

# Thesis

John DeCorato

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Background / Related Work</b>	<b>5</b>
2.1	Two-Dimensional Image Representation on Computers . . . . .	5
2.1.1	Raster Graphics . . . . .	6
2.1.2	Vector Graphics . . . . .	8
2.1.3	Adaptive Distance Fields: Mischief . . . . .	9
2.2	3-D Modeling in CAD . . . . .	11
2.3	3-D Sketching in CAD . . . . .	12
2.3.1	CATIA Natural Sketch . . . . .	12
2.3.2	ILoveSketch / EverybodyLovesSketch . . . . .	12

2.3.3	Hyve 3D . . . . .	14
2.4	3-D Sketching in 3-D . . . . .	16
2.4.1	Virtual Reality / Augmented Reality . . . . .	16
2.4.2	3-D Printing . . . . .	18
2.5	3-D Media Interaction . . . . .	18
2.6	Input . . . . .	20
2.6.1	Pen . . . . .	20
2.6.2	Touch . . . . .	21
2.6.3	Gesture . . . . .	21
<b>3</b>	<b>Strokes, the Base of the Sketch</b>	<b>24</b>
3.1	Creating a Stroke . . . . .	25
3.2	Definition of Splines . . . . .	25
3.3	Inverse Spline Calculation . . . . .	25
<b>4</b>	<b>Sketching in 3-D</b>	<b>26</b>
4.1	Ray Casting . . . . .	27

4.1.1	Implementation . . . . .	27
<b>5</b>	<b>Displaying Strokes</b>	<b>28</b>
<b>6</b>	<b>Usability and Feel: Bringing Physical Tools to the Virtual World</b>	<b>29</b>
6.1	Interacting with the 3D environment . . . . .	30
6.2	Combining Pen and Touch . . . . .	30
6.3	Turning the Page . . . . .	30
<b>7</b>	<b>Conclusion</b>	<b>32</b>

# Chapter 1

## Introduction

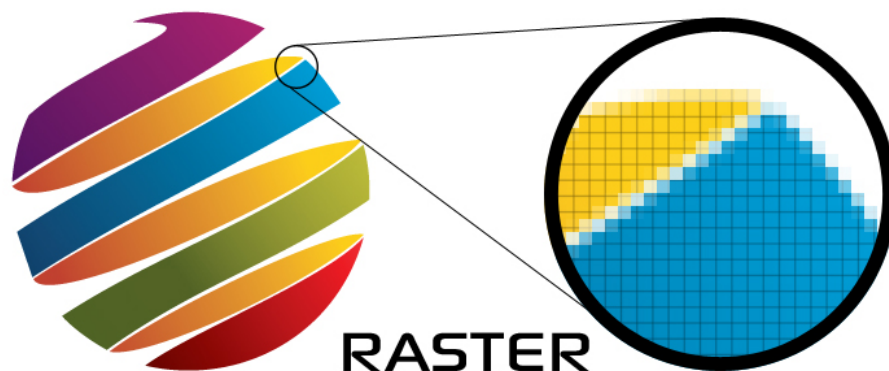
- My project is 3-D sketching
- Important because when using current CAD software you need to know the dimensions of the final object. Not creative
- Different because we plan on focusing on recreating the environment and tools a designer uses
- Novel contributions are 3D layer system, reprojected pen styles

# Chapter 2

## Background / Related Work

### 2.1 Two-Dimensional Image Representation on Computers

At its most basic form, a sketch can be described as an image drawn on a planar two-dimensional surface. With computer displays, there are two standard methods for working with two dimensional images: raster graphics and vector graphics. These underlying data structures have a large impact on the types of tools that can be designed to create, modify, and display the resulting images.

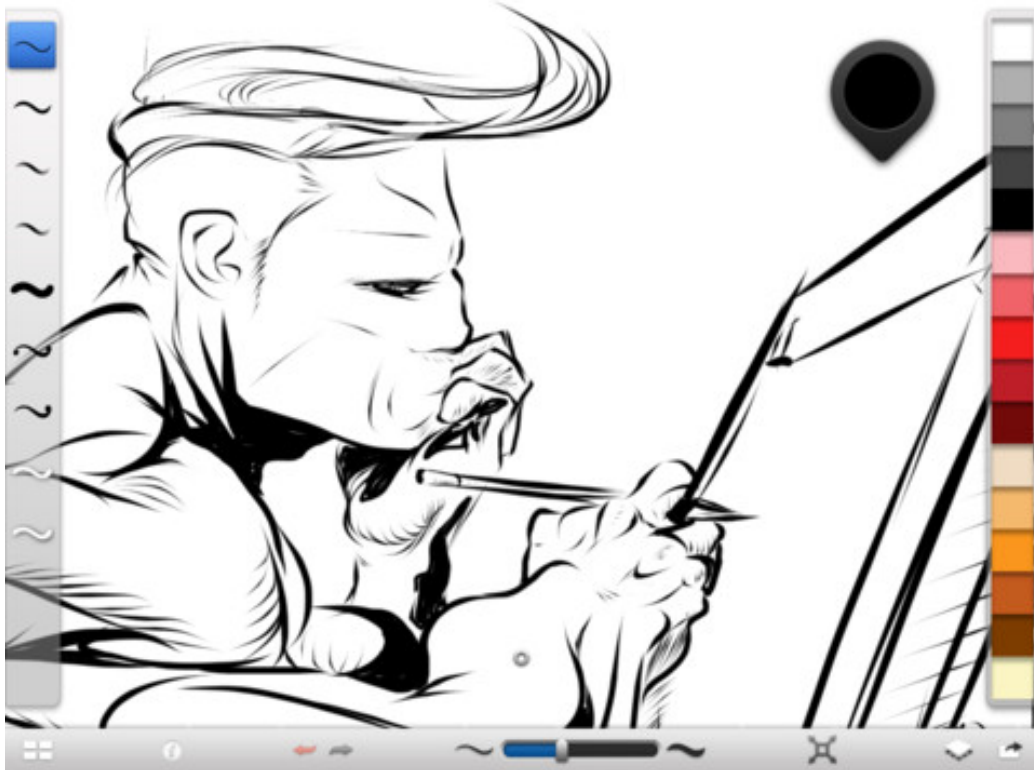


**Figure 2.1:** Zooming in on a Raster Graphics Image

### 2.1.1 Raster Graphics

Raster graphics is an image format that uses a two-dimensional grid to represent each pixel in the image. A raster image is characterized by its width and height in pixels and by its color depth, the number of bytes per pixel. The depth value specifies the color for each pixel, usually by identifying the magnitude of the pixel's RGB components. The reasoning behind representing images by this method is that today most computer monitors have bitmapped displays. Today, although there are many standard formats, almost all displays consist of rectangular arrays of square pixels, and the bandwidth from the display's memory is sufficient enough to dynamically render multi-megapixel images.

When creating and editing raster graphics images, the software directly manipulates pixel values, also known as pixel editing. This simplifies creating tools for editing raster graphics, since each tool can manually define



**Figure 2.2:** Creating a Raster Image in Sketchbook

how pixels are effected based on where and how an input occurs. Unfortunately, the ultimate quality of an image based on raster graphics is limited by the fact that the picture is resolution dependent. If you were to continuously zoom in on a raster image, eventually the image would suffer from image degradation.

Examples of popular raster graphics software are Corel Painter, Adobe Photoshop, Microsoft's Paint.NET and MSPaint, the open-source GIMP software, and Autodesk's Sketchbook.



### 2.1.2 Vector Graphics

Vector graphics is the representation of an image by the use of geometrical primitives such as points, lines, curves, shapes and polygons. Each of these primitives has a defined xy coordinate of the work space and determines the direction of the vector. Vectors can also be assigned a variety of properties such as its color and thickness. Because of their mathematical nature, they are theoretically similar to three-dimensional computer graphics, but the term specifically refers to two-dimensional images; in part to distinguish them from raster graphics. Vector graphics are primarily used for line art, images drawn with distinct straight or curved lines. For example, early CAD systems mostly used calligraphic black and white displays and rendered images in vector graphics formats. However, in modern times, vector graphics are now converted to raster graphics formats when used outside of vector specific editing software.

Vector graphics data structures offer a number of advantages compared to raster approaches. First, they are based on mathematical expressions, which means they are resolution independent. Zooming in on the image does not cause image degradation like in raster graphics; the image will remain smooth. Second, objects made using vector graphics are independent from their visual representation. This allows for easy and accurate editing of primitives, provided they are contained in a vector graphics workspace. For example, say we have an image of a circle covering a part of a square. In vec-



**Figure 2.3:** Zooming in on a Vector Graphics Image

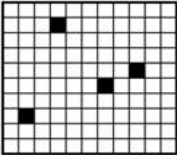

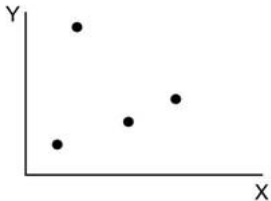
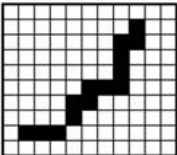

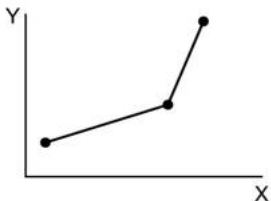
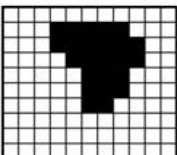
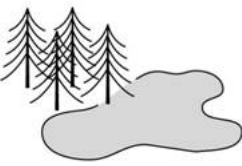
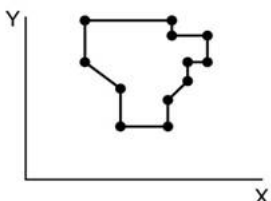
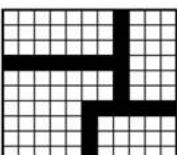
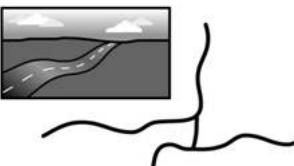
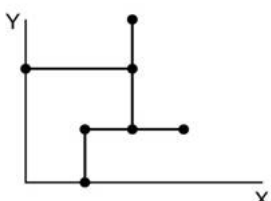
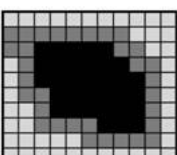

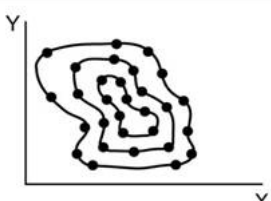
tor graphics, the circle can be moved without effecting the square beneath, because the system knows that the two objects are a circle and a square. This type of editing is not possible in a single raster graphics image, because with the underlying pixel representation there is no way for the data structure to know the definition of objects in the image.

Examples of popular vector graphics editing software are Adobe Illustrator, Corel Draw, and Inkscape.

### 2.1.3 Adaptive Distance Fields: Mischief

<https://www.madewithmischief.com/>

Mischief is a pseudo-vector graphics editor created by Made With Mischief, now owned by The Foundry. Although its systems are based on vector graphics, it does not allow for the precise editing of curves as seen in

The raster view of the world	Happy Valley spatial entities	The vector view of the world
	 x x Points: hotels	
	 Lines: ski lifts	
	 Areas: forest	
	 Network: roads	
	 Surface: elevation	

**Figure 2.4:** Raster definition of an image vs Vector definition

traditional vector graphics programs such as Adobe Illustrator. Instead it attempts to use vector graphics to simulate real world art techniques, similar to Sketchbook. This is made possible by using a data-structure called Adaptive Distance Fields, which stores vectors in a tree data structure instead of the traditional vector format. Since the tree data structure ends up looking like a pseudo-raster, raster brushes and pens can be implemented in this system by defining what shapes go inside of the data structure. Adaptive Distance Fields also greatly reduce the storage size of large scale vectors, making it possible for the implementation of their infinite canvas; their infinitely zoomable, translatable workspace. This unique data structure allows for incredible detail and unlimited scale.

**TODO:** Add examples of the infinite canvas (bug Nick)

## 2.2 3-D Modeling in CAD

Rhino

AutoCAD

Maya / 3DMAX

Sketchup

Not sure if this section is necessary on its own or if it should be

absolved into the one below it for examples of how 3-D cad software fail to accompany early stage design

## **2.3 3-D Sketching in CAD**

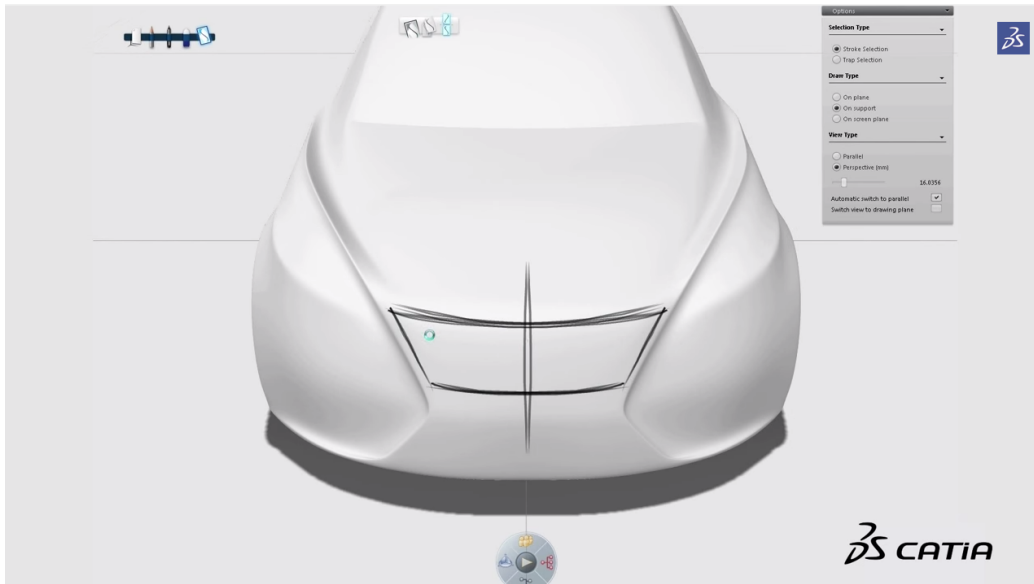
Not sure how to start this section. Probably need to clear out descriptions more before writing the intro.

### **2.3.1 CATIA Natural Sketch**

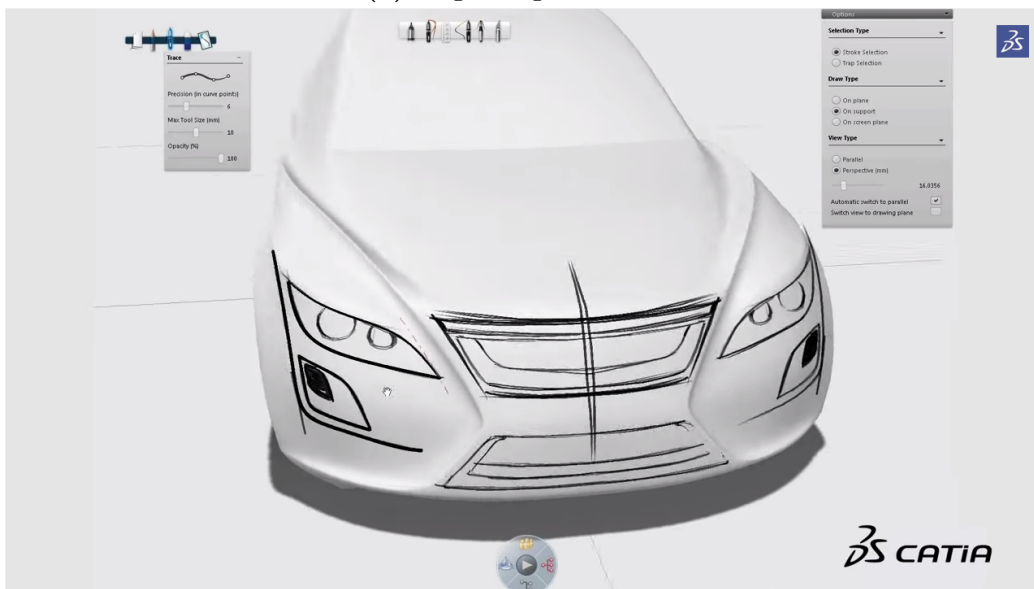
Natural Sketch is a feature inside of the CATIA modeling software by Dassault Systemes. Natural Sketch allows the user to draw on a virtual plane, a 3-D model, and the plane where the screen lies in the 3D environment. It features the abilities to alter the pen style, alter the number of control points used to make the post sketch curve, automatically change the camera view to align with the drawing plane if one is being used, copy and alter individual strokes, and generate models from the 3-D sketch.

### **2.3.2 ILoveSketch / EverybodyLovesSketch**

EverybodyLovesSketch is a 3D curve sketching system from the University of Toronto's Dynamic Graphics Project Lab. It features a pen based gesture



(a) Beginning the sketch



(b) Finishing the sketch

**Figure 2.5:** Using Natural Sketch to Add Detailing to a Car

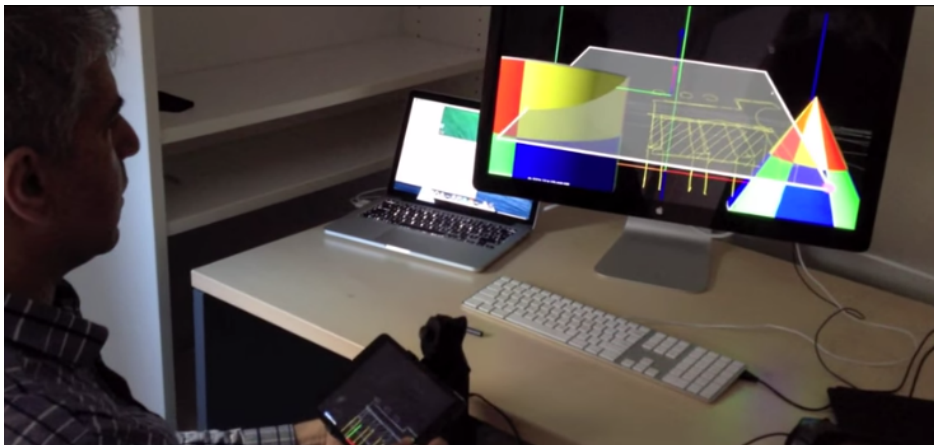
system, allowing the user to execute functions using rapid strokes, circles, and other defined gestures. Other features include dynamic sketch plane selection, single view definition of arbitrary extrusion vectors, multiple extruded surface sketching, copy-and-project of 3D curves, free-form surface sketching, and an interactive perspective grid. This project is based off of previous work by the same lab, ILoveSketch, which is the base 3D sketching functionality of the EverybodyLovesSketch project.

### **2.3.3 Hyve 3D**

Hyve 3D is an infinite virtual sketching environment from the University of Montreal. It uses two screens; a computer monitor to show the 3-D environment, and an iPad to draw. The sketching plane represented by the iPad is shown in the virtual environment, and is manipulated by moving and rotating the iPad in the real world. The user then pins the sketch plane in place and proceeds to draw at leisure. The advantage of this system is that it combines real world manipulation with virtual representation, eliminating the need for complex user interfaces and gestures. The disadvantage is that this kind of movement has no one-to-one feedback between the real world and the virtual, meaning that it is difficult to judge how your movements of the iPad effect the drawing place without confirming it visually.



(a) Using the tablet to draw



(b) Manipulating the drawing plane

**Figure 2.6:** Examples of Hyve 3-D in use



## **2.4 3-D Sketching in 3-D**

3-D content creation on a traditional computer screen is limiting. Any type of input or user interface can never overcome that one dimension of the workspace will always be inferred, due to the two dimensional output. The result has been leveraging a number of emerging technologies that deal with output that is experienced in three dimensions.

### **2.4.1 Virtual Reality / Augmented Reality**

Recently, there have been two key technologies developed with the intention to immerse the user in a virtual environment; virtual and augmented reality. Both of these involve head-mounted displays (HMDs) that display two dimensional images. Despite the images being flat, the system takes advantage of how humans see, such that the user feels the presence of actually being inside of the virtual environment. The advantage to using these virtual systems is the user can use their sense of depth to fully understand the space they are working in. The downside that these systems have is since their input methods are essentially drawing in midair, there is no tactile feedback similar to a pen pushing against a piece of paper. An example of an augmented reality approach to 3-D sketching is Gravity Sketch, and one of a virtual reality approach is TiltBrush by Skillman & Hackett.



**Figure 2.7:** Augmented Reality Sketching using the Gravity tablet and Headset

### **2.4.2 3-D Printing**

So far, we have focus on the transition from 2-D physical, to 2-D virtual, to 3-D digital. However, groups have experimented with using 3-D printing to create pens that can sketch simple 3-D models; examples being the Polyes Q1 Pen and the 3Doodler. These pens work similarly to hot glue guns, except instead of glue, the pens secrete ABS plastic that quickly hardens as it exits the tip of the pen. Like the digital approaches, these pens lack tactile feedback, and rely purely on the user's sight to sketch.

## 2.5 3-D Media Interaction

Interacting with 3-D Media using modern input devices

Gestures vs. Postures: ‘Gestural’ Touch Interaction in 3D Environments

[http://tobias.isenberg.cc/personal/papers/Isenberg\\_2012\\_GPG.pdf](http://tobias.isenberg.cc/personal/papers/Isenberg_2012_GPG.pdf)

A Survey of Interaction Techniques for Interactive 3D Environments <http://www.greys-eminence.org/papers/EG2013-STAR.pdf>

Interaction with 3-D environments using Multitouch Screens *http :  
//www.researchgate.net/publication/236304194\_Interaction\_with\_3D\_Environments\_using\_Multi-Touch\_Screens*

## 2.6 Input

- Basic input chapter
- Goal is to discuss where we are at in screen-based HCI
- Hardware overview and methods, as well as gesture diagrams, use cases, etc

### 2.6.1 Pen

Discusses how people have attempted to solve HCI problems with pen input, as well as hardware and how the pen works Pen Based Interaction

[http://www.academia.edu/2236260/Pen-based\\_Interaction\\_-\\_Next\\_Generation\\_User\\_Interface](http://www.academia.edu/2236260/Pen-based_Interaction_-_Next_Generation_User_Interface)

Pen-based User Interface

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1349146>

Experimental Analysis of Mode Switching Techniques in Pen-based User Interfaces <http://research.microsoft.com/en-us/um/people/kenh/papers/p226-li.pdf>

### **2.6.2 Touch**

### **2.6.3 Gesture**

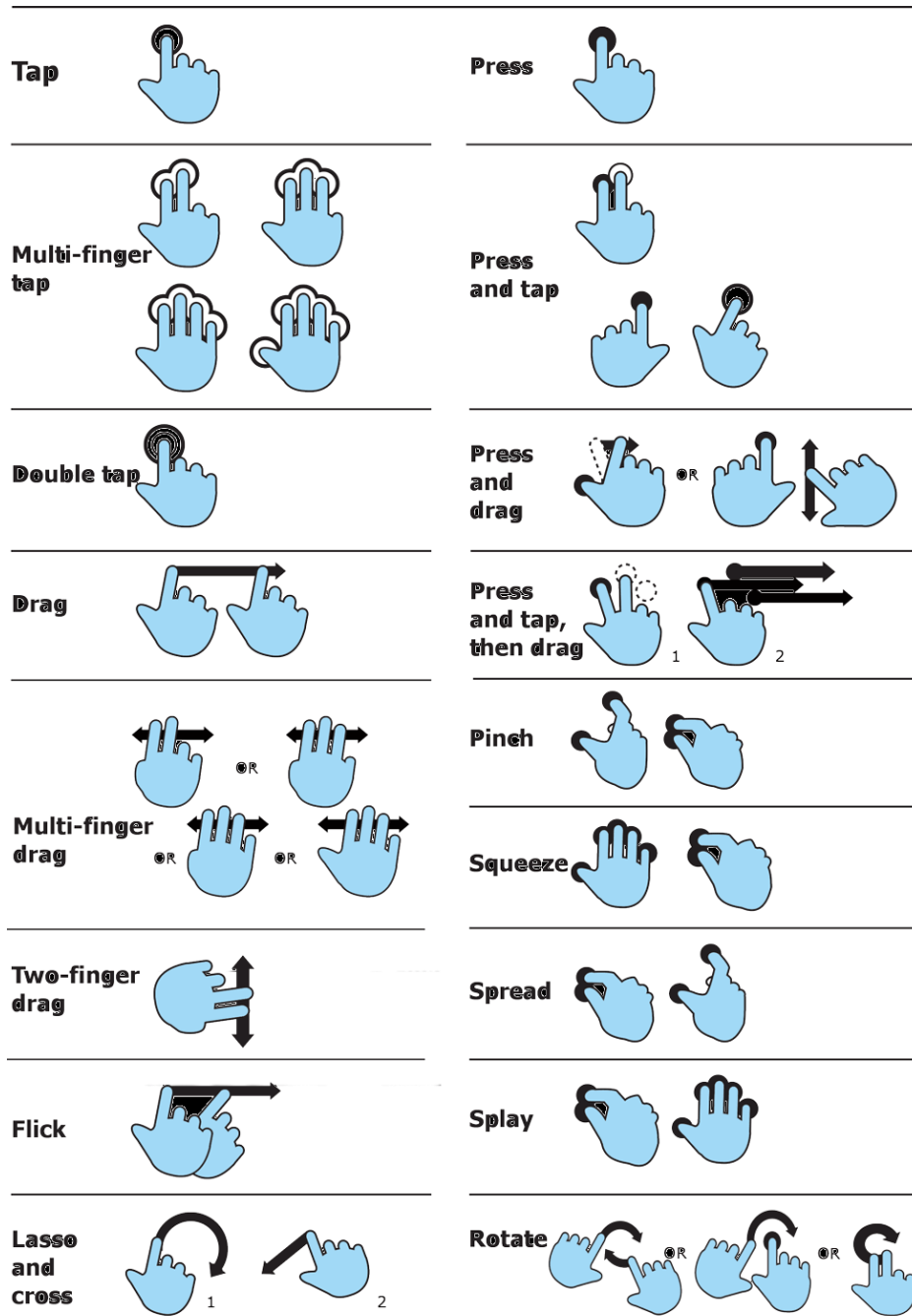


Figure 2.8: Touch Gesture Library

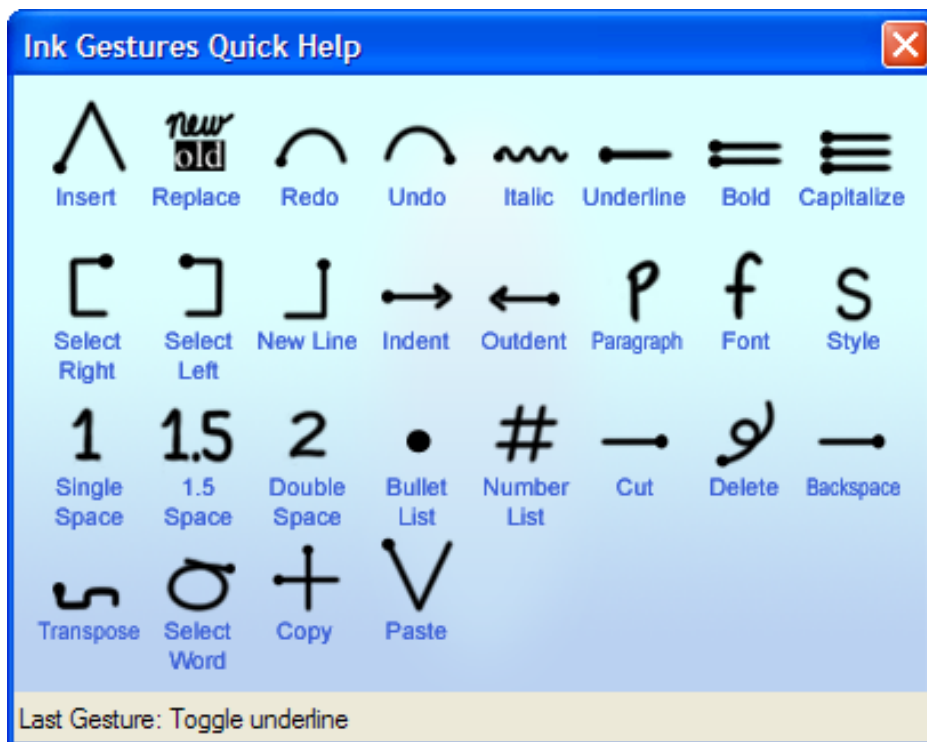


Figure 2.9: Example Pen Gestures



## Chapter 3

# Strokes, the Base of the Sketch

- Spline talk chapter
- Go over the inverse spline equation and the math behind the optimization
- Get into 2-D to 3-D projection of strokes, and storage



**Figure 3.1:** Rough Diagram of Optimizing a Curve using Control Points

Curve Global Interpolation <http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/INTERPOLATION/CURVE-INT-global.html>

Smooth Spline Through Prescribed Points <https://www.particleincell.com/2012/bezier-splines/>

### 3.1 Creating a Stroke

- user inputs a series of points on the screen
- storing all points too expensive, need to optimize
- solution: create a curve based on the points that is defined by a smaller set of control points

### 3.2 Definition of Splines

Math goes here

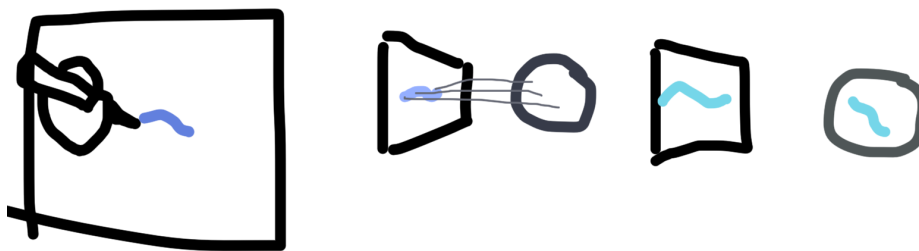
### 3.3 Inverse Spline Calculation

Math goes here

## Chapter 4

### Sketching in 3-D

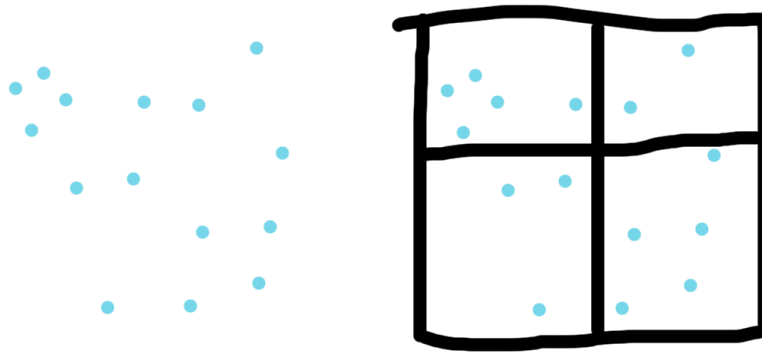
- Discuss how we get from the last chapter to 3-D sketching
- Discuss Ray Casting and drawing surfaces
- show system



**Figure 4.1:** Rough Diagram of Projecting Stroke onto Drawing Surface

## 4.1 Ray Casting

### Acceleration Structure



**Figure 4.2:** Creating an Acceleration Structure

### Intersection

Math goes here (Ray triangle intersection, Ray plane intersection, More in appendix? Reference?)

#### 4.1.1 Implementation

- Display algorithm used
- Discuss storage of strokes? (might not be particularly special)

# Chapter 5

## Displaying Strokes

Pen Styles section. This is getting complicated to the point that I think it needs it's own chapter

- Discuss reprojection of strokes into 2 space
- Discuss pen styles and how pen styles are created (probably input re-projected 2D curve to geometry shader)
- Discuss the what the actual styles are and the math behind them (possible appendix material after discussing one or two)

## Chapter 6

# Usability and Feel: Bringing Physical Tools to the Virtual World

- UX chapter
- talk about how people actually use the system tools
- talk about random quality of life features (page turning between layers, whatever else we come up with)

## 6.1 Interacting with the 3D environment

How do you move around the environment? How do you control things?

## 6.2 Combining Pen and Touch

Discuss modality of system

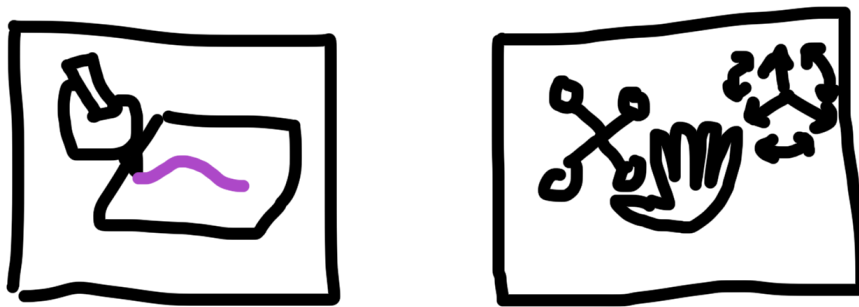
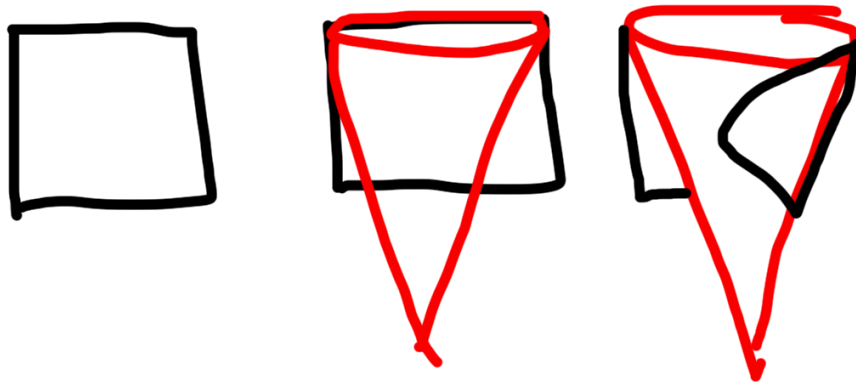


Figure 6.1: System Duality

## 6.3 Turning the Page

- Add reference to paper
- show math



**Figure 6.2:** Rough Diagram of math behind page turn



## Chapter 7

## Conclusion

Summary of everything goes here. More images of the system in use and example use cases.