

Three-Dimensional Sketching within an Iterative Design Workflow

A Thesis

Presented to the Faculty of the Graduate School
of Cornell University

in Partial Fulfillment of the Requirements for the Degree of
Master of Science

By

Nicholas Cassab-Gheta

May 2019

© 2019 Andrés E Gutiérrez A
© 2019 Nicholas Cassab-Gheta
ALL RIGHTS RESERVED

ABSTRACT

Starting in the 15th century, intellectuals and theorists began developing the techniques and visual language necessary to study, understand, review, and communicate spatial concepts through drawings and models. Drawing and drafting on paper has since remained the generally preferred method of conceptual design exploration, and designers and architects have honed the necessary techniques to explore, understand, and represent three-dimensional space with this two-dimensional medium. Today, design and especially conceptual design, has evolved into a highly iterative process that usually starts with rough doodles, sketches, and other methods for expressive, quick, and inexpensive exploration before moving the work into a more precise and less forgiving environment: the computer. Digital models already have the ability to offer designers more insights through the rapid exploration of variation, even automated iteration, and powerful simulation and validation, as well as the ability to routinely and to easily generate drawings. Even still, rough physical models and drawings are widely encouraged by educators and professionals prior to entering the digital realm, and drawing on paper endures as the de facto method of choice for designers to record and develop their ideas. Our experience and research suggests that the main issues preventing designers from fully embracing digital methods as viable alternatives or even improvements on their traditional mediums, particularly in the early-phases of the design process, are the limitations of the customary input technologies and methodologies as well as the subsequent interaction and representation capabilities available with and of the inputted geometry.

Pen and multi-touch technologies have made big leaps in recent years though hardware specifications and costs still remain among the biggest hindrances; most importantly, currently affordable display surface areas are not sufficiently large or of adequate resolution for design type explorations. Our prototype 3-dimensional drawing platform was developed using a large format Microsoft Surface Hub, which we believe is the closest hardware solution currently available to an ideal high-resolution, multi-

touch, drafting board. Moreover, Windows 10's intrinsic cross-platform characteristics provide a framework for our prototype to also work on more inexpensive personal computers, such as Microsoft's Surface Book, while the larger format displays remain cost-prohibitive.

This thesis recognizes that a viable digital alternative to drawing on paper must look and feel good and must support pen and multi-touch input. Of course, the full range of possibilities that pens, pencils, markers, and paper mediums offer designers are very difficult to accurately simulate and render digitally, especially when drawing is considered a personal experience that can vary greatly from designer to designer. While we acknowledge that our platform will not be able to provide the full range of possibilities available in expression through traditional drawing, we believe that the added benefit of a more integrated workflow will overcome some of the shortcomings in expression and demonstrate a clear path for future research and development.

Drawing plays different roles throughout the different stages of the design process and is also used to move ideas and intentions back and forth between those stages; from understanding others' existing designs, to recording and developing one's own ideas, to voluntarily misreading and reimagining those ideas. Trace paper or "red-lining" are popular common tools that allow designers and architects to draw and redraw over previous sketches or detailed drawings as they iterate, reference, and rethink past design decisions. This thesis demonstrates a more integrated and therefore more powerful way of drawing that recognizes the iterative nature of the design process and the apprehensions preventing designers from embracing digital drawing. By providing a flexible framework for iterative exploration within a 3-dimensional context, we demonstrate a valuable design solution and workflow that is not currently accessible by designers; one that blurs traditional distinctions between drawing and modeling.

This thesis proposes that drawing can be more powerful in 3-dimensions, while acknowledging that 3d modeling is not drawing. We propose a 3d drawing tool, built over an existing 3d modeling ecosystem, which allows designers to do drawing, but within a 3d environment. More specifically, we have developed a system that allows 2-dimensional and 3-dimensional sketching to happen within a 3d scene, one that

intrinsically provides a basis to transition back and forth more fluidly and intuitively between a hand drawing and a virtual model. We feel strongly that the best solution is one that works with existing tools that designers already find comfortable and that are an essential part of their workflows and processes. We therefore built our prototype 3d drawing platform (Cuttlefish 3D) over a popular 3d modeling software ecosystem (Rhinoceros 3D) that has a powerful feature set, a low cost of entry, and a passionate and dedicated user and developer base. In this way, a working drawing can directly become a working digital model and therefore be further and fully explored and developed within powerful 3d modeling software. We further acknowledge that most continued design explorations are part of an iterative process and that any viable solution should therefore provide the designer the ability to then sketch over and within a working digital 3d model and scene.

BIOGRAPHICAL SKETCH

Nicholas Cassab-Gheta was born in New York City on May 22nd 1991. He graduated from the prestigious Stuyvesant High School in 2009 and was admitted to the Bachelor's of Architecture program at Cornell University. While an undergraduate student, Nicholas engaged in research in 3D Printing, eventually publishing a paper in the journal 3D Printing and Additive Manufacturing. He was a president of Thumbnail, an editor of The Cornell Journal of Architecture Issue 9, and an editor-in-chief of Association Volume 6. Nicholas graduated from Cornell with honors, receiving the Charles Goodwin Sands Memorial Medal for best undergraduate thesis. Nicholas has worked for numerous Architecture and Design offices in New York City, including the office of Peter Eisenman, Volley Studio, and HDR.

This work is dedicated to my mother Marcia Cassab.

ACKNOWLEDGEMENTS

We would like to acknowledge everyone who helped make this Thesis a reality. Jenny Sabin, thank you for being a wonderful advisor, teacher, and friend. I hope to live by your example, to be incredibly curious, and incredibly caring. Andrea Simitch, your leadership within the Cornell community has been one of the most inspiring things to look up to. Don Greenberg, thank you for being a maverick, and for believing in mavericks. Hurf Sheldon, thank you for providing support for the lab, both emotional and technical. Joe Kider, thank you for showing me that being a jack of all trades is possible. Martin Miller, thank you for always keeping things light, and always being able to make me laugh. Caroline O'Donnell, thank you for being my first friend at Cornell. Cindy Bowman, thank you for always being incredibly helpful. Jan Allen, thank you for putting up with our constant e-mails. Mom, thank you for listening, and for being supportive no matter what. Dad and Raluca, thank you for being there for me.

I would also like to acknowledge those who were along for the journey but are no longer with us. Taylan Cihan, you were an inspiration to us who are still trying to draw the link between sound and image. Arthur Ovaska, your kindness and your wisdom were two of the greatest gifts I've ever received. Kevin Pratt, you have no idea how much your example influenced and inspired my interests and my career.

Lastly, I'd like to thank the Graduate School at Cornell, and all the people behind the scenes who support a wonderful and inspiring community to be a part of.

TABLE OF CONTENTS

ABSTRACT.....	i
BIOGRAPHICAL SKETCH	iii
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES.....	ix
1 OVERVIEW.....	18
1.1 Overview of Implementation.....	19
1.2 Thesis Structure	20
2 BACKGROUND.....	22
3 RELATED WORK.....	25
3.1 2-Dimensional Software.....	26
3.1.1 Adobe Illustrator	26
3.1.2 Adobe Photoshop	29
3.1.3 Sketch.....	31
3.1.4 Made with Mischief.....	32
3.1.5 Morpholio Trace.....	35
3.2 3-Dimensional Software.....	37
3.2.1 SketchPad+ by Moreno Piccolotto and Michael Malone.....	37
3.2.2 Collaborative Design Platform – Violin Yanev Thesis	38

3.2.3	T.E.D.D.Y. (Draw in 2D – Output in 3D)	39
3.2.4	3D Model-Assisted Sketching Interface	40
3.2.5	Napkin Sketch Surfacing in 3D.....	41
3.2.6	Rhonda Forever	42
3.2.7	Field 3D Freehand Drawing.....	43
3.2.8	I Love Sketch / Everybody Loves Sketch.....	44
3.2.9	uMake	45
3.2.10	Mental Canvas.....	48
3.2.11	DDDoolz.....	50
3.2.12	CATIA Natural Sketch	52
3.3	Other Notable Mediums	56
3.3.1	Sketch Furniture by FRONT.....	56
3.3.2	3-D Printing	57
3.3.3	Gravity Sketch.....	57
3.3.4	Tilt Brush.....	59
3.3.5	Oculus Quill and Oculus Medium.....	60
4	HUMAN COMPUTER INTERACTION.....	62
4.1	Hardware for Architectural Design.....	63
4.2	Interaction Design for Pen and Touch Interfaces.....	67
4.3	Touch Screen Devices	70
5	SOFTWARE LANDSCAPE.....	71

6	CHARACTERISTICS OF A SKETCH.....	79
6.1	Understanding Drawing	79
6.2	Visual Reasoning: Drawing is a Language.....	81
6.3	Characteristics of a Sketch	87
7	ADDING A 3RD-DIMENSION TO THE SKETCH.....	89
7.1	Important Features	89
7.1.1	Creating geometry in an empty scene.....	90
7.1.2	Creating geometry from existing geometry	93
7.2	Implementation	97
7.2.1	Creating Surfaces.	99
7.2.2	Selection and Snapping.....	99
8	RESULTS	101
9	FUTURE WORK	112
9.1	Introduction to Improvements	112
9.2	Improvements in Outputs	113
9.3	Improvements in Inputs.....	115
9.4	Visual History.....	116
9.5	Connecting to Distribution	118
9.6	Future of CAD	120
10	CONCLUSION	121

LIST OF FIGURES

Figure 2.1 A sketch over a classic picture of the original Cornell architecture studios where architects are working with traditional methods in an empowered way through the use of modern technologies and proposed techniques..... 22

Figure 3.1 Two screenshots of Illustrator showing the production of a graphic image through drawing at two different points in time. The screenshot on the left shows the underlying sketch work used by the artist to outline rough contours and the screenshot on the right illustrates how the artist creates final image with variable line qualities. 26

Figure 3.2 Screenshot of the Photoshop interface showing an .obj mesh file being painted on. On the left hand side is the unrolled mesh that allows for 2-dimensional painting on a 3-dimensional form shown on the right-hand side. Can input in either 2D or 3D space. 29

Figure 3.3 Example of Mischief in use. Notice the designer first drew a ghosted and rough preliminary sketch (a) before adding any detail or shading to the drawing (b)..... 32

Figure 3.4 Example of *Mischief* artwork by Carly Sanker demonstrating the power of ADF Technology. Here you can see the detail at different levels of zoom showing the scalability, infinite canvas, and sharp line strokes. 33

Figure 3.5 Marketing image of the Morpholio Trace application shows the user picking between different line thicknesses to apply over an initial sketch of a 2 dimensional representation of a 3d diagram... 35

Figure 3.6 Sketch planes are able to be placed to allow for input into a 3-dimensional scene. 37

Figure 3.7 A translucent layer resembling trace paper is used in the interface as a way for the user to toggle between drawing and camera manipulations. 38

Figure 3.8 Screenshots of *uMake* show the ability to work symmetry, the ability to edit splines with control points, and some basic surface modeling functionality..... 45

Figure 3.9 Wireframe of *uMake* interface on an iPad..... 46

Figure 3.10 The screenshots above taken from a video on the Mental Canvas website show the same mental canvas sketch from two different orientations. This image describes the way strokes in the Mental Canvas system are placed on canvases arbitrarily placed in 3D..... 48

Figure 3.11 Typical progression through CATIA using Natural Sketch. First create mood boards and organize inspiration, then initial 2-dimensional sketches, followed by 3-dimensional sketches, massing models, and finally a refined 3d model..... 52

Figure 3.12 A view of the inspiration and ideation boards collaboratively created and used by designers in CATIA at the beginning of the design process. Notice that these boards include 3D drawings as well as images. 54

Figure 3.13 A view of CATIA Natural Sketch’s user interface. 55

Figure 3.14 On the left is an image demonstrating the process of drawing the sketches with motion capture technology and on the right are the 3d printed outcomes..... 56

Figure 3.15 Above is an image depicting the original prototype of the Gravity Sketch system which features an arbitrary physical plane and a stylus, the combination of the two allow the user to position a plane in 3d space and input points on that plane. The user here has created an outline of a pair of sunglasses using this technique. 57

Figure 3.16 A view of Google’s Tilt Brush, a 3d painting application for the HTC Vive 59

Figure 3.17 A view of Oculus Medium, a 3d sculpting application for Oculus Rift 60

Figure 4.1 Image of the future of Architectural Design superimposed over a historic image of Architecture students drafting in one of the first Architecture Studios at Cornell University. The image depicts touch screen interfaces of all sizes, as well as a mixed reality headset experience, with a touch screen display. 63

Figure 4.2 Photo of a man using the 2016 version of the iPad Pro with the corresponding Apple Pencil.

The device is larger than its predecessors, portable, higher resolution, and has a faster processor. The Apple Pencil has multiple sensors built in to measure tilt and pressure 65

Figure 4.3 Photo of a man drawing on the 2016 version of the Surface Book by Microsoft. The device has

a special hinge and detachable mechanism design that allows the screen to be folded backwards over the keyboard to allow for higher processing and touch/pen interface..... 66

Figure 4.4 Thumbnails of common gestures from Windows 8 app development guidelines provided by

Microsoft. They show how the difference in interactions changes the look and feel and positioning of menus on screens. 67

Figure 4.5 Microsoft’s Universal Windows Platform is designed to easily allow developers to design and

develop applications that work on devices of all sizes and interaction logic. The devices above increasing in size from the smartphone on the left to the Surface Hub on the right are all Microsoft devices. 68

Figure 4.6 Photo of a man using the Surface Studio by Microsoft which was released for the first time in

2016. The device is designed to be a non-portable desktop. It has more processing power, and a higher resolution and screen size. The stand for the monitor can pivot from a drawing position (shown on the left), to an upright position shown on the right. 70

Figure 5.1 Our application has different modes depending on which device you’re using it with. The image

above depicts a diagram that shows the “breakpoints” the points in which the look and feel of the software is different depending on the screen size and hardware..... 71

Figure 5.2 Here is an image of our application as a standalone app for a tablet device. The application is

lightweight and allows for some of the functionality while being much more portable..... 72

Figure 5.3 Here is an image of our application as a Rhino Plug-in. The Rhino Plug-in version of our application is best suited for large format touch and pen displays. This is because access to a keyboard and mouse on one of these displays is relatively difficult, and so all the controls are modified to take advantage of pen and touch. 74

Figure 5.4 Here is an image of our application as Grasshopper plug-in. The Grasshopper plug-in version of our software is most suited when the user has a fast computer, an upright monitor and a pen input device. This allows the user to interact with the pen based interface while at the same time being able to interface with Grasshopper..... 78

Figure 6.1 This figure shows the traditionally labor intensive drafting process that designers embraced using large format papers and tools like trace paper, rulers, and splines to create smoothly varying curves..... 80

Figure 6.2 The drawing on the left shows iterations on trace while the image on the right shows a number of design iterations spread out along the wall in order to discuss the many different options explored. 83

Figure 6.3 This figure shows a quick red-lined sketch over a conceptual Google SketchUp model. The sketch provides a wealth of information impractical to communicate through normal modeling techniques..... 84

Figure 6.4 shows a freehand sketch of the south façade of the Denver Central Library by Michael Graves 85

Figure 6.5 shows a drafted drawing of the south façade of the Denver Central Library by Michael Graves 85

Figure 7.1 Sketching on Orthogonal Planes in Virtual 3D space (a) shows a line drawing sketch on a horizontally oriented plane (grey). (b) The abstraction is augmented by sketching on the vertical plane

(grey) which has been interactively translated from the origin. (c) In a similar way, a sketched line of a second vertical plane is drawn. (This can be more easily seen in the video accompanying this submission). 90

Figure 7.2 shows how hand sketched drawings exist in a 3-dimensional space. An “orthogonal drawing tool” is shown here that helps the designer work with 2D sketches orthogonally-oriented within the 3D scene. Quadrant a shows this orthogonal drawing tool; quadrant b shows how the designer can select and draw on a specific plane; quadrant c shows how the designer can move the tool to a desired location within the scene and choose the desired plane to draw on; and quadrant d shows how the designer can draw on this different plane within the 3D environment. 91

Figure 7.3 shows how the designer can construct arbitrary 2D planes from existing geometry in the 3D scene. There are three options for doing this: Figure a shows how the designer can construct a new drawing plane by selecting a new origin for the existing plane; Figure b shows how the designer can construct an arbitrary drawing plane by specifying a new origin and a second point along the y-axis; Figure c shows how the designer can construct an arbitrary drawing plane by specifying a new origin, a second point along the y-axis, and a third point that specifies the plane’s orientation about the x axis. 92

Figure 7.4 Drawing on a Plane perpendicular to the Line of Sight. When the user orbits the camera, the drawing plane follows, allowing the user to draw in 3D without having to constantly shift and manipulate the drawing plane separately. All the new strokes in this mode are perpendicular to the view plane of the camera and a preset specified distance from the camera. (a) The interactive sketch is shown on the physical drawing surface. (b) This figure shows the relationship between the drawing surface and the observer location at the apex of the pyramid of vision. 93

Figure 7.5 This figure shows a drawing method that uses the camera to define the orientation of a temporary drawing plane and object snapping to define the temporary plane’s origin at a given z-value: quadrant

a shows the original spline drawn on the XY plane; quadrant b shows a new spline starting from one of the end points of the spline drawn in quadrant a; quadrant c shows how the spline drawn in quadrant b was created on a plane perpendicular to the camera direction with an origin at the snap point; quadrant d shows how additional splines can be made in this manner with the new spline snapping one or both endpoints to existing geometry in the scene. 94

Figure 7.6 shows 3 methods for surface creation: figure a shows the ability to extrude a spline to create a surface; figure b shows the ability to loft between existing splines to create a surface; and figure c shows the ability to interpolate a surface from boundary splines. 95

Figure 7.7 shows the ability to use an arbitrary surface as a drawing surface and to create new geometry by trimming the surface with the drawn geometry: section a shows the ability to select surfaces as drawing "planes"; section b shows a spline drawn on the chosen surface; and section c shows the resulting surface after the middle section has been trimmed with the newly-drawn spline. 96

Figure 7.8 shows how you can add perspective dependent drawings over existing geometry within the scene. These drawings can be layered and draw a parallel with how designers work with trace paper using traditional analogue tools. 97

Figure 8.1 User experiments with strokes of different colors in Cuttlefish on the large Microsoft display. 101

Figure 8.2 User experiments with Cuttlefish and fluidly goes back and forth between modeling with a keyboard and mouse, and drawing on the display. 102

Figure 8.3 In this set of figures the user is drawing using Cuttlefish and has changed the orientation of the view in order to draw a different section of the object. 103

Figure 8.4 This figure shows a user "red-lining" or editing on top of an existing 3D Model on a mobile Surface Book tablet. 103

- Figure 8.5** This image depicts two users collaborating on an upright Surface Hub. The user in the background has the pen and is able to make edits on the fly while the user in the foreground has enough space to point and demonstrate areas in need of improvement. 104
- Figure 8.6** The figure above depicts a section drawing of the Elbphilharmonie in Hamburg, by Herzog & de Meuron drawn in Cuttlefish. It is shown here to demonstrate how the user can quickly sketch a drawing like this with many variations in line weights..... 105
- Figure 8.7** The figure above depicts a plan drawing of the MAXXI museum designed by Zaha Hadid Architects in Rome. The drawing was made in Cuttlefish and shows how figurative sketches and splines are quick and easy to create. 106
- Figure 8.8** This figure shows a perspective section of the Ford Foundation Headquarters in New York City designed by Kevin Roche. This drawing was created in Cuttlefish and demonstrates how a user can quickly create a sketch from a single perspective..... 107
- Figure 8.9** This figure is an image of a Cuttlefish 3D drawing of the Eames Case Study House in LA. The drawing is a 3D drawing and can be rotated and reoriented to discover different views..... 108
- Figure 8.10** This drawing is a creative interpretation of the Seattle Public Library by OMA. The drawing starts with a detailed section of the building oriented in 3D space. The user then took this as a starting point and created simple floorplates extending out. 108
- Figure 9.1** Many inputs many outputs. One of the reasons why our prototype is successful is because it accesses a larger array of inputs giving paths to a larger array of outputs. We believe that the most straightforward way to continue improving on this is to keep adding inputs and outputs. 112
- Figure 9.2** The images above depict the same geometry rendered with 3 different shaders. The geometry was created using Tilt Brush and is very similar to the geometry we get when creating in our system.

This image is used to explain how shaders have an effect on the overall look and feel of a 3D scene.
..... 113

Figure 9.3 The images above depict three different analytical sketches. These are 2 Dimensional sketches that our mind interprets as being in 3D. There have been many attempts to create algorithms that do the same kind of interpretation to create 3D drawings and models from 2-Dimensional sketching. 115

Figure 9.4 The images above depict an explicit history 3D Modeling system in which each iteration or phase of design of the model is stored in a visual history of the model making process. This allows for versioning, this allows for a quick assessment of multiple iterations, and it allows designers and engineers to intelligently manipulate their creations. 116

Figure 9.5 An artist’s depiction of a Web VR experience. Notice how the geometry is a stylized lightweight geometry with a low number of polygons. This allows users experiencing the scene with low bandwidth to enjoy the immersive presence of VR without the frame rate dropping too significantly.
..... 118

Figure 9.6 An infographic depicting what Autodesk believes to be the future of CAD. There’s a large focus and emphasis on connection. The infographic explains that Building Information Modeling and CAD software will serve to connect people to the information of the project as well as the project itself.
..... 120

Figure 10.1 Above is a screenshot of another software for early stage design called 123D Design by Autodesk. It’s shown here to indicate how tools for early stage design are becoming more and more user friendly because there’s a demand to make software of these capabilities accessible to people of all ages and professions..... 121

Figure 10.2 This is an image of Frank Gehry’s original sketch for the Guggenheim Museum in Bilbao. It’s depicted here in order to show the importance of the flexibility and speed of sketching in the process of Early Stage Design. 122

Figure 10.3 An image showing 6 different view styles of the same 3d model of the Villa Savoye by Le Corbusier, depicted here to demonstrate the many uses and advantages that come from obtaining a 3D Model. This model can now be analyzed in a number of different ways and displayed in many different systems. 123

Figure 10.4 The image above shows an architect’s set of iterations presumably to explain how they arrived at the “best iteration described in the lower right hand corner. These types of images of an architect’s process are common in the architecture industry when critiquing early stage design. 124

Figure 10.5 Two side-by-side images of schematic design documents for the Morgan Library Expansion by Renzo Piano in New York. The image on the left is a plan drawing, akin to the type of drawing that would be made in our software and the image on the right is a physical model of the same space. They are shown here to demonstrate the importance of context in the architectural design process. 125

Figure 10.6 Using the advantages and aspects of trace paper and combining them with the 3 Dimensional needs of early stage design, we’ve created a software that is specific to architectural use. The image above is of Morpholio trace, another software created specifically for architects, it is shown here to describe the validation of these needs in the architectural profession. 126

1 OVERVIEW

In this thesis, we present a position that explains why a majority of designers in 2016, particularly architects, still choose pen and paper over computers to explore, develop, and review conceptual designs. We examine the needs of architects and designers through the lens of their processes and reveal that early-stage-design remains an area with a lot of opportunity for improvement. We recognize the importance of freehand drawing within various stages of the design process and the need for further integration of the tools used throughout and between these stages. We further acknowledge that in order to best contribute to the related fields, this thesis must embrace existing workflows and avoid “reinventing the wheel”. We do this by leveraging an existing software solution with a powerful feature set, a low cost of entry, and a passionate and dedicated user and developer base. “In this chapter, we provide an illustrated overview of the system through a sequence of images recorded during a typical design session. We follow this with a cursory introduction to the various components that make up the system and then describe the layout of the remainder of this thesis.”

1.1 Overview of Implementation

Our system was developed for the growing Microsoft Surface product lineup, which includes Microsoft’s Surface Hub, Surface Book, and Surface Studio models. This hardware allowed us to integrate pen and touch functionality into the architecture pipeline, which exists almost entirely within the Windows environment. Rather than attempting to independently fulfill every aspect of our vision, we focus our implementation on the areas of research that can provide the biggest impact to designers, particularly architects. Our implementation focused on improved user interaction and overall experience within existing design workflows. We chose a popular early-stage 3D design software, McNeel’s Rhinoceros 3D, as the hook into the design pipeline.

The “sketches” our users create exist in three-dimensional virtual space and are created by converting elements on arbitrarily-oriented planes and surfaces into NURBS definitions which are stored in a Rhino document. Consistent with the early stage design process, we have created our system in a way that mimics the traditional acts of and methods for drawing and sketching. Our system encourages ease of use and iterative design procedures with free-hand and pen-based sketching routines, alongside real-world metaphors, including functionality that enables multiple layers of virtual tracing paper. Although drawing is most efficiently enabled on a large, flat, horizontal or near-horizontal tilted surface, the actual “virtual tracing paper” can be rotated to enable sketching in three dimensions. In fact, using canonical solids, or even spline surfaces, the sketching need not be restricted to planar surfaces. Combining virtual layers enables the merging of components from various sketches.

Our system can be subdivided into four main parts: the core module, the input module, the rhino module, and the rendering module. The most important part of our implementation is the core module as it stores and manipulates all the geometry that is rendered through DirectX. This module also manages the current state of the application, including all input parameters (i.e. stroke type, pressure, width, color, etc.), drawing guides and planes, trace paper, and object visibility. The input module reads the strokes of the pen input as well as interactions with our GUI and calls the appropriate functions in the core module. The rhino

module allows the core module to communicate with and use Rhino functions such as spline and surface creation and manipulation, nearest point algorithms, and object state linking (i.e. selection). Finally, the rendering module takes data from the core module and renders it in our window with DirectX.

1.2 Thesis Structure

This thesis, the related research, and much of this writing is the result of a close collaboration between Andrés E Gutiérrez A and Nicholas Cassab-Gheta. While each author has unique parts to their research and writing, both share the majority of the work and chapters included herein. Andrés Gutiérrez focused his independent research on the important history of drawing in design, architecture, and communication and the foundation-shattering impact of being able create 3-Dimensional drawings that are experienced virtually and tangibly as spatial objects. Nicholas Cassab-Gheta focused his independent research on the potential for 3-Dimensional drawing to be extended throughout the digital landscape of existing tools available for design development; for example, alternative design tools can allow for additional input parameters from drawing input to be evaluated and modified algorithmically.

Chapter 2 is a review of the background behind the thesis. This is a description of the context, reasons, backstory behind some of the decisions made during the thesis. Chapter 3 is a comprehensive review of the existing state of software products related to Computer Aided Design, 3D Modeling, and the use of Pen and Touch Displays. Chapter 4 provides an overview of the history and current state of the art in human computer interaction, and how it relates to the difficulties behind designing a 3D Modeling interface around Pen and Touch. Chapter 5 is a description of the Software Landscape, and how that landscape changes and in effect changes the way this software is used. This software has a number of different configurations for users depending on what kind of hardware set up they have. Chapter 6 describes the characteristics of a sketch and its utility in conversation both from the standpoint of the speaker and the listener. Chapter 7 provides a detailed overview of our system. It begins with an overview of our implementation's aspirations and the reasoning behind why certain approaches were taken. It proceeds to review our system's core features in details and provides figures that show our system's interface and

examples. The chapter ends by describing the technical details in how the features were implemented. Chapter 8 provides an overview of how we user-tested our system, the results that came out of those experiments, and a critical analysis of our system in context of the experiment and its results. Chapter 9 describes areas of future work, and potential directions for this software to explore in the future. While Chapter 10 concludes the thesis with some final takeaways about what is gained from the prototyping of a software like this.

This proposal documents the state of the art in hardware and software capabilities and puts forward a new way of interacting with digital 3D content, whether you are creating something new, exploring an idea, modifying or annotating existing content, and more. The project describes why drawing and sketching, particularly in 3D, has not yet been fully embraced by designers across different age groups, disciplines, etc. The thesis focuses on the art, the science, and the language of drawing (and sketching) as the preferred medium of exploration and communication for many designers and then describes the opportunities that are available to digital sketching both through new hardware solutions and through new software approaches.

2 BACKGROUND

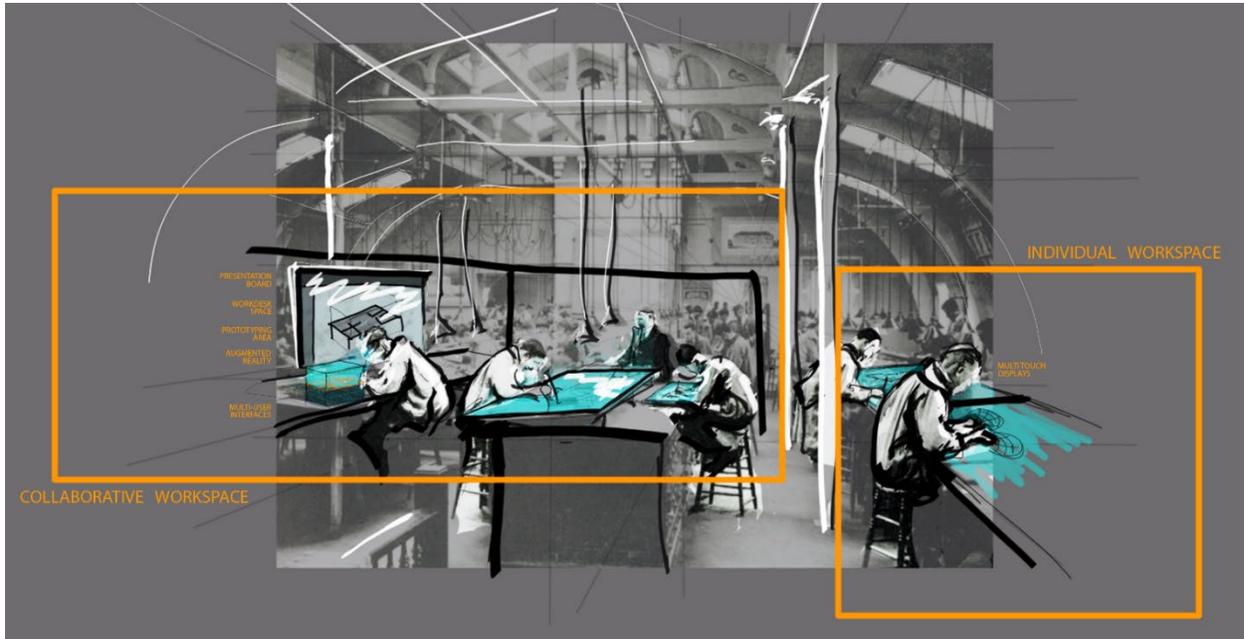


Figure 2.1 A sketch over a classic picture of the original Cornell architecture studios where architects are working with traditional methods in an empowered way through the use of modern technologies and proposed techniques.

From preliminary conceptual explorations to detailed specifications, the design process is currently segmented by the use of incompatible tools: traditional design methods including hand drawing and model making, software which constrains creative reasoning, and simulation technology that is expensive, difficult to use, and requires excessive specificity. With the improvements and proliferation of modern pen-input technologies and multi-touch displays (electronic drafting boards), we have the ability to maintain some of the best characteristics of hand drawing with the added benefits of computation, simulation, and 3-dimensional representation. Working hand drawings and working 3-dimensional digital models can now be blended and more closely integrated into a singular format that can be used to develop and communicate spatial expressions and renderings for aesthetic or functional judgements on one device. Beyond the theoretical revolutionary implications that 3-dimensional hand drawing capabilities have on our understanding of design, and more specifically the language of design (typically drawing), fully integrating

early explorations into systems that can automatically and continuously provide useful feedback and even suggestions through computational approaches promises to upend the way we think about and go about designing almost everything.

“To grasp the magnitude of the changes we face, it is important to realize that knowledge created with computer assistance goes well beyond classical knowledge formation—arising from computer processing of digital information resources on a scale that could not be achieved by all peoples of the earth acting in concert using all their cognitive powers.” – Constableⁱ

For many disciplines, mouse and keyboard technologies have been an adequate form of human computer interaction; the same is not true for design. Written languages have long provided people with a way of recording, formalizing, and communicating concepts, ideas, proposals, and more. In general, professionals like mathematicians, journalists, and businessmen have been able to integrate computers into their workflows through mouse and keyboard technologies with much less friction than designers. Where a mouse and keyboard works relatively well with languages based on text and numbers, it becomes very unintuitive and disconnected when working with graphic languages, for example drawing. Of course, the powers and benefits of computation have succeeded in persuading many designers to partake in the joys and woes of computer aided design, though we must acknowledge the histories of the disciplines and the mindsets of creative exploration, and realize that for a designer to connect with and reason through his or her work, there has to be a shift in the way he or she interacts with their designs through a computer.

“Indeed, as the next generation of photographic cameras will produce 3-D objects rather than 2-D images, and print out three-dimensional physical replications of the originals they portray, at any scale of our choosing – the primacy of two-dimensional projectival images, as we have known them since Alberti and Gaspard Monge, will soon be challenged by new technologies.” – Neil Spillerⁱⁱ

New possibilities for 3-dimensional interaction and understanding are on the horizon as a result of the ever-increasing rate of investment, development, and consequent convergence of a variety of technologies. Moore’s law has promised and delivered exponential growth in processing power for decades and graphics processors have been advancing at an even faster rate. Low cost, easily accessible, and

powerful sensors, cameras, flash storage, and touch screen displays have saturated markets as a result of the mobile revolution. Further increases in connectivity and communication, both interpersonal and mechanical, through continued growth in internet penetration, speed, and bandwidth continue to disrupt all aspects of modern civilization and the collection and organization of enormous amounts of data has yielded new insights, increased automation, and excitement and concern surrounding the prospects of artificial intelligence. Geometry capture technologies, propelled by Hollywood, virtual reality consumer products, and initiatives like Google's driverless car, have already begun dominating media channels and permeating markets and new 3d printing services, consumer products, and industrial prototypes continue to emerge at alarming rates. Unsurprisingly this list goes on; the point is that while these technologies have been advancing and developing in their own ways, their convergence begins to promise an imminent 3d revolution that will encompass a number of industries and that threatens to impact fundamental paradigms established as early as the renaissance.

As design becomes further democratized and every person gains the ability to capture, adapt, and reproduce 3-dimensional forms regardless of any technical or formal training, the purposes, processes, and even business models behind design, drawing, craftsmanship, and other associated fields are brought into question. This chapter and the sections within serve to examine and explain the current disconnect between the capabilities of design technology and the needs and desires of 3-dimensional designers, particularly architects. Moreover, this chapter provides the foundational knowledge necessary to properly consider the role of drawing in the world to come.

3 RELATED WORK

While not perfect, many of the hardware requirements needed for 3-dimensional digital drawing with pen and multi-touch input have become available and easily accessible over the past few years. These include flat large format displays, with fast accurate multi-touch technology, and the ubiquity of processing power in the form of high powered GPU's. As hardware options and specifications continue to improve and costs continue to decrease, there are a number of software developers that have taken advantage of the arising opportunities. Some of these developers are large corporations that have been developing graphics hardware for decades, others include startups and small teams pursuing academic research. This chapter reviews the current state of the art in related work and research and development trajectories.

There are three sections below. The first two sections are dedicated to software solutions for traditional computer hardware and tablets that take advantage of multi-touch and pen input. The related work reviewed in these first two sections is subdivided into software that is mostly intended for 2-dimensional graphics work in the first, and software mostly dedicated to 3-dimensional in the second. The third section in this chapter is not directly relevant to the pen and multi-touch 3d drawing implementation described in this thesis, but is nonetheless included because it offers perspective into the rapidly changing creation, interaction, and understanding of drawing.

3.1 2-Dimensional Software

A lot of very impressive 2-dimensional graphics software, for drawing and sketching has been created, packaged, and sold for many years. This thesis does not seek to be a replacement for these 2d platforms, but given that designers have always been trained in expressing ideas and forms through drawing in 2-dimensions it is important to understand the existing software solutions for 2-dimensional graphics. This section describes why and where designers and creative people enjoy the different existing platforms by evaluating their strengths, limitations, points of frustration, and aspirations.

3.1.1 Adobe Illustrator

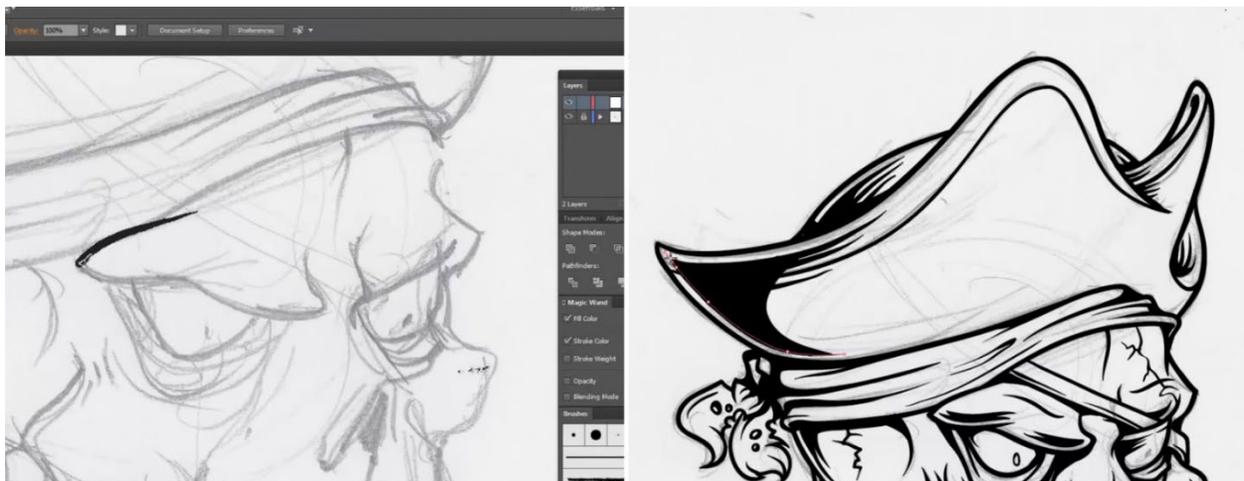


Figure 3.1 Two screenshots of Illustrator showing the production of a graphic image through drawing at two different points in time. The screenshot on the left shows the underlying sketch work used by the artist to outline rough contours and the screenshot on the right illustrates how the artist creates final image with variable line qualities.

Adobe Illustrator is a vector graphics editor currently in its seventeenth product generation. Originally developed in the late 1980s for Apple Macintosh as a ‘commercialization of Adobe’s in-house font development software’, *Illustrator* has grown to be the de facto vector graphics platform of choice for designers and artists working in a number of different fields.

Illustrator contains many different powerful features. This review does not attempt to provide a comprehensive evaluation of *Illustrator* within all the contexts that it can be used. Instead this review

attempts to draw focus to some of the application's key features that we believe are worth exploring within a 3-dimensional sketching context and communicates where some of the issues may lie in moving these features into a 3-dimensional space.

While it is true that *Illustrator* has some 3-dimensional capabilities, and may even appear to have some modeling functionality, it has not catered to exploring 3-dimensional form and space, but rather has focused on generating the illusion of 3-dimensional objects for a 2-dimensional scene. This distinction is quite important and is evidenced in *Illustrator's* inability to export 3d content in a way that other 3d applications can understand. The application does however offer three main features – rotate, revolve, and extrude/bevel – for the simulation of simple 3-dimensional models. Each one offers some perspective transformations as well as lighting and shading controls. Texture mapping is also enabled. One intended purpose of this semi-3d functionality is to allow 2-dimensional artists the ability to easily represent 3-dimensional objects within their scenes without having to draw them from scratch; consider an example where a designer creates a label for a wine bottle and wants to visualize his/her design wrapped around the bottle. The bottle and perspective distortion can be quickly and easily simulated using *Illustrator's* 3-dimensional techniques.

Beyond its limited 3d functionality *Illustrator* offers powerful 2-dimensional vector and spline functionality. The ability to add, remove, or change control points, to choose between different spline representations, and to map and render variable widths over the length of the spline, are but a few of the many features offered. Importantly *Illustrator* defines most of its splines using Bezier splines and is therefore able to read and edit 2-dimensional lines, vectors, hatches, splines, and other geometry from popular CAD file types and software. As a result of this ability to open, edit, and export 2-dimensional CAD files, *Illustrator* is widely used by architects and other 3d artists in the production and manipulation of drawings.

In exchange for a loss of control over the creation and/or modification of 3-dimensional forms, designers working within *Illustrator's* 2-dimensional ecosystem have much more control over the visual

representation and expression of their designs through drawing. One particular feature that outlines and greatly strengthens *Illustrator's parametric* spline representation is the user's ability to create their own "pen" or stroke styles. Through Adobe's Create Cloud suite and mobile apps ecosystem, a user can draw a particular stroke on paper, take a photograph of it with a mobile device and an Adobe app will transform it into a usable stroke for splines and vector graphics within Illustrator.

Finally, *Illustrator CC (Creative Cloud)* ships with a "Touch" workspace that behaves as its own miniature-program within *Illustrator* and is simultaneously quite intuitive and straightforward; most people familiar with *Illustrator* should not have a hard time grasping the basics. It modifies the traditional user interface by removing many features and simultaneously putting forward traditional and new *Illustrator* features for multi-touch and pen experiences. Despite there being far fewer buttons, *Illustrator's* "Touch" workspace works quite well by limiting its functionality to the key commands that are needed for drawing in the more traditional sense. From pen pressure, to larger control points and larger buttons, to undo, redo, and delete buttons, the "Touch" interface incorporates all the commands needed to work without a keyboard or mouse.

There are two features within *Illustrator's* "Touch" workspace that stand out: "the Stencil tool" and a new variation on the pen tool that is called "the Curvature tool". The Stencil tool introduces guide geometry in the form of rulers and French curves to the drawing environment irrespective of stroke styles or line representation. This guide geometry behaves exactly as one would expect in real life: there are two hit zones for multi touch that allow the user to move and scale the ruler or French curve with two fingers. As the user moves the ruler around, it can snap to existing geometry in the artboard. The second tool is the Curvature tool that allows spline drawing through the placement of individual control points. Using a pen or mouse for input, the spline previews before the user places a control point. This allows the user to dynamically preview the shape of a curve before settling on a fixed control point, rather than placing a control point to only later decide that it should be modified.

Also worth mentioning within the “Touch” workspace are *Illustrator’s* “Join tool,” that can join and fillet multiple strokes into a single spline, and *Illustrator’s* “Shaper tool,” that can understand and transform loose drawings into (editable) perfect geometric shapes. The “Shaper tool” further allows the user to trim strokes by scribbling on one side of an intersection of two or more strokes. This trimming behaves generatively by masking the undesired portions of the geometry definition and allowing the user to access and edit the original geometry as well as elements of the final form like fill colors and stroke types.

3.1.2 Adobe Photoshop

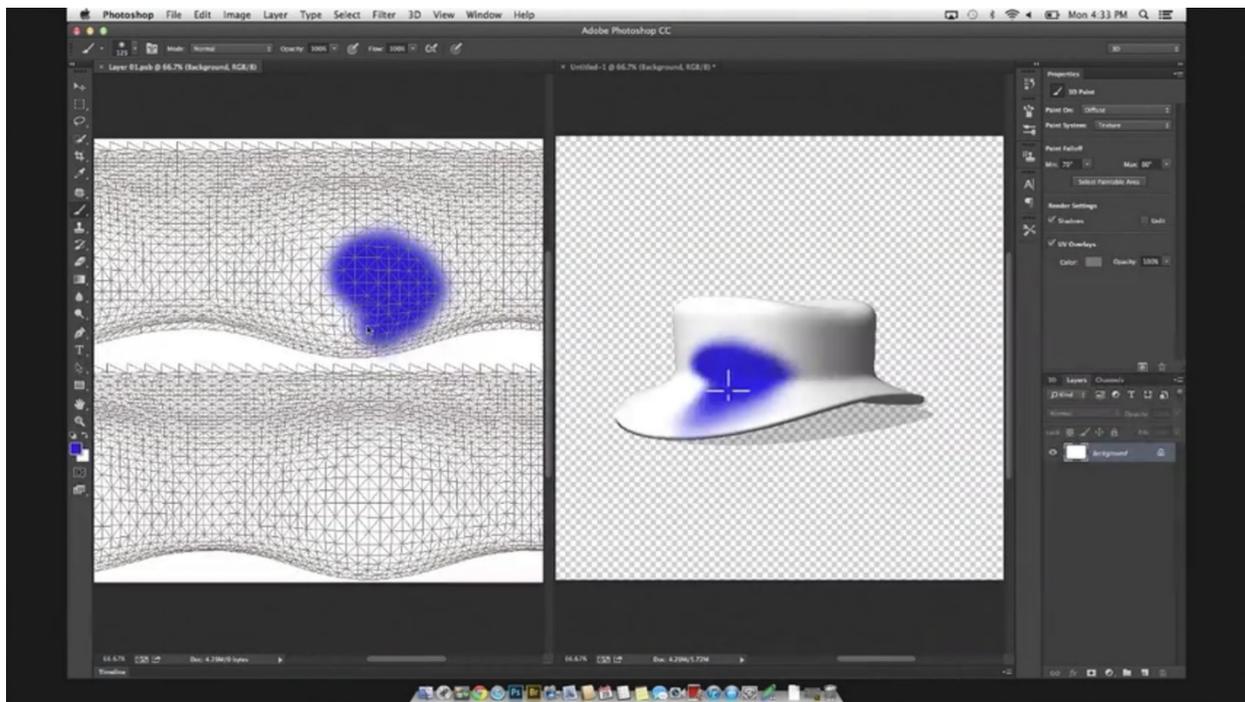


Figure 3.2 Screenshot of the Photoshop interface showing an .obj mesh file being painted on. On the left hand side is the unrolled mesh that allows for 2-dimensional painting on a 3-dimensional form shown on the right-hand side. Can input in either 2D or 3D space.

Adobe *Photoshop* has been the industry’s de facto raster graphics editor since it was created by Thomas and John Knoll in 1988. It has a powerful layer manager that allows for the composition of raster images and has features including masking, alpha compositing, filtering, image analysis, scripting, drawing,

vector text and shapes, video editing, recording and saving strings of commands, and much, much more. This review, will focus on *Photoshop's* 3-dimensional functionality.

Unlike *Illustrator*, *Photoshop* can create, import, and export 3d content compatible with other powerful modeling and rendering suites through common file types like .obj, .3ds, and .kmz. Included are basic 3d tools for object rotation, translation, and scale as well as some basic camera controls. Editing capabilities for 3-dimensional form in *Photoshop* are largely limited to placement and transformations of scene geometry and a merge objects command that begins to suggest the potential for standard Boolean operations. *Photoshop* also offers the ability to generate and place basic canonical forms like cones, cubes, cylinders, tori, etc. within the scene. The real value behind *Photoshop's* 3d capabilities however does not lie in the creation or editing of the geometry but rather in its ability to create textures and other 2-dimensional graphic images like renderings. In this way, *Photoshop* leverages its core strengths and maintains its focus on the creation, editing, and compositing of raster images.

To allow the mix of different types of content within the same interface *Photoshop* stores any 3d content created or imported as a 3d layer within its powerful layer manager. For users familiar with *Photoshop's* more traditional feature set, 3d layers are different and yet comfortably similar to normal layers. They have a thumbnail, a description, and effects in their traditional locations and styles. Normal operations like opacity or blending changes are also applied to 3d layers in the same way. The magic behind *Photoshop's* 3d functionality becomes apparent when you realize that you can paint and apply rendering and lighting effects directly on the 3d models. In Figure 3.2 we can see two views of the same object, a hat. On the left is a flattened mesh of the hat that updates in real time if you draw on the model; of course if you draw on the texture map the hat will update. In this way users can paint detailed textures on their models for rendering or use in other software. Alternatively, users that prefer compositing and creating realistic images in *Photoshop* could do it all without ever leaving the software.

3.1.3 Sketch

Sketch is a lightweight and quite powerful 2-dimensional vector-based application for Mac. It is a very common tool amongst designers who are building digital products and who need to block out, outline and design user interfaces. It is strongly targeted at graphic, interface, and experience designers who design computer or web applications and thereby restricts what might traditionally be considered ‘sketching’ in other areas of design. Rather than focusing traditional tools such as a pen or a pencil, *Sketch* provides a number of simple vector-based features that work very well for a specific purpose. It focuses on manipulating basic shapes and fonts through transformations, colors, opacities, shadows, and so on. *Sketch* also has a fairly powerful layer manager, more like Photoshop than Illustrator, where each ‘object’ is in its own layer and layers can be grouped into folders. Like Illustrator, *Sketch* has artboards which can be scaled or sized as the user decides, however unlike Illustrator, *Sketch* is much more lightweight. It can support larger workspaces, to the point that one might consider the project space an ‘infinite canvas’ within which there can exist a substantial amount of artboards.

Of course, given the lack of pen input, and given the general scope and interest of this thesis, *Sketch* is not an existing solution to the problem we are exploring. That being said, we include *Sketch* as part of this research because we believe there is great value in parts of the solution they offer that could inform and potentially justify our decisions behind the product. This is particularly true in the interaction between different artboards that are within a project’s infinite canvas, a characteristic which is quite useful in practice. It should also be noted that the lightweight nature of the stored geometry allows for a very responsive experience, for example zooming in and out of different designs and explorations. This responsiveness of scale and variety allows the exploration of alternate designs simultaneously and has proven crucial to *Sketch*’s success.

It will be interesting to see how *Sketch*, a leading user-interface design application, evolves. As it stands, as a two dimensional drawing application with limited functionality, it works really well in a specific context. As that context itself evolves, and as user interaction design shifts into 3-dimensional space with

technologies like virtual and augmented reality, it will be interesting to follow how very targeted applications like *Sketch* adapt and incorporate the third dimension.

3.1.4 Made with Mischief

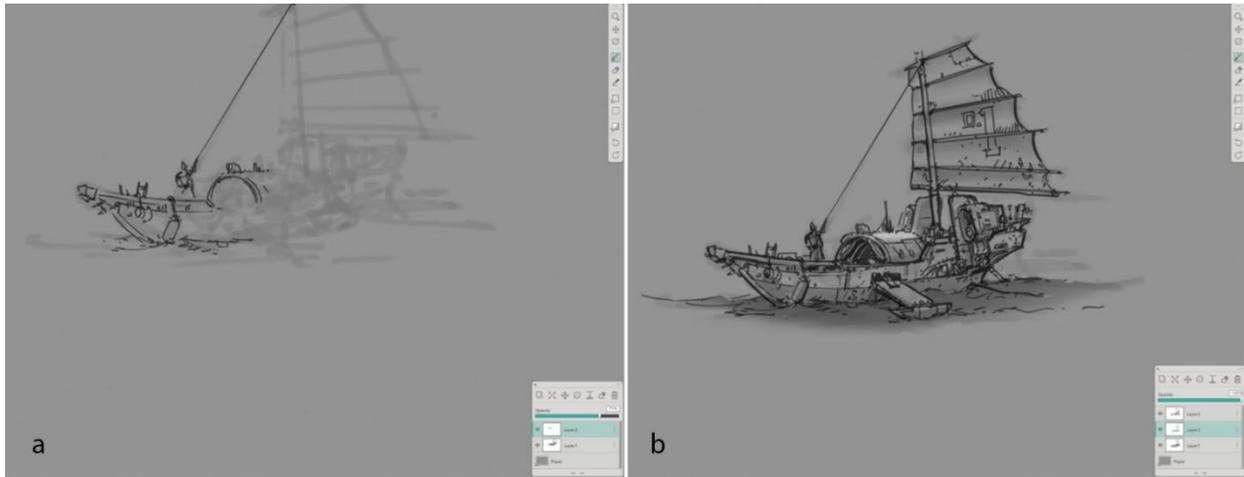


Figure 3.3 Example of Mischief in use. Notice the designer first drew a ghosted and rough preliminary sketch (a) before adding any detail or shading to the drawing (b).

Mischief is a 2-dimensional drawing application for Windows and Mac OS X that was spun out of research by two Scientists at Mitsubishi Electric Research Labs (MERL), Dr. Sarah Frisken and Dr. Ronald Perry. The technology behind *Mischief* is protected by over 50 patents, and in late 2014, the company 61 Solutions along with the product and the intellectual property portfolio MERL was acquired by the Foundry, who specializes in creating high-end software for the visual effects industry.

What *Mischief* set out to do, and has managed to do quite well, is to provide a powerful 2-dimensional drawing application that has the natural feel and control of raster graphics without losing the scalability of vector graphics. The user is able to draw very intuitively, with pen and touch input, and interact with their artwork in a similar fashion as you might expect from a pixel based editor. The differentiating factors are the application's infinite canvas and revolutionary stroke representation that allow designers to scale or zoom without running out of space, without aliasing problems, without loss of edge definition, and all while maintaining small file sizes.

“ADFs are a new digital representation of shape which provide numerous advantages including high-quality anti-aliasing, very fast rendering, very small file sizes, multi-scale rendering, support for massive parallelism, and the ability to succinctly represent variable-width, scalable, textured strokes.”

– Made with Mischiefⁱⁱⁱ

Dr. Frisken and Dr. Perry each hold dozens of patents in Adaptively Sampled Distance Fields (ADF) Technology. ADFs are the magic behind Mischief and they make the program feel more like a lightweight app than the powerful program it is. In 2016, the software retailed at a very affordable \$25 for the full version and there is a free version with limited features also available to all users. Moreover, the installation is quick and straightforward and the software only takes up 7MB on your hard drive; it really feels more like a lightweight app than a full application. From a usability standpoint, working with ADFs enables very small file sizes and fast response times while working with a zoom factor of *50 trillion-to-1*. Neither pixels nor vectors, Frisken describes working with this revolutionary shape representation is “like sitting on the moon and looking at a single flower on Earth and then drawing on a petal of that flower”.^{iv}



Figure 3.4 Example of *Mischief* artwork by Carly Sanker demonstrating the power of ADF Technology. Here you can see the detail at different levels of zoom showing the scalability, infinite canvas, and sharp line strokes.^v

One of the major benefits of using ADF Technology, beyond the technology itself, is some of the tangential functionality that it allows; the most important of which is a truly responsive infinite canvas, one of *Mischief's* core strengths. While 3d modeling environments have traditionally employed this infinite workspace concept, 2-dimensional programs generally tend to more closely follow the paper or screen analogy where borders are part of the constraints. Removing the borders and losing a sense of scale provides a certain freedom of exploration that is more difficult to achieve in traditional software applications. *Mischief* puts forward an unconventional software approach where the scale of the output, to either paper or to a 2-dimensional image, comes after the artwork is created. This flexibility enables different connections between the artist and the digital artwork - usually more experiential, more emotional, and less technical or formal.

This sketchy, in some ways non-technical, and scale-less approach is what sets *Mischief* apart, and the flexibility and power of ADF technology is what allows it. More than simply being able to represent strokes sharply throughout a large range of scales, ADFs allow a certain expression of emotion and intent that traditional vector representations struggle with. In their press release The Foundry proclaims that ADF Technology “can accurately represent much richer and more complex shapes than traditional vector-based stroke representation.”^{vi} Pen pressure, pen angle, line and fill opacities, gradients, erasing, etc. are interacted with in a more natural way than with other vector-based applications, in some ways blurring the lines between vector and raster graphics. This results in an experience that an artist can more easily relate to and begin to easily use, as they might in the real world on pen and paper, or even in other raster based graphics drawing/painting applications. The last major hugely positive differentiating feature *Mischief* offers falls in line with this approach. They leverage existing practices within drawing-based professions, such as the use of translucent trace paper, and enable the functionality within the interface. In other words, the entire application can be made translucent; this has very large implications and applications for designers and artist in practice, something easily seen in public feedback about *Mischief* online.

“From the very start that was very much one of our passions – Mischief should be easy to use. You don’t need a hundred menus; you don’t need a manual. You just open it and you start to draw.” – Frisken^{vii}

Despite all of these innovations and all *Mischief’s* promise, the application is currently still restricted to drawing in two dimensions. Oddly enough, the hypothetical leap in to 3d should not be a technical hurdle for the *Mischief* team. The technology that drives *Mischief* (Adaptively Sampled Distance Fields, or ADF) was created for 3-dimensional medical imaging, and given that most of The Foundry’s services are for 3d content, it is not a stretch to see some of this technology finding its way into the 3d sketching market. The question of implementation and *how to use ADFs to draw in 3 dimensions* remains.

3.1.5 Morpholio Trace

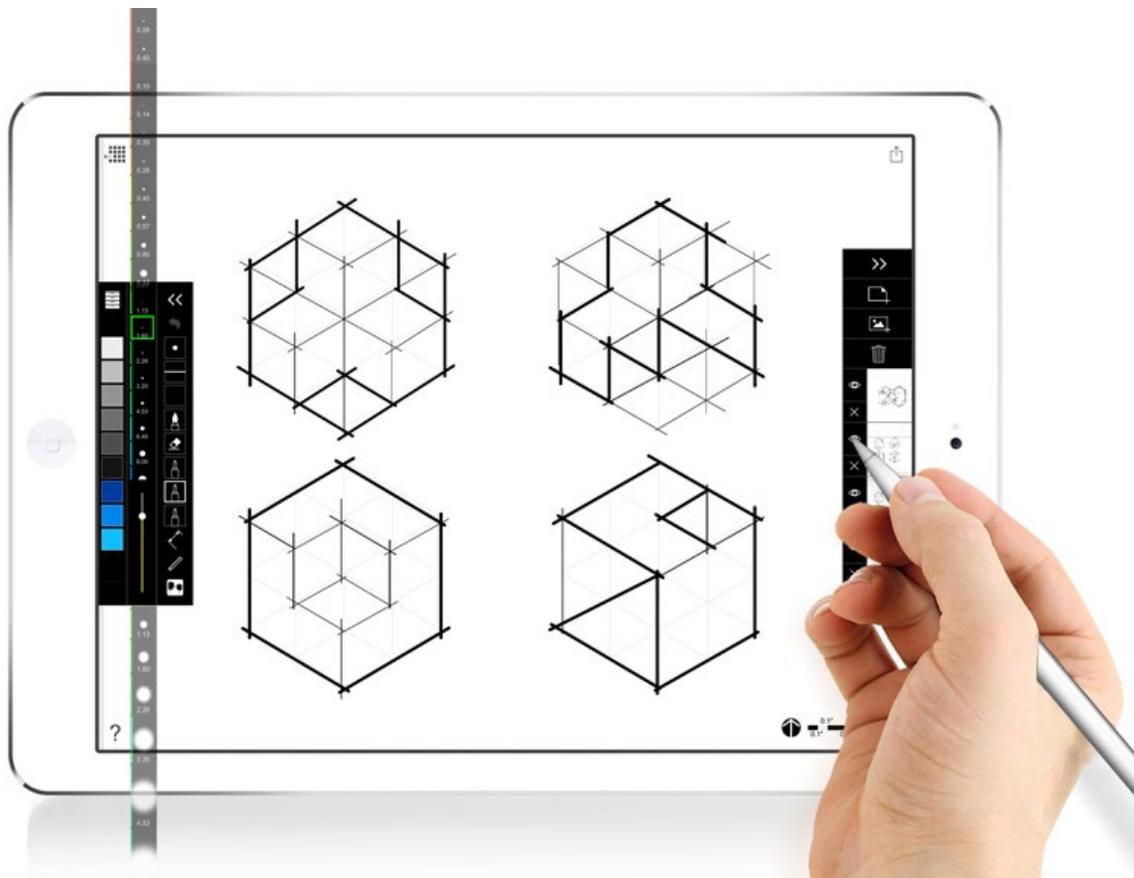


Figure 3.5 Marketing image of the Morpholio Trace application shows the user picking between different line thicknesses to apply over an initial sketch of a 2 dimensional representation of a 3d diagram.

Morpholio Trace is part of a suite of applications that have been developed for the iPad that cater specifically to designers and architects. *Trace* is specifically designed to mimic the user experience that is familiar and comfortable to designers who use trace to iterate and revise. The app is a fully functional drawing application that allows the user the added control of manipulating a layer of trace on top of an existing image or drawing. The user has the added freedom of being able to manipulate multiple layers of trace in one drawing session. The user can rotate, and translate using multi-touch gestures that are common to the iPad interface, and they can use a pinch gesture to zoom in or out of their drawings.

The latest version of *Morpholio Trace* introduced yet another physical metaphor or hallmark of Architectural drawing, the technical pen. This tool harkens back to tools that were originally used to draw starting as early as the 1960's. The appeal being that you can now control different line-weights and line thicknesses in your drawings in order to convey more information. The application also automatically adapts line-weights based on the magnification level of the drawing.

3.2 3-Dimensional Software

This section reviews software that has been developed specifically for 3d content. There are a variety of different projects with different scales and budgets, some are academic research proposals and others are extremely powerful and exclusive software solutions. We include here the software that has had the greatest influence on our understanding and aspirations for 3d drawing using multi-touch and pen input technologies.

3.2.1 SketchPad+ by Moreno Piccolotto and Michael Malone

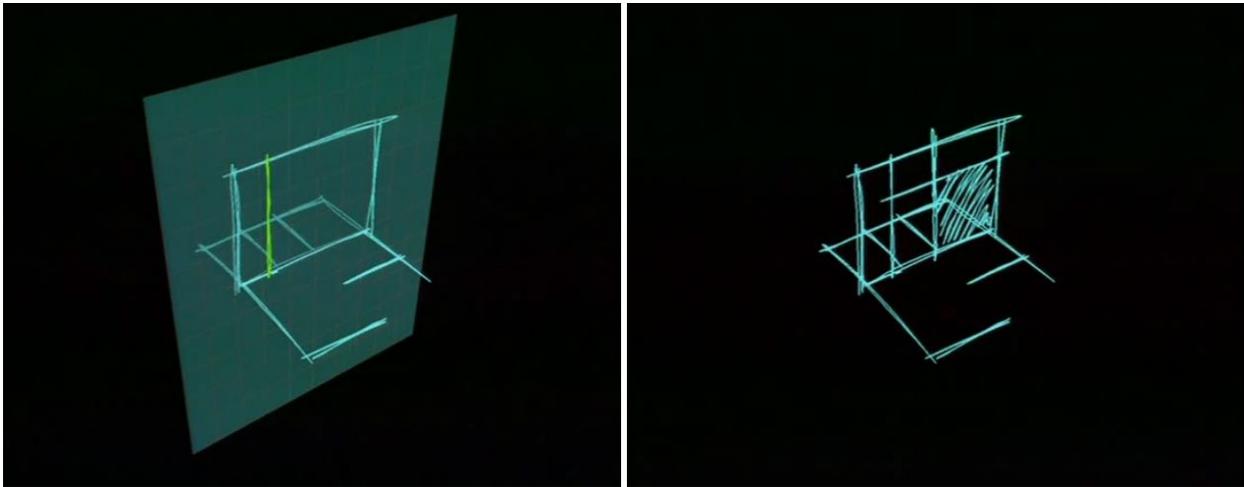


Figure 3.6 Sketch planes are able to be placed to allow for input into a 3-dimensional scene.

We include this early proposal by two Master of Science graduates of the Program of Computer Graphics at Cornell University. The thinking and objectives behind this thesis have greatly influenced the continued work and research in 3d drawing by students and alumni of Cornell's Program of Computer Graphics. Unfortunately, at the time this thesis was presented in 1998, adequate hardware was simply not available to realize the full extent of their vision. That being said, Malone's and Piccolotto's research offers insights into some of the hesitations that some architects have with computer aided design and begins to suggest strategies that can offer designers the best of both worlds. We have expanded on many of these insights in Chapter 2 of this thesis.

3.2.2 Collaborative Design Platform – Violin Yanev Thesis

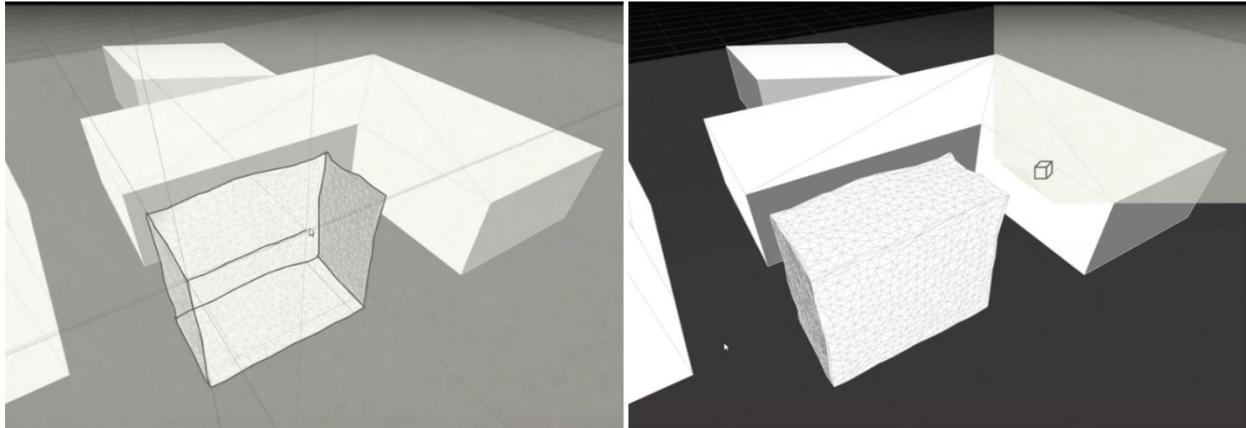


Figure 3.7 A translucent layer resembling trace paper is used in the interface as a way for the user to toggle between drawing and camera manipulations. ^{viii}

Inspired by a 1999 paper (Urp: A Luminous-Tangible Workbench for Urban Planning and Design) written by John Underkoffler and Hiroshi Ishii out of MIT Media Lab’s Tangible Media Group, researchers in the Technical University of Munich formed a group in 2011 called the Collaborative Design Platform Research Group. The group is led by research director Dr.-Ing. Gerhard Schubert and is supported by the department chair for “architectural computer science” (Architektural Informatics) Dr.-Ing. Frank Petzold and the department chair for augmented reality Prof. Gudrun Klinker, Ph.D. The group’s mission is to “resolve the current discrepancy between familiar, analogue ways of working in the early architectural design stages and the ever increasing use of digital tools in office practice.” ^{ix}

Violin Yanev is currently a software engineer at BMW in Munich. In 2012, Yanev graduated with a Master of Science in computer science from the CDP group in the Technical University of Munich; his master’s thesis, a 3d sketching application aimed at architectural design, is titled “3D virtuality sketching: a freehand sketch tool for conceptual urban design in architecture.” The application allows for traditional camera controls within a 3d modeling environment and uses a trace paper metaphor to allow for functional differentiation between basic 3d sketching and traditional 3d modeling interactions. Figure 3.7 shows a layer of trace paper can be overlaid on the viewport in order to enable sketch input and edit sketch geometry.

3.2.3 T.E.D.D.Y. (Draw in 2D – Output in 3D)



Figure 3.8 In the above image, the user of T.E.D.D.Y. demonstrates how a 2 Dimensional sketch in one orientation becomes a 3 Dimensional object in a different orientation, through algorithms that interpret the 2 non-planar strokes as descriptors of a 3D object. ^x

T.E.D.D.Y. was a drawing application that appeared out of the research group at the University of Tokyo, and the Tokyo Institute of technology. The application allowed for the creation of freeform meshes that would be rendered using line strokes and automatically shaded depending on orientation. The software proved to be intuitive for the creation of arbitrary 3D Objects. Our interest in T.E.D.D.Y. is in the choice of representation of the 3D object, in which the mesh is rendered as a set of outlines and shaded in order to look like a pencil drawing. This software is also notable for its interface which assumes the user is performing 3D modeling operations based on the position and relationship of the stroke and the existing geometry.

In T.E.D.D.Y. the user is working on the silhouette of an object. By starting with this assumption, T.E.D.D.Y.'s creators are able to interpret strokes in relation to the silhouette as strokes that are performing “cutting” operations, or “creation” operations. The user starts by outlining a silhouette, and the software then interprets the silhouette in to a smooth shape that creates that silhouette. If a stroke is drawn across the silhouette of an object, the computer interprets that as a cutting operation and trims the original object.

3.2.4 3D Model-Assisted Sketching Interface

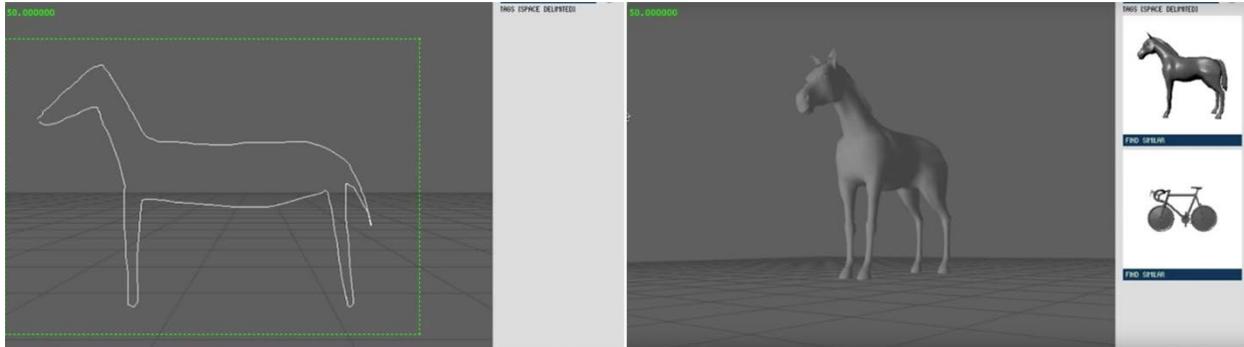


Figure 3.9 The image above demonstrates how a crude sketch generated by the user, selects the most similar 3D object from a known database. In this case, the rough outline of a horse, is replaced by a more accurate 3D Model of a horse, at the same location and scale. ^{xi}

This work is from 2012 and is included here because it demonstrates a very interesting implementation in which hand drawn input is being recognized by a computer and is then used to present relevant information from a database of 3d models. In this scenario, a rough shape is drawn and the interface retrieves a set of similar 3d models that can be imported. The application was developed for the goal of helping the user draw by providing the ability to reference and even trace over an external 3d model within a 3d scene. Beyond the ability to import custom guide and reference geometry from an external library, the significance of providing relevant information and even suggestions as a computer analyzes sketched geometry are particularly interesting.

Limitations of the above system occur when there are no pre-existing samples for what the user is attempting to create. This means the software is limited to 3D object “selection”, over 3D Object creation. Even in the sample above, if a user wanted to create the model of a horse rearing its head, the system would have to rely on previously containing that specific 3D model, and even then the horse may still not be in the position the user intended.

3.2.5 Napkin Sketch Surfacing in 3D

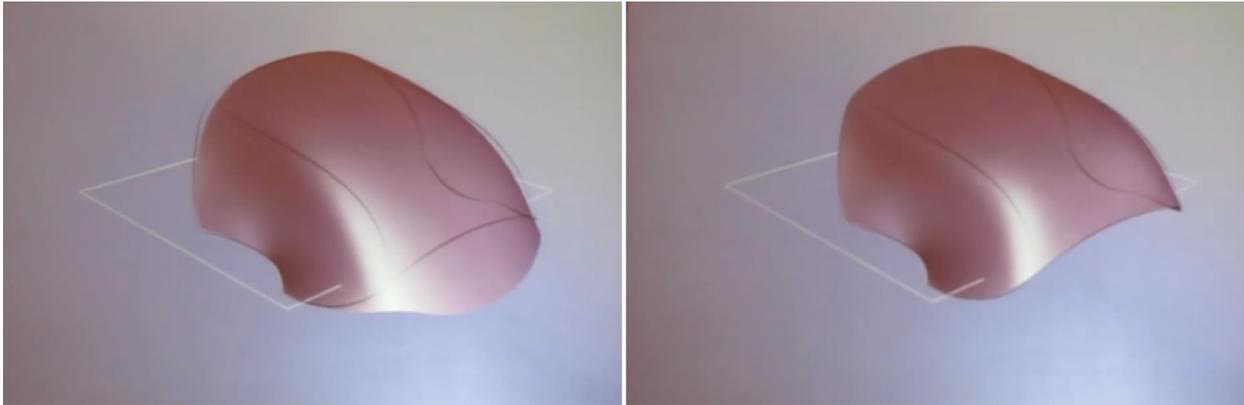


Figure 3.10 The image above shows how a “trim” operation is performed on a surface model in the aforementioned software. The user draws a stroke near the edge of the surface and the computer system removes the portion of the surface on one side of the new stroke. ^{xii}

Napkin Sketch Surfacing in 3D is a pen-based 3D modeling platform that interprets sketches as 3-Dimensional surface geometry. The interface allows for the intuitive creation of freeform surfaces through sketch based input. The system automatically interprets 2-Dimensional pen-stroke inputs in relation to the existing surface geometry in the scene and from this information it is able to modify the surface geometry to more appropriately align its silhouette with the latest stroke input in the system. Not much else is known about the software included above, but it is included in this chapter for its ability to interpret sketch input in order to create surface geometry that could potentially be used in a traditional 3D Modeling environment.

3.2.6 Rhonda Forever



Figure 3.11 Two screenshots of James Paterson, one of the contributors to the *Rhonda Forever* project, are shown drawing a teddy bear within *Rhonda*.^{xiii}

Rhonda Forever is a 3d drawing tool originally developed by Amit Pitaru around 2003 and has been contributed to by four others over the next six years. Around 2009 or 2010, development for *Rhonda* stopped after being shown in a number of galleries, museums, festivals, and conferences. *Rhonda* works in a straightforward and easily understandable way. There is always a drawing plane parallel to the screen and the user uses pen input to draw on the drawing plane. The camera controls allow the user to orbit and pan, and intersection points between geometry and the drawing plane parallel to the screen are highlighted with red dots. At any point in time the user can continue to input line geometry with the stylus or pen. One issue with this approach is the lack of control intuitively available to the user in establishing precise drawing planes for accurately drawing between two specific arbitrary points.

3.2.7 Field 3D Freehand Drawing

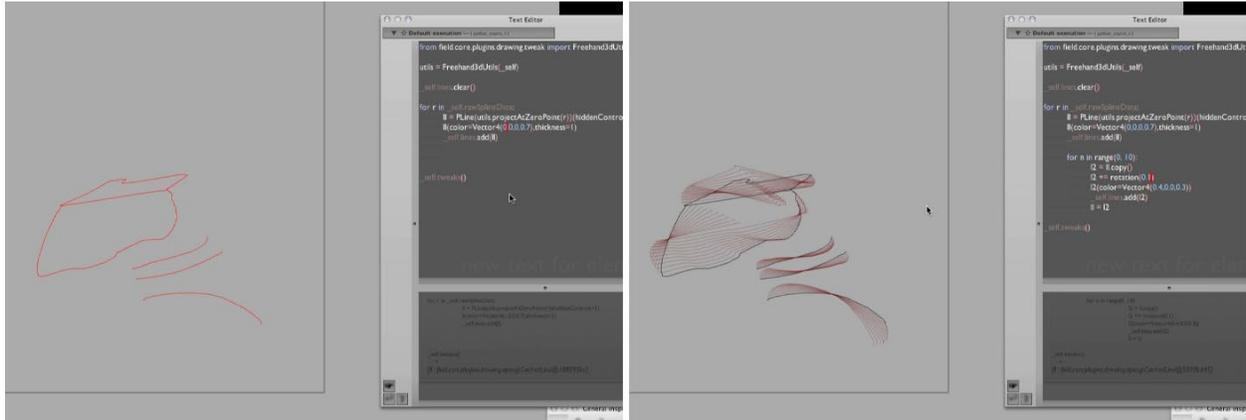


Figure 3.12 The above images show a set of strokes in the field 3D drawing environment, before and after they've been further manipulated by code written in the panel on the right. In both instances the red strokes are the new strokes and the black strokes are strokes that were created before the current step. ^{xiv}

Field 3D Freehand Drawing is a tool that reveals the strokes created in a 3D interface to a simple code editing panel that allows for direct manipulation of the strokes by simple algorithms that augment the original 3D drawing. This tool was included in the survey because of its focus on a simple and intuitive interface which focuses on the creation of initial strokes and splines, in order to be later manipulated by algorithms created by users. This software shows the potential for users of different skill sets to collaborate with each other using an amalgamation of both drawing tools and scripting tools. The tool is also important in this discussion in that it demonstrates a potential for a tool that plugs-in to an existing set of tools and workflow, while maintaining its utility in a standalone form.

Some limitations of the tool include the inability to support surface geometry. This limits the tool to creating outlines of geometry that can later be adapted to mesh, or surface geometry. It also limits the tool from potentially creating complex scenes as lines fail to be occluded when they appear behind geometry.

3.2.8 I Love Sketch / Everybody Loves Sketch

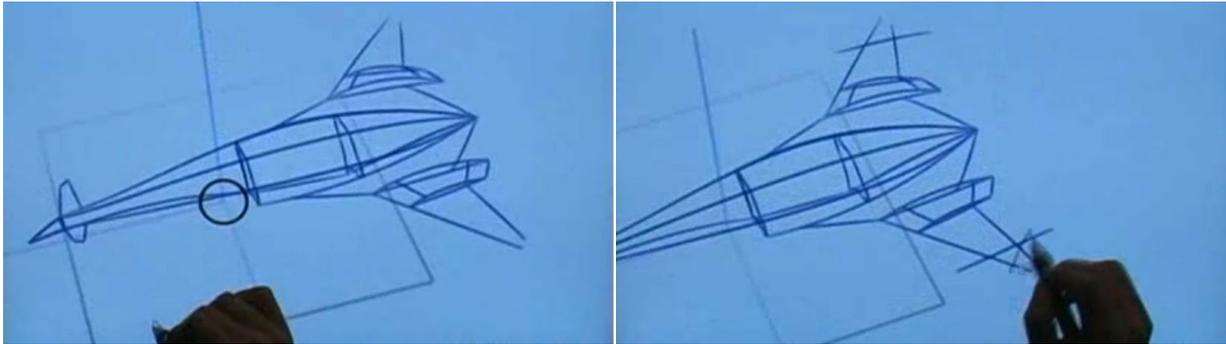


Figure 3.13 Shown side by side are two images of the 3D Sketch based interface I Love Sketch being used by potential users. The image on the left shows a user creating the 3-Dimensional outline of a complex object using a pen and tablet interface, while the image on the right shows the software being used by novice users without a background in 3D Modeling.^{xv}

I Love Sketch is a prototype 3D Sketch based interface who's input algorithms and interface form the basis behind uMake. It is one of the more robust tools developed to create 3D geometry from a pen based interface. As such, the uMake section will delve deeper in to the individual pieces of functionality that emerged from I love Sketch, and its follow up program entitled Everybody Loves Sketch. I Love Sketch uses an interface that allows users to define the position of a curve in 3D space through the use of 3D "scaffolding". I Love Sketch's algorithm's rely on a hallmark of analytical sketching which shows a designer defining the position of curves and geometry in 3D space by using "scaffolding" curves or "construction" curves that always represent rectilinear flat geometry as a position marker or orientation point for the 3-Dimensional sketch. In this way the user is able to position their strokes in the scene in an intuitive manner that takes advantage of the user's ability to draw arbitrarily positioned 2D planes.

Limitations of the software include the inability to use the created 3D geometry in other design software. Collaboration not being the main goal of the program, the geometry that was created in I Love Sketch could only be accessed and viewed through the program itself.

3.2.9 uMake

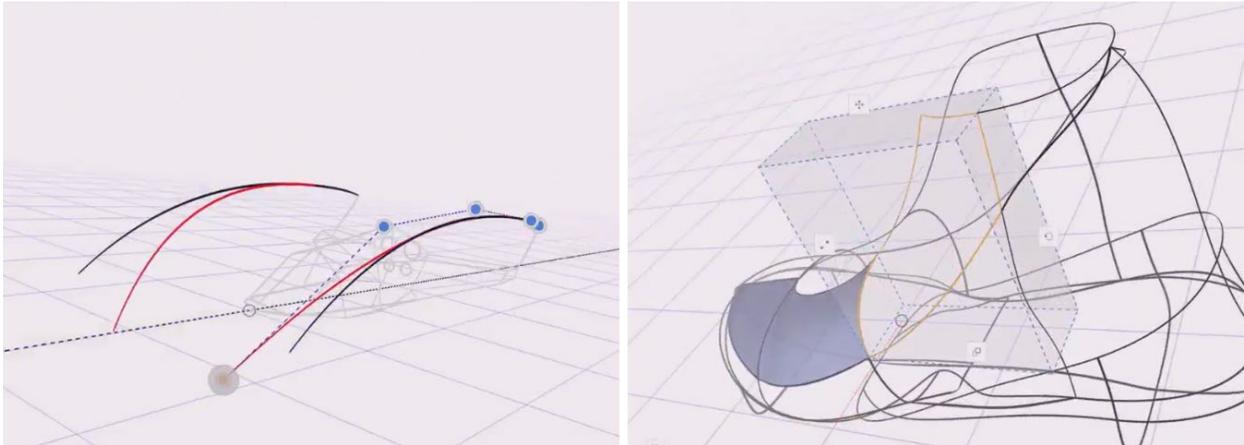


Figure 3.8 Screenshots of *uMake* show the ability to work symmetry, the ability to edit splines with control points, and some basic surface modeling functionality.

In September 2015, Apple announced the iPad Pro and the Apple Pencil stylus. To demonstrate the implications of their announcement, Apple drew attention to one app in particular: *uMake*. *uMake* is an award winning consumer software product/service for iOS written and sold by the former Autodesk employees responsible for AutoCAD 360 mobile and web apps. The product appears to be closely related to or at least influenced by the 8+ years of research behind *I Love Sketch* and *Everybody Loves Sketch*. *uMake* is focused on delivering an intuitive and easily understandable, standalone application that allows for 3-dimensional conceptual exploration and refinement using pen and touch input.

“One of the main outcomes that we can see is that you can explore the shape and form while you’re in a 3D environment – this is a huge benefit while exploring ideas. Additionally, it saves a lot of time by getting complex curves and surfaces without the hassles” – Eviathar Meyer, Co-founder and CEO of uMake

uMake is a robust application for 3-dimensional exploration using sketching techniques. It provides a mixture of 2-dimensional, view-dependent, and fully 3-dimensional input techniques. The platform allows users to easily sketch on easily defined virtual 2D planes in 3D space, import and reference photos in the application, leverage spline technology by using easily editable control points, and use simple modeling

commands like move, rotate, array, mirror, etc. Importantly, *uMake* also has some basic surface modeling commands that allow you to visualize and explore your design more profoundly than a simple wireframe. Overall the app is well thought-out and executed; the lines are responsive and render well and there appears to be no latency or buffering issues.

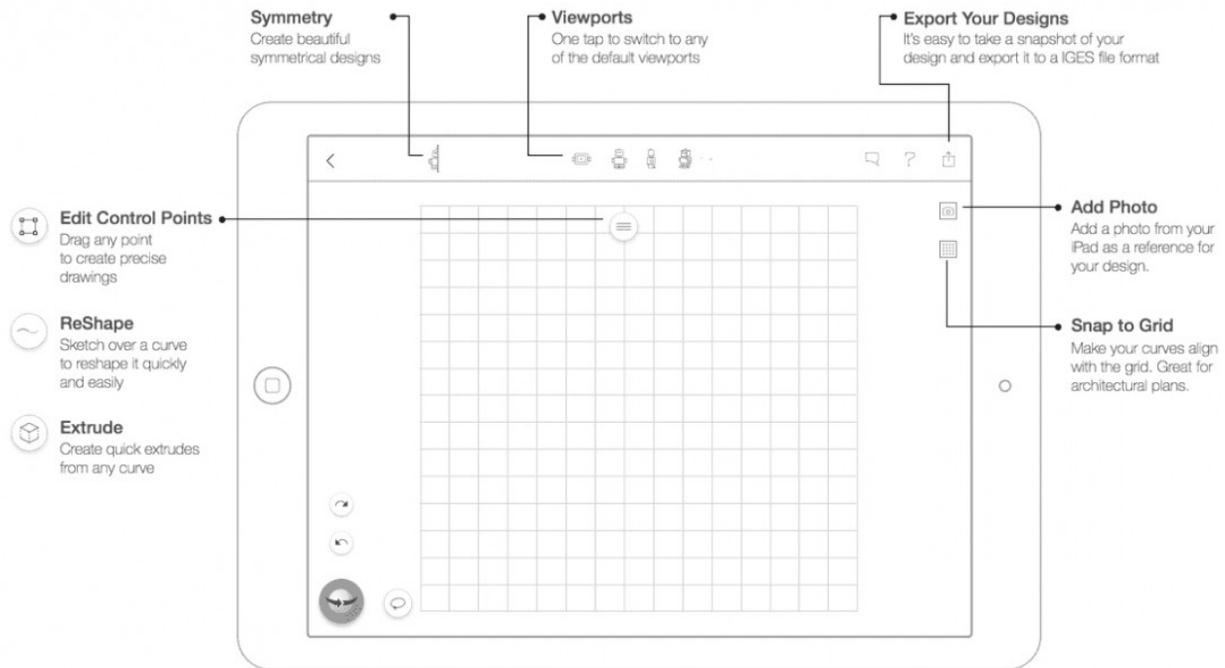


Figure 3.9 Wireframe of *uMake* interface on an iPad

The features included and the experience cleverly delivered by the *uMake* team allow most people to explore their design ideas in a new and more natural way. It appears that through this platform they intend to introduce 3d printing services and/or functionality as a next step in their attempt to democratize the currently complicated and expensive 3d virtual and prototyping spaces. In this context we are completely aligned with *uMake's* decisions; however we believe that there remains a weakness in the realm of 3d sketching that a standalone iOS app like *uMake* will have difficulty capitalizing on. This is because of hardware limitations from a tablet device.

uMake has decided to leverage consumer trends and has chosen to deliver beautiful and well thought out software on widespread and reliable hardware: Apple's iPad. While the reasons for doing this

make sense in some contexts, they ignore other just as important considerations. *uMake* made a decision to require that designers create and explore their 3d sketches in an app environment where modeling functionality is quite limited. In order to move forward with design development, designers must then move their designs out of the app environment and into more robust 3d software on a desktop operating system like Windows 10 or to a lesser degree Mac OSX. This approach doesn't allow for the continued inclusion of hand drawing in the back and forth iteration that may be required as new ideas are sketched over modified designs further down the pipeline. In forcing people to transition between software platforms and operating systems, some information and editability of geometry is usually lost; this is especially true if the designs are exported as meshes, such as the .obj files.

By forcing the designer to export and reimport their designs into different software, *uMake* forces designers to make a decision between losing editability and functionality in developing their models or losing the ability to explore form through intuitive sketching routines. This innately influences the work to flow through a generally linear pipeline.

Recall from Chapter 2 that drawing is not modeling and that modeling is not drawing. When working with *uMake*, designers must choose to be either in a sketching environment with some modeling functionality or they must leaving sketching behind all together, they cannot easily go back and forth as they reconsider ideas and make new decisions. This is crucial to the creative process and by delivering a standalone, marketable product that they can wholly own and control, *uMake* also risks limiting the scope of where and how they can impact design. It is true that sketching is generally the first step in design, though it is just as important to recognize that it is common for designers to use sketching throughout the latter parts of the design pipeline.

3.2.10 Mental Canvas

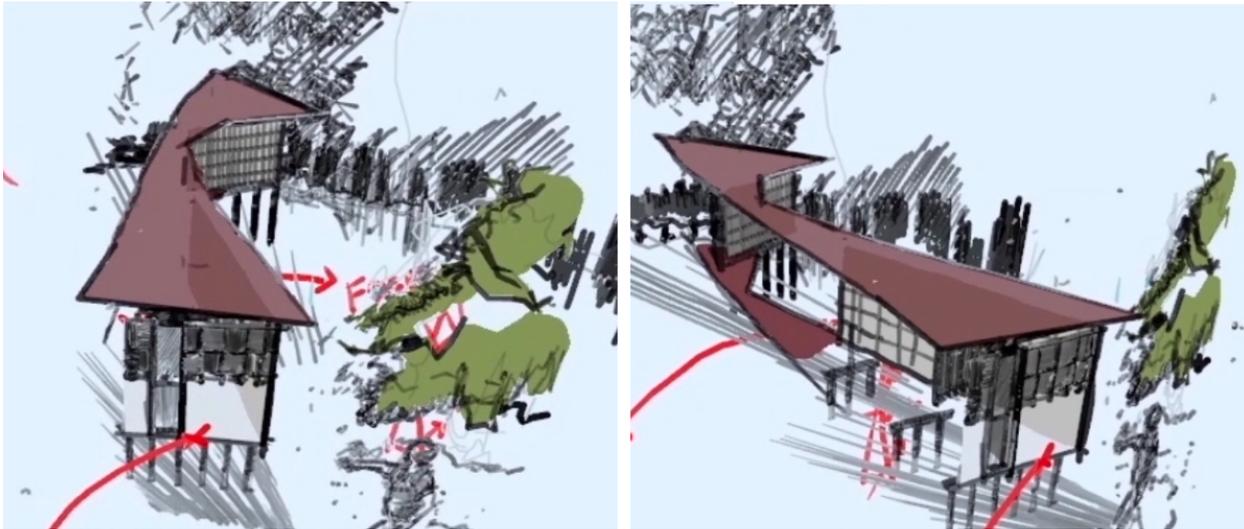


Figure 3.10 The screenshots above taken from a video on the Mental Canvas website show the same mental canvas sketch from two different orientations. This image describes the way strokes in the Mental Canvas system are placed on canvases arbitrarily placed in 3D.

Mental Canvas is a 3D Drawing application built for Windows 10 Surface devices. The software is largely based on technology from *The Mental Canvas: A Tool For Conceptual Architectural Design and Analysis*^{xvi} that was published in 2007 for the Pacific Graphics Conference. Julie Dorsey was one of the first authors on the original paper and is currently the CEO and founder of Mental Canvas the company. Mental Canvas is a drawing application that allows users to position, and draw, sketch-like strokes in 3D. This allows the user to create drawings that when viewed in 2 dimensions look like beautifully hand drawn conceptual sketches, but with the added ability to pan, zoom, and rotate around and into the sketch. This allows artists to easily create 3D content with a tool that resembles a similar touch and feel to drawing on paper. Mental Canvas produces illustration style content, in a 3-Dimensional interactive interface. Using this software, the Mental Canvas team was able to reimagine *The Other Side* an illustrated book written and illustrated by Istvan Banyai into an immersive experience for the iPad. The experience allows users to interact with their content in an entirely new way, in some cases the user can interact with the “pages” of the 3D scene and view them from entirely new orientations.

The goal of the software is to allow traditional illustrators and designers to easily create 3D content without having to be constrained by the rigid inputs of traditional 3D Modeling software. Mental Canvas uses pen and touch input, as well as a new input device created by Microsoft called the Surface Dial. The formfactor and design of the mental canvas interface allows for quick and easy creation of analytical sketches as well as 3-Dimensional illustrations

Some limitations of the software are that it does not input directly into traditional 3D Modeling software to be used as a starting point for 3D Modeling. This is because the software was never intended to be used to create 3D Models. Mental Canvas is interested in exploring the realm of creation that is outside of the area of producing traditional Mesh and Nurbs models. In the discussion and future work section of their 2007 paper Dorsie et al. describe this limitation. They also go on to describe how an additional limitation is the software's inability to "store and represent the temporal evolution of a design." Such a feature would certainly allow for a designer to more easily use Mental Canvas for preliminary design. This is because the initial design process is extremely iterative, and the designer needs to be able to go back and forth between multiple design options.

3.2.11 DDDoolz

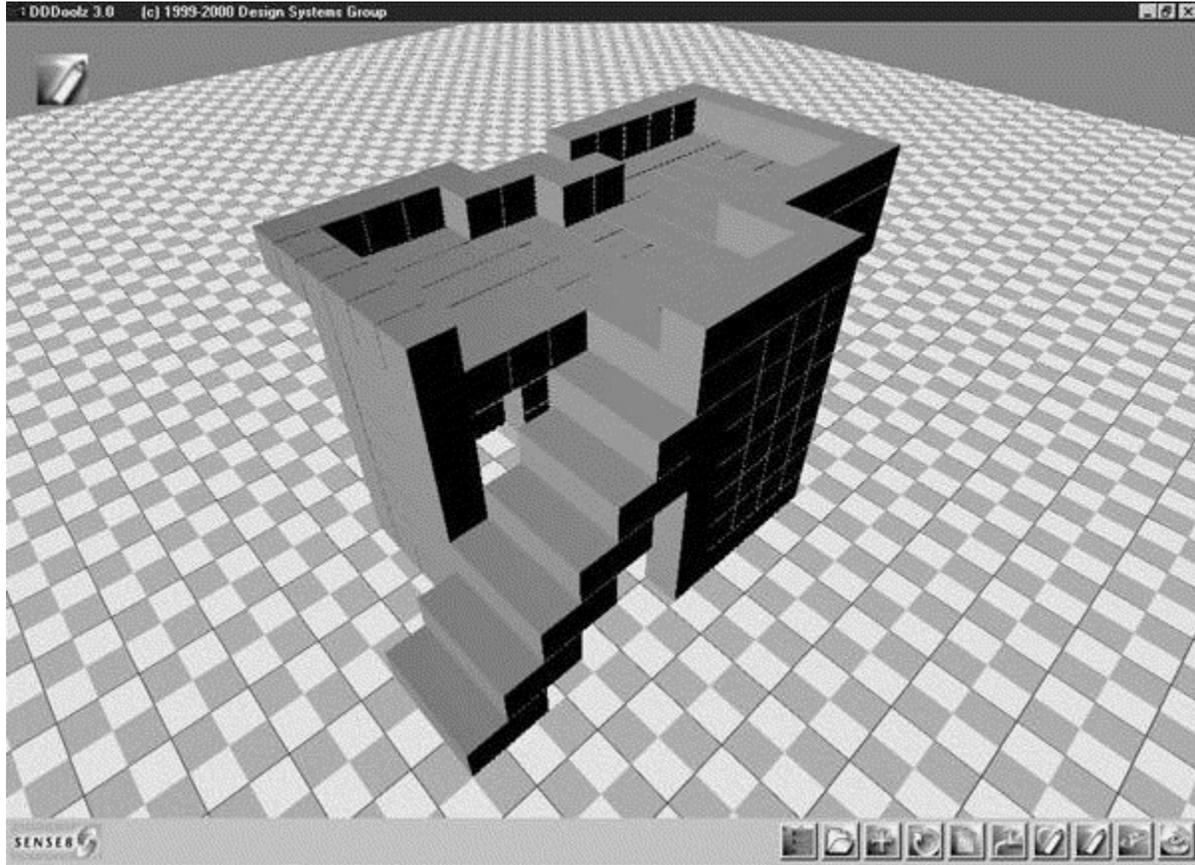


Figure 3.11 The figure above shows an example of a model produced by DDDoolz, a 3D sketching and early design modeling tool for architects. This specific figure demonstrates the software’s ability to create a variety of 3-Dimensional designs using a cube as the main primitive geometry and simply placing a number of these cubes in a specific orientation.

DDDoolz: “designing with modular masses” is an extensive study on the needs of an early stage design tool for Architecture published in 2002. The authors of the study evaluated existing modeling and CAD software through the lens of sketching and early stage design. They hypothesized that a tool such as DDDoolz would not be able to replace pen and paper, but that it may provide modeling features that are not possible in the physical world and yet would be very valuable to early stage design. They then developed a prototype tool that allowed designers to draw uninhibited, in 3D, with sets of cubes capable of drawing in different orientations, as well as in different scales. While developing such a tool, the authors tested the tool with a class of students at the Eindhoven University of Technology. The feedback they received was

very positive. The students posited that the main advantage and feature that DDDoolz had was a quick learning curve. Students with very little experience could pick up the tool and demonstrate ideas and designs in 3D in a matter of minutes.

DDDoolz is an interesting case study and analyses of the need for early stage design tools that are easy to use, computer based, and yet not restricted by the computer's need for a high level of accuracy for input. While their analyses were accurate, their focus was much more concerned with solving the problem of traditional 3D modeling software's unintuitive nature. While mention was given to the importance of using a 3D modeling tool for presentation, presentation was never a focus of the work and therefore they were not concerned with creating outputs that were visually appealing. Our work builds from their correct assessment of the limitations of traditional 3D Modeling software packages, mainly their difficulty to use, and their inability to serve the phase of early stage design. Where our work takes a departure from these initial primary goals, is in the question of specifically using pen based computing to provide a tool for early stage design, and to create an output which is useful and visually appealing.

3.2.12 CATIA Natural Sketch

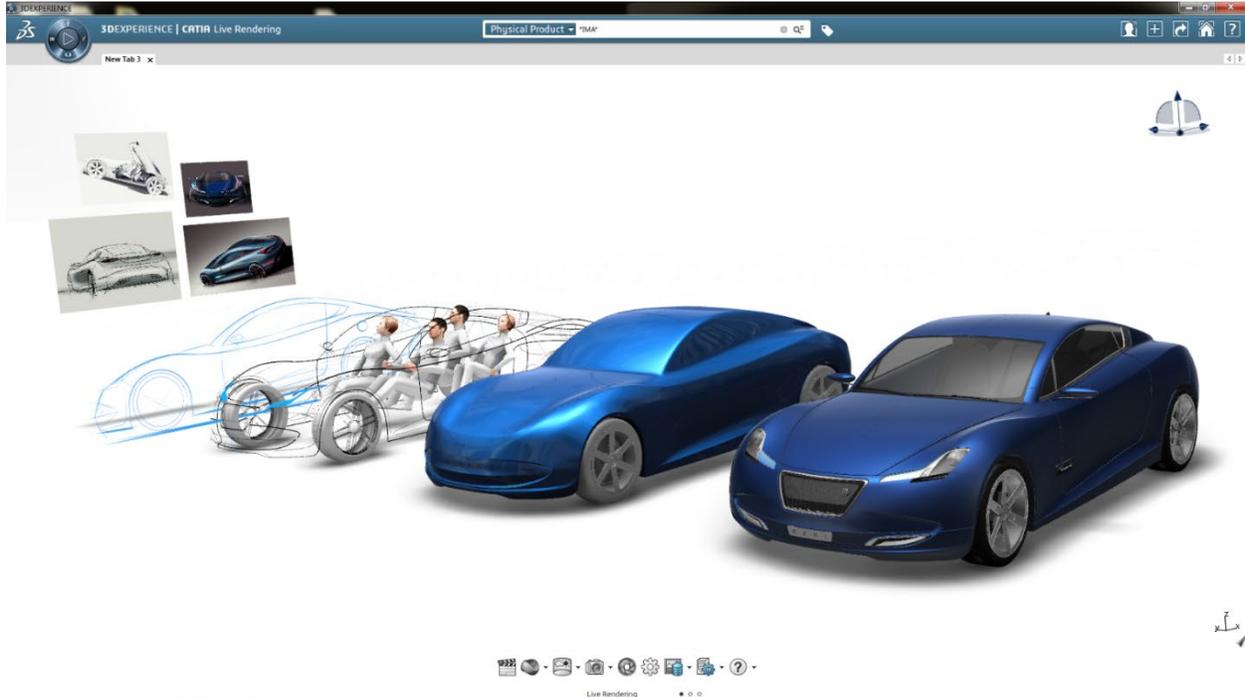


Figure 3.11 Typical progression through CATIA using Natural Sketch. First create mood boards and organize inspiration, then initial 2-dimensional sketches, followed by 3-dimensional sketches, massing models, and finally a refined 3d model.

In 2011, Dassault Systèmes announced a new interface for 3d sketching within CATIA v6 called *Natural Sketch*. We repeatedly tried to get access to CATIA v6 and *Natural Sketch* over a period of 12-13 months and were ultimately unsuccessful. In 2012, the price per seat of CATIA v6 ranged from \$9,000 to \$65,000 USD with an additional 18% yearly maintenance fee.^{xvii} The high price, along with our experience and difficulty in obtaining an evaluation version of the software makes us confident that *Natural Sketch*, as promising and revolutionary as it appears, is too inaccessible to impact the design process for all but a few enterprise users. This review is based on usage experience with CATIA v5 and indirect access to *Natural Sketch* through screencasts, reviews, and other information about the software sourced through online research as well as discussions with Dassault Systèmes representatives.

“CATIA Natural Sketch complements the CATIA for Creative Designers solution to address the complete industrial design workflow with a unified environment, from ideation and concept to

*refinement and design validation, from the first sketch to the final production-ready 3D product.” –
Dassault Systèmes^{xviii}*

CATIA (Computer Aided Three-Dimensional Interactive Application) was initially developed as a CAD/CAM solution for aerospace in 1977 and since has branched out to other industries, namely automotive, shipbuilding, and industrial equipment. Moreover, architect Frank Gehry developed an application over CATIA v5 specifically for architecture called Digital Project that has been used in a number of complex and innovative projects. To justify the high price tag, CATIA v6 offers many powerful features and packages/modules within a single environment.

CATIA is generally “referred to as a 3D Product Lifecycle Management software suite that supports multiple stages of product development, including conceptualization, design, engineering, and manufacturing. CATIA facilitates collaborative engineering across disciplines around its 3DEXPERIENCE platform, including surfacing and shape design, electrical fluid and electronics systems design, mechanical engineering and systems engineering.”

Dassault Systèmes believes that design is confusing and non-linear and that the best decisions involve a number of people with different backgrounds. In order to enable these decisions to be appropriately made, CATIA provides a design framework in which design, engineering, manufacturing, and other aspects of the product lifecycle influence decisions in each other’s areas at various stages of the process. Of course this means that the designs, models, information, etc. needs to be easily accessible and adaptable as the project develops.

It is clear that CATIA offers a much broader range of functionality than the scope of our proposal; therefore the remaining part of this review is focused purely on design conception and design development using *Natural Sketch*. *Natural Sketch* is offered as part of CATIA Creative Design Solution which provides what Dassault Systèmes refers to as a “*Unified Industrial Design workflow* solution”. This means that within one platform a number of tools and services are offered starting with early inspiration and ideation, to conceptual modeling, surface refinement, and detail design, and finally to virtual and physical prototypes. This process is enabled by straightforward 3d sketching tools, powerful surface modeling tools, and

importantly, tools that help designs and engineers streamline their collaboration by working on the same platform.

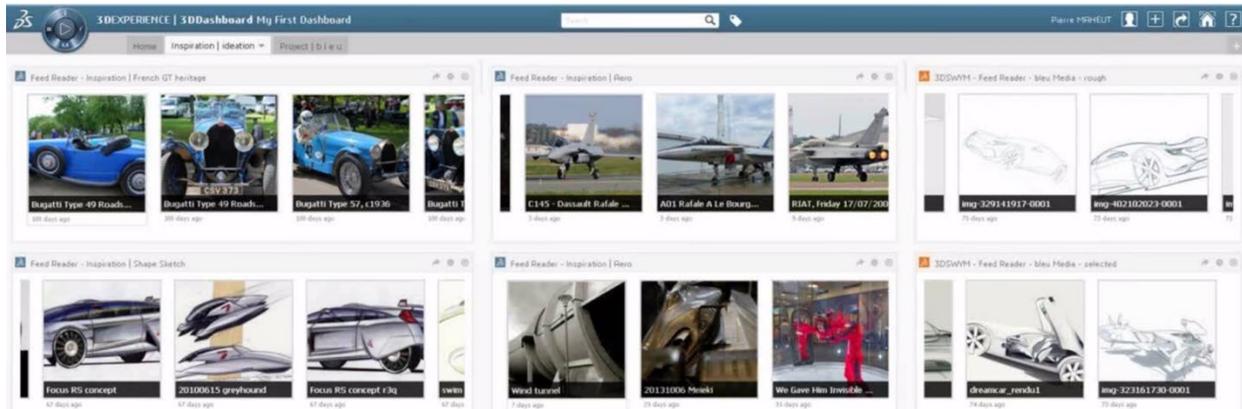


Figure 3.12 A view of the inspiration and ideation boards collaboratively created and used by designers in CATIA at the beginning of the design process. Notice that these boards include 3D drawings as well as images.

The CATIA design process begins with a design brief and the creation of inspiration and information boards. Through collaborative refinement, these inspiration boards are transformed into more formal mood boards with tags and text as shown in Figure 3.14. These searchable boards include the aggregation of different kinds of information and precedents including images, drawings, text, and sketches. The design team then enters the 3d sketching interface and leverages the content within their mood boards to begin their drawings. They can import any of the previously sourced and categorized inspiration and/or sketches into their drawing space to help them with their drawings and models. Of course, at any point in time, the mood boards can be adapted and changed as needed.

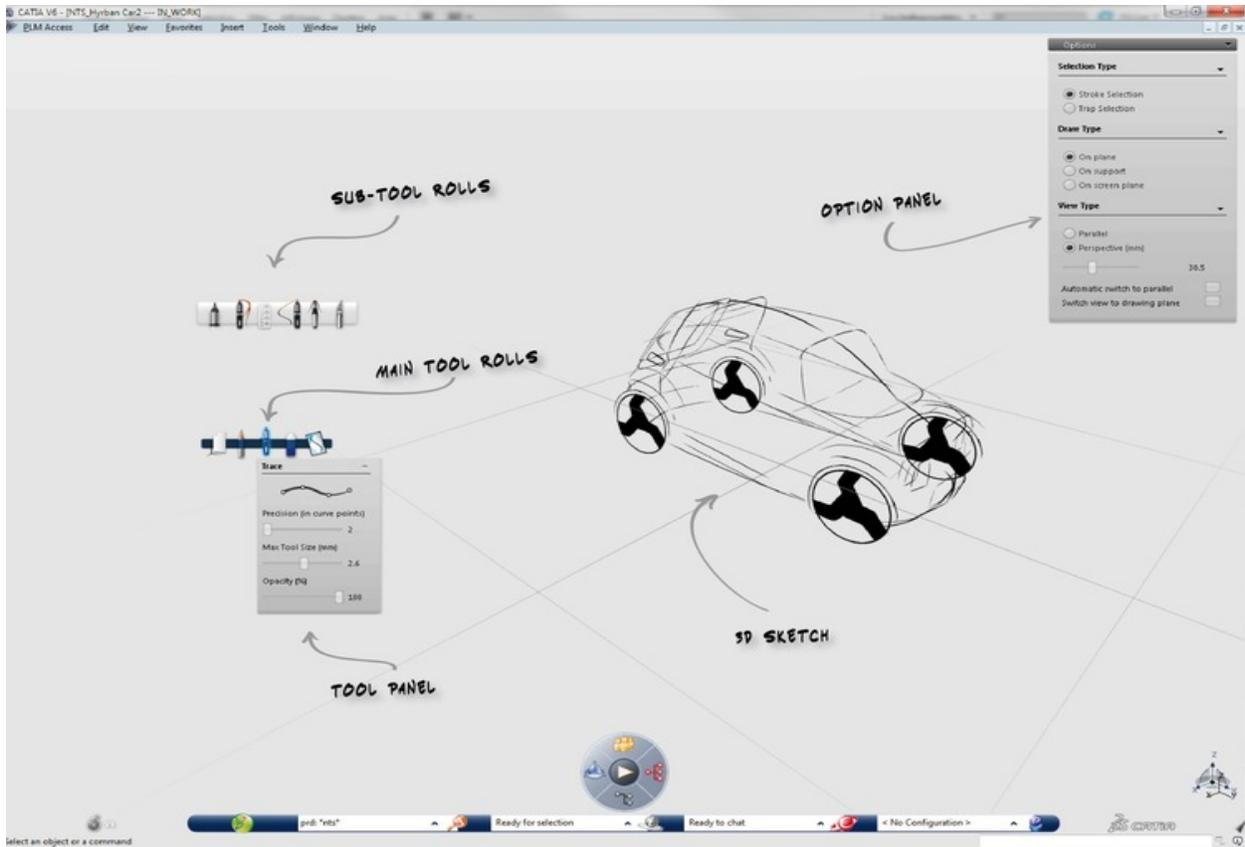


Figure 3.13 A view of CATIA Natural Sketch’s user interface.

Natural Sketch offers designers a way to leverage drawing, their generally preferred medium of exploration and communication, by allowing designer to integrate the sketching process into the modeling pipeline. *Natural Sketch* allows designers to sketch in both 2d and 3d and with a variety of line styles. Moreover *Natural Sketch* allows designers to transform 3d line work into 3d surfaces and then even draw on those surfaces after the fact. *Natural Sketch* is aimed at the aviation and automobile design and manufacturing communities, and it is unclear how well some of its features could apply to other scales and practice areas.

For all the promise that CATIA *Natural Sketch* offers through Dassault Systemes sponsored videos and other marketing material, and the fact that it was introduced as what appears to be a comprehensive solution far before any other platform, it is unclear why it hasn’t been more widely recognized by certain disciplines yearning for integrated sketching solutions like architecture. Moreover, it is further unclear why

some of the lessons and functionality developed for and offered by *Natural Sketch* have not been used or replicated elsewhere.

3.3 Other Notable Mediums

As mentioned earlier there are other technologies being developed that directly impact the world of 3d drawing. While this thesis focuses on 3d drawing through pen and multi touch input, in this section we provide and very briefly comment on a few notable examples using other technologies like geometry capture and 3d printing for reference.

3.3.1 Sketch Furniture by FRONT

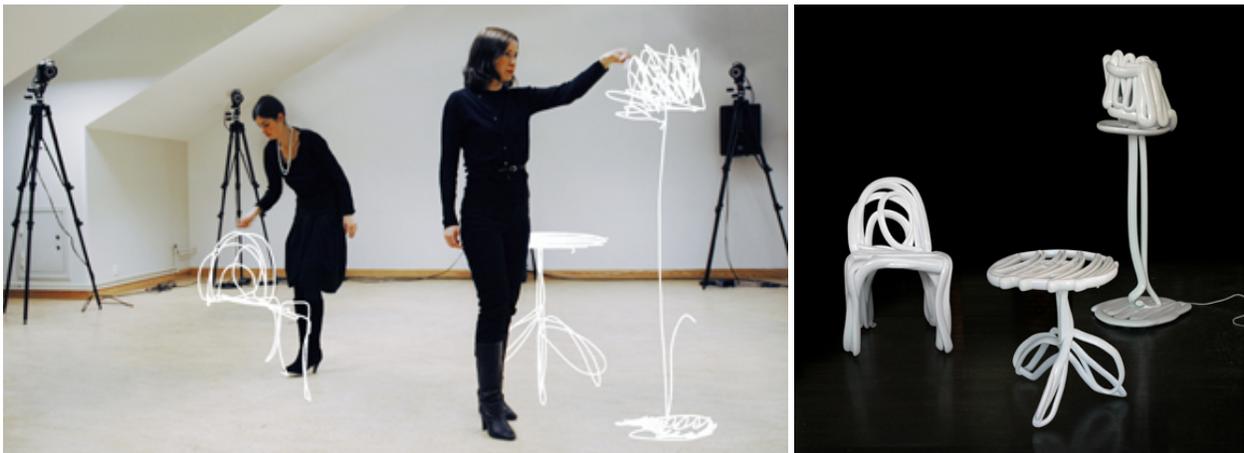


Figure 3.14 On the left is an image demonstrating the process of drawing the sketches with motion capture technology and on the right are the 3d printed outcomes.

In 2007, a Swedish design group called FRONT “developed a method to materialize free hand sketches. They make it possible by using a unique method where two advanced techniques are combined. Pen strokes made in the air are recorded with Motion Capture and become 3D digital files; these are then materialized through Rapid Prototyping into real pieces of furniture.”^{xix}

3.3.2 3-D Printing

There have been a number of hand-held consumer 3d printers that are fashioned after a pen and let you draw with extruded plastic. Some examples include 3Doodler, 7TECH 3D Drawing Pen, and GENESIS 3D Printing Pen.

3.3.3 Gravity Sketch



Figure 3.15 Above is an image depicting the original prototype of the Gravity Sketch system which features an arbitrary physical plane and a stylus, the combination of the two allow the user to position a plane in 3d space and input points on that plane. The user here has created an outline of a pair of sunglasses using this technique.

The makers of Gravity Sketch have stated their goal to make 3D content creation easy fun and intuitive. They are trying to appeal to the widest consumer base for the widest number of applications. That is why in the past number of years they have made strategic partnerships and developed multi-platform apps to take advantage of most of the current hardware that exists for the 3D market. They have software that integrates with iPad's and touch screens as well as commercially available Virtual Reality headsets such as the Vive. In terms of output, they've integrated with both sketchfab, a web service for viewing

models in 3D and Shapeways, a commercial service for direct to home 3D printing. Some of the differences they have from our approach is they are not interested in targeting specifically to architects and so they are not conscious of the architect's needs. They aren't focusing on lineweights, trace paper, iteration, or the process of constructing a model. Their goal is to make it easy enough for an eight year old child to pick up a tablet and 3D Print a toy. As a consequence they are choosing to forego a set of features that Architects would consider necessary for them to be able to use the software in early stage design.



Figure 1.3.4 The current version of Gravity sketch has an iPad app version that allows the user to create 3D Geometry using pen and touch very intuitively. The image above shows a user creating a 3D Model of a chair on the iPad Pro with Apple Pencil.

3.3.4 Tilt Brush



Figure 3.16 A view of Google's Tilt Brush, a 3d painting application for the HTC Vive ^{xx}

Tilt Brush is a 3d virtual reality drawing and sculpting application developed by Google for the HTC Vive. As one of the first consumer grade sketching and drawing apps for virtual reality, it is in a class of its own. The tool allows users to draw 3-Dimensional scenes simply by positioning their hand and pulling a trigger. The XYZ position of the controller is tracked by the system's base station and reported to the computer. This allows the user to access the 3rd dimension simply by pushing the brush away and towards the body. No longer is it necessary for the user to translate 2-Dimensional strokes from an arbitrary plane into 3D. The user then uses the virtual reality headset to walk around their creation. The drawing is fixed to a 3-Dimensional world coordinate system, while the headset provides a window into that coordinate system. This allows the user to interact with their drawing in a spatial context. The experience is akin to sculpting without material constraints or gravitational constraints. Drawing aids that are not constrained by gravity have also been incorporated in order to experiment with novel ways to draw. One such example is of a mirror plane that reflects any part of the drawing on to the other side of the plane. You can grab this plane and position it anywhere in 3D Space. You can also tilt and spin the plane so that whatever is getting

reflected follows the rotation of the plane as it is drawn. This makes for a truly unique drawing experience that allows individuals and users to create inherently 3-Dimensional creations.

Tilt Brush, in its current form, has some limitations. Firstly, drawing is uncomfortable and inaccurate because your hand is not resting on a surface, which makes the experience feel more like painting on a canvas or spray painting a mural. Currently, there is an inability to change scale, and the content being created is currently only for consumption through a vr headset or as an animation. Lastly, the quality of drawing doesn't allow you to make engineering shop drawings for the use of digital fabrication.

3.3.5 Oculus Quill and Oculus Medium



Figure 3.17 A view of Oculus Medium, a 3d sculpting application for Oculus Rift ^{xxi}

Oculus Medium is a Virtual Reality sculpting app. Similar to Tilt Brush, Medium uses a Virtual Reality headset and Oculus Touch controllers to orient you in a 3D space, allowing you to physically sculpt virtual objects. The main advantages are its accessibility, ease of use, and gentle learning curve, especially when compared to traditional keyboard and mouse 3D modeling applications. Oculus Medium uses a spatial

User Interface in much the same way Tilt Brush uses one, allowing you to switch tools the way a painter would do with their palette. One of the main differences between Tilt Brush and Oculus Medium, is Medium's, for lack of a better word, medium. Oculus Medium treats the creation of 3D objects like sculpting with clay. Created on principles of Voxel-Mesh based modeling, the program allows the user to push and pull surfaces away from each other much in the same way a sculptor might carve, press and kneed clay in to a subject's likeness. Medium also has an interesting education and tutorial model where it's possible to bring instructor and student together in a virtual room, allowing them to interact and sculpt with one another in the same virtual space. The software shows a lot of promise and its potential is extremely apparent to character animators and sculptors. In order for the software to be used by Architects, however, there would need to be a lot of industry specific development. It is unclear whether the software would be able to handle models of an extreme amount of geometry, and the freeform direction of the tool would only satisfy a small subset of designers. Buildings, after all, are not sculpted from clay.

4 HUMAN COMPUTER INTERACTION

When creating software for the architecture industry it's important to survey and take note of all the different ways certain hardware is used in the architecture industry. Devices that range from computers that are mobile and have positioning capabilities for job sites, to others that allow for the creation and manipulation of different three dimensional objects. Our thesis is focused mainly on the ability for the touchscreen display, when connected to a computer with fast enough 3D capabilities, to aid the architect in the creative process. To understand how these technologies are affecting the field of architecture it's important to comprehend what these hardware devices can do, and what they used to not be able to do until very recently. This chapter will touch upon the many different tools that are in use today and how they are connected to the system that was developed for this thesis. This chapter will focus primarily on the challenges and advantages of using a touchscreen display with pen input, especially when manipulating three dimensional objects. This chapter will also talk about devices that take advantage of 3d content by manipulating the data and displaying it in unique and innovative ways.

The first technology worth discussing is the ubiquity and ease of use of capacitance touch displays to the point in which they've become the dominant mode of input. The most prolific computing devices currently are mobile phones with touch screen displays. These displays have not yet completely usurped the market of the desktop computer because of form factor and scaling issues. However, new users of computers are becoming familiar with touchscreens before they become familiar with mouse and keyboard interfaces. The second technological advancement that is contributing to an ideal time to release this software is the increased power of processing, specifically regarding GPU's and graphics processing, allowing us to manipulate and view 3-dimensional content at interactive rates on practically any device. This convergence has both created a demand for 3-dimensional content, and created a demand for software that is used using a touch display. While these demands have occurred in opposite directions, higher power processing for desktop interfaces, and touchscreen interfaces for mini-computers, the two technologies are progressing towards each other. Processing is becoming more efficient, and available for

smaller computers, while touch screen displays are scaling up to desktop size. In addition to these two primary drivers for the technology, this chapter will also talk about an emergence of a hardware ecosystem that is more reliant on spatial computing it requires 3-Dimensional content to be useful, and there have been many means of acquiring this 3-Dimensional data, one of which is through modeling.

Touch screen displays are integral to this thesis. The current generation of touch screen displays rely on a technology known as capacitive touch. This technology does not require a stylus to interact with the screen, and it can process multiple inputs at the same time. This allows for several different modes of interaction for computers in the form of gestures, and touch interactions. It's important to discuss how this technology works, and how it can improve human-computer interaction because it is a core driver behind the thesis. Direct input is necessary for this project, it is what allows users to interact freely with this system, and it lets the computer acquire sketch data much more effectively than ever before.

4.1 Hardware for Architectural Design

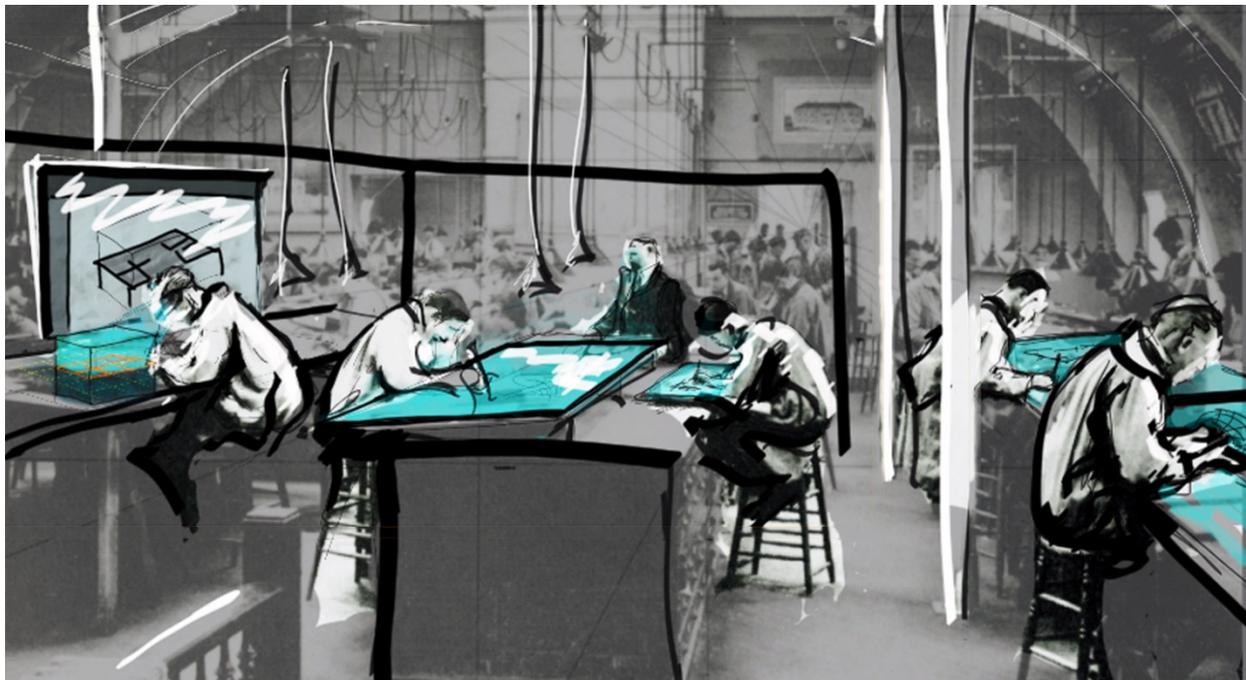


Figure 4.1 Image of the future of Architectural Design superimposed over a historic image of Architecture students drafting in one of the first Architecture Studios at Cornell University. The image depicts touch screen interfaces of all sizes, as well as a mixed reality headset experience, with a touch screen display.

This next section describes some of the history of touch screen displays and how they've been used in the past, in typical computing environments and specifically in designing contexts. Touch displays have existed for a few years. One of the first examples of software on a touch screen display was demonstrated by Ivan Sutherland in 1963 with the creation of Sketchpad. While Ivan Sutherland's program was, groundbreaking and led the way for many different advances in graphical user interfaces and the field of computer graphics, the world would have to wait 50 years before computers of this caliber and capability would be universally available and affordable. In addition to affordability, there were many technologies that increased the practicality of such a system, including faster graphics processing, and thinner display technology. Some of the first examples of ubiquitous touch display technology occurred on smaller devices. Screen real estate was at a premium at that point, and the technology was not scalable to larger display sizes. Once touch and stylus technology began to expand to larger displays, it became apparent that drawing would be one of the first use cases. Wacom tablets and displays took advantage of the processing power from other computers and acted primarily as peripheral devices. They were specifically used by artists as secondary devices. Add on to the fact that the Wacom displays could only handle input from a stylus, and they couldn't handle multiple simultaneous inputs. During this time period there exist some prototypes and proofs of concepts for modeling interfaces using the Wacom device as an input. One example of this is the tool Modeling Animation with Wacom.

Although touch panel displays had been prototyped and created using acoustic technology, total internal reflection, and a host of other techniques, it wasn't until the combination of LCD displays, and touch capacitance that Apple popularized the touch display when it debuted the iPhone in 2007. One of the last major advancements to push the use of touch display technology to the forefront of computing was the responsiveness of the graphics. The content on the screen would immediately react to a touch. Design-wise, it was the incorporation of a language of physical objects that made the experience even more enjoyable. Adding inertia to the system made it so that users believed they were sliding tablets of data on ice. In addition to inertia to keep the plane of information scrolling, and friction to slow it down over time, the interface also had an elastic collision when the list a user was scrolling through would

reach the end. This way the interface had the appearance of something physical and natural. With the advent of multi-touch displays, new gestures entered the realm of universal vocabulary. A pinch to zoom in and out became so ubiquitous that babies in the year 2016 would pinch newspapers and television screens thinking that these sources of information could zoom in and out in the same way. Rotation was another command added to this new lexicon. Both interactions needed no explanation or user manual, and became the default mode of interaction very quickly.



Figure 4.2 Photo of a man using the 2016 version of the iPad Pro with the corresponding Apple Pencil. The device is larger than its predecessors, portable, higher resolution, and has a faster processor. The Apple Pencil has multiple sensors built in to measure tilt and pressure

Apple continued its consumer market success when it brought its interaction design and its hardware to ever larger displays. The iPad debuted in 2010. The iPad allowed multi-touch functionality on a larger format display. This encouraged the creation of content on a multi-touch display, whereas the iPhone was primarily used to consume content while on the go. Thus, the multi-touch device shed its primary role as a communication device, and became a device for creation. With the introduction of the iPad, pioneering companies started to design styluses for the device to turn the iPad into a tool used for

drawing. The creation of the first Apple created iPad drawing device came out in 2015, with the iPad Pro. The iPad Pro was a larger format iPad, with a high-resolution display. The resolution of the display used to not be an issue for viewing content from a safe distance, but drawing pushed the specifications of the iPad to another level. For tablets to match traditional pen and paper, screens and styluses would have to be capable of thinner lines. When the Apple Pencil was released, it allowed for several new interactions involving much more than pressure and position. Pressure and position were taken as givens, and the Apple Pencil expanded into tilt as well.



Figure 4.3 Photo of a man drawing on the 2016 version of the Surface Book by Microsoft. The device has a special hinge and detachable mechanism design that allows the screen to be folded backwards over the keyboard to allow for higher processing and touch/pen interface.

Microsoft debuted their first personal computer designed in house to compete directly with the iPad. The Microsoft Surface was built on technology developed by Jeff Han years earlier. The devices initially used Total Internal Reflection. The Microsoft Surface debuted in 2012, and it was marketed as a hybrid tablet computer. Microsoft continued its success with the hybrid model by introducing the Surface Studio and the Surface Book in subsequent years, each with faster computer processors and graphics

processors. Included in the development of the surface line were larger format surface displays, at the 55” and the 80” size. The 80” size has been upgraded to a higher resolution, while the 55” size has not. This thesis was generously supported by Microsoft, as a lot of the hardware we used to test the software was donated to the Program of Computer Graphics. The 55” device was the device used initially to prototype this software. The 55” device was originally designed to be oriented vertically. During this thesis, it was exclusively mounted at the orientation of a drafting board. At the time the drafting board did not have a high enough resolution to support drawing for architecture. At the time of the writing of this thesis, Microsoft has released the Surface Studio desktop PC with a sufficient resolution and a 28” screen size.

4.2 Interaction Design for Pen and Touch Interfaces

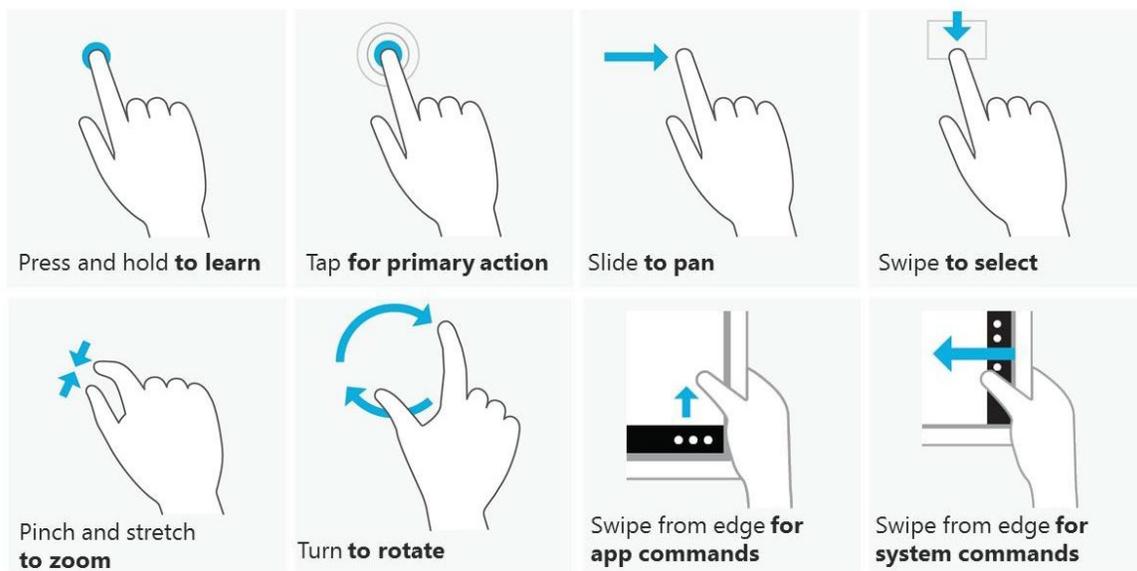


Figure 4.4 Thumbnails of common gestures from Windows 8 app development guidelines provided by Microsoft. They show how the difference in interactions changes the look and feel and positioning of menus on screens.

3D Modeling and conventional CAD programs have existed for traditional desktop systems for a number of years. AutoCAD was initially released in 1982, and although initially it did not deal with the creation of 3-Dimensional models, the current landscape of 3D Modeling software is mature. Due to the fact that touch displays for desktop computers are only starting to become ubiquitous or standard, the

landscape of software that allow for pen and touch functionality for 3D Modeling is limited. This poses an interesting experience design problem. Adapting an existing tool to be used with a fundamentally different mode of interaction. Going from mouse and keyboard to pen and touch is advantageous to the software design in many ways, but in many other ways it poses a complex design problem.

Microsoft's guidelines for designing for touch show how touch is a fundamentally different interaction than mouse. Some of the ways in which the pointer functionality is fundamentally different is purely in size. The mouse pointer is small and highly accurate and relies on a preview of the position of the click before the action is performed. When using touch, the finger is the preview. There is no recognition of the hovering position of the finger on the screen, and if there was it would be fundamentally uncomfortable to hold a finger above the screen for any amount of time. In addition to simply tapping on a screen, dragging is also fundamentally different. Due to the friction between the finger and the surface of the screen, it becomes really difficult to drag a finger across the screen a very short distance. Certain new gestures like rocking back and forth with a finger have been introduced to circumvent some of these initial issues. Taking advantage of these gestures is advantageous for a system like this one because in a lot of ways it allows users to more quickly interact with 3D-Modeling in a way that seems more natural and direct. For instance, pinching to zoom in to a model, or rotating a set of fingers to change a camera's orientation.



Figure 4.5 Microsoft's Universal Windows Platform is designed to easily allow developers to design and develop applications that work on devices of all sizes and interaction logic. The devices above increasing in size from the smartphone on the left to the Surface Hub on the right are all Microsoft devices.

In addition to fundamentally changing the way a user might interact with a screen with a pointer, touch screens also complicate the use of a keyboard. Keyboards take up the position that a touch display would take up on a desk, unless the touch display is uncomfortably oriented in an upright position. The alternative is that a digital keyboard is displayed on the touch screen. This poses several issues. In a lot of instances the touch screen keyboard is more difficult to use than a traditional keyboard because of the lack of tactile response, and feedback. In addition to the lack of a tactile component, the keyboard takes up valuable real estate on the screen itself, making it difficult to see the content that is being created and manipulated. Lastly, the digital keyboard does not work in the same way that a traditional keyboard works, specifically when it comes to certain modifier keys that would typically be used extensively in a 3D Modeling environment. These reasons make it apparent that 3D Modeling designed for pen and touch must be designed to be used only with pen and touch. The user can't be expected to use a keyboard or press a modifier key to access a different portion of a menu. This poses a challenging design problem as the keyboard has large number of keys that can be used to perform different commands. Some popular 3D Modeling programs make extensive use of command lines to access command routines. One of the ways we tackled this issue was incorporating a scrollable list where users can access many different commands.

Designing interfaces for touch screen devices also proves challenging because of the need to design for screens of different sizes. Menus and features now constantly fight for real estate on touch screens. The system described in this thesis uses some of the many techniques of hiding menus until their use is activated to create more real estate for drawing. In the future, high resolution touch displays are only becoming larger and more affordable. OLED technology is improving, making thinner displays at ever larger dimensions. AMOLED technology is also improving size performance and resolution of displays for tablet devices and phones. Advances in higher pixel densities and pressure sensitivity have also increased the quality and usability of touch devices. In addition to pixel density and pressure there are a couple of advancements in display technology that have allowed for "flat" touch devices, the first being the development of the ease of manufacturing of flat plate glass, and LCD technology which replaced rear projection technology. Flat plate glass being developed by Corning and Samsung increased

the ability for computer screens to go ever thinner and wider while LCD technology allowed for confined screen hardware.

4.3 Touch Screen Devices



Figure 4.6 Photo of a man using the Surface Studio by Microsoft which was released for the first time in 2016. The device is designed to be a non-portable desktop. It has more processing power, and a higher resolution and screen size. The stand for the monitor can pivot from a drawing position (shown on the left), to an upright position shown on the right.

The current state of the art for touch screen computers is at a crossroads. As powerful GPU's become more available and more affordable on one end, the high end of processing devices. Touch displays become more ubiquitous on smaller, cheaper and lower end devices. As they continue to progress, the point in which they cross is when touch displays and high powered GPU's will exist on the same device. Right now, the world is starting to see the emergence of devices that both have a touch screen and are capable of intense 3d graphics. What follows is a list of touch screen input devices with their advantages and disadvantages.

5 SOFTWARE LANDSCAPE

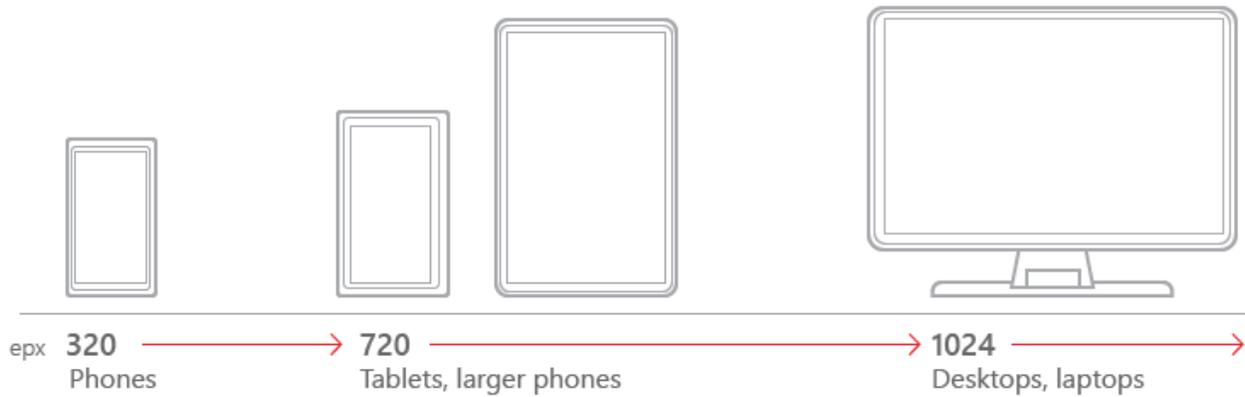


Figure 5.1 Our application has different modes depending on which device you’re using it with. The image above depicts a diagram that shows the “breakpoints” the points in which the look and feel of the software is different depending on the screen size and hardware.

The software system described in this thesis is primarily a Rhino Plug-in. It is also available as a standalone application, and a Grasshopper Plug-in. This chapter will go through the different skins, so to speak, of the application, and explain the thinking behind the many different variations. During the production of this thesis and the software application that resulted from this thesis there was always an acute awareness of the hardware being used to interface with the application. Originally the software was written exclusively for the 55” Surface Touch Display developed by Microsoft. Seeing as how this was a prototype device that did not reach commercial use, there were a number of variations that were developed and created simultaneously for other form factors. This simultaneous production resulted in three different applications. The first is a standalone application. This standalone application is lightweight, runs without Rhino, and can be used to draw splines in 3D. The second is a Rhino Plug-in. This application can not be run independently from running Rhino. All of the geometry information streams back and forth between both applications when it is running. The third and last application is a Grasshopper Plug-in that allows users to access the core libraries that were used to generate splines from pen input, and play around with the creation of these splines.



Figure 5.2 Here is an image of our application as a standalone app for a tablet device. The application is lightweight and allows for some of the functionality while being much more portable.

The standalone version of this software is a lightweight 3d sketching application that takes advantage of open source libraries to allow the user to sketch freely in 3D with a transparent background. One major caveat of the standalone application is that the splines are created with a large amount of control points. The standalone application doesn't have access to a curve fitting algorithm that can properly reduce the number of control points. One of the advantages that the standalone application has is that it is built using entirely open source libraries. The importance of using open source libraries in this portion of the project is that Rhino is a for purchase application. One can't use tools that were developed for Rhino without first purchasing Rhino, and one can't further develop the plug-in without keeping the Rhino API tightly intertwined in the production of the software. It was important, for the posterity of the project, to create a version that was decoupled from Rhino in order to let communities and students to continue developing the tool without incurring a cost or financial burden. Using open source libraries also

reveals more of the core functionality to the development of the tool, one can add features that Rhino can't support such as the creation and development of custom geometry shaders and stroke rendering. The application is lightweight enough, and it sits on top of a library that is compatible for web. In order to make the tool more robust it was not compiled to be used on the web, but this is something that it is firmly believed is going to be a large part of 3D creation applications in the future, multiple users signing in and collaborating on a project in real time. Moreno's thesis referred to the potential of this possibility, and it is currently a reality in many creation based sandbox style video games similar to Minecraft. Making a version of the tool compatible for the web and lightweight also lets the use of such a tool extend to tablets. This is especially valuable to architects who would like to use the software on job sites, either to make changes or to explain complicated 3-Dimensional concepts.

A large requirement for the project was to be able to make the background completely transparent. This allows the user to use this application analogously to transparent trace-paper on top of the internet. The trace paper aspect was a feature that was discovered in mischief and kept coming back to as a killer feature and need for architectural applications. Rhino is a commonly used 3D design software because it is easy to use and it can import and export between a number of different file formats. Architects need to be able to suggest changes and make marks on existing file formats and content that are all primarily visual. In order to ensure that this application would be able to load every 2-Dimensional/3-Dimensional file, the strategy was to make all files visible to the user while they use the application. This allows the user to very easily orient their model to the correct perspective of the content being viewed to make a quick sketch of that content. This is useful for 2-Dimensional images, but it's particularly more useful for 3 Dimensional content that can be viewed, but can't be downloaded and directly edited, like the models from Google Earth.

Sketching freely in 3d is handled by having the user define the plane or line that the pen's ray will intersect with in order to place the spline in a 3d environment. The form-factor of the software allows the software to force the user to manipulate geometry using touch and multitouch, and draw using the stylus/pen. This combination/form-factor of transforming a guide object in 3d and snapping the drawing to that 3d guide

object is one that is very familiar to architects and designers. The process is analogous to the use of French curves and t-squares, on traditional drafting boards. The description of how this interaction works is the core basis of the software, most of the other features expand upon this starting point and interaction.



Figure 5.3 Here is an image of our application as a Rhino Plug-in. The Rhino Plug-in version of our application is best suited for large format touch and pen displays. This is because access to a keyboard and mouse on one of these displays is relatively difficult, and so all the controls are modified to take advantage of pen and touch.

The Plug-in version of the software expands on the core functionality from the standalone while creating a near seamless loop between geometry created in this software, and geometry created within Rhino. This allows for more robust interaction, and is meant for use with a fast computer with a large amount of tablet real estate. The plug-in mode of the software opens up in a side by side window with Rhino. This allows the software to maintain many of the key aspects of the standalone version. Including the ability to have a transparent window, and custom lineweights. It also allows the thesis to demonstrate a seamless geometry loop between Rhino and the software, to expand the potential modeling functionality. In initial use cases, the process of switching between types of media and content more

seamlessly allows users to complete their work faster. The process is very fluid. As a consequence of being tied into Rhino, the plug-in mode allows users to leverage commands and functionality in Rhino to more easily go from a sketch to a model. This includes lofting, sweeping, network surface, extrusions, and lofting curves. This mode also allows for the editing and annotation of existing Rhino models in a presentation or critique setting. In a format more akin to red lining, multiple users can more easily discuss a 3d Model at the pace of conversation. In traditional drawing with pen and paper, drawing can happen at the pace of conversation, allowing for an idea or a shape to evolve with the back and forth feedback of multiple individuals. 3D Modeling software is too cumbersome to be used at the pace of conversation. Typically it takes a user a number of years to get up to speed, and even then the speed doesn't match the speed of communication. With a sketching application, the drawing is happening immediately.

Another feature of the Plug-in is that custom line weights allow the user to more expressively communicate ideas that otherwise communicate very little when depicted in equal line weights. This allows the user to create quick 3d drawings in the interest of communicating a point. These quick sketches can then be quickly and easily turned into 3d Models for further development. This seamless geometry loop within Rhino allows the software to quickly integrate with other plug-ins in Rhino's ecosystem. Plug-ins such as Vray and Grasshopper. Vray allows users to very quickly render geometry that has been sketched, allowing the user to quickly visualize what an object might look like when it interacts with different lights and materials. With Grasshopper one can enter into an automation loop or a parametric workflow that properly iterates and generates feedback. The capability then allows the user to edit a sketch and have the whole geometry update. Again, this particular mode of interaction with the software, i.e. side by side with Rhino, can take many different forms depending on the form-factor of the hardware. Desk critiques (a common practice in Architecture), reviews, and general production modes can all be accommodated, and are specific to certain hardware configurations. The first configuration worth discussing involves the 55" Surface display, mounted at the orientation of a drafting table. This mode has many advantages, and a number of limitations. For desk critiques, this form-factor is perfect. Having two people hover over a design that they can manipulate and annotate on the fly is extremely useful and

entertaining. The experience feels more natural when compared to fighting over the mouse and keyboard or requesting that the designer orbit and zoom a 3D Model one way or the other. Simple interactions follow the same form-factor that people are growing more accustomed to with pinching to zoom and panning and orbiting with their fingers, while annotating and sketching with a pen is also very intuitive. The limitations come when a user is trying to go into a more isolated work centric version of production over the content. These limitations come from the fact that the tablet is not designed for use with a mouse and keyboard, while traditional 3d Modeling is almost entirely done through the use of a mouse and keyboard. The cure to such a dilemma is as simple as porting all of the commands from one mode of interaction to a mode that is more suited to a giant touch screen. This porting however involves a certain amount of translation, and even still, not all features or interactions can be expected to work the same way. Clicking with a mouse is almost always going to be more finepoint than tapping, and sometimes also more accurate than a stylus. And searching will always feel easier when one can comfortably type on a keyboard. Areas for improvement certainly include incorporating voice search functionality. However one can imagine a scenario where hearing co-workers around one another shouting commands to their computer screens would be frustrating.

The second mode is a traditional desktop form-factor with a touch/pen tablet similar to a wacom. This can also be achieved by docking a Surface Book with an upright external monitor with a keyboard and a mouse. This form-factor is particularly noteworthy because of many articles that cite that the Surface Book is extremely useful to architects. This trend seems to stem from the fact that being able to bring the laptop or tablet to the construction site, means that the architect would also like to use the same device like a desktop replacement, when back in the office. This form factor allows the user to have a hybrid mode of interaction which engages the traditional means of using the keyboard and mouse for 3D Modeling interactions while at the same time maintaining the capability to draw and interact with the 3D content by using the touch screen. This also allows the user to more easily interact with certain plug-ins in the Rhino ecosystem that are demanding of a more keyboard and mouse type interface. The second screen

augments this type of interaction and facilitates the production of automated and parametric style workflows.

The third and final mode uses an upright 80 inch touchscreen designed for presentations and meetings. The two-way touch allows for a user to use a smaller tablet to screen and edit their model while it is being presented to a group of people, and for an individual to sequentially walk up to the board and create annotations or change the viewpoint of the model in much the same way that they were able to, when using the board in a drafting table orientation. The presenting and co-creation of a model on a large touch screen also facilitates collaborative modeling exercises. Some brain storming sessions require that an upright board be the main focus for a small group of up to 10 people. This mode is also conducive for an activity akin to 3-Dimensional whiteboarding. This configuration could also be used in the classroom as a form of instruction. An instructor can explain the details of a 3-Dimensional concept by drawing on the screen and asking students to interact with the concept by either having them contribute to the diagram or ask questions about the diagram. Combined with the possibility to incorporate sketching in 3D with video conferencing or screen sharing, this mode could facilitate a lot of design meetings when there are multiple parties involved. A networked version of the same tool would allow people to not just see what others have created but edit those creations on the fly and to draw and contribute to the model at the pace of conversation.



Figure 5.4 Here is an image of our application as Grasshopper plug-in. The Grasshopper plug-in version of our software is most suited when the user has a fast computer, an upright monitor and a pen input device. This allows the user to interact with the pen based interface while at the same time being able to interface with Grasshopper.

The Grasshopper plug-in version of this software is a simple addition that allows users to create their own algorithms of translating 2-Dimensional curves to 3D curves, and 3D curves to geometry. This plug-in reveals the initial underpinning of the code so that advanced Grasshopper users can develop their own plug-ins for the sketching tool. This creates a positive feedback loop in which the software improves over time by gaining more functionality and uses. One example of an interaction that was created using this tool was a piping script, that can automatically turn 3D splines into 3D printable mesh geometry. The Grasshopper version has very basic functionality, it includes redo, undo and delete, and the strokes are input into Grasshopper with corresponding pressure values, and as a tree so that algorithms can be made to interact with sets of strokes.

6 CHARACTERISTICS OF A SKETCH

6.1 Understanding Drawing

Starting in the 15th century, intellectuals and theorists began developing the techniques and visual language necessary to study, understand, review, and communicate spatial concepts through drawings and models. Drawing and drafting on paper has since remained the generally preferred method of conceptual design exploration, and designers and architects have honed the necessary techniques to explore, understand, and represent three-dimensional space with this two-dimensional medium. Today, design and especially conceptual design, has evolved into a highly iterative process that usually starts with rough doodles, sketches, and other methods for expressive, quick, and inexpensive exploration before moving the work into a more precise and less forgiving environment: the computer.

It is important to understand that architecture did not begin as a drawing profession. Up until the 15th century, architects were largely artisans, “who labored and toiled on building sites, cutting stones, laying bricks and sawing timber”.^{xxii} During this time a very important man of the renaissance, Leon Battista Alberti, introduced a concept that would fundamentally change the role of designers and architects, and thereby establish the foundation for several concepts central to this thesis. One of Alberti’s core contributions was recognizing a distinction between the conception and description of an idea and its future execution and manifestation. In other words, architects and designers could now work through abstractions and representations rather than in the final mediums. This is incredibly important to recognize because it marks an important and fundamental shift, where the role and value of the architect/designer moved from being purely an artisan to an intellectual. Where the value of design no longer existed purely in physical objects, but in the intellectual property and original descriptions behind those objects as well.

Alberti’s idea that the physical objects should be entirely designed prior to being built, and that designers should be seen as the intellectual authors of the objects they did not actually make, was in time, as we now know full well, amazingly successful. This is how, in the course of the 16th century, architecture

became one of the three modern fine arts, or 'arts of drawing' (painting, sculpture, and architecture); and drawings, rather than bricks and mortar, became, and remain to this day, the main tool of the architect's trade. - Mario Carpo^{xxiii}

In the centuries following the renaissance, architecture as a whole evolved into and has remained a professional service that primarily sells the information required to create a building or object, rather than the building or object itself. The construction of the designs / the execution of the instruction sets created by architects has fallen to different categories of professionals, contractors and construction managers, and the de facto method of communication remains drawing. Hand drawing and drafting emerged as the de facto medium and language of choice for designers and architects to record, develop, and communicate their ideas (Figure 5.1).



Figure 6.1 This figure shows the traditionally labor intensive drafting process that designers embraced using large format papers and tools like trace paper, rulers, and splines to create smoothly varying curves.

Over time, different standards of drawings arose to improve not just the definition and communication of these ideas, but also their conception and exploration. Furthermore, these drawings and standards were developed and practiced through the dissemination of technologies arising out of the renaissance: from the rise of printing and the invention of copyrights and royalties to the spread of literacy, reading, writing, and drawing with pens, papers, and rulers.

This paradigm remained largely unchallenged, with minor exceptions, until the dawn of computation and digital tools. Even then, the new digital tools were clunky and difficult to interact with. The act of drawing digitally had to be interpreted by the computer using a mouse and keyboard, and the designer had to further abstract his or her intent through a series of commands and mathematically-defined geometry. Through this development in computation and digital tools, computers became increasingly useful for the latter stages of design. These tools allowed architects to manage larger sets of information and data. Therefore, they became more useful in phases involving construction documents and design development. These tools, however, did not improve or relate to drawing in the early stages of design. Early stage design remained within the domain of free-hand drawing.

6.2 Visual Reasoning: Drawing is a Language

As explained above, architects and designers have long been trained in the art of drawing, and drawing has therefore been fundamentally incorporated into their workflows and thought processes. A designer's ability to understand his or her ideas and mental images drastically improves as they are described with ink on paper. As a result, most design processes typically start with doodles, sketches, and even back-of-the-envelope diagrams. Frequently, to help with the creative reasoning processes, models of clay, cardboard, or paper are made, but even then, realizations and discoveries tend to be documented through drawing, and ultimately, as designers further develop their designs, their associated drawings will continue to grow in complexity and specificity. Manifesting ideas through drawing allow the designs to be communicated and evaluated through, for example, engineering calculations or aesthetic and humanistic considerations.

Drawing can therefore be seen, more than an art form, as a language, and languages are not just forms of communication, but ways of reasoning. Through language people can develop, communicate, refine, validate, and execute their ideas.

"That language is an instrument of human reason, and not merely a medium for the expression of thought, is a truth generally admitted" - Ken Iverson ^{xxiv}

During the conceptual phase, general ideas and intentions are first explored through a visual language in rough and ambiguous ways. Since most important design decisions are made in the early design stages, it is also extremely beneficial to be able to accurately simulate many of the engineering behaviors of the virtual designs during preliminary design. For example, structural analyses and earthquake design, daylighting studies, and energy consumption and efficiency can all be simulated and would improve early design decisions. These earlier studies arguably have the largest overall influence on the final outcome.

In many cases, the concepts and solutions explored are merely untested assumptions that a designer makes in the effort to move forward with his or her vision. These largely uninformed decisions, especially early in the design process, have immense impact on the final deliverable. The problems and issues that arise from untested assumptions are usually only discovered further along the design and production pipeline at which point they are too expensive to be fully addressed (and the solutions, as afterthoughts, don't tend to reach the project's full potential).

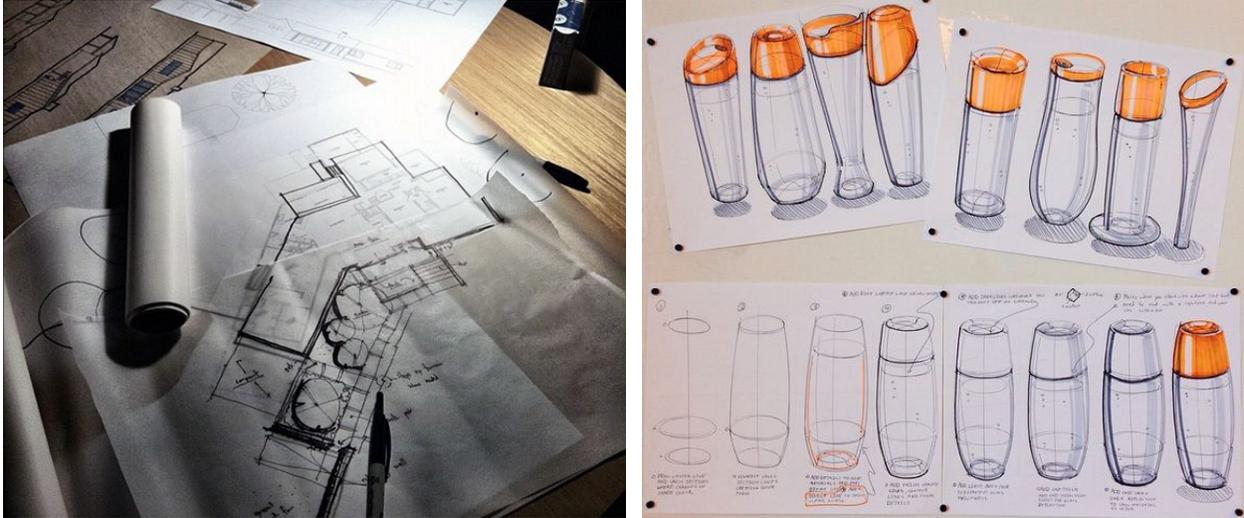


Figure 6.2 The drawing on the left shows iterations on trace while the image on the right shows a number of design iterations spread out along the wall in order to discuss the many different options explored.

It is not unusual for architects to place layer upon layer of tracing paper drawings on top of each other, modifying and combining portions of each layer to derive a preliminary design. Unfortunately, at this stage of the design the most important decisions are made without having the answers to the questions that a designer would like to ask, since they are too difficult to obtain rapidly and require too much specificity to obtain correct solutions. Therefore it is safe to say that architectural drawing has many stages, and many levels of specificity, detail and abstraction.

According to Michael Graves, an influential late 20th century American architect, architectural drawing can be broken up into three main categories: the referential sketch, the preparatory study, and the definitive drawing.

The first sketch is not the final the drawing, much like the first proposal is not a dissertation. There are too many unknowns, too many discoveries, and too many decisions that need to be made along the way. It is important to sketch out ideas through various stages of the design process. Some of these ideas may be well developed and others will only suggest the beginnings of an idea. Each idea further suggests various potential avenues of exploration, and only some fraction of these ideas can be partially or extensively explored within a project’s schedule and budget. The architect or the designer will iterate through these

drawings and ideas, creating relationships between previous versions and newer versions by overlaying the information. Trace paper and “Red-lining” are popular common tools that allow designers to draw and redraw over previous sketch or detailed drawing as they iterate, reference, and rethink past design decisions. This way of working through drawing also allows designers to communicate and collaborate over a design’s many issues and complexities.

An electrical engineer can quickly infer from a circuit diagram - by inspection, a short circuit, a misplaced transistor - what would be tedious and laborious to determine from a list of connections or verbal description. – Constable^{xxv}

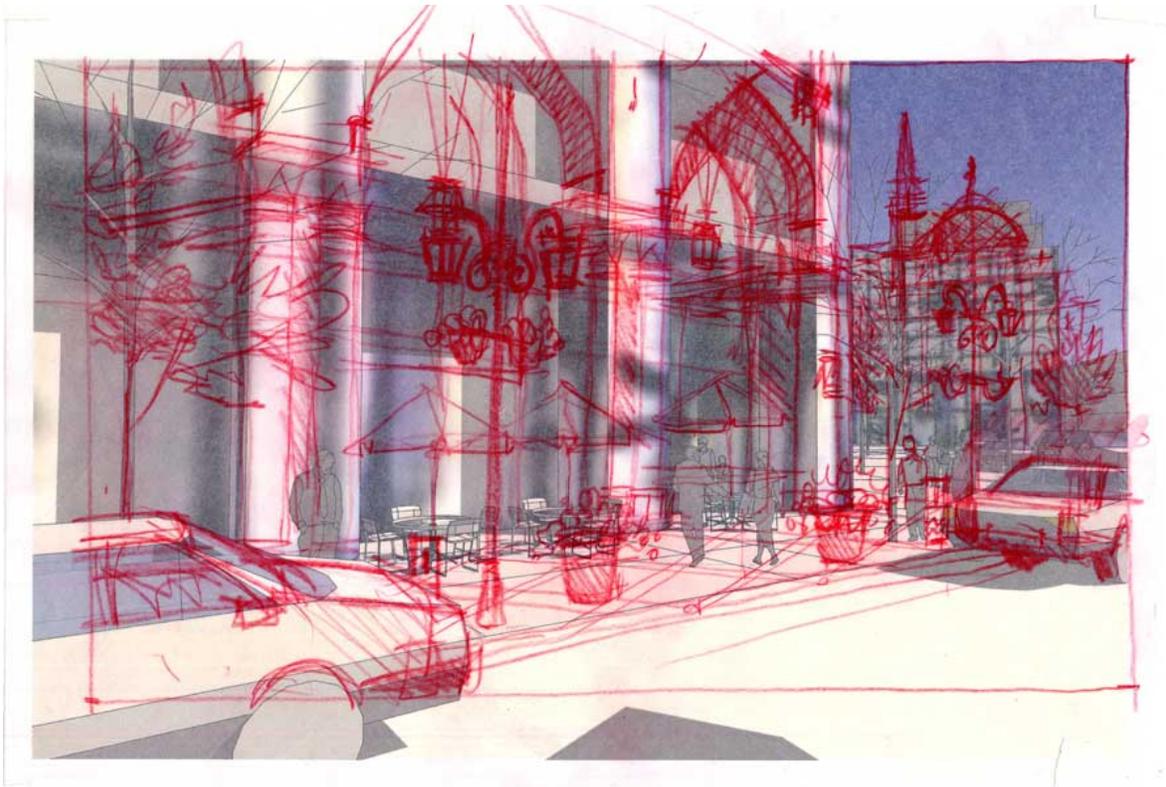


Figure 6.3 This figure shows a quick red-lined sketch over a conceptual Google SketchUp model. The sketch provides a wealth of information impractical to communicate through normal modeling techniques.

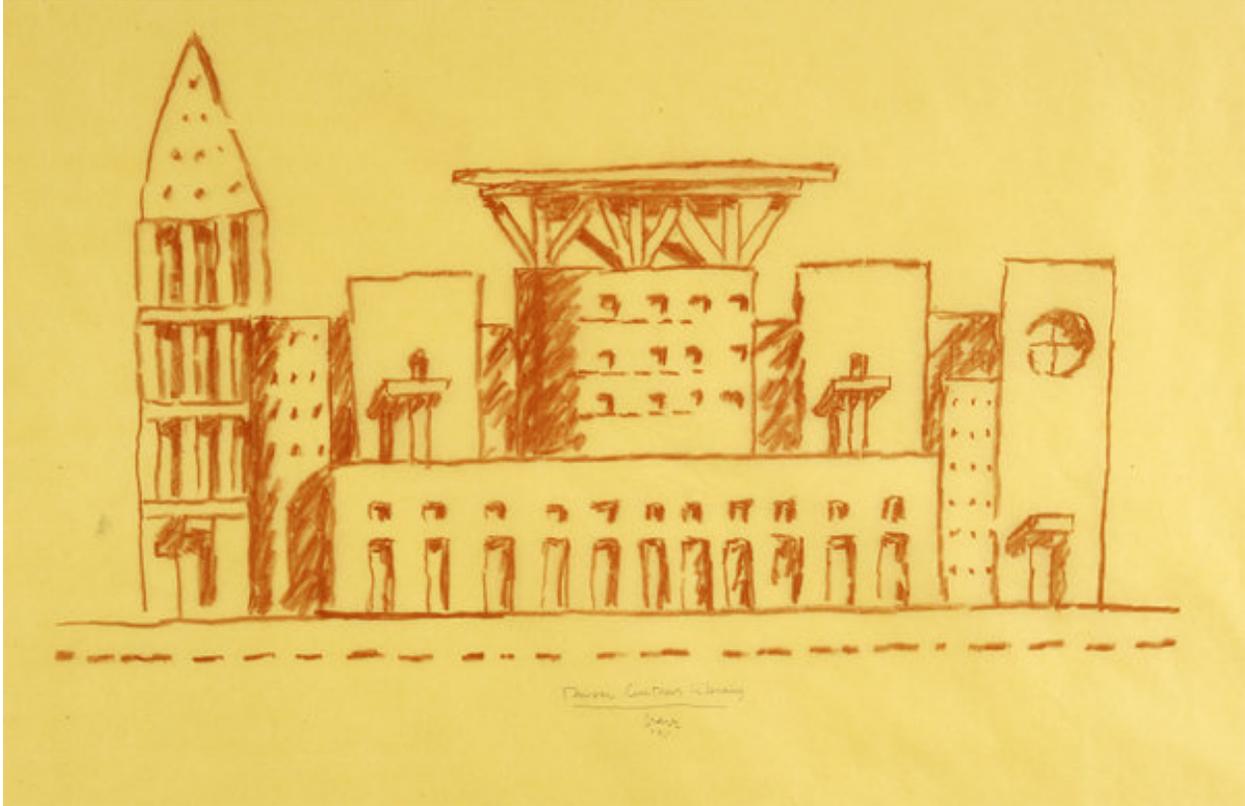


Figure 6.4 shows a freehand sketch of the south façade of the Denver Central Library by Michael Graves ^{xxvi}



Figure 6.5 shows a drafted drawing of the south façade of the Denver Central Library by Michael Graves

Drawing plays different roles throughout the different stages of the design process and is also used to move ideas and intentions back and forth between those stages; from understanding others' existing designs, to recording and developing one's own ideas, to voluntarily misreading and reimagining those ideas.

The act of drawing on paper does not simply involve an automatic transcription on surfaces of ideas that are already clear in the architect's mind. Working on paper is a way for the architect to mediate the act of making, following the Vitruvian precept that making architecture is "a continuous mental process completed by the hands." – Marco Frascari^{xxvii}

Very importantly, an early exploratory sketch for an idea must have the flexibility to be more open-ended than its later counterparts. This early sketch must be able to have a characteristic ambiguity that encourages interpretation, reinterpretation, and even misinterpretation of the concepts and solutions embedded in the drawing. In other words, it needs to facilitate the creative reasoning of its subject matter.

Of course, not having the full answer or even a partial right answer at first is expected or encouraged. The faster a designer can explore potential solutions, the more he or she can learn, and the larger the solution set, he or she can make decisions from, at this early yet influential part of the process.

Even then the exploration done on an idea is limited by the designer's training and technical knowledge. The criteria for evaluating the various design decisions is therefore largely skewed and quite biased. Is there a way to better inform designers throughout this process in a way that empowers them to make the best decisions? We believe there is. We believe that a computer can help a designer simulate and evaluate a design based on a number of criteria, without the design needing to be too developed. This is why drawing on a computer would allow for more informed decision making. The computer would not only allow the designer to communicate with more stakeholders and consultants in the process, but it would also allow the designer to use the computer to gain insight and knowledge about specific criteria of the design. In order to encourage designers to use the computer to draw, there must be an adequate reason or advantage. Drawing on the computer in early stages of the design process must be no more difficult to do

than drawing on paper. This thesis recognizes that a viable digital alternative to drawing on paper must look and feel good and must support pen and multi-touch input.

6.3 Characteristics of a Sketch

When developing a set of experiments focused on adding drawing and sketch functionality to a 3D Modeling environment, we needed a goal to determine whether the experiments were successful and a framework for evaluation. While the overall goal was to bring sketching within a 3D Modeling environment, this goal forced us to clarify what sketching is, and what are the characteristics of a sketch that we wanted to preserve, within a 3D Modeling ecosystem. The characteristics of a sketch that this thesis grew to focus on are as follows. (1) A sketch is abstract. (2) The act of sketching itself is easy. (3) Sketching allows for the ability to modify and iterate. (4) Sketches are drawn on a flat surface. Most importantly, we need the ability to unambiguously add 3D information while maintaining all of the above benefits of sketching environments. With these goals in mind we set out to create many different kinds of sketching routines within a 3D Modeling environment in order to test their usability and utility for architects and 3D designers. This was done to test whether sketching within a precise 3D environment would allow for novel forms of interaction and content. In testing these experiments we soon learned that building the software entirely within an existing 3D Modeling environment created restrictions and constraints that ran contrary to the characteristics of a sketch. All in all, simply building an add-on or (plug-in) to an existing 3D Modeling environment would not have allowed us to change the appearance of the strokes on the display. This development encouraged us to run another set of experiments in which we created a separate modeling environment that was fully synced with the movements, interactions, and geometry of a 3D Modeling environment. We learned a lot from these experiments including what was necessary for sketching to be a part of a 3D Modeling environment, and what kinds of advantages this type of interaction affords the user. We also learned where the “pain points” exist when working with sketch data and 3D modeling data and when creating a proper pipeline between the two. The following text describes in greater detail one of the methods employed in running these experiments. The 3D drawing system developed within the 3D

Modeling environment is named Cuttlefish, and the description of its functionality and implementation is briefly outlined below.

7 ADDING A 3RD-DIMENSION TO THE SKETCH

The driving interest behind this thesis was to explore how to leverage the full extent of 2-Dimensional pen and multi-touch input to create more intuitive and integrated drawing, modeling, and iteration workflow for the creation of 3D objects. To do this several experiments were undertaken in how to best understand the intended 3rd dimension using only 2-Dimension input. The unique aspect of this prototype is that the sketch is not only an abstract representation of a 3D model, **the sketch actually exists in three dimensions.**

In this chapter experiments and strategies for the following use cases are explained in further detail: (1) if the user is creating geometry in an empty 3D scene, the user is provided with several controls to position and manipulate the drawing plane in 3D; (2) if the user is creating geometry in a scene with existing geometry, the user is provided with additional functionality that lets him/her base new input off of existing geometry; (3) if the user not creating new geometry, but rather manipulating and transforming existing scene objects, he/she needs additional functionality for manipulation of content with pen and touch.

7.1 Important Features

The software offers four main ways of 3D drawing in space using 2D input: the user can manipulate orthogonal 2D planes within the 3D scene and then draw directly on the chosen plane (drawing plane); the user can define arbitrary drawing planes within the 3D scene; the user can choose from and draw on the surfaces of existing objects in the scene; and the user can draw on a plane with a normal that is parallel to the camera direction. These methods are slightly different between the two independent parts of these thesis projects and are therefore explained in more detail under section with the figures under this section 8.1.

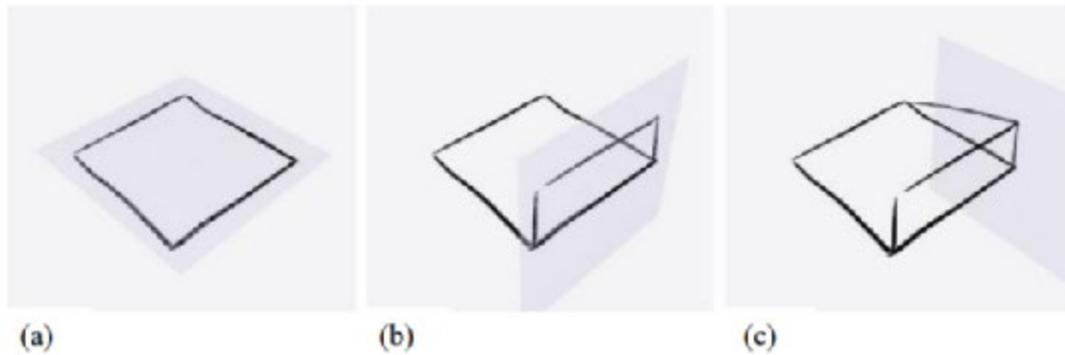


Figure 7.1 Sketching on Orthogonal Planes in Virtual 3D space (a) shows a line drawing sketch on a horizontally oriented plane (grey). (b) The abstraction is augmented by sketching on the vertical plane (grey) which has been interactively translated from the origin. (c) In a similar way, a sketched line of a second vertical plane is drawn. (This can be more easily seen in the video accompanying this submission).

7.1.1 Creating geometry in an empty scene

The first method is an orthogonal drawing tool consisting of three planes oriented perpendicular to each other, which allows the user to select the active virtual drawing surface (Figure 8.2). This is accomplished by a simple touch or tap on a specific drawing plane with the pen, which then displays the orientation of the virtual drawing plane but makes the other planes disappear. The three planes can be translated and rotated with touch interactions to help the designer easily draw in 3D.

The second method shows how the designer can easily construct arbitrary two-dimensional planes over existing scene geometry, such as a three-dimensional sketch (Figure 8.3) and/or model. Furthermore, to prevent difficult perspectives that present users with oblique and unproductive views of the drawing plane, the system automatically changes the drawing plane after a threshold by rotating it 90 degrees along its X, Y, or Z axis.

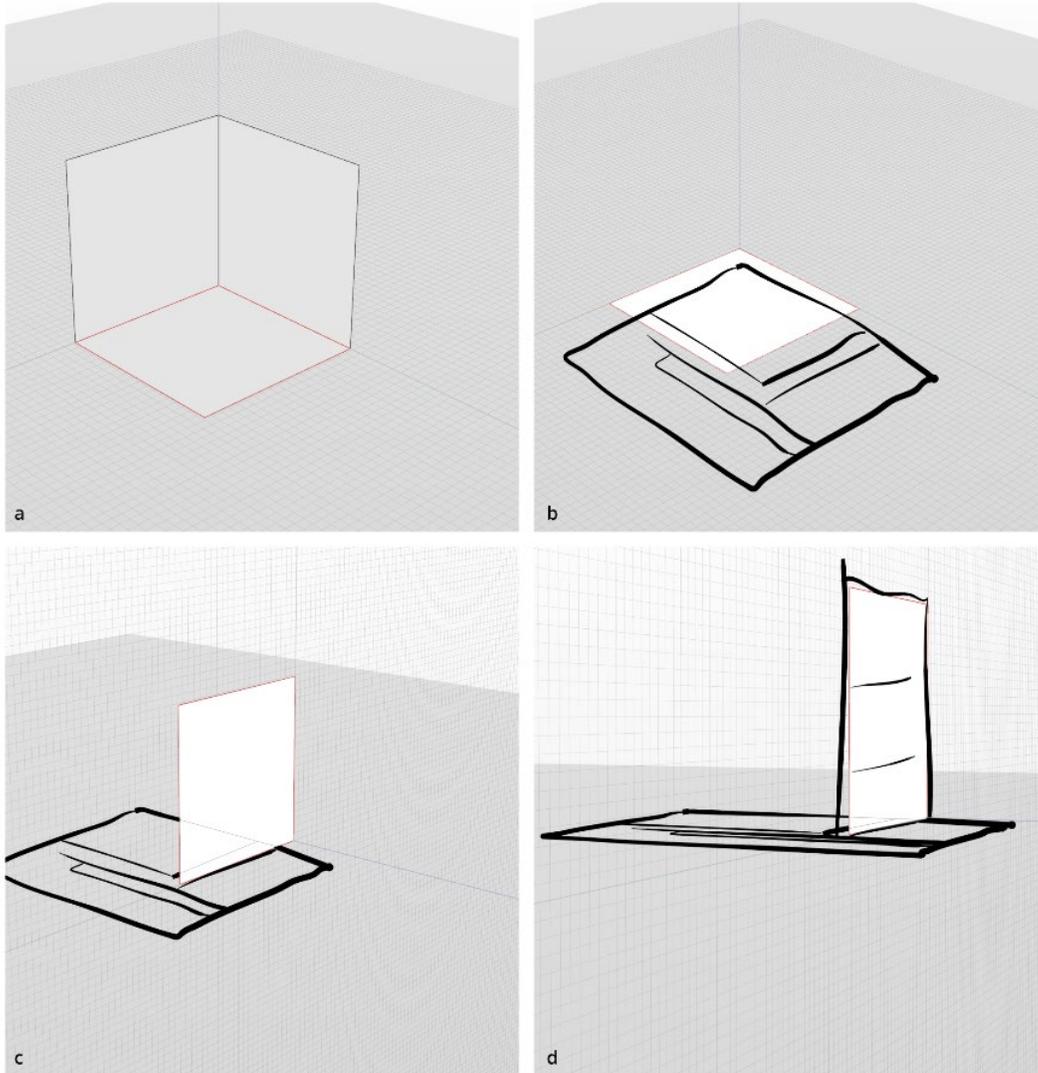


Figure 7.2 shows how hand sketched drawings exist in a 3-dimensional space. An “orthogonal drawing tool” is shown here that helps the designer work with 2D sketches orthogonally-oriented within the 3D scene. Quadrant a shows this orthogonal drawing tool; quadrant b shows how the designer can select and draw on a specific plane; quadrant c shows how the designer can move the tool to a desired location within the scene and choose the desired plane to draw on; and quadrant d shows how the designer can draw on this different plane within the 3D environment.

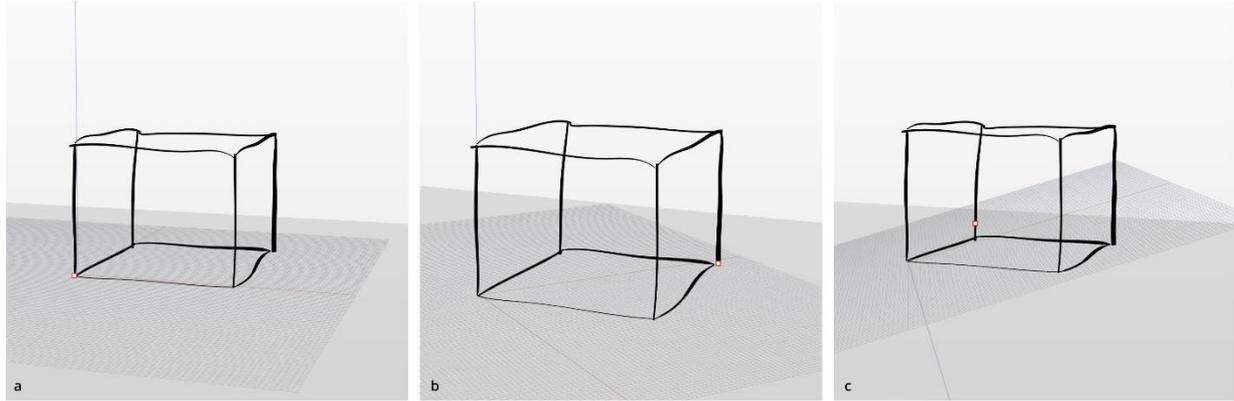


Figure 7.3 shows how the designer can construct arbitrary 2D planes from existing geometry in the 3D scene. There are three options for doing this: Figure a shows how the designer can construct a new drawing plane by selecting a new origin for the existing plane; Figure b shows how the designer can construct an arbitrary drawing plane by specifying a new origin and a second point along the y-axis; Figure c shows how the designer can construct an arbitrary drawing plane by specifying a new origin, a second point along the y-axis, and a third point that specifies the plane's orientation about the x axis.

A third method allows users to draw freely within the scene by using the line of sight from the virtual camera to define the orientation of a temporary drawing plane whose normal is co-linear with the view direction, and whose origin is remains fixed. (Figure 8.4). This feature attaches the plane of intersection to a plane perpendicular to the view and within the frustum of vision. In other words, when the user orbits the camera, the drawing plane follows, allowing the user to draw in 3D without having to constantly shift and manipulate the drawing plane separately. Since the plane updates with any movements to the camera, it provides the impression to the user that the user is drawing in 3D space and is beneficial for making annotations. Lastly, there are some minor variations on these methods. For example, one can specify that the drawing plane only rotates at orthogonal intervals so that one can draw in a more controlled manner.

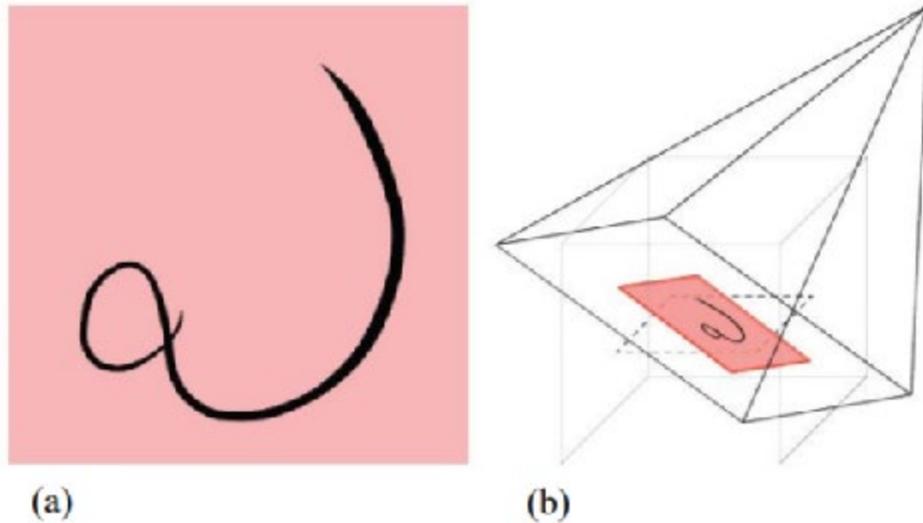


Figure 7.4 Drawing on a Plane perpendicular to the Line of Sight. When the user orbits the camera, the drawing plane follows, allowing the user to draw in 3D without having to constantly shift and manipulate the drawing plane separately. All the new strokes in this mode are perpendicular to the view plane of the camera and a preset specified distance from the camera. (a) The interactive sketch is shown on the physical drawing surface. (b) This figure shows the relationship between the drawing surface and the observer location at the apex of the pyramid of vision.

7.1.2 Creating geometry from existing geometry

One of the more important features that proved to be necessary early on is “snapping”. Figure 8.5 shows a variation of the third method described in section 8.1.1 with added snapping functionality so that the line of sight from the virtual camera is used to define the orientation of a temporary drawing plane whose normal is co-linear with the view direction, and whose origin is created by “snapping” to existing referencing geometry. Thus, the composite 3D sketch could consist of sets of sketching input on multiple orientations of three-dimensional planes in space creating a full 3D line drawing in virtual space and can be easily transformed into a watertight 3D model.

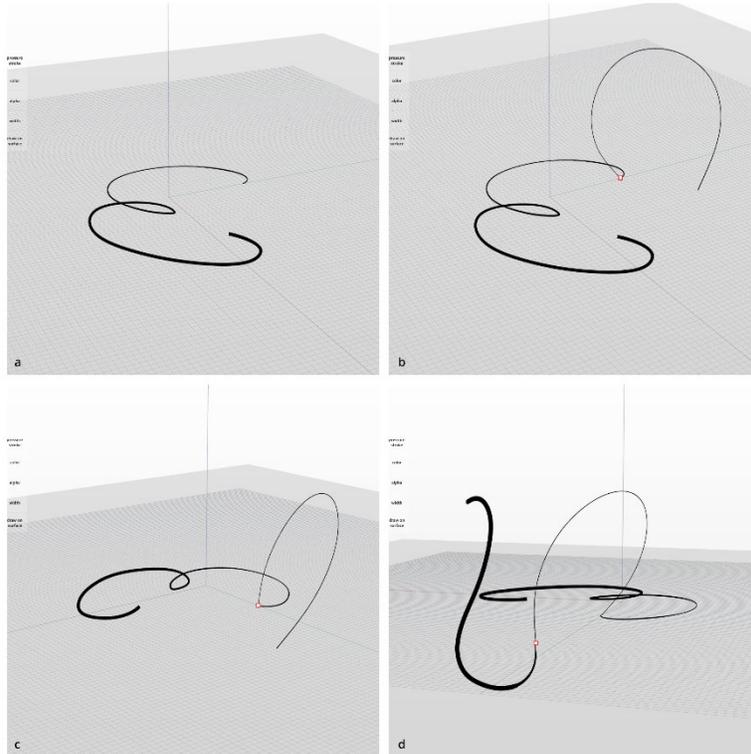


Figure 7.5 This figure shows a drawing method that uses the camera to define the orientation of a temporary drawing plane and object snapping to define the temporary plane's origin at a given z-value: quadrant a shows the original spline drawn on the XY plane; quadrant b shows a new spline starting from one of the end points of the spline drawn in quadrant a; quadrant c shows how the spline drawn in quadrant b was created on a plane perpendicular to the camera direction with an origin at the snap point; quadrant d shows how additional splines can be made in this manner with the new spline snapping one or both endpoints to existing geometry in the scene.

The software uses mathematical descriptions of NURBS to represent strokes, and the user can therefore also create NURBS surfaces from arbitrary sketches (Figure 8.6). The system first records the sketch of a curved line in three-dimensional space with standard digitizing procedures, then transparently converts the sketch into a Bézier spline so it has an accurate mathematical description. It then offers several approaches which can be used to define the surface. The user can extrude the spline, can draw a second spline and loft between those two curves, or can add two additional boundary splines to interpolate between the four boundaries that form a NURBS surface.

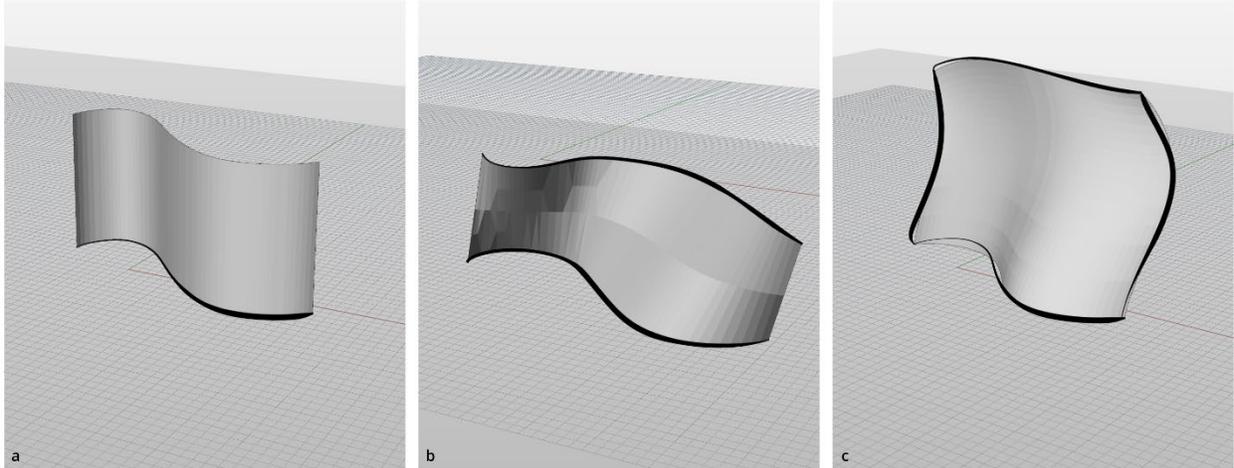


Figure 7.6 shows 3 methods for surface creation: figure a shows the ability to extrude a spline to create a surface; figure b shows the ability to loft between existing splines to create a surface; and figure c shows the ability to interpolate a surface from boundary splines.

The system also provides other forms of drawing that allow the user to break out the 2D drawing constraint imposed by standard 2D drawing planes. Since the user can select any existing geometrically defined surface and use it as the drawing surface, primitive surfaces such as a cylinder, cone, or sphere as well as any surface created through sketching can be used (Figure 8.7). After defining a geometric surface as a drawing surface, the user input's z is exactly at the camera ray's intersection with the geometry. Ultimately, any sketched splines on the chosen surface, can then also be used to alter the surface, for example by trimming it (Figure 8.7).

Consistent with the goal of bridging analogue and digital tools, the system provides the user with simple interactions that allow primitives and other surfaces to be used as drawing guides or rules for drawing. This functionality resembles the system's ability to manipulate and transform (translate, orient, rotate) the 2-Dimensional drawing planes to restrict input to a certain place in 3D space. Similarly, this functionality also resembles the ability to draw on a surface, but different in that the designer's input is restricted to the boundary of the chosen surface from camera's perspective. Users can use any 3-Dimensional primitive surfaces or to import their own surfaces or stencils.

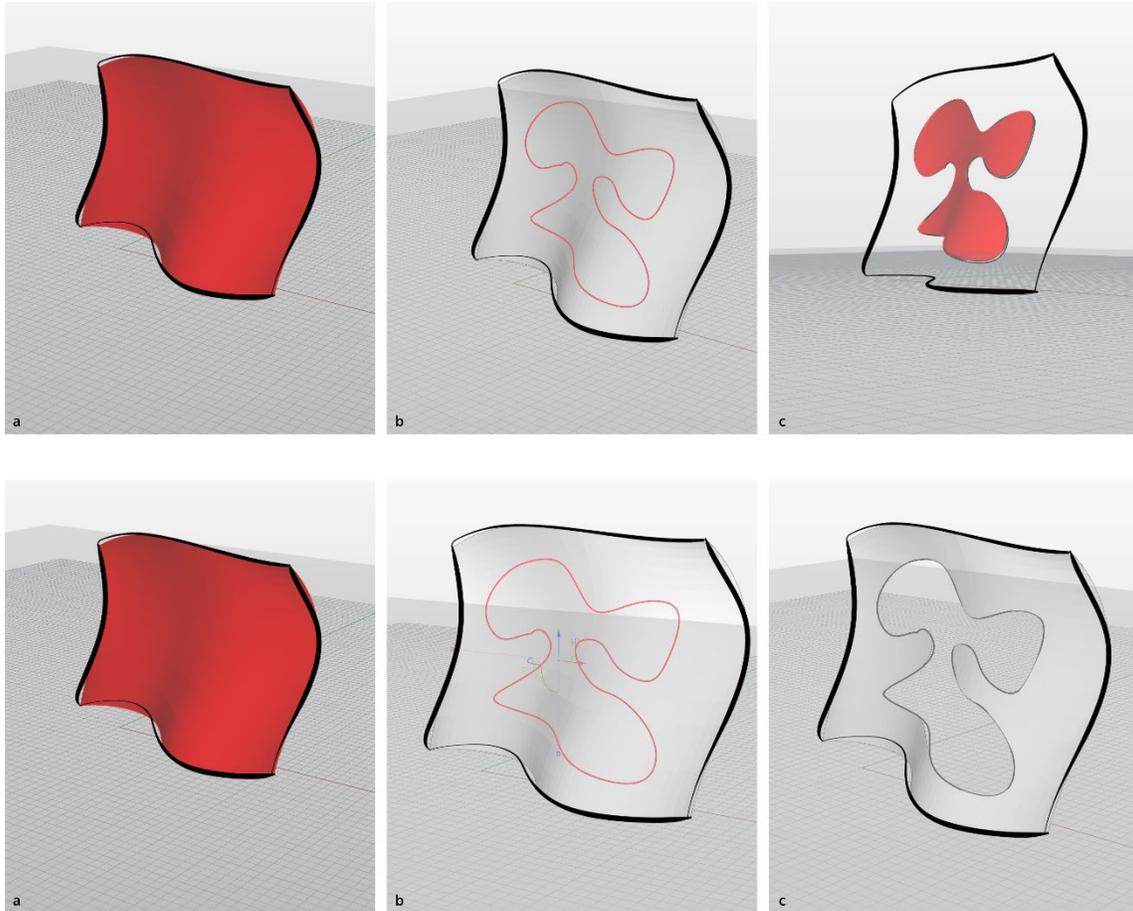


Figure 7.7 shows the ability to use an arbitrary surface as a drawing surface and to create new geometry by trimming the surface with the drawn geometry: section a shows the ability to select surfaces as drawing "planes"; section b shows a spline drawn on the chosen surface; and section c shows the resulting surface after the middle section has been trimmed with the newly-drawn spline.

A main goal of the system is to bring new functionality to designers' existing workflows, and to blur the traditional boundaries between analogue and digital tools. Drawing with guides and rulers on sheets of trace paper is among the most popular and influential of these tools and methods. The following are a couple explorations of the implications of these tools for 3D drawing.

This system provides a digital version of drawing on tracing paper that allows the users to more quickly iterate through concepts using 2-dimensional drawing within a 3-dimensional scene. A sheet of trace paper has a saved drawing plane and saved camera position that allows the user to sketch and trace from a given viewpoint. The digital trace paper provides two additional features that provide unique

functionality for 3D tools: first is overall window transparency that allows the user to trace over media from other applications; and second is the ability to layer the versions of these drawings as if they were actual sheets of tracing paper.

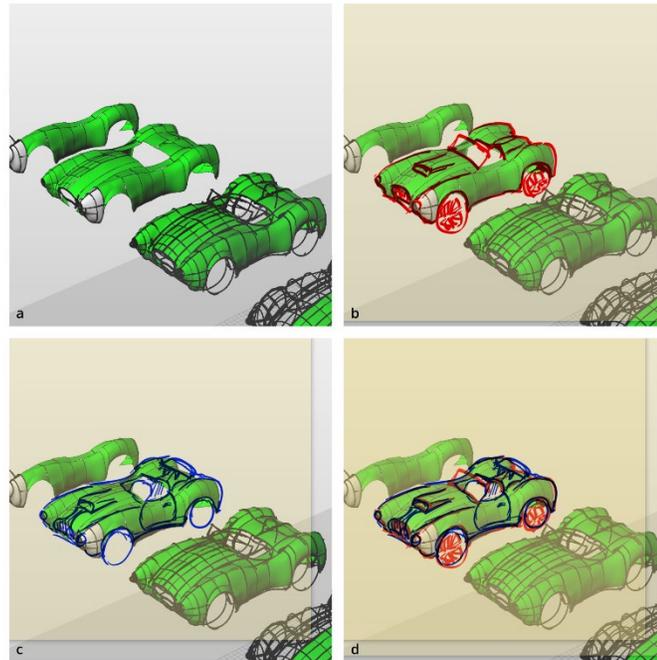


Figure 7.8 shows how you can add perspective dependent drawings over existing geometry within the scene. These drawings can be layered and draw a parallel with how designers work with trace paper using traditional analogue tools.

7.2 Implementation

Creating a new geometry and representation storage document and replacing the standard Rhino rendering pipeline also meant adapting a number of the shared interaction methods like selection and camera controls. Every action the user makes has to impact both data structures regardless of where the action came from. For example, if the designer selects a stroke in Cuttlefish and deletes it from the Cuttlefish document, that stroke also needs to be selected and deleted from the Rhino data. Independent Cuttlefish scene objects like cameras and drawing planes had to be created in the Cuttlefish document and the transformations made to match with the Rhino document whenever the user changes the view or the drawing

plane. This ultimately impacts the ways in which 2-dimensional input through a pen and multi-touch display is registered in Cuttlefish and in Rhino. Lastly, the additional functionality described herein is built over traditional and well accepted 3D tool basics such as the ability to translate, rotate, and scale geometry. Other standard modeling features like trim and Boolean operations are also available. Snapping new input to existing geometry (i.e. spline endpoints) also plays a key role in helping the user more cleanly define geometry in the tool

In the end, the designer is able to use the Cuttlefish system's improved stroke representation without fundamentally limiting the user's ability to interact with the geometry inside of Rhino because any modifications made to the Rhino data by the end-user would be reflected in the Cuttlefish document and vice-versa..

The hybrid Cuttlefish-Rhino system can be subdivided into four main parts: the core module, the input module, the rhino module, and the rendering module. The most important part of the implementation is the core module as it stores and manipulates all the geometry that is rendered through DirectX. This module also manages the current state of the application, including all input parameters (i.e. stroke type, pressure, width, color, etc.), drawing guides and planes, trace paper, and object visibility. The input module reads the strokes of the pen input as well as interactions with the GUI and calls the appropriate functions in the core module. The rhino module allows the core module to communicate with and use Rhino functions such as spline and surface creation and manipulation, nearest point algorithms, and object state linking (i.e. selection). Finally, the rendering module takes data from the core module and renders it in the window with DirectX.

The software was developed in C# because Microsoft's Windows Presentation Form (WPF) and McNeel's RhinoCommon API, an interface into Rhino 3D, are both offered in C#. The interaction capabilities available in Rhino3D were expanded through WPF, which offers pen and touch input libraries and natively supports window transparency. DirectX was chosen to render the geometry in real time since

it preserved the window transparency capabilities in WPF. To use DirectX in C#, a DirectX wrapper called SharpDX was used.

All the sketched geometry is stored and organized in two places: in the Rhino document, as well as in the Cuttlefish document's core module in groups called sketches. Each sketch stores splines, surfaces, and/or other geometry. Sketches can have their visibility turned on or off and they exist independent of any layers in the traditional modeling environment. There are two types of sketches in the core module: normal sketches or trace paper sketches. If a sketch is marked as trace paper, the camera location and a perpendicular drawing plane is saved along with the sketch.

7.2.1 Creating Surfaces.

Once the user has created curves, there are 3 main ways in which the user can use these curves to create a new surface. The first method is a simple extrusion, which defines NURBS surface by extruding a spline. The second method is a loft where splines are used as cross-sections of the new NURBS surface and the values in between are interpolated. The third method is known as "Network Surface." The user must select three to four curves that define a closed region within a given tolerance that depends on the zoom factor of the camera. The system then interpolates the curves until they meet at the ends and in between an interpolated mesh is created to define a surface.

7.2.2 Selection and Snapping.

The implementation offers two different types of selection: selection by tapping and selection by dragging. With selection by tapping, the system chooses the object closest to the place tapped, within a certain tolerance that is dependent on the zoom factor of the camera. A naive implementation was used to find the closest distance from scene objects to a ray cast through the tapped selection point and sorts these by distance from the ray cast line. Since we have not seen noticeable performance issues with running the function on points constantly for snapping, we have not yet implemented a more efficient algorithm such as B.V.H. (bounding volume hierarchy) or k-d (k-dimensional) trees. The selected object is the first one in

the list. Selection by dragging or box selection, uses the 3D modelers built in standard selection function to select the objects.

Snapping is incredibly important for the user experience and is required to draw water tight meshes. The version of snapping is an extension of selection by tapping where only the closest existing endpoint or midpoint to the ray cast line is chosen. This chosen point is then put at the front of the list of points used to create a new spline.

8 RESULTS

We experimented with different types of users when testing out our application. Some of these users were experienced in 3D modeling software and other users were design oriented and could draw but had never used a 3D modeling package before. We also took some novice users who had never drawn using computer software before. As we demonstrated the capabilities of the software, we would notice and fix bugs that were made apparent during the demo process.

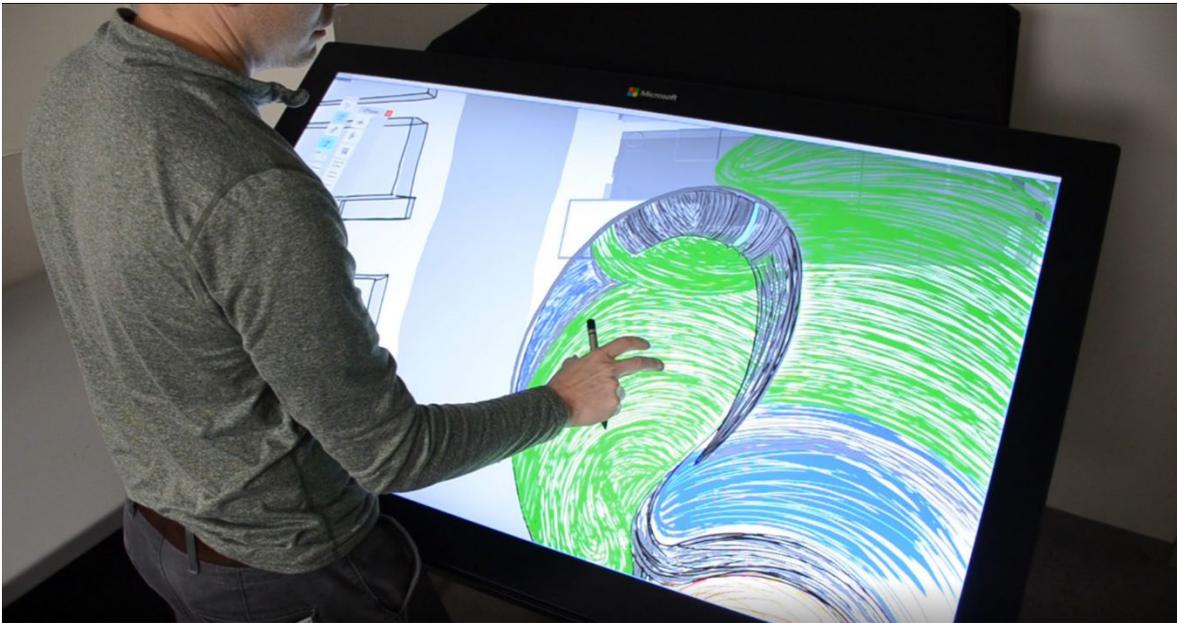


Figure 8.1 User experiments with strokes of different colors in Cuttlefish on the large Microsoft display.

For the users who were accustomed to 3D modeling packages, using pen and touch to navigate around the scene was a bit jarring at first, and there was an initial desire to constrain or create curves and lines that were extremely accurate. We instead encouraged all of the users to not worry about accuracy and instead to create a loose diagram of a 3D object.

Most of the unique value our system provides is revealed when the drawing and modeling environments merged. In other words, most users with a background in 3D modeling found being able to draw in 3D over site models and massing blocks of their designs very useful. The geometry the users input

was not directly used to create a water-tight model, but rather, to explore form and intent, and then model that in detail with a mouse and keyboard which is the workflow they are more familiar in.



Figure 8.2 User experiments with Cuttlefish and fluidly goes back and forth between modeling with a keyboard and mouse, and drawing on the display.

The same was true when starting drawings from scratch, users tended to explore designs too loosely and roughly to allow the resulting geometry to be practically converted into a 3D model. That being said, there were cases when users preferred modeling with pen-input and touch-input. This tended to be the case when creating looser and non-orthogonal models. Our “guides” functionality was not powerful enough to be practically used in creating precise drawings that could be transformed into clean models. This is an area for improvement.

Sketching in 3D on a large format canvas was particularly successful. Not only was the surface of the drawing board for the Microsoft Surface Hub large and spacious, but the 3D Modeling environment itself had an infinite space to draw within. This allowed users to develop their drawings without fear of running in to an edge of the drawing space. If the user found themselves reaching the edge of the viewport, the user could always pivot, pan and zoom to bring the focus of their drawing back to the center. Being able

to change the perspective, or viewpoint of the geometry really allowed users of the system to design and draw with less limitations.

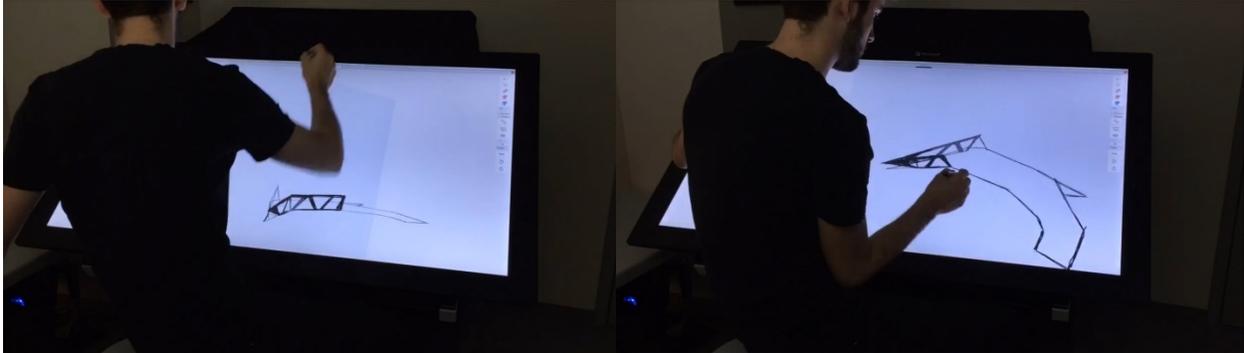


Figure 8.3 In this set of figures the user is drawing using Cuttlefish and has changed the orientation of the view in order to draw a different section of the object.

Users could opt to view and draw on models on a number of different devices. Using a tablet device to draw meant that the users needed functionality that was much more necessary for mobile purposes. Whether that be drawing during a site visit, or drawing a simple diagram to explain or communicate an idea.



Figure 8.4 This figure shows a user "red-lining" or editing on top of an existing 3D Model on a mobile Surface Book tablet.

Users also experimented with drawing on upright presentation boards. This method of using Cuttlefish proved ideal for users to collaboratively work through a design problem together, provide feedback, and or to present a given concept or idea.



Figure 8.5 This image depicts two users collaborating on an upright Surface Hub. The user in the background has the pen and is able to make edits on the fly while the user in the foreground has enough space to point and demonstrate areas in need of improvement.

What follows are a number of descriptions of objects that were created using our software and the techniques that were employed to create these objects. Most users were surprised to discover that they could create surface geometry from their input, but also recognized that in order to make an extremely accurate 3d model a lot of the geometry would change and adapt later on in the pipeline.

The first and simplest use case to describe is the creation of the 2-Dimensional sketch within Rhino. Previously, Rhino could support 2-Dimensional sketches, but these sketches lacked visible variation in line weight. One could adapt Rhino in such a way that a user could see a slight variation in

line weights, where each line is a fixed width. In our plug-in, users were able to create sketches that had large variation in line weights, with lines that changed in thickness along the length of the curve. This allowed users to quickly draw as if they were drawing in a sketchbook.

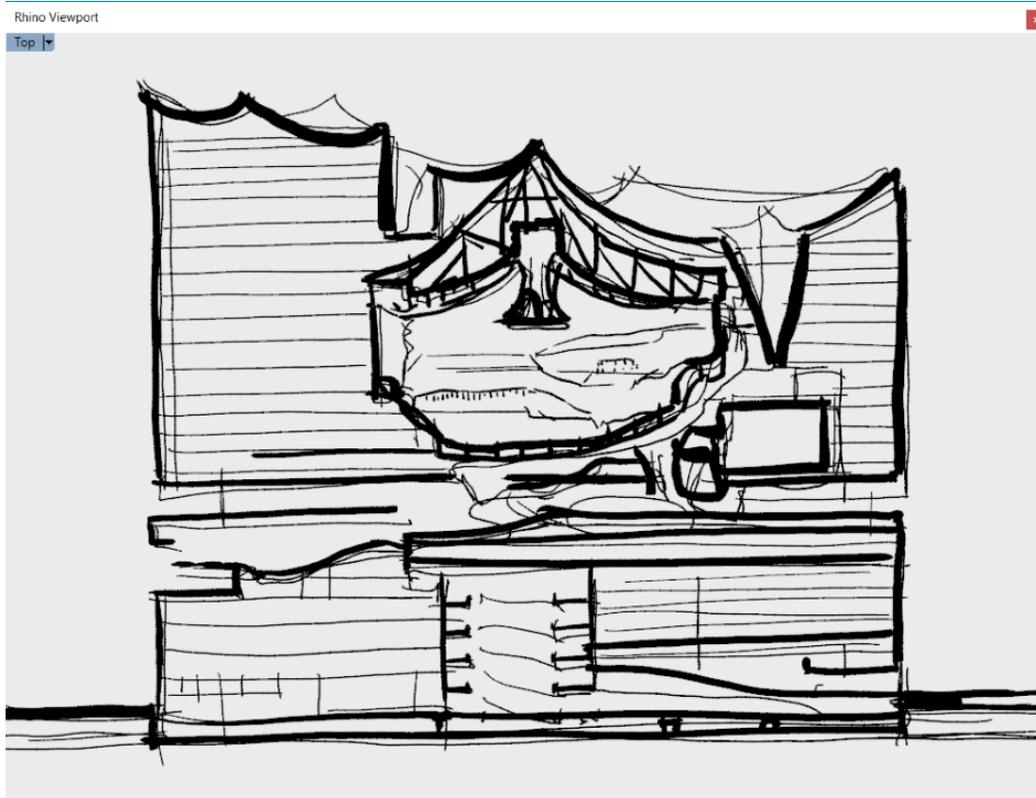


Figure 8.6 The figure above depicts a section drawing of the Elbphilharmonie in Hamburg, by Herzog & de Meuron drawn in Cuttlefish. It is shown here to demonstrate how the user can quickly sketch a drawing like this with many variations in line weights.

These new techniques of using Rhino allowed users to create complex drawings very quickly. A lot of users chose to create pretty complex sections, but the drawing platform was also capable of allowing the users to create plans as well. Whether a user was creating a plan or a section they could always orient

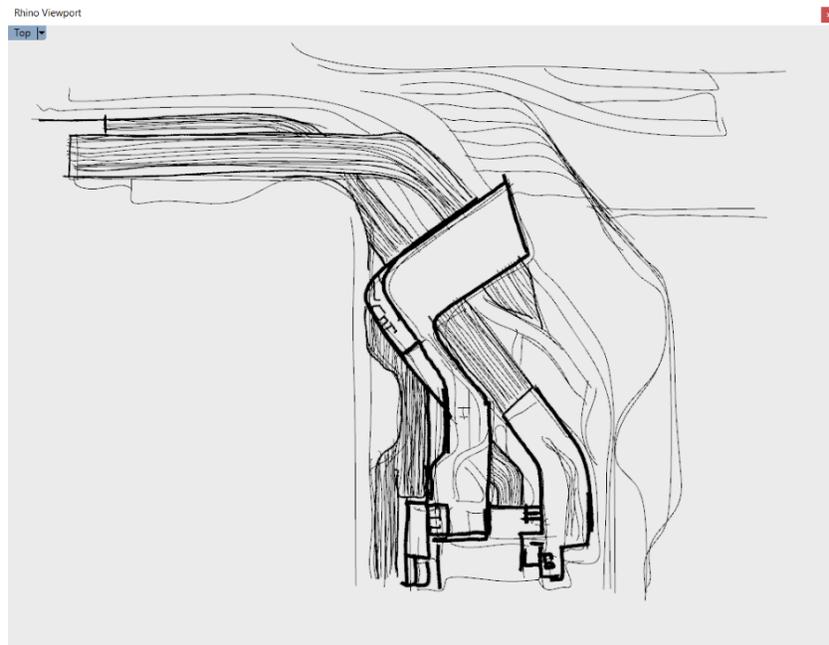


Figure 8.7 The figure above depicts a plan drawing of the MAXXI museum designed by Zaha Hadid Architects in Rome. The drawing was made in Cuttlefish and shows how figurative sketches and splines are quick and easy to create.

their drawing plane to the correct position so that the drawing could be used as a plan or a section once the user decided to add 3D information to the sketch. Users who added 3D information to the sketch did so in two ways. One way which was more figurative and faster, was a view dependent sketch where the final position of the viewer was known. This allowed the user to focus on creating a drawing that would look good from a fixed viewpoint. Using perspective and detail, they were able to create drawings that conveyed 3-Dimensional information.

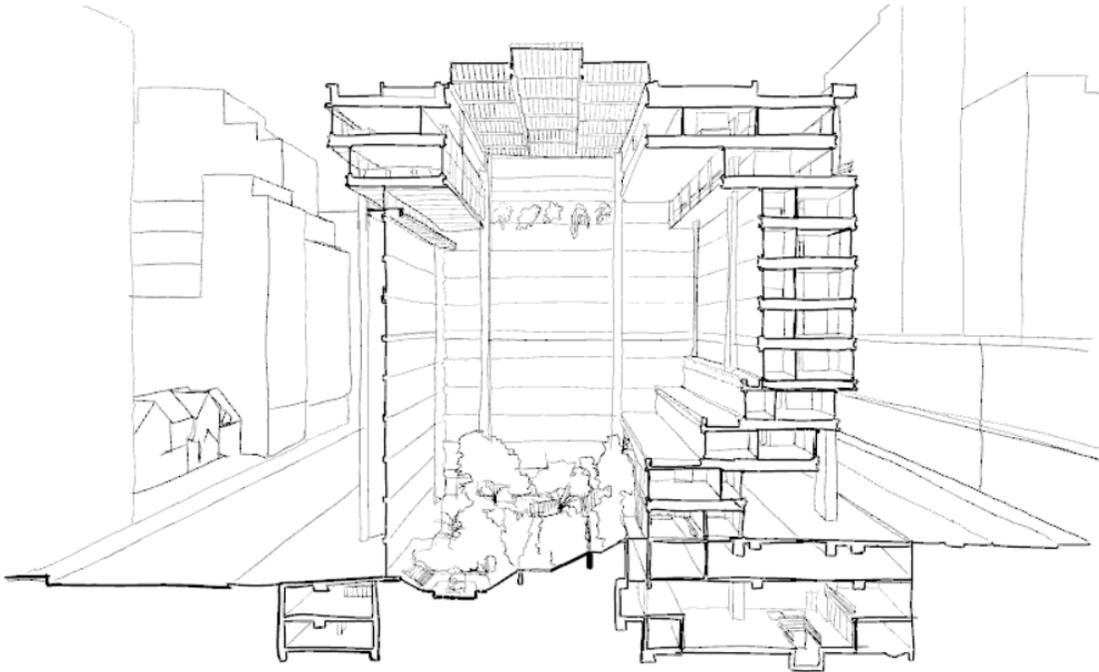


Figure 8.8 This figure shows a perspective section of the Ford Foundation Headquarters in New York City designed by Kevin Roche. This drawing was created in Cuttlefish and demonstrates how a user can quickly create a sketch from a single perspective.

In examples where the user drew primarily in 3D, without first picking a specific viewpoint, users were able to build up 3-Dimensional sketches in such a way that it allowed them to communicate special qualities of a design in a small amount of time. Interestingly enough, users were able to quickly realize that what they were trying to convey was an idea or a concept rather than a finished and complete set of drawings. This allowed for communication and collaboration to develop over the drawings as no one felt that the choices being displayed were set in stone. In addition, the drawings themselves were so easily and quickly adaptable, that critiques and changes could be made on the fly and at the pace of conversation. Even figurative elements such as trees were included in the sketch in such a way that they simply depicted where they would be and their relative size. The user was not concerned with the accuracy of the tree, and they were not interested in leaving it out, because in fact the tree was necessary to communicate an idea in the drawing.

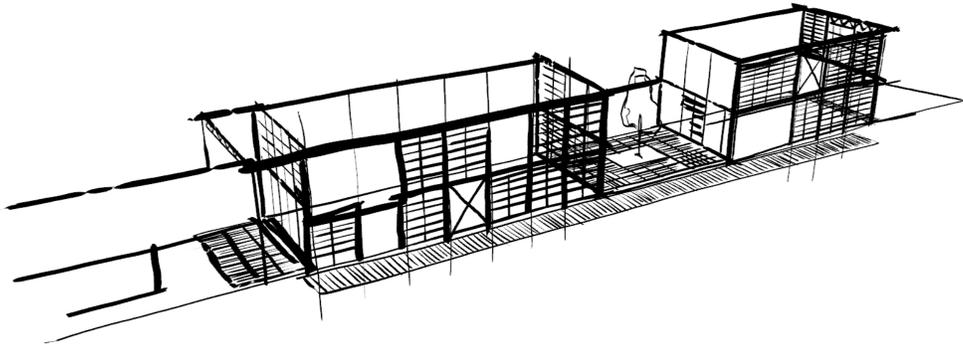


Figure 8.9 This figure is an image of a Cuttlefish 3D drawing of the Eames Case Study House in LA. The drawing is a 3D drawing and can be rotated and reoriented to discover different views.

Users also experimented with creating surfaces and planes within the drawing. Often times a user would create a complex and detailed drawing from a single orientation and then add to the drawing in another dimension by creating planes and sketches in a different orientation, and then moving the camera to see the assembly in perspective.

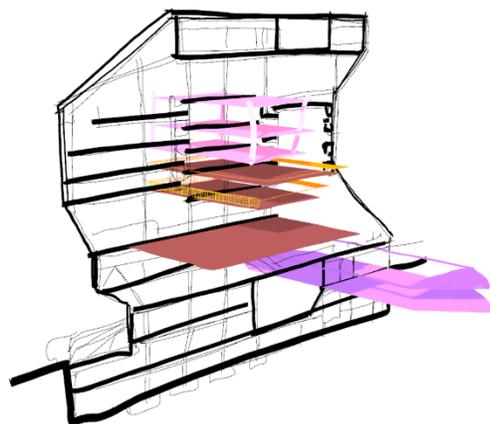


Figure 8.10 This drawing is a creative interpretation of the Seattle Public Library by OMA. The drawing starts with a detailed section of the building oriented in 3D space. The user then took this as a starting point and created simple floorplates extending out.

Once a user spent enough time with the software and became accustomed with the interface and the method of drawing and model making that worked best, users were able to create complete drawings and diagrams. Using a combination of colors, planes, sketches, fills, and all in different orientations, users

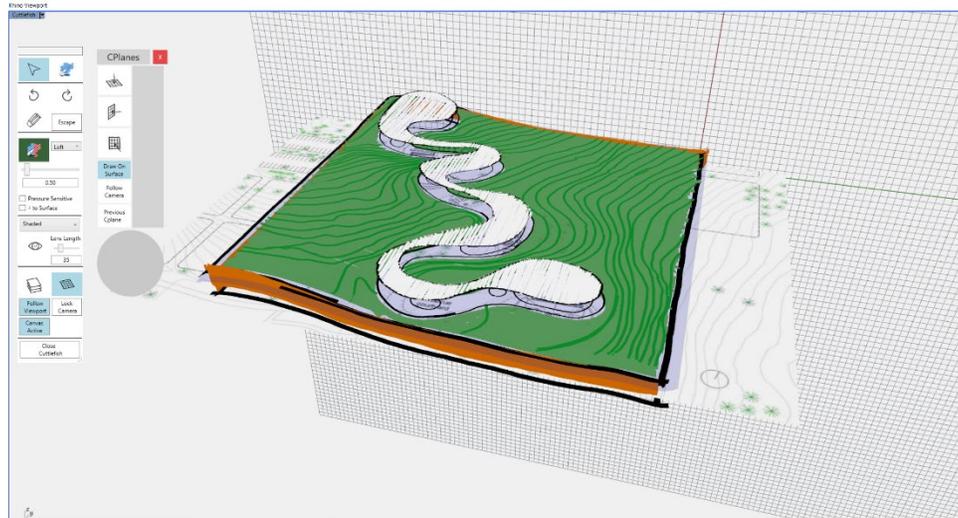


Figure 8.11 This drawing is a 3D representation of the SANAA River Building at Grace Farms. The drawing depicts a building that sits dynamically in the landscape. This drawing was made in Cuttlefish. The application interface is shown on the left.

are able to create models, and drawings that can depict a concept from many different viewpoints and at many different scales.

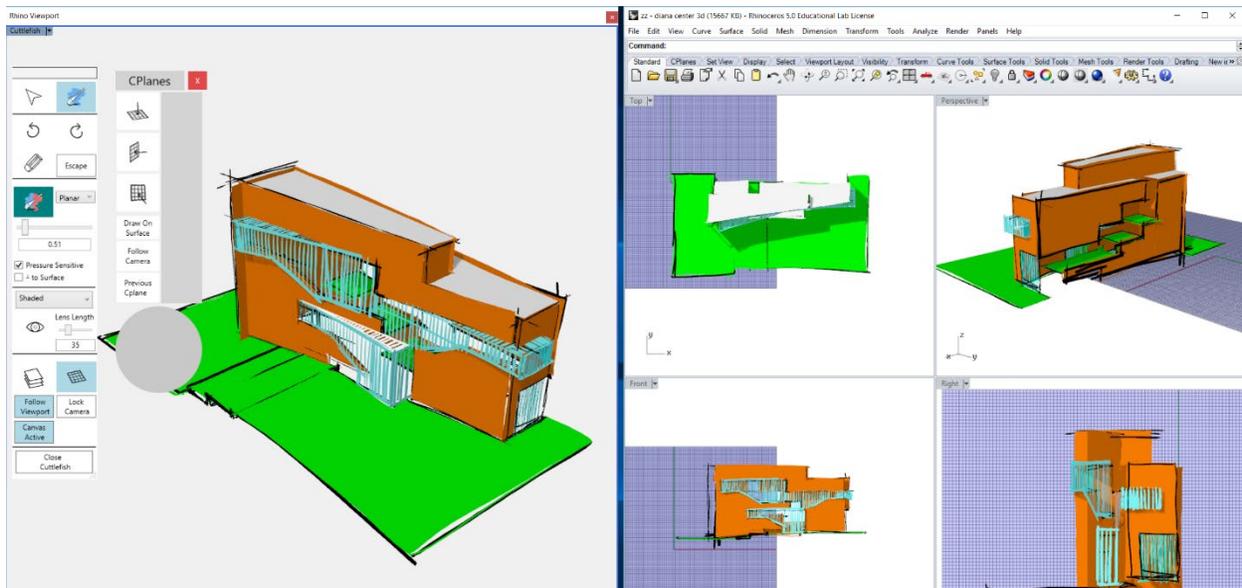


Figure 8.12 This Cuttlefish drawing of the Diana Center at Barnard College shows how a user can rely on the application to create a model quickly and easily, using planes, strokes and colors.

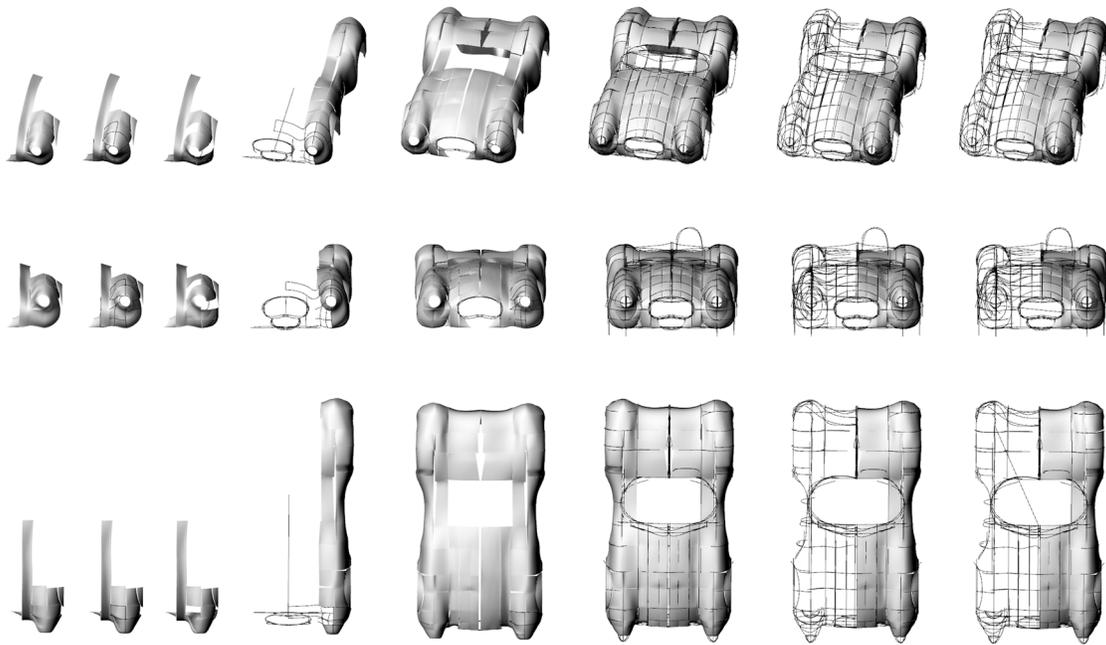


Figure 8.13 This drawing depicts a sequence of steps for the modeling of a car surface using Cuttelfish. The user created the surface model by outlining the various patches of the geometry and merging them together using the 3D modeling program.

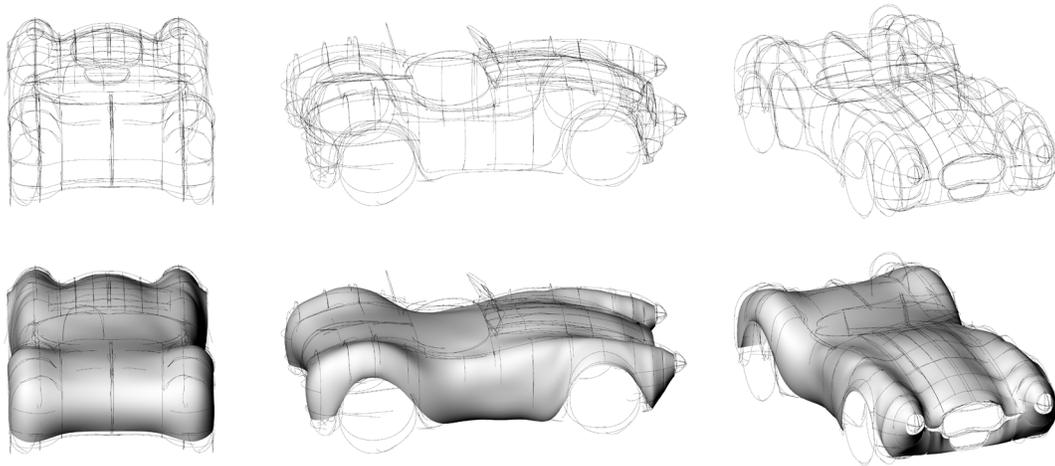


Figure 8.14 This figure shows a set of curves for the creation of a surface model. The figures above depict the curves from different orientations while the images below depict the surface created from the same curves, drawn in Cuttlefish.

9 FUTURE WORK

9.1 Introduction to Improvements

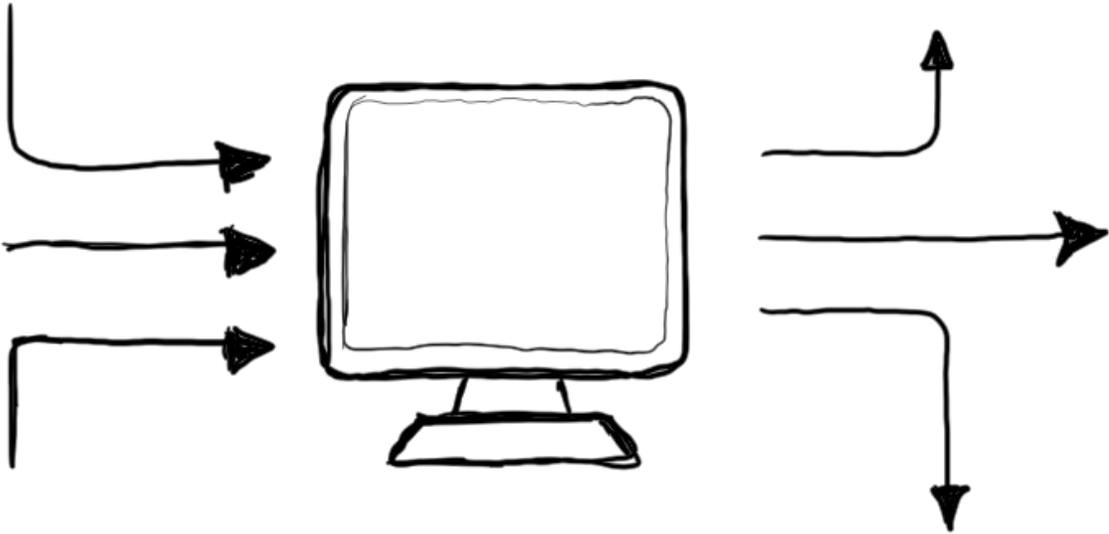


Figure 9.1 Many inputs many outputs. One of the reasons why our prototype is successful is because it accesses a larger array of inputs giving paths to a larger array of outputs. We believe that the most straightforward way to continue improving on this is to keep adding inputs and outputs.

This thesis, and the software that was developed as a product of the thesis, Cuttlefish, is a prototype software for early stage design. On the topic of future work when discussing prototypical software there is typically a laundry list of features and additions that could be made. There exist directions or paths the software can go down at different stages in development. It is important to say first, that this software set out to accomplish four core tenets that make it extremely useful for early stage design. Those four tenets are making sure the drawing looks good, encouraging quick iteration, providing a path to go from sketch to model, and allowing drawings to deal with context using multiple forms of media. Any one of these four core features, could be improved upon and expanded, and there could exist any number of unique core features that could add to the experience of an early stage design tool. As part of this thesis, a number of areas of improvement were discovered and given priority or importance over

countless other options. Those areas of improvement are as follows. First, is an increase in the flexibility of the output of our system. Second, is an increase in the number of inputs this system can use and interpret. Third, is a fine tuning the tools that allow for iteration within the software. Lastly, fourth, expanding the number of different contexts the system can exist in, mainly with advancements in different capture and display devices. One can see that a careful decision was made to maintain the software within its core mission. Thus, maintaining the belief that the four core features listed above form the basis of a useful early stage design tool.

9.2 Improvements in Outputs

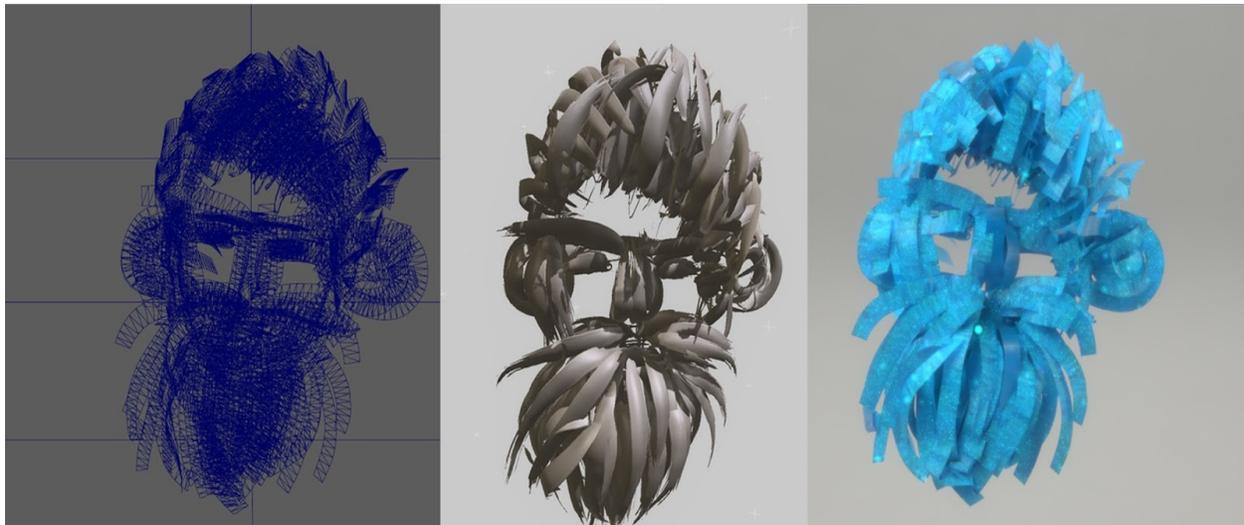


Figure 9.2 The images above depict the same geometry rendered with 3 different shaders. The geometry was created using Tilt Brush and is very similar to the geometry we get when creating in our system. This image is used to explain how shaders have an effect on the overall look and feel of a 3D scene.

Cuttlefish is a very simple software once most of the details are abstracted away, however, it is in these details that improvements could be made to expand the potential uses of the software. On the topic of details, the details behind the way strokes are displayed and rendered on the screen have a number of decision points and choices that could have gone any number of different ways. Right now, the software allows for the manipulation of the thickness, color and transparency of a stroke. However, there are a number of decisions that were made before the stroke could be colored or made wider or more

transparent. At any one point of those decisions, a different stroke could have been created, and should be created. In addition to the construction of the stroke, which would change the fundamental appearance of the stroke, strokes can be given different material qualities based on the way they interact with lights in the scene. In the Computer Graphics world shaders control the material qualities and characteristics of geometry in a scene. These shaders can do anything from making a stroke appear metallic and reflective, to making a stroke appear as if it's glowing or emitting light into the scene. Ideally Cuttlefish would have a number of different ways for a user to add, manipulate, and create their own shaders for the strokes they're drawing. The ultimate end goal of such a feature would result in the creation of a library of shaders, where users can trade, sample and use shaders created by other individuals all over the web. In addition to creating shaders that inform how light interacts with the strokes. There are shaders that can inform how the physical stroke is created in between the points that were input by the user. This means that if a user wanted to create a pen that draws tree-branch like curves wherever the pen goes, or 3-Dimensional I-Beam extrusions that curve with the spline, that would be entirely possible as well. The first step in allowing for the full-fledged version of this is a flexible version of the code that is broken up into modules that can be independently manipulated. In order to accomplish this within an ecosystem like Rhino, we use a plug-in for Rhino called Grasshopper. In this plug-in, we're able to remake our plug-in in such a way that it allows novice users of Grasshopper a way to manipulate and interact with our code.

9.3 Improvements in Inputs

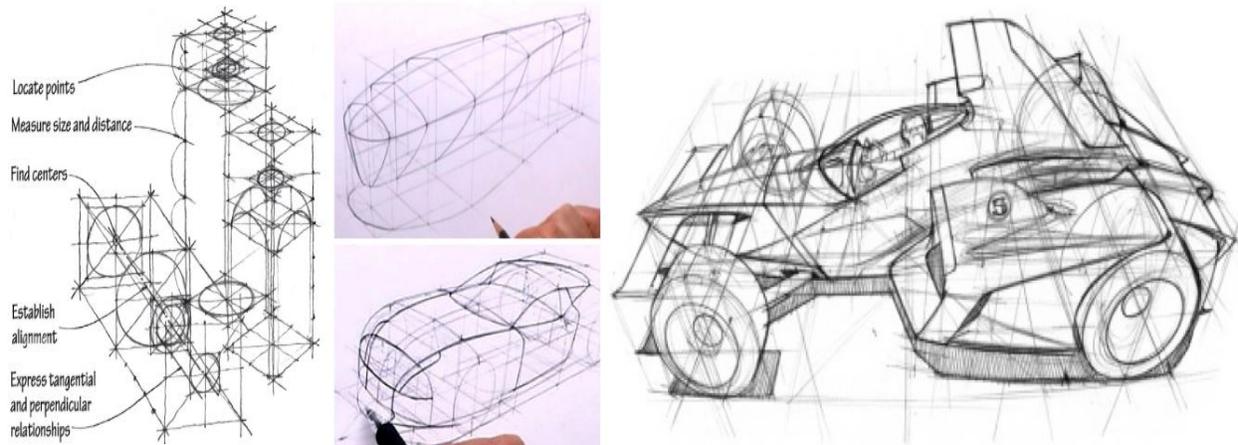


Figure 9.3 The images above depict three different analytical sketches. These are 2 Dimensional sketches that our mind interprets as being in 3D. There have been many attempts to create algorithms that do the same kind of interpretation to create 3D drawings and models from 2-Dimensional sketching.

Another aspect of drawing that is a large area of future work is in the act of input. When architects and designers draw on a 2-Dimensional surface, there are a number of ways and techniques that are used to make that 2-Dimensional drawing appear 3-Dimensional. One area that remains unexplored in the scope of this thesis is the ability to interpret those 2-Dimensional drawings of 3-Dimensional objects, into 3-Dimensional objects. This would be extremely useful for designers and architects who are trained in analytical drawing, or in axonometric drawing. This is because it allows them to more quickly draw their 3-Dimensional ideas and get them down “on paper” so to speak. In addition to being fundamentally useful to a core user group of this software, the software would also benefit from an expanded use of different inputs. If this software would be able to more easily interpret 3-Dimensional models from 2-Dimensional content, designers and artists could use this software to creatively interpret and translate fundamentally 2-Dimensional examples of visual art, opening the doors for new forms of expression. In the architecture field there are a number of practical uses that this feature would serve. For one, and 3D drawings created before the advent of computers and computer graphics could be easily interpreted and translated in order to make analytical models, or site models of certain contexts. Lastly, allowing users to

draw in 2-Dimensions, the way a user might be typically trained, and have those drawings automatically generate complex 3d models provides a benefit to the user without requesting too much buy in from the user. In the current iteration of the software, the user has to draw a little bit differently than they're used to in order to take advantage of the system's benefits. This is a major hinderance to anyone testing out or trying the software for the first time. If, instead, the users were able to continue doing what they're typically doing with a traditional pen and paper, and have the computer do the rest, there would be no barrier to using this new tool.

9.4 Visual History

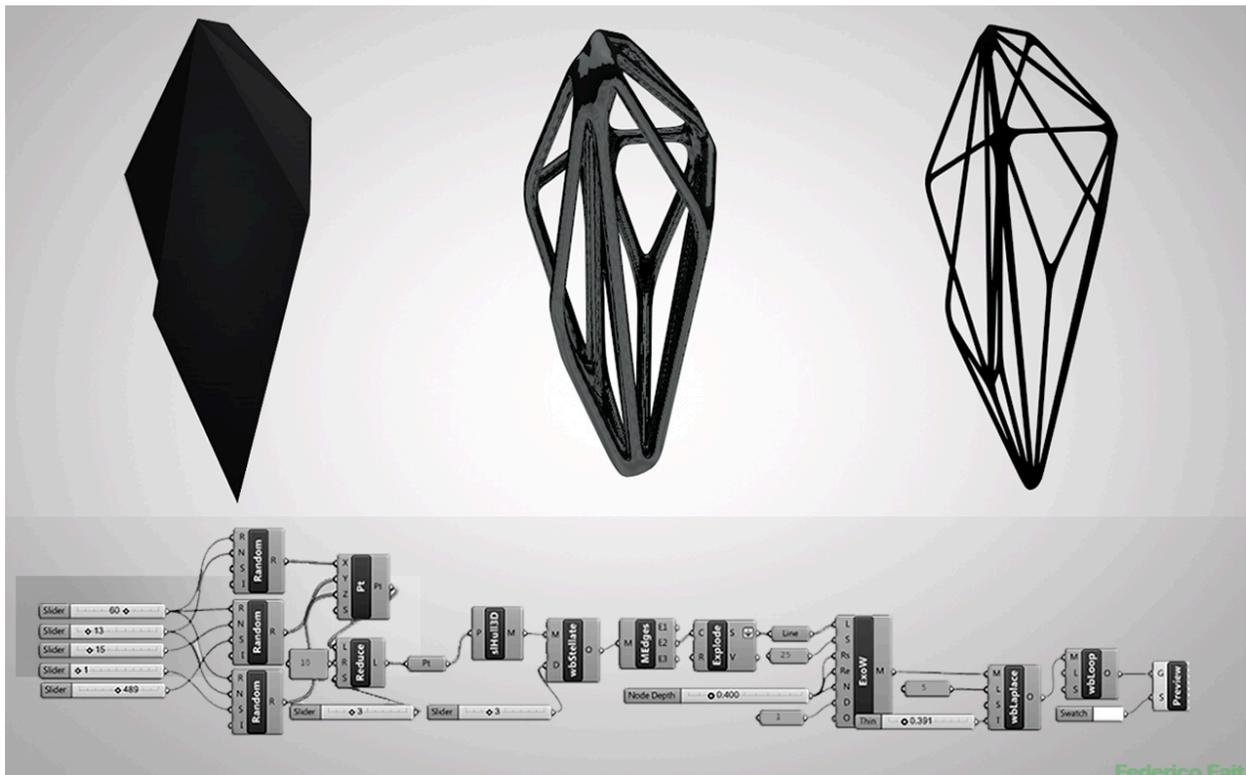


Figure 9.4 The images above depict an explicit history 3D Modeling system in which each iteration or phase of design of the model is stored in a visual history of the model making process. This allows for versioning, this allows for a quick assessment of multiple iterations, and it allows designers and engineers to intelligently manipulate their creations.

A feature that would augment the system's ability to allow for quicker iteration would be one that allows the system to visualize and understand the history of the drawing. This visualization could be browseable, selectable, and editable, in such a way that a user can explore different points in the design process and choose to go down a different path if they so desire. In addition to creating a feature that would allow users to more quickly iterate and explore different options, this feature would also allow for the vast reduction in storage requirements. Versioning is something that is known to decrease demands on file storage, and this feature is a type of versioning that would be more useful to users who specifically engage in the design process. Increasing the density of information in each individual file has many advantages for designers, from being able to more easily store and search projects, to being able to more easily send and collaborate on projects. Not only is this a feature that is proving to help users from all types of professions, but specifically for designers, it tends to be a harder task to accomplish and a feature that is more often requested. This is because the design process is fundamentally iterative, and through iteration a number of solutions are discovered that might not work for this particular case study, but some of those ideas could work really well for a different project or problem. Often times users go back to designs that were in the beginning of the design process either to understand better where they have ended up, to change course, or to synthesize from a number of different options from within the search space. The information collected from a browseable history can also be used for projects further down the road. Not only are different details and solutions literally reused, or rehashed in older projects, but many times, those old solutions can be reexamined from this new reference frame that allows the designer to discover a better solution from the synthesis of an old idea and a new problem.

9.5 Connecting to Distribution



Figure 9.5 An artist's depiction of a Web VR experience. Notice how the geometry is a stylized lightweight geometry with a low number of polygons. This allows users experiencing the scene with low bandwidth to enjoy the immersive presence of VR without the frame rate dropping too significantly.

Context is one of the most important aspects of design. When designing a new design software one must consider not only the different contexts designers are exposed to, with their projects, but also the different contexts a designer might experience within a hardware and software ecosystem. Within the Computer Graphics industry, two of the most important contexts are the contexts of displays, and the contexts of distribution. In the realm of displays we have virtual reality devices, and Head Mounted Displays, in which the user is choosing the orientation or the vantage point to view the content at all times. In the context of distribution, the internet and the Web has dramatically shifted not only how software obtains information, but how software itself is distributed and how it functions. Some of the most ubiquitous software in this day and age are lightweight browser based tools, because they can be easily accessed by multiple devices. Not only can applications on the web be accessed by multiple

devices simultaneously, they can also be accessed by multiple, different, devices simultaneously.

Allowing for users to display the spoils of their 3D Model from a phone to some friends, while a co-worker toils away on some PC halfway across the globe to edit and update the model in real time. The ultimate goal of connecting an application like this to the web, is to connect the application, to an infinite array of peripheral devices. All of these peripheral devices being able to use the 3-Dimensional information in their own way to give the user access to more tools and more processes. There is another reason why extending the application to exist in an online format would be valuable. Having an online version of the tool, and opening up the development of that online tool in the form of creating an API for individuals to make plug-ins, allows the tool to grow and expand into new use cases. One potential use case is the ability to use the software as an environment for creating an online marketplace to sell custom designed objects. An example of a system that does this is Nervous System's jewelry designing tool. With new information fed to the system from a user, and that user's interaction with a user interface, the tool is able to generate new 3-Dimensional. Once that tool has generated a new 3D-Model, it now has access to a number of websites and services that can be used for the creation and manufacturing of those 3-Dimensional objects.

9.6 Future of CAD

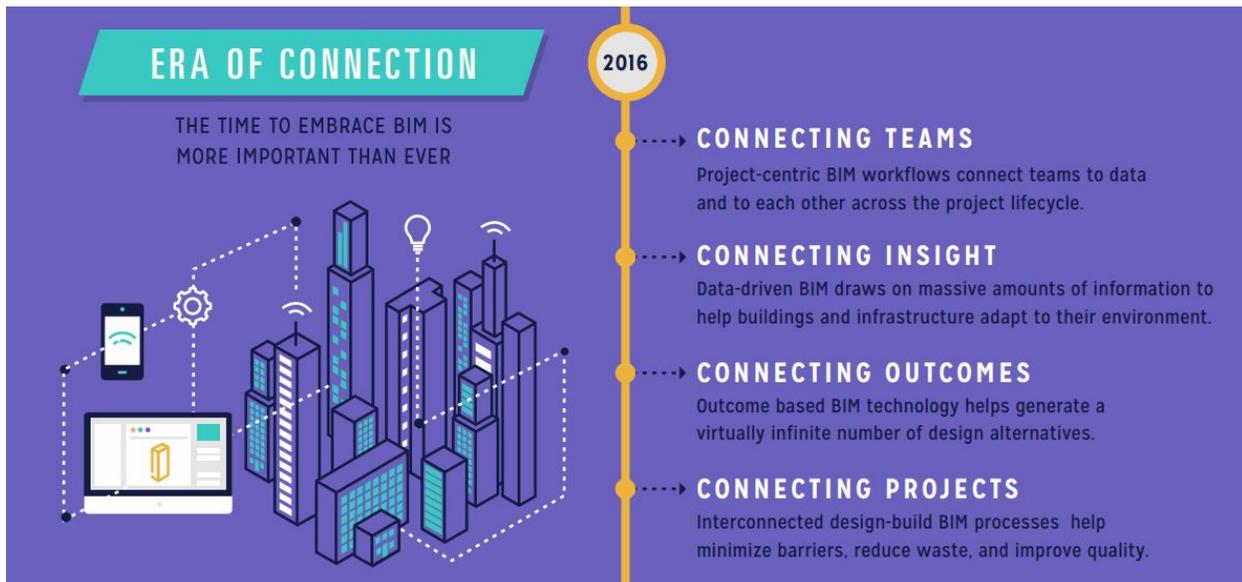


Figure 9.6 An infographic depicting what Autodesk believes to be the future of CAD. There's a large focus and emphasis on connection. The infographic explains that Building Information Modeling and CAD software will serve to connect people to the information of the project as well as the project itself.

Luckily, it appears that the CAD industry shares the ambitions of this thesis. With many of the same ideas put forth in this thesis being expressed as directions the CAD industry wants to go in. CAD is becoming much more ubiquitous and intuitive to use, while the output is being used for more and more applications. It is continuously allowing users to iterate faster and access design history in ever more powerful ways, and the software landscape and context as it relates to CAD is uncovering new uses day to day for 3D content in creation and in storytelling.

10 CONCLUSION

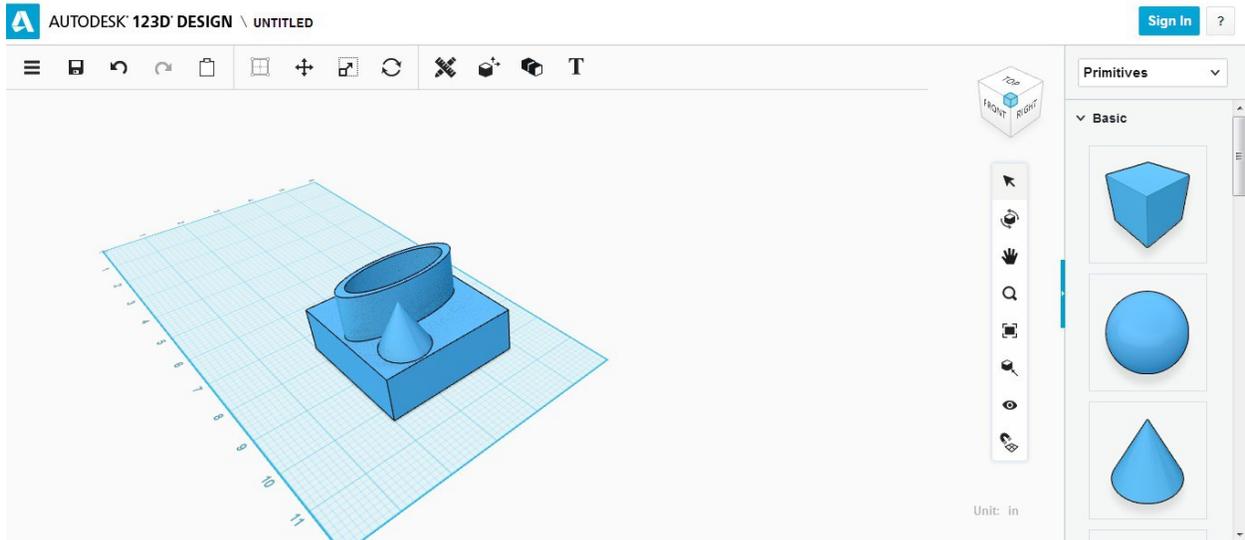


Figure 10.1 Above is a screenshot of another software for early stage design called 123D Design by Autodesk. It's shown here to indicate how tools for early stage design are becoming more and more user friendly because there's a demand to make software of these capabilities accessible to people of all ages and professions.

Our primary goal was to provide an unconstrained flexible environment for sketching ideas and concepts for early stage architectural design. By simulating pencil and paper hand drawing routines on a large scale raster display, we were able to replace the cumbersome standard mouse and keyboard input routines with an easy-to-use drawing and touch interface. Upon the completion of this thesis a number of things became apparent about software and specifically how software relates to the early stage design process. Evidently, there are only a handful of things that a software package must do really well, in order to be used in the early stage of design. First, the content the software produces, and displays, needs to look good. Second, the design of the interface, needs to encourage iteration. Third, the software must allow users to go from a sketch to a model. Lastly, the software must allow for the use, manipulation, and reference of multiple forms of media. This conclusion goes through these items one by one. It describes how architects view these features, and why they need them. It describes this need in the context of the early stage design process. This conclusion also describes these features from the perspective of the software.

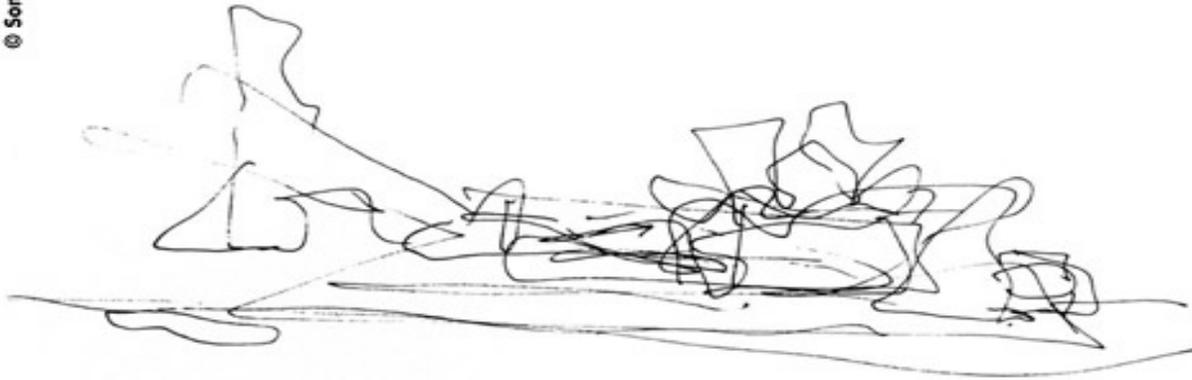


Figure 10.2 This is an image of Frank Gehry's original sketch for the Guggenheim Museum in Bilbao. It's depicted here in order to show the importance of the flexibility and speed of sketching in the process of Early Stage Design.

An architect will spend a lot of time, making a drawing look good. An architect does this not simply by controlling the orientation, position, and form of a line, but also by the quality of the line. It became very apparent that a feature that was missing in a lot of Computer Aided Design software was the ability to control the thickness of a line directly with the input. Being able to add pressure as an input for an architect when they draw, allowed the software to capture an entirely new type of architectural drawing. A design that an architect produces, must work, but it also must be presented beautifully. This holds true for drawings at all stages of the design process, but it is most true for early stage design. Drawings from early stage design typically have less lines, and that is what makes each line more important at an early stage. The relationships between these few lines have to be clearly legible, and one of the best ways to make a drawing more legible is to allow the architect to manipulate and change line weights fluidly. A bi-product of this feature is another dimension of style. Four architects could be asked to draw the same box, with the same dimensions, and no two boxes would look alike. This was made possible by introducing thickness and pressure sensitivity to 3D Modeling / 3D Drawing. In order to do this properly, the software had to project

the image of the curves to the screen and add thickness after the projection. Some complications arise from doing it this way, but allowing the user a What You See Is What You Get style of input/output, allows the architect to quickly make decisions about the appearance of a drawing.



Figure 10.3 An image showing 6 different view styles of the same 3d model of the Villa Savoye by Le Corbusier, depicted here to demonstrate the many uses and advantages that come from obtaining a 3D Model. This model can now be analyzed in a number of different ways and displayed in many different systems.

Allowing the architect to quickly iterate and evaluate their drawings was essential to the software from the very beginning. The system allows the user to quickly iterate by incorporating a layering system that is analogous to traditional trace paper. This allows the user to progress through the development of a design much faster. The user can iterate through multiple options faster, and they can develop or refine an idea using trace. Trace paper layers allow the architect to very quickly see what has been drawn already, and what is being drawn new. The ability to differentiate between the different layers very quickly was also extremely important. Lastly, the software also allows the architect to switch between layers of trace very quickly and fluidly. This ability to transition back and forth and to see the same project at multiple stages of development allows the architect to work through a design, rather than just replicate one for the sake of creating construction documents.

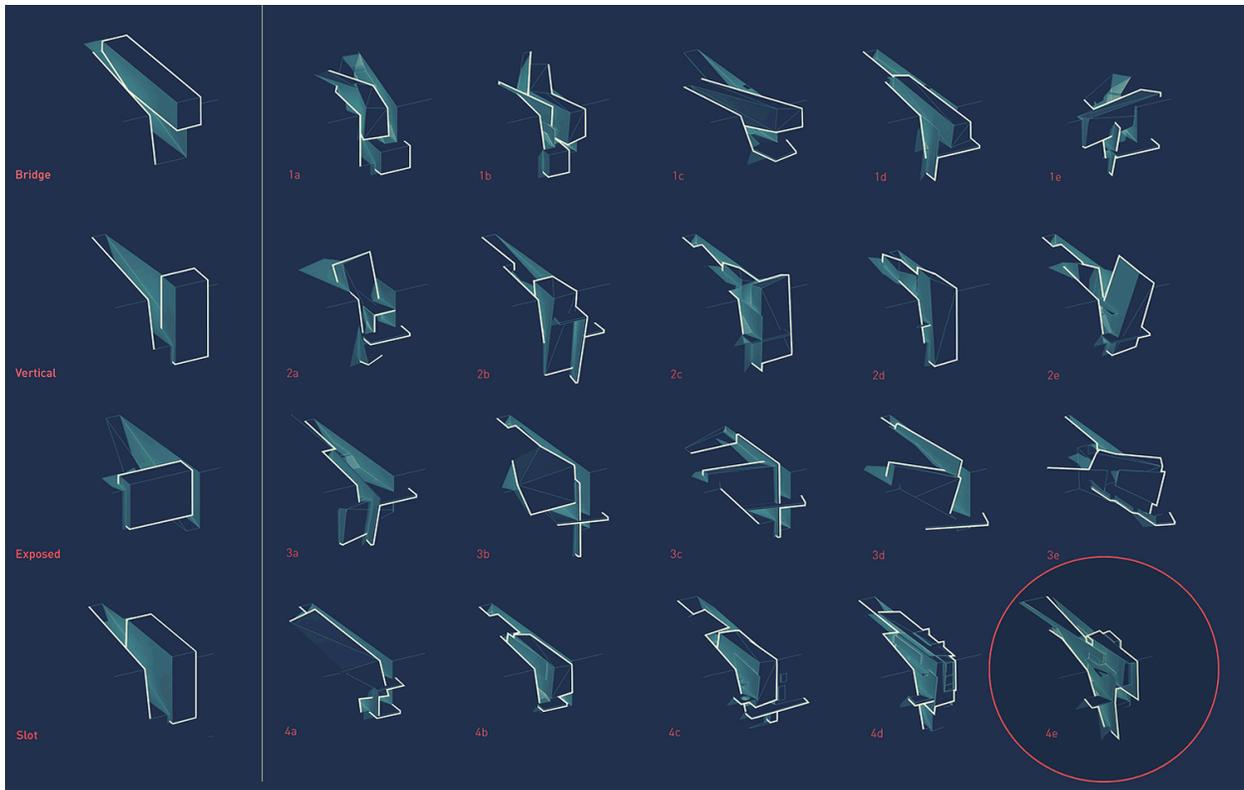


Figure 10.4 The image above shows an architect’s set of iterations presumably to explain how they arrived at the “best iteration described in the lower right hand corner. These types of images of an architect’s process are common in the architecture industry when critiquing early stage design.

One of the most important features of this system, and the primary reason that this system can straddle the lines of traditional drawing and CAD programs, is that it allows the user to transition from a sketch to a model entirely within the software. The system achieves this by capturing 3-Dimensional input, and refining it into 3-Dimensional geometry. There are several examples of systems that try to interpret 3D information from 2D data to generate a model. A lot of those systems are largely unsuccessful because the problem is an indeterminate problem. There are a few correct solutions for any given drawing. Our system avoids this problem entirely by making it relatively easy for a user to sketch in 3D. Once our system obtains a sketch in 3D, there are many tools that allow us to create surface geometry from the sketch curves. At every step along the way there are several assumptions that are being made between the user’s intent and the geometry that’s being created in the scene. We allow for these assumptions to be manipulated so that a curve can have a smooth look to it, or if the user decides, it can also be a lot rougher. It is by allowing for

multiple forms of interaction, creation, and development, that we encourage this tool to be used and misused in many different ways.

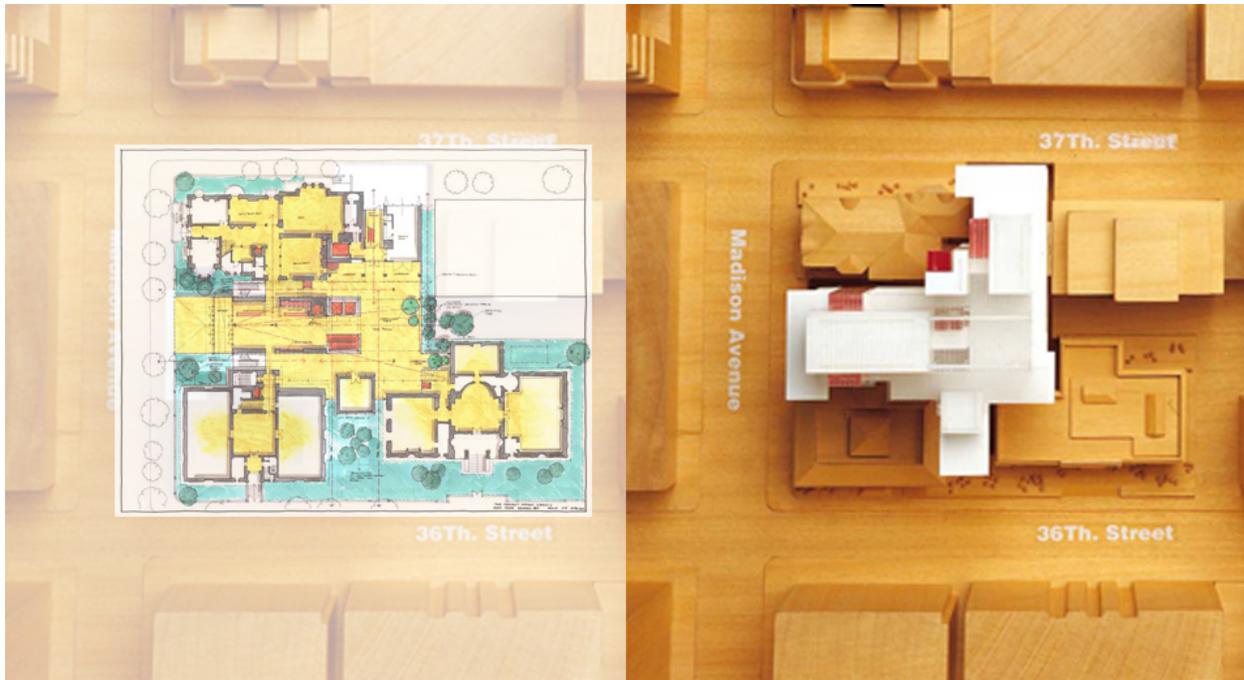


Figure 10.5 Two side-by-side images of schematic design documents for the Morgan Library Expansion by Renzo Piano in New York. The image on the left is a plan drawing, akin to the type of drawing that would be made in our software and the image on the right is a physical model of the same space. They are shown here to demonstrate the importance of context in the architectural design process.

With the thought in mind that this software would encourage multiple different uses, it became very important that the software was able to reference media from many different, existing forms of media. The easiest way to do this was to both, create a method for the software to become transparent and to overlay itself over any form of media a computer can visualize, while at the same time building this tool within an ecosystem that already interprets and utilizes many different file formats. The ability for Rhino, the software environment we're developing our tool in, to open and import many different 3D files is a feature that a lot of Rhino's users like about the software. We wanted to expand upon that by building our plug-in in such a way that the plug-in works within the 3D Modeling tool, and extends the ability of the 3D Modeling tool by creating a transparent window. This allows the user, to draw/trace on their desktop,

without having to resort to taking a screenshot and importing the image file into the software. It is with a tool like this, that tracing a 3-Dimensional object being rendered on the screen, or tracing a still from a video becomes second nature for the software.



Figure 10.6 Using the advantages and aspects of trace paper and combining them with the 3 Dimensional needs of early stage design, we've created a software that is specific to architectural use. The image above is of Morpholio trace, another software created specifically for architects, it is shown here to describe the validation of these needs in the architectural profession.

At the end of the day, no single one of these features would be enough for the software to change the way architects typically use software for early stage design. However, the addition of all four features, and the way they interplay with each other, that is what we think will change the way people use early stage design. Allowing for more control of a drawings line weights, encouraging quicker iteration by incorporating more aspects of trace into the software, creating a pipeline for the user's data to go from a sketch to a model, and then allowing the user to use all forms of multimedia as a reference for the drawing; that is what will change early stage design.

-
- ⁱ Constable, Robert L. "Transforming the Academy: Knowledge Formation in the Age of Digital Information." *Cornell University Library*. Cornell University, n.d. Web. 2 Nov. 2015.
- ⁱⁱ http://www.amazon.com/Drawing-Architecture-Neil-Spiller/dp/1118418794/ref=sr_1_1?s=books&ie=UTF8&qid=1459078000&sr=1-1&keywords=drawing+architecture
- ⁱⁱⁱ <http://surfaceproartist.com/blog/2013/7/21/no-trouble-running-mischief-on-surface-pro>
- ^{iv} <http://community.wacom.com/en/inspiration/blog/2015/january/50-trillion-to-one-the-magic-behind-mischief>
- ^v <http://carlysancker.tumblr.com/post/33277100477/hello-art-world-artmuse-that-program-that-ive>
- ^{vi} <http://www.thefoundry.co.uk/about-us/news-awards/the-foundry-acquires-made-with-mischief/>
- ^{vii} <http://www.thefoundry.co.uk/about-us/news-awards/the-foundry-acquires-made-with-mischief/>
- ^{viii} <https://vimeo.com/53403722>
- ^{ix} <http://cdp.ai.ar.tum.de/project>
- ^x <https://vimeo.com/53403722>
- ^{xi} <https://vimeo.com/53403722>
- ^{xii} <https://vimeo.com/53403722>
- ^{xiii} <http://rhondaforever.com/>
- ^{xiv} <http://rhondaforever.com/>
- ^{xv} <http://rhondaforever.com/>
- ^{xvi} Mental Canvas Paper by Julie Dorsey
- ^{xvii} <http://www.worldcadaccess.com/blog/2012/05/whats-the-price-of-catia.html>
- ^{xviii} <https://www.youtube.com/watch?v=mwp2dWzm2rI>
- ^{xix} <http://www.frontdesign.com/category.php?id=81&product=91>
- ^{xx} <http://geekpress.co.uk/now-were-thinking-with-htc-vive/>
- ^{xxi} <http://www.roadtovr.com/oculus-unveils-medium-creators-tool-lets-sculpt-vr-using-oculus-touch/>
- ^{xxii} <http://onlinelibrary.wiley.com/doi/10.1002/ad.1646/pdf>
- ^{xxiii} <http://onlinelibrary.wiley.com/doi/10.1002/ad.1646/pdf>
- ^{xxiv} Constable, Robert L. "Transforming the Academy: Knowledge Formation in the Age of Digital Information." *Cornell University Library*. Cornell University, n.d. Web. 2 Nov. 2015.
- ^{xxv} Constable, Robert L. "Transforming the Academy: Knowledge Formation in the Age of Digital Information." *Cornell University Library*. Cornell University, n.d. Web. 2 Nov. 2015.
- ^{xxvi} Graves, Michael. "Architecture and the Lost Art of Drawing." *The New York Times*. The New York Times, 01 Sept. 2012. Web. 02 Nov. 2015.
- ^{xxvii} <http://www.fondation-langlois.org/devicesofdesign/e/speakers.html>