emerged in the 90s as Architects began to use CAD software.[57]

[57] Notably the Lewis Residence by Frank O. Gehry & Associates. See Creators. (2017). *The Fathers of Digital Architecture Are Reunited In a New Exhibition*. Available at: https://creators.vice.com/en_uk/article/the-

The 3D model has a manipulability that far exceeds that of its physical counterpart. It exists in a dynamically changing world exempt from physical repercussions.[58] Further still, the computer model can be visually manipulated at ease and with incomparable precision; a user can zoom and move through the model uninhibited. Architects are no longer drafting flat representations of buildings, they're experiencing their creations inside software, and thus through the lens of software.[59] Rhino, as a design environment, is characteristic of software elements; digital data and programming functionality.

The logic of Rhino is the logic behind every tool, it cannot compute a foreign logic. Physical tools don't have such closely related methods; cutting is very different to drawing, casting is very different to fixing. *A designer uses all kinds of logic to manipulate physical tools, whereas in Rhino, all tools obey the logic of software.*

fathers-of-digital-architecture-are-reunited-in-a-new-exhibition [Accessed 19 Apr. 2017].

[58] "Initially, a parametric definition was simply a mathematical formula that required values to be substituted for a few parameters in order to generate variations from within a family of entities. Today it is used to imply that the entity once generated can easily be changed." Yessios, C., Bonn, M. and Jordan, W. (2003). *Form-Z RenderZone form-z RadioZity*. Columbus, Ohio: Autodessys, p.263. And "Design is change. Parametric modelling represents change" Woodbury, R. (2010). *Elements of Parametric Design*. Oxford: Routledge 2010. p. 7.

[59] There is much discussion regarding representation of architectural ideas. Architect Michael Graves (an architect and an emeritus professor at Princeton) asks "What has happened to our profession, and our art, to cause the supposed end of our most powerful means of conceptualizing and representing architecture?... The computer, of course." Graves, M. (2017). *Opinion | Architecture and the Lost Art of Drawing*. Nytimes.com. Available at: http://www.nytimes.com/2012/09/02/opinion/sunday/architecture-and-the-lost-art-of-drawing.html?pagewanted=all [Accessed 7 Apr. 2017].

"My central argument is that the relationship between design and reality is undergoing a shift from representation to simulation and that this shift has many profound implications for architecture." See Sheer, D. (2014). *The Death of Drawing, Architecture in the Age of Simulation*. Oxford: Routledge. p. 20.

Chapter 4: Software Mediums

"The unique properties and techniques of different media have become software elements that can be combined together in previously impossible ways… The computer metamedium" not only contains "a set of separate mediums, it also contains a larger set of smaller building blocks that unite to create hybrids." (Manovich, 2014)

In the computer "metamedium", techniques and mediums prior to software are combined with data manipulation techniques and data formats within software. Due to the compatibility of software and data, innovation in techniques and mediums converge; the design process becomes one of allocating different 'parts' of the design to different software applications. Each 'part' is worked on in a different way according to the type of data specific to the program (e.g 3D modelling, scripting, rendering, image editing etc), although *always in and through software*. The design process is divided, each stage orientated toward a different representation. The result is a compilation of software techniques and data types; media hybrids.

File formats are fundamental to the storage and access of data. A particular format is a layout for the organization of data in a particular structure.[60] Files can only be read if the software application knows how to read them, i.e. data in a Rhino file (.3dm) is structured in a particular way so that Rhino can gain access to and edit it. Standardization of file formats is an essential condition for the use and manipulation of files across various applications. File formats legible to software applications must stay relatively constant or else the user's files would quickly become illegible/obsolete. Although new tools can be added in Rhino fairly regularly (with each new software release), the structure of the data they operate on is more stable.[61]

During a session of Rhino, all geometry data is stored and managed by the core (Rhino.Exe).[62] Selecting a piece of the geometry, equates to selecting a piece of data within a structure.[63] If a

---

[60] WhatIs.com. (2017). *What is file format?* Available at: http://whatis.techtarget.com/definition/file-format [Accessed 5 Feb. 2017].

[61] Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 216.

[62] Rhino Forum conversation with Steve Baer. See McNeel Forum. (2017). *Rhino software architecture*. Available at: https://discourse.mcneel.com/t/rhino-software-architecture/33170/18 [Accessed 2 Mar. 2017].

[63] This Data Structure can be traced back to the first CAD software (APT). In response to US Air Force requirements programmers at MIT's Servo Lab formulated a comprehensive multi-axis machining technique for a Numerically Controlled Milling Machine. The process of programming the machine manually on a reel of punched tape had transitioned to a program on a computer. This was enabled by a subroutine technique which evolved into a procedural programming language (APT). Douglas Ross was responsible for the software data

particular type of geometry can be isolated from a range of other geometries (e.g. 'SelectSurfaces'), there must be something about the data for this geometry that makes it identifiable as a surface. Rhino can select geometry by type, boundary, history, attributes and state, because it knows the structure of its stored data and the procedures for 'calling' this data (an 'Address' indicative of these criteria).[64] Therefore, properties of mediums depend on the application they 'exist' in. Their data is structured according to the way the software stores and accesses it. For example, Rhino can export an image of a 3d model enabling it to be accessed by photoshop. The file format changes so that the data is primed for a particular type of manipulation.

Similarly, data can be restructured to improve its processing speed. The 'SurfaceToMesh' tool, translates a NURBS surface into a mesh[65]. This way of describing 3D geometry is used for rendering and animation.[66] See fig 2.3.

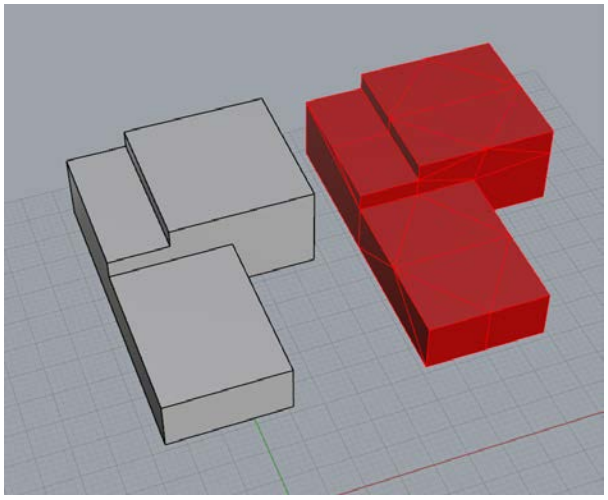

Figure 2.3 Rhino V5 Screen Shot: 'SurfaceToMesh'

Hybridity is built into Rhino's software architecture. Plug-ins, are part of the Rhino 'jacket', these

structure which enabled the user of a CAD system to select an item for further operation. See Reintjes, J. (1991). *Numerical control: Making a New Technology*. New York: Oxford university press. p. 78-81.

[64] Sutherland developed a 'ring' like structure; a string of pointers (pieces of data that point to other pieces of data) that eventually closed back on itself. This was key to Sketchpads application software; it enabled the user to insert new elements arbitrarily anywhere in the displayed drawing (eg.point and click); to remove elements and have the software close the logical gap created by that operation (e.g close curve); to merge several data files (eg.blocks or nested symbols); to perform auxiliary operations on the data in either forward or reverse order (eg.untrim or edit blocks) and to set geometric constraints (eg. orthogonal/planar/Osnap).

[65] In computing, a mesh is a network of processors (See Glossary "Mesh") this technique has become a method of describing surfaces.

[66] 3D meshes are required by artists using commercial suites such as Maya and 3D studio max.

are software components which add features to a program. They enable third-party developers (authors of different software species) to easily add features to Rhino. Rhino operates independently of the plug-ins which makes it easy to add and update them dynamically without needing to make changes to Rhino itself.[67] Unlike most applications, Rhino does not merely provide a platform for plugins, they are part of the development model.[68] This allows tools from other software species to migrate easily into Rhino and enables plasticity in design across multiple applications.[69] Plug-in categories include; 20 for import and export; 20 for rendering and visualization; 8 for animation; 15 for drafting and illustration; 71 for analysis and simulation and many more.[70] Another breed of plug-in are the scripting plug-ins.[71] Theses gain access to Rhino, the core libraries and even other plug-ins through the RhinoScript plug-in. Grasshopper, a graphical scripting plug-in, takes hybridization a step further. There are 170 plug-ins which use the Grasshopper plug-in as a platform to access Rhino. Essentially, *if a tool can be written for one software application (whether it be programming, animation or visualization etc), it can be implemented in Rhino through a plug-in*. A designer in Rhino has the ability to use tools originating in all kinds of professions, including those otherwise unrelated to design. When designing a building, an architect uses techniques originating in programming, animation, web design and media compositing. Such plasticity is impossible in the real-world, where every medium has a different form and cannot be reformatted or made compatible with a tool.

Rhino has some tools which are designed to work with data formats different to its native type such as Bitmaps. This enables design using different types of data, created in other software programs. For example, Rhino may be used to create a 3D model onto which a rendering software 'maps' textures, effects and environment etc. to create a 'photorealistic' representation of the model. Vice versa, an image created in another software application can become a 3D model in Rhino. 'HeightField', for instance, creates a NURBS surface based on grayscale values of an image or 'BackgroundBitmap' provides an image for tracing. Import/Export tools mean a model can easily be exported as a .ai (for graphical editing), a .3ds (for rendering) or .stl (for 3D printing). Not only does the product (drawings, models, designs etc.) express the properties of

[67] A common proverb in the software industry states that you can easily shoot yourself in the foot with programming, but you can take your whole leg off with C++. The plugin model 'protects' Rhino from this type of damage.

[68] McNeel Forum. (2017). *Rhino software architecture*. Available at: https://discourse.mcneel.com/t/rhino-software-architecture/33170/18 [Accessed 2 Mar. 2017].

[69] The 'CrossCad' plugin, for example, enables Rhino to import and export more CAD file formats.

[70] Food4Rhino. (2017). *Food4Rhino*. Available at: http://www.food4rhino.com/ [Accessed 10 Apr. 2017].

[71] The RhinoScript plug-in implements and extends the Microsoft Visual Basic Scripting language at the front end, while tapping into all the core Rhino resources at the back end.

multiple software applications, *it also expresses the hybridity of modern software* (seen in a drawing compiled of vectors, textures and photographs).[72]

These media hybrids are a product of software 'craftsmanship'; *the orchestration of design through different software applications each orientated toward a different type of representation according to its native data type*. The result will not be a .3dm (Rhino File) but an image format such as Jpeg, PNG or Bitmap. This is in part due to the fact that software has not yet 'come up' with a way to easily access and 'show' 3D models, although there are clearly innovations in this direction (Google glass and Sketchfab use virtual reality to move around 3D spaces), the image is still the most popular form of architectural representation. In turn, *the 3D model becomes a means to create an image, build a physical model, or generate an animation*. Models in Rhino become images in other software applications. *The hybrid nature of design in software aids the production of an image*.[73]

[72] Media Hybridization is a term used by Manovich for media created by multiple software applications. Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 161.

[73] "The Aestheticization of the world induces a form of numbness. It reduces any level of pain to the seductive image. What is at risk in this process of aestheticization is that political and social content may be subsumed, absorbed or denied. The seduction of the image works against any underlying sense of social commitment… The world becomes aestheticized and anaesthetized." Leach, N. (1999) The anaesthetics of architecture. Cambridge, Mass.: MIT Press. p.45. And, for general discussion, see Rattenbury, K. (2002). *This is Not Architecture: Media Constructions*. Oxford: Routledge.

Part III *Where is it going? Where is it taking us?*

This section will explore the condition of the digital model and the evolutionary direction of Rhino software. The intention is to show how, through software, the process of design is becoming more 'softwarized'. An evaluation of Rhino's development in relation to the wider software industry, will expose the direction of change and its causation. This will lead to an analysis of the impact software's trajectory has on design and speculation over its future.

Chapter 5: Editability and Extendibility

The layer feature is common to CAD, media editing and GIS software. In Rhino, this allows you to work on one/some parts of the model without disturbing the rest.[74] One might assume that layers are similar to tracing paper or acetate, but they are much more than an overlay or visual filter, *they redefine the way a designer thinks about a model. A single entity becomes disembodied parts*. Building skins can be thrown off, walls flattened, details hidden and so on. Instead of working with an indivisible whole, where each change would immediately (and in the case of a physical model; irreversibly) alter the design, a designer now plays with, edits and modifies a collection of elements. If another layer is visible but 'Locked', the 'Current Layer' geometry can be referenced against the 'Locked Geometry', he/she is free to manipulate the model without consequences.[75] The concept of a layered model (an aggregate of enclosed groups), connects to the general principles of C++ programming; assembling data, controlling access to that data and protecting the data from outside interference.[76] In Rhino a designer's method of working becomes more like the software process.

---

[74] "Layers are a way of organizing objects so you can manipulate them all at once or keep track of them in some way. When objects are on a layer, you can turn them all off at once, change their wireframe display color, and select them all with one selection." Docs.mcneel.com. (2017). *Layer | Rhino 3-D modeling*. [online] Available at: http://docs.mcneel.com/rhino/5/help/en-us/commands/layer.htm [Accessed 3 Feb. 2017].

[75] Manovich, L. (2014). *Software takes command*. 1st ed. New York [u.a.]: Bloomsbury. p. 142.

[76] See Glossary 'Encapsulation' and 'Data Hiding'. Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse. It led to Data Hiding concept see
https://www.tutorialspoint.com/cplusplus/cpp_data_encapsulation.htm
Inheritance makes it easier to create and maintain an application, see
https://www.tutorialspoint.com/cplusplus/cpp_inheritance.htm

Blocks[77] provide another degree of editability. Nesting is a computing term used to describe the placement of one or more objects within another.[78] Blocks, the C++ equivalent to 'Nested Symbols', are logically connected statements surrounded by braces, treated by the compiler as if it were single statement. They can be invoked by this statement at any point in the program.[79] The Rhino 'Block' tool is conceptually the same; geometric elements of any kind can be grouped together to form a symbol and then placed anywhere in the model at different sizes and orientations, editing the original symbol edits them all. A user does not need to be certain of the final form this 'Block' might take, rather he/she can distribute the blocks at will and then 'test' a variation on this block at any stage of the design process. Unlike layers, Blocks can be placed anywhere in the model (even inside other blocks), at any scale and on any layer whilst containing elements from different layers. If the layer attributes change, the block definition is unaffected. The contained geometry can remain 'in sync' with its layer, despite being implemented in another. Changing the layer attributes of the block, will not affect geometry within the block. These characteristics are paramount to their function in programming; they contain multiple statements but are effectively a single statement.[80] Again, *software method becomes design technique*.

Editing existing geometry in Rhino is slightly different to the typical tool method. A user can turn on the control points (markers that describe a piece of geometry) of an object and change their location. An object's control point equates to part of its data stored in the core. Rather than constructing or operating on a piece of geometry (like 'Box' or 'Split') you operate on the control point/s which describe this geometry. Each control point is weighted according to the governing parameter.[81] Rhino features local support meaning a single control point only influences those intervals where it is active. It allows distortion of a surface at specific locations.[82] The user has

---

[77] Also known as nested symbols. See Glossary 'Blocks'. Sketchpad was the first software to feature graphical Blocks.

[78] Computerhope.com. (2017). *What is nest?*. [online] Available at: http://www.computerhope.com/jargon/n/nesting.htm [Accessed 3 Feb. 2017].

[79] Learn C++. (2017). *4.1 — Blocks (compound statements)*. [online] Available at: http://www.learncpp.com/cpp-tutorial/41-blocks-compound-statements/ [Accessed 4 Apr. 2017].

[80] Learn C++. (2017). *4.1 — Blocks (compound statements)*. [online] Available at: http://www.learncpp.com/cpp-tutorial/41-blocks-compound-statements/ [Accessed 4 Apr. 2017].

[81] See Glossary 'Weighted Function'. A weight function is a mathematical device used when performing a sum, integral, or average to give some elements more "weight" or influence on the result than other elements in the same set. The term *rational* in NURBS refers to these weights. En.wikipedia.org. (2017). *Weight function*. Available at: https://en.wikipedia.org/wiki/Weight_function [Accessed 3 Apr. 2017].

[82] En.wikipedia.org. (2017). *Non-uniform rational B-spline*. Available at: https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline#Control_points [Accessed 4 Apr. 2017].

access to the data structure, he/she can change member values by changing the variable inside the member function (control point). Variable members with variable members.

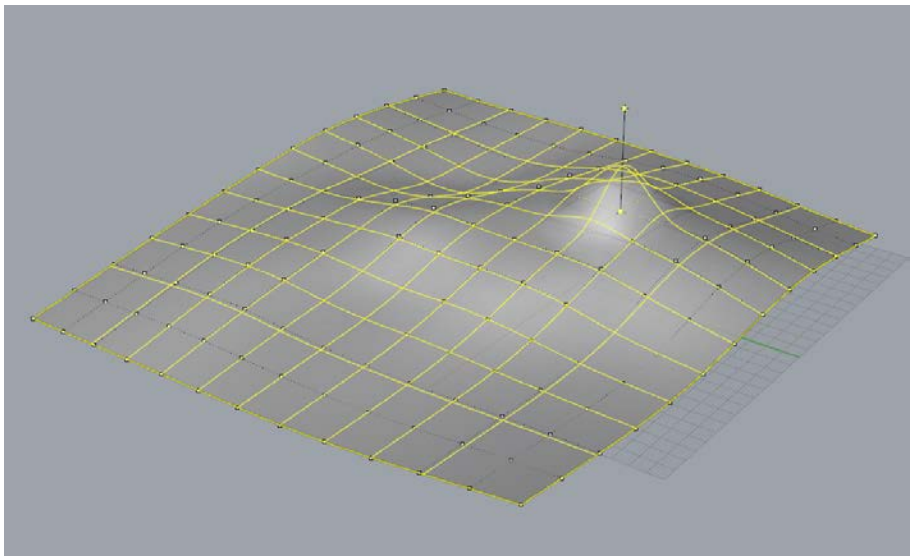

Figure 3.0 Rhino V5 Screen Shot: Control Points of a NURBS Surface.

Due to the structuring of data, the real-world predicament of making irreversible design decisions is absent in Rhino. *All configurations are editable and every operation is undoable*, geometry can even be 'Unjoined', 'Untrimmed' or 'Unwelded'. This transitions to the designer's cognitive process; he/she can test something that would have a risk associated to it in the real-world. The digital model can always be edited, one's design 'History' can be resurrected at any time and thus, according to Lunenfeld, it exists in a "state of perpetual beta"[83]; *in a constant testing phase, never finished and always susceptible to further editing*.

In the process of design this "encourages endless tweaking rather than a commitment to the discrete project"[84]. Lunenfeld goes on to describe this digital condition as "an aesthetic of unfinish, an understanding both literal (the work is never out of beta) and metaphoric (the digital is always in flux flitting about and flickering on the grid)"[85]. No elements of the digital model are set, no conclusive decisions have to be made, the entirety of the model could be edited by editing a single element. Combined with its ability to move from application to application*, as it exists inside software, a design is permanently extendable*.[86]

[83] Lunenfeld, P. (2011). *The secret war between downloading and uploading*. Cambridge, Mass.: MIT Press. p. 36.

[84] Ibid. p. 35.

[85] Ibid. p. 136.

[86] Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 157.

Chapter 6: Software Development

Design techniques enabled by/through/in Rhino are a result of more than just software paradigms but also the practice of software development. Decisions made by individuals and companies who develop software are heavily influenced by culture, client's, economics and competition.[87] However, according to Marx, culture and social actors alone cannot account for the history of a particular technology.[88] *Technological autonomy holds the ubiquity, "interwoveness" and power of our existing technological systems responsible for technological process.*[89] Rhino is not exempt from these forces. Its creators and their choices are subject to the wider technological environment. These factors coalesce in the establishment of software protocols and conventions (such as commands mentioned previously built into all modern GUI's). When a particular tool which appears in one application, becomes popular with its users, this tool can 'spread' into other applications and sometimes take on a new function.[90] The presence of tools in applications can also be a result of software industry economics, such as when one software company buys another and merges its existing product with the newly purchased one.[91]

The extent of tool migration can be seen in all types of software. Emoji's have spread from a single model of Japanese mobiles to almost all phones and social media apps. A list of possible 'Effects' are present in all image editing software, one of which, 'Noise', mimics the degradation of communication signals.[92] Tweening, is a technique used in virtually all types of animation. The software tool generates intermediate frames between two images to create a visually smooth

---

[87] Lecture by IBM Fellow. Booch, G. (2016). *The History (and Future) of Software*.

[88] Marx "technological determinism". See Winner, L. (2001). *Autonomous technology*. Cambridge, Mass.: MIT Press. p. 83.

[89] Ibid. p. 74.

[90] Although visual layers have become a common feature in Adobe Products and all CAD software, mostly for the purpose of editing, they were first implemented in GIS maps and used for spatial analysis. The GIS system is used by many software applications including Google Maps and equivalents. GIS Geography. (2017). *The Remarkable History of GIS - GIS Geography*. Available at: http://gisgeography.com/history-of-gis [Accessed 2 Mar. 2017].

[91] Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 148.
This was the case for, Rhino (an offshoot of AutoCad). Economic and internal dynamics surrounding AutoDesk, led to purchasing of Sun Systems in an effort to expand their consumer market. They approached Rhino because they needed help integrating the newly purchased NURBS library into AutoCad. Weisenberg, D. (2008). *The Engineering Design Revolution*. [ebook] p.12. Available at: http://www.cadhistory.net/toc.htm [Accessed 7 Jan. 2017]. ch. 4. p. 8.

[92] WhatIs.com. (2017). *What is noise?* Available at: http://whatis.techtarget.com/definition/noise [Accessed 15 Apr. 2017]. A similar mutation occurred in Rhinos evolution. Vector graphics displays were first used for air defense systems, followed by GIS, computer games, automobile design and eventually Rhino.

transition from one image to the next.[93] This tool has migrated into Rhino. The 'TweenSurfaces' tool generates an 'in-between' surface that is halfway (in form and position) between two formally different surfaces, in other words; a transitional stage from one surface to the other. *Animation technique becomes a Rhino technique*.[94]

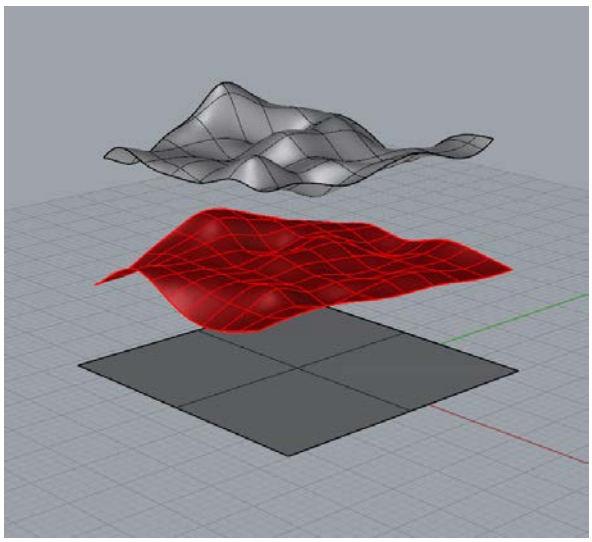

Figure 3.1 Rhino V5 ScreenShot: 'TweenSurfaces'

In summary, software techniques and mutations are written into applications by groups of people who refine and expand it in response to its users and the applicable market. Manovich aptly describes the vehicles behind software "the result of intellectual ideas by pioneers working in larger labs, the actual products created by software companies and OpenSource communities, the cultural and social processes set up when many people and companies start using it, and the software market forces and constraints."[95]

Software updates allow Rhino to remain in Lunenfelds 'perpetual beta state'; new functionality can be added and changes made indefinitely. The rate of development behind the interface is undetectable to the low-level user. The first version of the software (v1) is unrecognizable to the

[93] Webopedia.com. (2017). *What is Tweening?* Available at: http://www.webopedia.com/TERM/T/tweening.html [Accessed 19 Mar. 2017].

[94] By this I mean the conceptual operation of a tool.

[95] Manovich, L. (2014). *Software takes command*. New York: Bloomsbury. p. 149. Note Alan Turing's influence: The Stored Program Computer, conceived by Turing, was the first instance of hardware with a layer of software on top; it stored instructions for controlling the machine inside the computer's data store (memory). Turing demonstrated the abilities of this Universal Machine; one that could simulate a large network of other machines and perform any task for which a program could be written. This was the basis for software.

programmer working on today's version (v6), however the GUI is relatively consistent.[96] Unknown to the user, the software itself is rapidly changing. With each update the user may notice adjustments or improvements (such as bug fixes and new tools), however their overall legibility of the software is retained. This corresponds to a shared belief in the software industry, that "when done well, software is invisible"[97].

Most applications provide plug-in support for 3rd party developers, Rhino goes a step further.[98] Rather than adding code to Rhino itself, Rhino programmers use the same tools as 3rd party programmers. They prefer to write a plug-in. This reduces collateral damage in case of a slip up.[99] This model provides a way of adding 'foreign' features to Rhino without disturbing the resource itself. Plugins must be Rhino legible but can have new behaviours unfamiliar to Rhino. *They can be less 'like' Rhino and more 'like' another software applications.* This strategy of software development contributes to hybridization. Furthermore, *with each software update, the compatibility of Rhino is extended*. "Rhino 5 includes 9 new import/export file formats, and more than 50 new compatibility enhancements"[100]. Evidently, Rhino's development is cultivating software 'craftsmanship'. Computer-aided design is not limited to one species of software, rather, *it has become a process of harnessing the resource provided by one software application to aid the next stage of design using another application*.

[96] According to "dale" and "wim" on the Rhino Forum. See McNeel Forum. (2017). *How was/is Rhino created?*. Available at: https://discourse.mcneel.com/t/how-was-is-rhino-created/41445/2 [Accessed 18 Feb. 2017].

[97] According to Bjarne Stroustrup, the designer and original implementer of C++. Wired.com. (2017). *C++ Adds to programming*. Available at: https://www.wired.com/2010/10/1014cplusplus-released/all/1 [Accessed 27 Feb. 2017].

[98] Technical term; "eating your own dogfood"

[99] Software is difficult to 'fix''. A common proverb in the software industry states that you can easily shoot yourself in the foot with programming, but you can take your whole leg off with C++. It also means that the SDK (Software Development Kit, used to build plugins) is rigorously tested internally and there is no need to maintain and support a separate product.

[100] New in Rhino 5. See Rhino3d.com. (2017). *Rhinoceros Feature Overview*. Available at: https://www.rhino3d.com/features [Accessed 3 Jan. 2017].

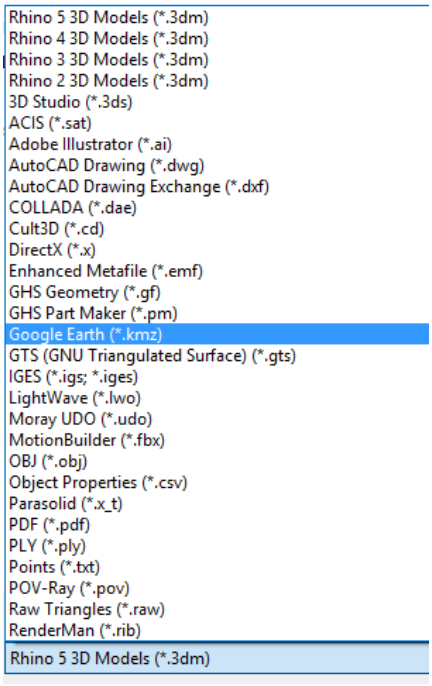

Figure 3.2 Rhino V5 Screen Shot: Export options including 'Google Earth (*.kmz)'

Rhino programmers claim to be neutral enablers, responding to the needs of the client and prioritizing their demands in the development process.[101] However, programmers are influenced by forces beyond their control. For instance, v5 introduced a new way of editing the model. The method; select an element of an object (curve, surface, solid etc.) manipulate (move, rotate, scale etc.) that element and in turn, distort the object accordingly. Similarly, 'CageEdit' allows all control points of an object to be manipulated. Other CAD software (SketchUp for instance) already had comparable methods for editing a model. Overtime, this technique proved popular and, to stay competitive, Rhino developers chose to integrate it. *Rather than neutral enablers, Rhino's authors are 'contaminated' by mutations and new species spawned out of technological autonomy*. This has implications for design. Essentially, elements that define geometry (points, edges and faces) are made editable, thereby making the 3D model *'hyper' editable*. There is even less reason to 'Delete' a mistake or remodel an unsatisfactory design, when all geometry can be 'tweaked' and 'adjusted' to one's liking. *The state of the 3D model is even less stable, less 'locked' and more 'unfinished'.*

The phenomenon of exaptation poses a difficulty for those who write software. When creating a tool, the programmer must be careful not to 'over design' it, thereby limiting its potential use. If a

[101] 'Error reports' (client feedback system) are how developers "prioritise their work". According to "Dale" on the McNeel Forum. (2017). *How was/is Rhino created?*. Available at: https://discourse.mcneel.com/t/how-was-is-rhino-created/41445/2 [Accessed 18 Feb. 2017].

tool can be used *for* more things and *on* more things, it will, by nature, be used more. The programmer must walk a line between providing a tool which will fulfil an operation, without hindering its future evolution by writing an overly restrictive method. This approach maximises tool potential and cultivates tool dexterity (the ideal traits for a tool to spread into other software applications). Rhino programmers are 'writing' hybridity into the software.

Rhino is becoming more like other software applications (in particular, 3D rendering programs such as 3DsMax and Maya). This is evident with the extensive addition of 'Texture Mapping' tools (required for rendering). 'Drafting' tools (including dimension styles, hatching, detail layers, a clipping plane and attribute options) have migrated from other CAD software programs (Microstation, Vectorworks, AutoCad etc.). Unlike the species model in the natural world, where different species evolve independently, *the software species absorbs the evolutionary traits of other software species, their trajectories are incredibly close*. With the addition of plug-ins, there is little sense to having two separate programs, one for rendering one for modelling, when the entire functionality of Keyshot, Artlantis, Maxwell Render (rendering software) has been engulfed by Rhino (facilitated by plug-ins). Tools established in Grasshopper are now Rhino tools (e.g. 'IntersectTwoSets' and 'RebuildUV'). Clearly, *software tools are infectious*; a plugin is a point of entry, if a tool is pervasive, it infects the host software. However, there is no 'immunity', Rhino has no limit to the number of plugins. Scripting plugins have also enabled a new way of designing tools; using software. Every aspect of design is being absorbed by software, this trajectory seems certain. In Rhino, designers' tools are essentially variations of functions and their mediums data structures, therefore, *the activity of design is immersed in the activity of software, tied to its evolution and defenseless from infection*.

Conclusion

Through software, the activity of architectural design becomes synonymous with the software process; using algorithms on data. The design environment is softwarized; already existing design techniques extended and new ones introduced. Tools require explicit values and follow software logical procedure. Mediums are 'unstable', their properties depend on their data structure and the application used to access them. The fundamental elements of design are changed, their properties are those of the underlying software.

Inside Rhino, designs have no physicality; all geometry (lines, surfaces and solids) are material-less. Consequently, the properties of materials are absent. Tools are not chosen for their ability to work with particular mediums (scalpel, saw, pliers), nor does a designer chose a medium (card, wood, steel) for its performance properties (strength, softness, density). An understanding for how a tool works and how different mediums behave is necessary in the real-word. This is not a requirement in Rhino, there exists a degree of abstraction between designer and his/her creation. *A designer can use the resource without having to fully understand it because software does the work for them*. Proficiency in Rhino depends on a different kind of knowledge. A carpenter's skill does not translate into software; *individual tools cannot be used 'well' or with 'expertise' because there is no variation in how an algorithm performs.*

All geometry is manipulated in a similar way, the method for which is defined by the software process; call a function, assign variables and follow a sequence of steps. With softwarization, diverse methods required to use specialized tools on relatively few mediums become analogous methods required to use unspecialized tools on many mediums. Contrary to the real-world, a tool is required for the most primitive manipulation of a model such as 'Move', 'Select' and 'Rotate'. Rather than mastering a handful of physical tools, designers must 'learn' how to operate a large number of tools to both navigate and realize their designs; *all interaction is through software.*

Software tools and techniques are not specific to mediums or profession, their real-world uniqueness is dissolved. However, different software applications have native file formats according to their purpose (3d modelling, rendering, image compositing etc.). Due to the hybridity of modern software, variations of the digital model can be exported/imported across these different applications. This significantly changes the properties of the medium and in turn its representational meaning.

Rhino is becoming more compatible and therefore more capable of generating hybrid designs. Its evolution cultivates software 'craftsmanship', whereby Rhino contributes to the design of an image or animation. Effectively, *design is orientated toward representation, rather than the building itself.* The designer is also artist, animator, programmer etc. with expertise in many software applications. This has significant implications for architects; software adeptness becomes imperative to the design process. *The skill of an architect or student is partially attributed to his or her proficiency in software craftsmanship.*

Due to the species model, techniques native to a particular industry (programming, film, and rendering) have become Rhino design techniques (boolean, tweening and texture mapping). Software practices (such as the structuring and protection of data) have become design practices (blocks and layers). Furthermore, Rhino's plug-in approach, increases the potential for exaptation and exposure to 'foreign' tools. This method of development makes Rhino permanently extendible. Potentially, all elements of the design process can be turned into algorithms or data. This has repercussions for how architects create and think about design; *their knowledge or skill is acquired inside software and therefore redefined.*

Rhino is subject to outside forces; evolutionary mutations and new species that emerge heavily influence its development. The contagious nature of software tools is, in part, responsible for Rhino's hyper editability. This means designs are not fixed; they exist on the verge of change, susceptible to further editing. *Editability fosters indecisiveness in the mind of the designer and influences the way in which digital designs are processed cognitively*. Like software, the digital model is in a perpetual state of beta; it's form unfinished and development indefinite.

Rhino software is deeply interwoven with other software species. Individual software applications are subject to the evolutionary trajectory of software as a whole. Developers can pluck tools out of the mix and recalibrate them. Each application is a selection of techniques with traits and origins common to other fields of use. In this way, Rhino becomes more like other applications; it has absorbed their functionality. Software, otherwise unrelated to design, can change the techniques used by architects to design. *Through CAD, Architecture hands over a degree of its autonomy; it is dependent on something beyond its control.*

Software manifests itself in the process of design by/in/with software. The human legible interface masks the activity beneath and removes necessity for designers to understand their tools. In CAD, designers have creative freedom only within the confines of software. Techniques conform to a

single methodology and are subject to a foreign agency. A designer's authorship is appropriated by software at both the local (programmatic) and global (autonomous) scale. As Feyerabend argues, "all methodologies... have their limits"[102], immersing the activity of design inside software has been proved limiting.

[102] Feyerabend, P. and Hacking, I. (2010). *Against method*. London: Verso. p. 32.

Glossary

Boolean - Boolean refers to a system of logical thought that is used to create true/false statements. A Boolean value expresses a truth value (which can be either true or false).
Boolean. (2017). Available at: https://www.techopedia.com/definition [Accessed 24 Apr. 2017].

Bitmap - A digital image composed of a matrix of dots. Each dot corresponds to an individual pixel on a display. Bitmap. (2017). Available at: https://techterms.com/definition [Accessed 23 Apr. 2017].

C++ - A general-purpose object-oriented programming (OOP) language. The main highlight of C++ is a collection of predefined classes, which are data types that can be instantiated multiple times.
C++ Programming Language. (2017). Available at: https://www.techopedia.com/definition [Accessed 24 Apr. 2017].

CAD - Computer-aided design. CAD. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Class - Classes are an expanded concept of data structures: like data structures, they can contain data members, but they can also contain functions as members. Class. (2017). Available at: http://www.cplusplus.com/doc/tutorial/classes/ [Accessed 24 Apr. 2017].

Command - An instruction or signal causing a computer to perform one of its basic functions. Command. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Command Line Interface - A command line interface (or CLI) is a text-based interface used for entering commands. Command Line Interface. (2017). Available at: https://techterms.com/definition [Accessed 23 Apr. 2017].

Command Object - An object that represents a command.

Constructors - A constructor is a special method of a class or structure in object-oriented programming that initializes an object of that type. Constructor. (2017). Available at: https://www.techopedia.com/definition [Accessed 24 Apr. 2017].

Command Pattern - the command pattern is a behavioral design pattern in which an object is used to encapsulate all information needed to perform an action or trigger an event at a later time. This information includes the method name, the object that owns the method and values for the method parameters. Command Pattern. (2017). Available at: https://en.wikipedia.org/wiki/ [Accessed 23 Apr. 2017]. **Receiver** - knows how to perform the operations. **Invoker** - asks the command to carry out the request. **Client** - creates an instance of an object and sets its receiver. Oodesign.com. (2017). Command Pattern Available at: http://www.oodesign.com/command-pattern.html [Accessed 23 Apr. 2017].

Core - a core is the processing unit which receives instructions and performs calculations, or actions, based on those instructions. A set of instructions can allow a software program to perform a specific function. Core. (2017). Available at: http://www.computerhope.com/jargon.htm/ [Accessed 23 Apr. 2017].

Data Structures - The form in which a collection of data is organized, typically allowing for efficient access or manipulation of the data. Data Structure. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Exaptation - The process by which features acquire functions for which they were not originally adapted or selected. Exaptation. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Exon - A segment of a DNA or RNA molecule containing information coding for a protein or peptide sequence. Exon. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

File format - A defined structure for the processing, storage, or display of data. Format. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Functions - A basic task of a computer, especially one that corresponds to a single instruction from the user. Function. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Graphical User Interface - a user interface that includes graphical elements, such as windows,

icons and buttons. Graphical User Interface (2017). Available at: https://techterms.com/definition [Accessed 23 Apr. 2017].

GIS - A geographic information system (GIS) is a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data. Geographic Information System. (2017). Available at: https://en.wikipedia.org/wiki/ [Accessed 23 Apr. 2017].

Interface - Refering to User Interface or UI - the means in which a person controls a software application or hardware device. User Interface. (2017). Available at: https://techterms.com/definition [Accessed 23 Apr. 2017].

Libraries - A collection of programs and software packages made generally available, often loaded and stored on disk for immediate use. Data Structure. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Medium - The material or form used by an artist, composer, or writer. Medium. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Members - Either data or function declarations within a body of declaration whose access is specified. Members. (2017). Available at: http://www.cplusplus.com/doc/tutorial/classes/ [Accessed 23 Apr. 2017].

Mesh - A computer network in which each processor is connected to a number of others, especially so as to form a multidimensional lattice. Mesh. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Null - When a variable has no value, it considered to be null. Having a null value is different than having a value of 0, since 0 is an actual value. Null. (2017). Available at: https://techterms.com/definition [Accessed 23 Apr. 2017].

NURBS - Non-uniform rational basis spline (NURBS) is a mathematical model commonly used in computer graphics for generating and representing curves and surfaces. Non-uniform rational basis spline (NURBS). (2017). Available at: https://en.wikipedia.org/wiki/ [Accessed 23 Apr. 2017].

Object - An object is an instantiation of a class. In terms of variables, a class would be the type,

and an object would be the variable. Object - Object. (2017). Available at: http://www.cplusplus.com/doc/tutorial/classes/ [Accessed 23 Apr. 2017].

Parameter - A numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation. Parameter. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Perpetual beta - Perpetual beta (or 'banana principle') is the keeping of software or a system at the beta development stage for an extended or indefinite period of time. It is often used by developers when they continue to release new features that might not be fully tested. Perpetual beta. (2017). Available at: https://en.wikipedia.org/wiki/ [Accessed 23 Apr. 2017].

Plug-in - A software plug-in is an add-on for a program that adds functionality to it. Plug-in. (2017). Available at: https://techterms.com/definition [Accessed 23 Apr. 2017].

Procedural Programming - Procedural programming is a programming paradigm that uses a linear or top-down approach. It relies on procedures or subroutines to perform computations. Procedural Programming. (2017). Available at: https://www.techopedia.com/definition [Accessed 23 Apr. 2017].

Program-code-template - A template is a C++ programming feature that permits function and class operations with generic types, which allows functionality with different data types without rewriting entire code blocks for each type. Template. (2017). https://www.techopedia.com/definition [Accessed 23 Apr. 2017].

Programming - The process of writing computer programs. Programming. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Programming Paradigm - A style or "way" of programming. Some languages make it easy to write in some paradigms but not others. Programming Paradigms. (2017). Available at: http://cs.lmu.edu/~ray/notes/paradigms/ [Accessed 23 Apr. 2017].

Real-world - The existing state of things, as opposed to one that is simulated. Real world. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

SDK Source Development Kit - . A software development kit (SDK) is typically a set of software development tools that allows the creation of applications for a certain software package. Source Development Kit. (2017). Available at: https://en.wikipedia.org/wiki/ [Accessed 23 Apr. 2017].

Script - An automated series of instructions carried out in a specific order. Script. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Sidebar - a sidebar is a user interface element that displays a list of choices. It typically appears as a column to the left of the main content, though it can appear on the right side as well. Sidebar. (2017). Available at: https://techterms.com/definition [Accessed 23 Apr. 2017].

Software Species - Software species are types of applications which share the same fundamental data structure and therefore designed to work on particular types of mediums. Examples: vector graphics editors, raster graphics editors, 2D animation and motion graphics software, 3D computer graphics software, sound editors, text processors and HTML editors. Manovich, L. (2014) Software takes command. New York: Bloomsbury. p. 177.

Software Tool - A operation available to the user in a software program such as 'Move'.

Software Hybrid - A software application in which techniques and representational formats of previous physical and electronic media forms and the new information manipulation techniques and data formats unique to the computer are brought together in new combinations. Manovich, L. (2014) Software takes command. New York: Bloomsbury. p. 176.

Softwarization - Manovich's term used to describe the translation of the real-world into (design techniques, communication, media etc.) into the domain of software. Manovich, L. (2014) Software takes command. New York: Bloomsbury. p. 164.

Software Process - The procedural process of routines and subroutines that occurs behind the interface and is seemingly 'invisible' to the user.

Tool method - The software procedure behind the operation of a tool.

Variables - A data item that may take on more than one value during the runtime of a program. Variables. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Viewport - A framed area on a display screen for viewing information. Viewport. (2017). Available at: https://en.oxforddictionaries.com/ [Accessed 23 Apr. 2017].

Wrapper - A wrapper encapsulates the functionality of another class or component. class?, W. (2017). Available at: http://stackoverflow.com/questions/889160/what-is-a-wrapper-class [Accessed 12 Apr. 2017].

Bibliography

Book

Bryant, L. (2014) Onto-cartography. Edinburgh: Edinburgh University Press.

Burry, J. and Burry, M. (2012) The new mathematics of architecture. London: Thames & Hudson.

Campbell-Kelly, M. (2004) From airline reservations to Sonic the Hedgehog. Cambridge, Mass.: MIT.

Carpo, M. (2011) The alphabet and the algorithm. Cambridge, Mass.: MIT Press.

Ceruzzi, P. (2003) A history of modern computing. London, Eng.: MIT Press.

Englebart, D. and English, W. (2003) 'A Research Center for Augmenting Human Intellect', in N. Wardrip-Fruin and N. Montfort (eds) The New Media Reader, London: MIT Press: 231-246.

Farin, G., Hoschek, J. and Kim, M. (2002) Handbook of computer aided geometric design. Amsterdam: Elsevier.

Feldman, C. (1968) 'Subsets and modular features of standard APT', in Proceedings of the Fall Joint Computer Conference. Thompson Books.

Feyerabend, P. and Hacking, I. (2010) Against method. London: Verso.

Fuller, M. (2008) Software Studies. Cambridge: MIT Press.

Fuller, M. (2003) Behind the blip. New York: Autonomedia.

Fuller, M. (2006) Softness: interrogability; general intellect; art methodologies in software. Århus: Center for Digital Æstetik-forskning.

George Stepanek. (2012) Software Project Secrets: Why Software Projects Fail. New York: Apress.

Goldstine, H. (1972) The computer from Pascal to von Neumann. Princeton, N.J.: Princeton University Press.

Heims, S. (1991) The cybernetics group. 1st ed. Cambridge, Mass.: The MIT Press.

Joasia, K. (ed.) (2006) Software Actions. In: Curating Immateriality, New York: Autonomedia.

Johnson, S. (1997) Interface Culture: How New Technology Transforms the Way We Create and Communicate. New York: HarperEdge.

Johnson, T. (1963) 'Sketchpad III A Computer Program for Drawing in Three Dimensions', in Spring Joint Computer Conference. Spartan Books.

Kay, A. (1990) 'User Interface: a Personal View', in B. Laurel (ed.) The Art of Human Computer Interface design. Reading, MA: Addison Wesley: 121-131.

Kilian, C. (2006) Modern control technology. Clifton Park: Delmar/Thomson Learning.

Knuth, D. (1992) Literate Programming. Stanford, Calif.: Center for the Study of Language and Information.

Kolarevic, B. (2005) Architecture in the digital age. New York: Taylor & Francis,.

Leach, N. (1999) The anaesthetics of architecture. Cambridge, Mass.: MIT Press.

Leach, N. (2002) Designing for a digital world. Chichester: Wiley-Academic.

Licklider, J. (1960) Man-Computer Symbiosis. IRE Transactions on Human Factors in Electronics, HFE-1(1), pp.4-11.

Llach, D. (2015) Builders of the vision. New York: Routledge.

Lunenfeld, P. (2011) The secret war between downloading and uploading. Cambridge, Mass.: MIT Press.

Lynn, G. (ed.), (2013) Archaeology of the Digital. Berlin: Sternberg Press.

Manovich, L. (2010) The language of new media. Cambridge, Mass.: MIT Press.

Manovich, L. (2014) Software takes command. New York: Bloomsbury.

Meyers, R. (ed.) (1995) Molecular Biology and Biotechnology: a comprehensive desk reference. New York: VCH.

McCarthy, J. and Levin, M. (1965) LISP 1.5 programmer's manual. Cambridge, Mass.: M.I.T. Press.

McWilliams, C. and Reas, C. (2013) Form + code in design, art, and architecture. New York: Princeton Architectural.

Noble, D. (2013) Forces of production. New York: Knopf.

Parsons, J. and Oja, D. (n.d.) New perspectives, computer concepts, 2014. [ebook]

Rattenbury, K. (ed.) (2002) This is not Architecture: Media Constructions, Oxford: Routledge.

Reintjes, J. (1991) Numerical control: Making a New Technology. New York: Oxford university press.

Rheingold, H. (1985) Tools for thought. New York: Simon [and] Schuster, Computer Book Division.

Ross, D. and Rodriquez, J. ‘Theoretical Foundations for the Computer-Aided Design System’, in Spring Joint Computer Conference. Baltimore: Spartan Books.

Sheer, D. (2014) The Death of Drawing, Architecture in the Age of Simulation. Oxford: Routledge.

Suchman, L. (1987) Plans and Situated Actions: The Problem of Human-Machine

Communication. Cambridge: Cambridge University Press.

Sutherland, I. (1963) ‘Sketchpad: A Man-Machine Graphical Communication System’, in Spring Joint Computer Conference, Baltimore: Spartan Books.

Teuscher, C. (2004) Alan Turing: Life and Legacy of a Great Thinker. Berlin: Springer.

Weisenberg, D. (2008) The Engineering Design Revolution. [ebook] Available at: http://www.cadhistory.net/toc.htm [Accessed 7 Jan. 2017].

Winner, L. (2001) Autonomous technology. Cambridge, Mass.: MIT Press.

Journal/Article

Bezier, P. (1998) ‘A View of the CAD/CAM Development Period’, IEEE Annals of the History of Computing, v.20, no.2: 37-40.

Eglash, R., Bennett, A., O'Donnell, C., Jennings, S. and Cintorino, M. (2006) ‘Culturally Situated Design Tools: Ethnocomputing from Field Site to Classroom’, American Anthropologist, v.108, no.2: 347-362.

Grier, D. (1996) ‘The ENIAC, the Verb ‘to program’ and the Emergence of Digital Computers’, IEEE Annals of the History of Computing, v.18, no.1: 51-55.

Kay, A. (1984) ‘Computer Software’, Scientific American, v.251, no.3: 52-59.

Kay, A. and Goldberg, A. (1977) ‘Personal Dynamic Media’, New Media Reader, v.10, no.3: 31-41.

Licklider, J. (1960) ‘Man-Computer Symbiosis’, IRE Transactions on Human Factors in Electronics, v.HFE-1, no.1: pp.4-11.

Mateas, M. (2005) ‘Procedural literacy: educating the new media practitioner’, On the Horizon,

v.13, no.2: 101-111.

Scaife, M. and Rogers, Y. (1996) ‘External cognition: how do graphical representations work?’, International Journal of Human-Computer Studies, v.45, no.2: 185-213.

Turing, A. (1950) ‘I.—Computing machinery and Intelligence’, Mind, v.49, no.1: 433-460.

Verplank, B., Canfield Smith, D., Irby, C., Kimball, R. and Harslem, E. (1982) ‘Designing the Star User Interface’, Byte Magazine, v.7, no.4: 242-284.

Lecture

Booch, G. (2016) The History (and Future) of Software.
https://www.youtube.com/watch?v=OdI7Ukf-Bf4
[Accessed 1 Feb. 2017].

Eisenman, P. (2013) The Foundations of Digital Architecture: Greg Lynn with Peter Eisenman.
https://www.youtube.com/watch?v=hKCrepgOix4
[Accessed 1 Feb. 2017].

Leach, N. (2013) IaaC Lecture Series 2013-2014 "Adaptation".
https://iaac.net/fall-lecture-series-2013-neil-leach/
[Accessed 2 Feb. 2017].

Manovich, L. (2015) Lev Manovich SVA Lecture.
https://vimeo.com/123215850
[Accessed 1 Feb. 2017].

Website

Archinect. (2017) ‘firms that use Rhino’
http://uk.archinect.com/forum/thread/1051/firms-that-use-rhino?ukredirect
[Accessed 18 Apr. 2017].

class?, W. (2017) ‘What is a wrapper class?’
http://stackoverflow.com/questions/889160/what-is-a-wrapper-class
[Accessed 12 Apr. 2017].

Cprogramming.com. (2017) ‘Tutorials - The Static Keyword in C++’
http://www.cprogramming.com/tutorial/statickeyword.html
[Accessed 2 Mar. 2017].

Docs.mcneel.com. (2017) ‘Layer | Rhino 3-D modeling’
http://docs.mcneel.com/rhino/5/help/en-us/commands/layer.htm
[Accessed 3 Feb. 2017].

En.wikipedia.org. (2017) ‘Command pattern’
https://en.wikipedia.org/wiki/Command_pattern
[Accessed 11 Feb. 2017].

En.wikipedia.org. (2017) ‘Data (computing)’
https://en.wikipedia.org/wiki/Data_(computing)
[Accessed 10 Mar. 2017].

En.wikipedia.org. (2017) ‘Weight function’
https://en.wikipedia.org/wiki/Weight_function
[Accessed 3 Apr. 2017].

En.wikipedia.org. (2017) ‘Non-uniform rational B-spline’
https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline#Control_points
[Accessed 4 Apr. 2017].

Food4Rhino. (2017) ‘Food4Rhino’
http://www.food4rhino.com/ [Accessed 10 Apr. 2017].

GIS Geography. (2017) ‘The Remarkable History of GIS - GIS Geography’
http://gisgeography.com/history-of-gis
[Accessed 2 Mar. 2017].

Graves, M. (2017) ‘Opinion | Architecture and the Lost Art of Drawing’
http://www.nytimes.com/2012/09/02/opinion/sunday/architecture-and-the-lost-art-of-drawing.html?pagewanted=all
[Accessed 7 Apr. 2017].

Iconeye.com. (2017) ‘Distribution warehouses - Icon Magazine’
https://www.iconeye.com/architecture/features/item/10477-distribution-warehouses
[Accessed 12 Mar. 2017].

Kay, A. (1993) ‘The Early History of Smalltalk’
Available at: http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html/
[Accessed 17 Apr. 2017].

Krishnamurti, S. (2006) ‘Programming Languages: Application and Interpretation’
Available at: http://www.cs.brown.edu/~sk/Publications/Books/ProgLangs/
[Accessed 17 Apr. 2017].

Learn C++. (2017) ‘4.1 — Blocks (compound statements)’
http://www.learncpp.com/cpp-tutorial/41-blocks-compound-statements/
[Accessed 4 Apr. 2017].

Learn C++. (2017) ‘7.1 — Function parameters and arguments’
http://www.learncpp.com/cpp-tutorial/71-function-parameters-and-arguments/
[Accessed 5 Mar. 2017].

Rhino3d.com. (2017) ‘Rhinoceros Feature Overview’
https://www.rhino3d.com/features
[Accessed 3 Jan. 2017].

Reimer, J. (2005) ‘A History of the GUI’
https://arstechnica.com/features/2005/05/gui/
[Accessed 6 Mar. 2006].

Techopedia.com. (2017) ‘What is a Command Line Interface (CLI)?’
https://www.techopedia.com/definition/3337/command-line-interface-cli

[Accessed 18 Apr. 2017]

Visualarq.com. (2017) ‘What is Rhino?’
http://www.visualarq.com/info/what-is-rhino/
[Accessed 18 Apr. 2017].

www.tutorialspoint.com. (2017) ‘C++ Classes and Objects’
https://www.tutorialspoint.com/cplusplus/cpp_classes_objects.htm
[Accessed 2 Mar. 2017].

www.tutorialspoint.com. (2017) ‘Computer Programming Arrays’
https://www.tutorialspoint.com/computer_programming/computer_programming_arrays.htm
[Accessed 8 Mar. 2017].

WhatIs.com. (2017) ‘What is Boolean?’
http://whatis.techtarget.com/definition/Boolean
[Accessed 12 Mar. 2017].

WhatIs.com. (2017) ‘What is file format?’
http://whatis.techtarget.com/definition/file-format
[Accessed 5 Feb. 2017].

Wired.com. (2017) ‘C++ Adds to programming’
Available at: https://www.wired.com/2010/10/1014cplusplus-released/all/1
[Accessed 27 Feb. 2017].

Webopedia.com. (2017) ‘What is Tweening?’
http://www.webopedia.com/TERM/T/tweening.html
[Accessed 19 Mar. 2017].

Rhinoceros Resources

Cheng, R. (2008) Inside Rhinoceros. Clifton Park, NY: Delmar Learning.

Cheng, R. (n.d.) Inside Rhinoceros 4. Clifton Park, NY: Delmar Learning.

Cheng, R. (n.d.) Inside Rhinoceros 5. Clifton Park, NY: Delmar Learning.

Developer.rhino3d.com. (2017) 'Rhino.Geometry Namespace'
http://developer.rhino3d.com/api/RhinoCommonWin/html/N_Rhino_Geometry.htm
[Accessed 19 Apr. 2017].

Rhino3d.com. (2017). Rhinoceros Feature Overview.
https://www.rhino3d.com/features
[Accessed 3 Jan. 2017].

Primary Research/Interview

McNeel Forum. (2017) 'How was/is Rhino created?'
https://discourse.mcneel.com/t/how-was-is-rhino-created/41445/2
[Accessed 18 Feb. 2017].

McNeel Forum. (2017) 'Command-line driven design software'
https://discourse.mcneel.com/t/command-line-driven-design-software/33081
[Accessed 18 Apr. 2017].

McNeel Forum. (2017) 'Rhino software architecture'
https://discourse.mcneel.com/t/rhino-software-architecture/33170/18
[Accessed 2 Mar. 2017].

Command: _Move

Point to move to <13.945> :