Speedy Sketching

Kenneth Pak-Kiu Lam

Supervised by Toby Howard

University of Manchester

27 April 2009

Abstract

Speedy Sketching by Kenneth Pak-Kiu Lam
Supervised by Toby Howard

27 April 2009

Nowadays, a lot of applications exist for the purpose of constructing 3D models. However, many of them put too much emphasis on detail and bog down usability and the speed of construction. Speedy Sketching aims at providing a very simple interface for sketching simple shapes quickly, preferably with a pen or mouse interface, which will provide freedom and naturalness for the users. The lack of details will encourage quick iterative adjustments, and designers will focus more on the overall structure.

The idea behind Speedy Sketching is that any complex shape can be composed of simple shapes. After drawing simple 2D silhouettes, the user will be able to convert them to 3D shapes one by one as they draw, or altogether after drawing. It is hoped that this will greatly increase the speed and user-friendliness of 3D model construction.

Contents

1 Introduction

1.1 Aims and Objectives

What makes a perfect drawing application? In the author's opinion, it is one that gives the illusion that you are drawing freely with a pen and a piece of paper. When we were small, we used to draw everything based on its front view, which is the easiest way to represent the object. With these two ideas in mind, Speedy Sketching will allow users to sketch objects in their front, top, or side view on a flat surface, then convert them to 3D shapes by pushing or pulling. By clicking specific icons, users will also be able to convert front views of common 3D shapes to a 3D shape (Figure 1). By combining simple 3D shapes, the user can obtain the desired 3D object, and place it into the scene.

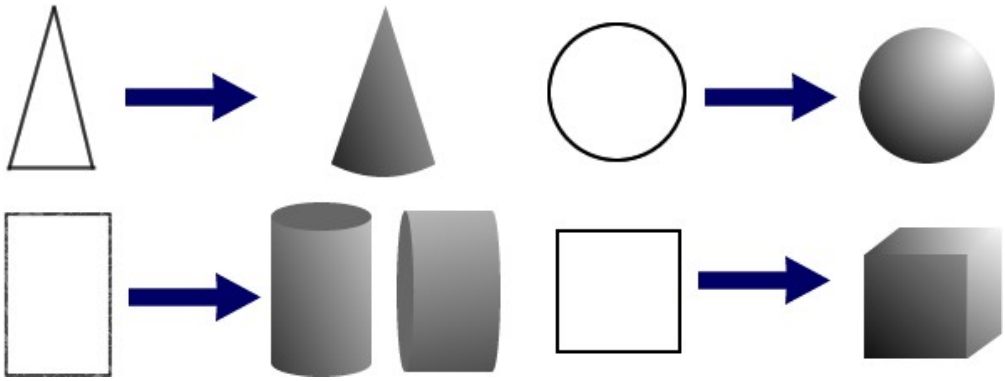

Figure 1: 2D silhouette to 3D conversion

The interface will be designed with a strong focus on simplicity – most users should be able to understand the software fully within 5 minutes of use. The number of key strokes required for an operation will be reduced to a minimum. It is hoped that with a simple interface, users will spend most of their time drawing but not switching between views and modes.

1.2 Target Audience

Speedy Sketching is directed to any discipline which needs quick brainstorming and sketching of ideas in 3D models. E.g., Computer animators who need to make 3D storyboards to express their ideas. It can also be used in art classes in school curriculum. It should be noted that Speedy Sketching is not intended for extremely detailed final models. Instead, we focus on the first step of the creative process – generating and comparing variations of design models quickly. These models are crude, alignment is not perfect but they stimulate thinking and discussions.

1.3 Project Proposal

Speedy Sketching will be developed based on the following aims and objectives:

1.3.1 Interface Basics

The interface will be so simple a user of any age can pick it up in a fairly short amount of time.

The application will be operational with just a mouse or pen interface. Keyboard shortcuts will be available for advanced users but not as a requirement for operation, because a keyboard is not always available in a Tablet PC. All possible operations will be shown as buttons in the GUI and achievable with one click. The icons will be relatively bigger than those found in a common application, because users will often found themselves carrying a Tablet PC and sketching in an unstable environment.

The application will have 2 main windows, one for showing the main scene (Scene View) and the other for drawing objects (Drawing Panel), which will be introduced in the next section.

1.3.2 Drawing Panel

The Drawing Panel will handle all object editing and sketching in Speedy Sketching. With the left mouse button (or the equivalent), users can draw shapes consisting of one or more line strokes. As they draw, all lines drawn on the Drawing Panel will be smoothed and simplified automatically to reduce unwanted fluctuations. A colour panel will allow users to change the colour of 2D and 3D shapes to be drawn, and a grid can be toggled to allow snapping of points so users can worry less about precision. All drawing will take place in the x-y plane and the drawing panel can be slid in the z-axis to cover all space (Figure 2). It will be semi-transparent and so will the objects in front, allowing the user to see the panel and the objects behind.

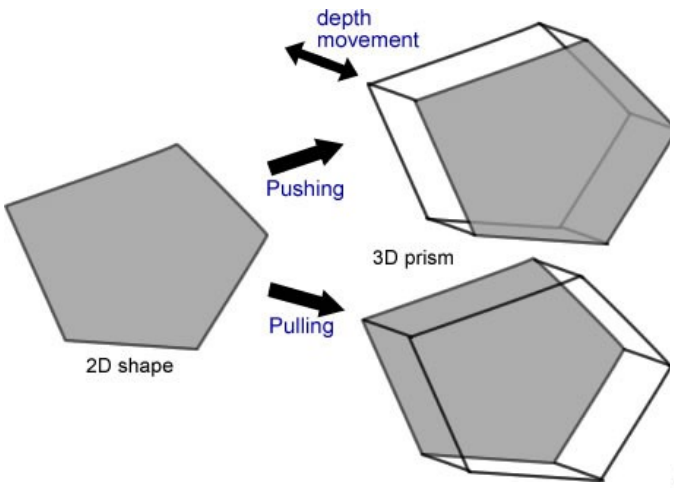

Figure 3: Turning a 2D shape into a 3D prism. If pushing, the 3D prism will lie behind the Drawing Panel, otherwise it will lie in front of the Drawing Panel

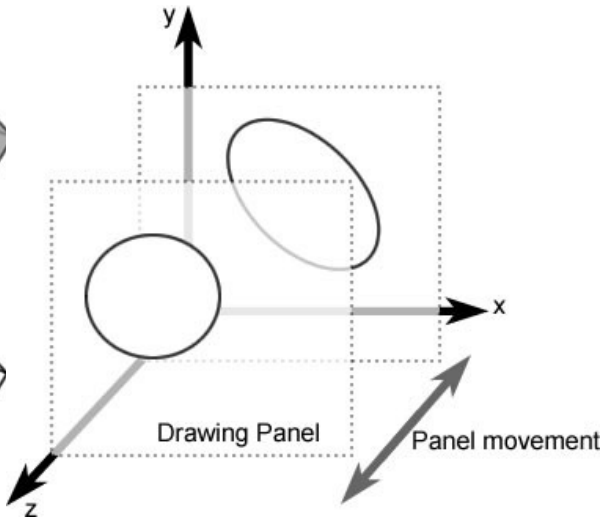

Figure 2: Movement of the Drawing Panel in 3D space

Speedy Sketching will feature two 2D to 3D conversion modes. Firstly, any *closed* 2D shape can be pulled/pushed to form a 3D prism (Figure 3). The user will have the option to centre the shape at the Drawing Panel regardless of it being pushed/pulled. Secondly, the user can select the basic 3D shape they want, and draw the front views in 2D (Figure 4). Speedy Sketching will automatically determine the parameters (size, width, height, radii) of the 3D shape, increasing the sketching speed and also providing a way to fix human errors (straighten lines, fix wobbly curves). As with pulling and pushing of shapes, the resulting 3D shapes can be placed in front, behind or centred at the Panel. Additionally, users will be able to draw a destructive shape and remove parts of other shapes already created (Figure 5).

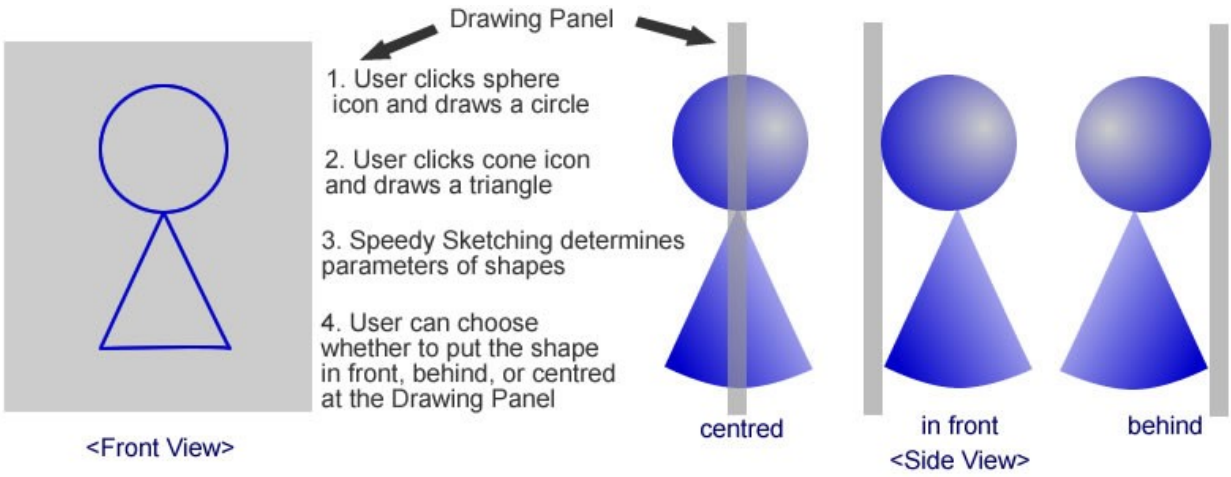

Figure 4: Speedy Sketching automatically converts common 3D shapes from 2D silhouettes

The Drawing Panel will save a small list of previously drawn shapes. The user will only need to rotate the scene or move the used object in the x-y plane to reuse the shape. This will be useful when there are multiple shapes of the same size.

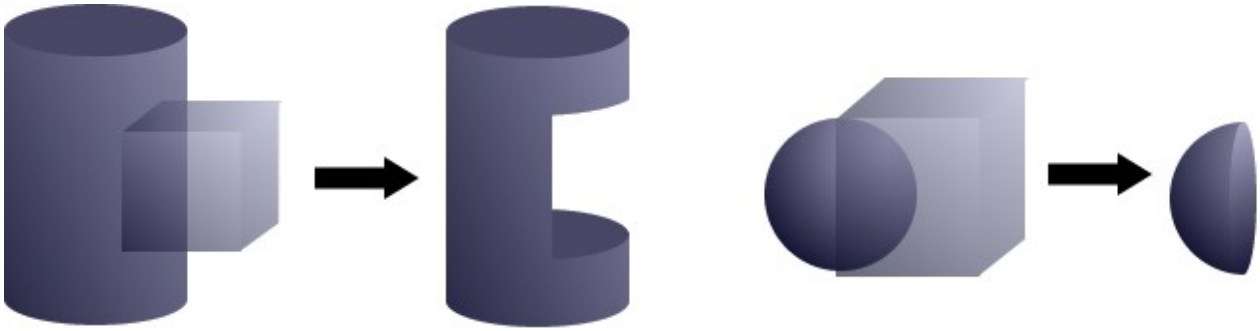

Figure 5: Destructive cube removing parts of created volumes

After drawing and constructing, the user will be asked to place the object in the scene. This will be passed on to the Scene Panel.

Editing can be performed using the right mouse button at any time during construction. Individual shapes (2D/3D) can be selected and merged, scaled, moved or rotated. Also, by double-clicking on an object, individual faces can be selected then pulled or deleted (Figure 6). Additionally, users can use a scissors tool to cut and remove parts of a 3D shape. A grouping operation will be available to join shapes together, which will help defining parts of objects.

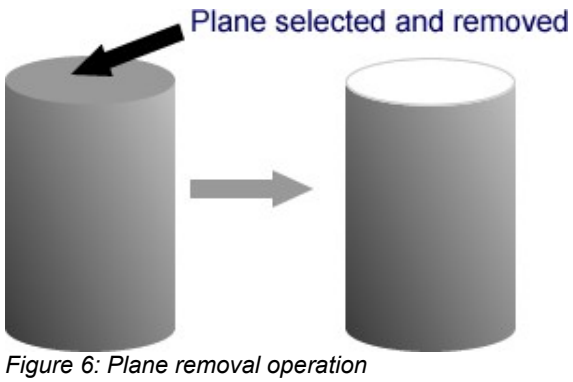

Figure 6: Plane removal operation

Camera

The camera is designed that users will be able to rotate the scene/objects in the x, y and z axes quickly with 3 sliders even when drawing. Unlike virtual track ball, sliders will make sure only one axes is rotated each time. Users will be able to rotate the scene quickly without worrying about accuracy. The sliders will allow rotation of -180 to 180 degrees and snap angles to the nearest predefined minimum angle. Sliders will bounce back to 0° position after each movement, so each time the movement is relative to the current orientation. Unlike popular software, Speedy Sketching will *not* feature 3 views of an object under construction because it is potentially confusing and restricts the user to 3 views only. Also, we want the users to feel that they are drawing on a piece of paper.

1.3.3 Scene Panel

The Scene Panel is for showing the entire scene which consists of different separate objects. The whole scene can be rotated, zoomed or panned freely. Individual objects can be selected by hovering on them, and a bounding box will appear. Buttons will appear for rotating or scaling the object.

Single clicking on an object will activate a pop-up window featuring a zoomed-in view of the object, which can be rotated in several predefined directions. Double-click will activate the Drawing Panel for editing the object.

1.3.4 Model Importing/Exporting

Speedy Sketching can save unfinished scenes in XML format for specific information about common 2D/3D shapes. It will be also able to import and export 3D scenes and models in .obj format which is commonly used in design software. This allows designers to modify their sketched objects in detail in after the initial creation process.

2 Background

2.1 Sketch Interfaces

2.1.1 Introduction

Traditionally, computers have been more machine-oriented and worry less about the convenience for users. But as computing power improves two-folds every 2 years [2], computer scientists have more in stock for boosting user experience. Nowadays, the computer can process properties like ambiguity, creativity and informal communications, thereby allowing support for user-oriented tasks like drawing, speech and design [1]. This gives rise to sketch interfaces which allows users to scribble ideas at will with minimum constraints.

2.1.2 Sketching and Creativity

With minimum constraints comes great innovations. It is shown that sketching and gesturing are 2 modes of informal and perceptual interaction which are valuable for creative design tasks [3]. It is argued that the ability to rapidly sketch objects with uncertain types, sizes, shapes and positions is important to the creative process. The ambiguity encourages designers to explore different ideas without being burdened by colours, fonts, precise alignment and formality. In a study performed by V. Goel [4], designers were asked to solve design problems either by sketching on paper or using traditional computer design applications. It is found that designers quickly created variations of an idea in free-hand sketch. In contrary, designers who used a computer-based drawing program spent more time fiddling with the initial design, thus limiting creativity.

The above study does not imply computers should be avoided for creative design work. Instead, we should combine the freedom provided by a sheet of paper and the efficiency of computers to create powerful applications – *electronic sketching* systems. The strength of such systems lies in the ability to recognise graphical elements common to a particular domain when they are drawn [1]. For example, in MathPad^2, mathematical equations, numbers, functions, trigonometry scribbles and even simple diagrams can be recognised (Figure 7) [5]. With a paper-like sketch interface, users are more encouraged to brainstorm. It does not matter if a mistake has been made, because a gesture is provided to easily rub out sketches [6]. The fault-friendly interface enhances scientific discussions where different ideas are expressed.

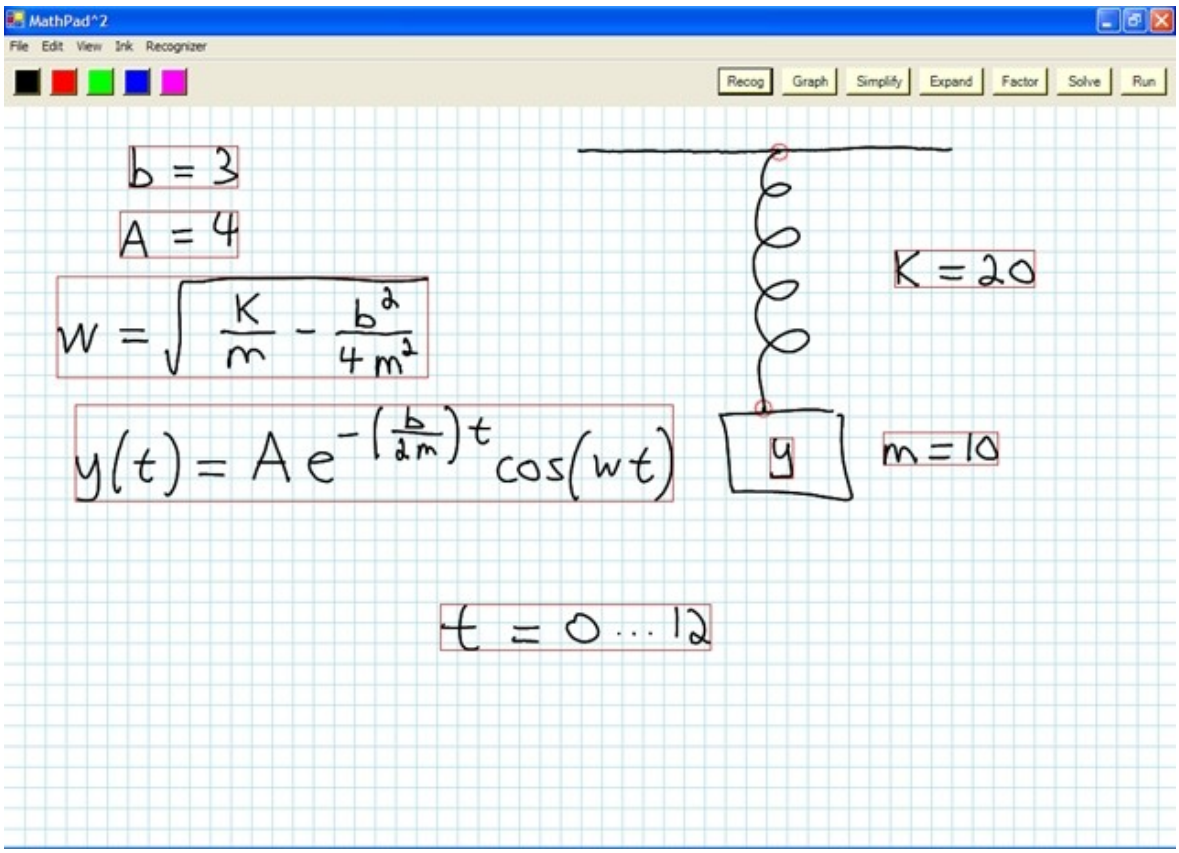

Figure 7: Mathematical equations, symbols and diagrams can be recognised in MathPad² (image from [5])

2.2 Cartographic Generalization

2.2.1 Introduction

Cartography – the making of maps – has close relationship with sketch interfaces. It is because they both deal with arbitrary input in the form of points, lines and shapes. Digitalisation of maps allows applications to display them at different scales, sizes and levels of detail according to requirements. However, it would make sense to display the map at a lower resolution only if the amount of data processed is reduced correspondingly. Otherwise, computing time would not be significantly improved. Traditionally, map generalization is a tedious task for cartographers, who have now turned to technology to automate the process. With the help of computers, map generalization can be achieved more uniformly, precisely, rapidly and with lower cost [7].

2.2.2 Line Simplification and Smoothing

A number of generalization techniques are used by cartographers, but only 2 of them are of interest to Speedy Sketching – line simplification and smoothing. Line simplification refers to a reduction of density by selecting a subset of the original point pairs, retaining points most representative of a line [8], which will reduce computing time and storage. Smoothing, on the other hand, relocates or shifts coordinate pairs in an attempt to plane away small perturbations and capture the most significant trends of a line, which will result in a more aesthetically pleasing representation [9]. These two techniques are useful for reducing the human errors introduced when using a pen interface and increasing the speed of calculations. Figure 8 below illustrates

line simplification and smoothing operations.

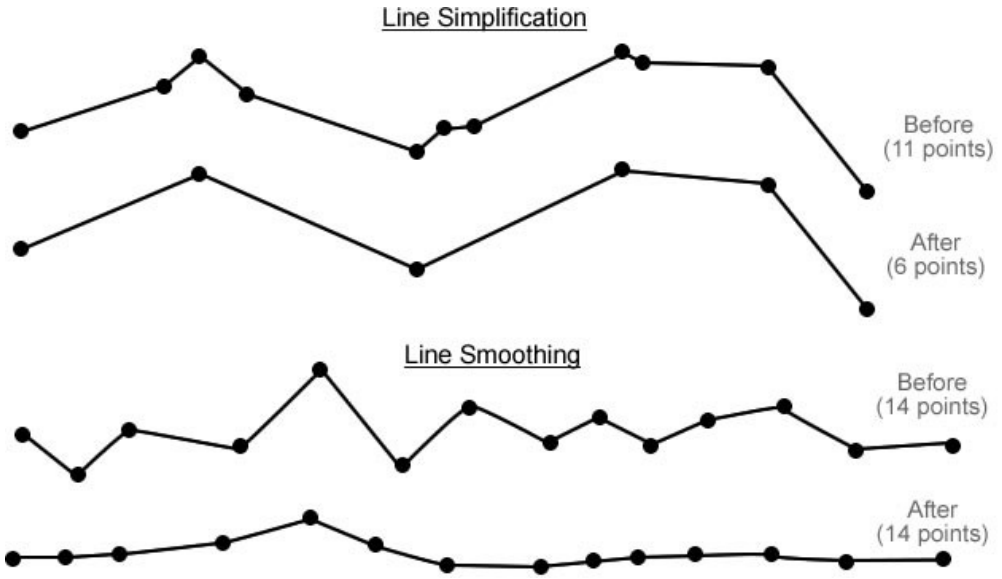

Figure 8: Demonstration of line simplification and smoothing. Note the change in number of points.

2.3 Previous Work

2.3.1 Teddy/SmoothTeddy

Teddy is a Java application by Takeo Igarashi (1999) which allows drawing of rotund 3D models in a sketch-based environment. Despite with a very simple user interface, Teddy detects user's free-form strokes and presents a number of complex operations like extrusion, cutting, digging, line painting, and erasing of lines. Initially, the engine takes a circular 2D shape and predicts the thickness of the object. In general, wide areas become flat and narrow areas become thin [10]. Figure 9 demonstrates a number of operations in Teddy.

In 2003, an update to Teddy – SmoothTeddy – was released. Apart from the original features of Teddy, it now automatically generates a hierarchical structure of the object, which can be exported in VMRL format and used in animation software like Alice [11]. A main improvement is that polygon meshes which are originally rough and with uneven triangulations are refined to increase the smoothness [12]. Figure 10 shows a screenshot of SmoothTeddy.

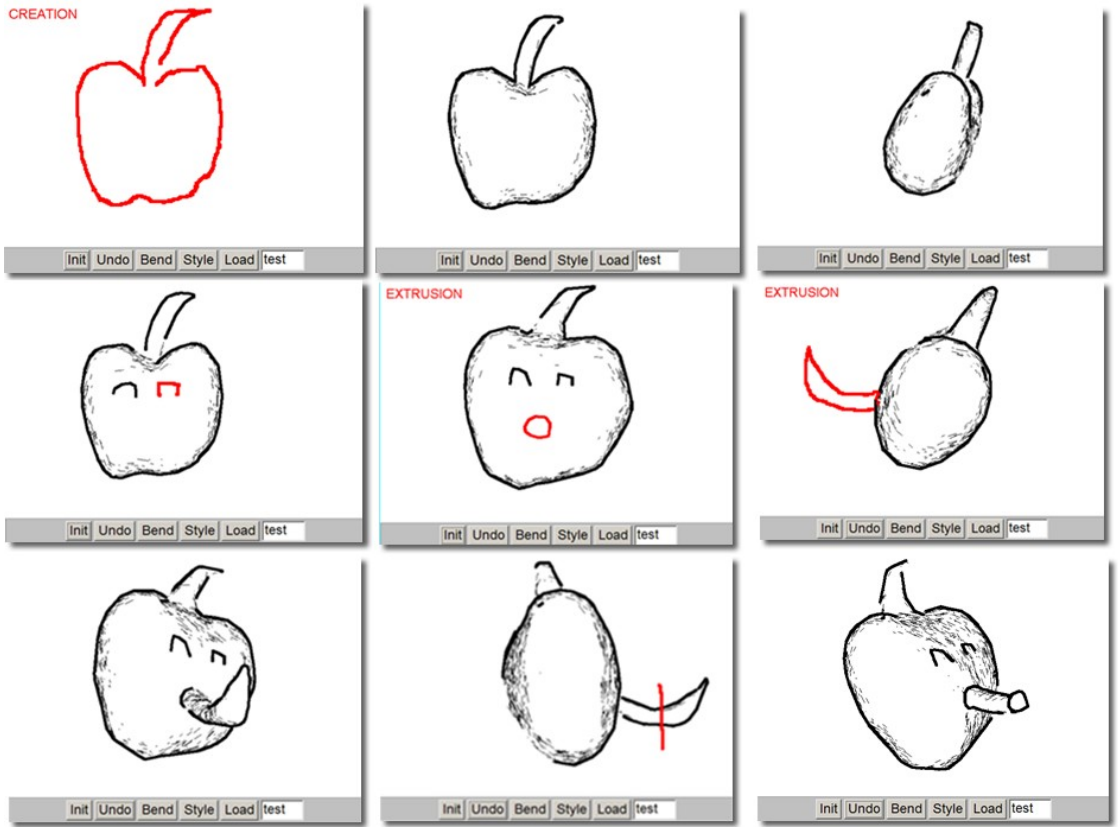

Figure 9: Basic operations of Teddy

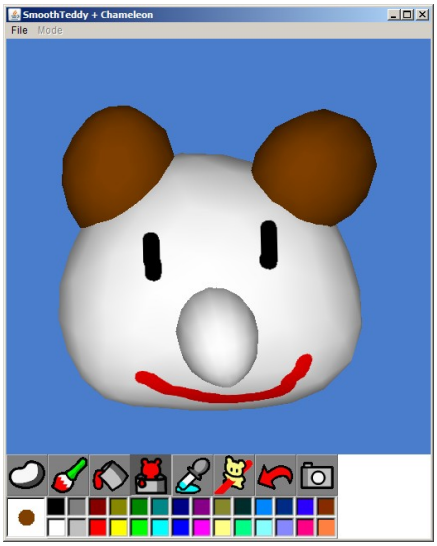

Figure 10: SmoothTeddy features much smoother models

Advantages

- The prediction algorithm is quite robust for natural objects like rocks and cartoon animals
- The application truly turns 3D modelling into 2D. Users have no need to worry about the depth of the objects, and can paint directly on a 3D surface
- The interface is extremely easy to learn as gestures are available
- It allows exporting of models to animation packages

Disadvantages

- Although SmoothTeddy was developed 6 years ago, it is still premature. Error messages are seen on the provided console almost every time the application is run
- It only works for rotund 3D objects. It is impossible to draw cubes, cylinders, rectangular prisms and the like with sharp edges in Teddy
- All objects must be connected. It is impossible to create an object with a "floating" component
- As shown in Figure 9, the body of the "apple" appears to be flat and there is no way to make it spherical

2.3.2 3D Journal

3D Journal is an interactive tool for freehand sketching developed by Cornell University (2005). It aims at combining the ease and speed of freehand sketching with the flexibility and analytical abilities of Computer-aided design (CAD) tools. To sketch a 3D model, the user will draw 2D sketches of straight and curved strokes, which will contain all edges as if the object is transparent. An algorithm will then determine the angular distribution of the strokes and offer a z-value to each vertex to create an orthogonal 3D axis system [13]. New strokes can be added onto constructed objects, which increases the flexibility of the system. This flexibility allows the initial sketch, reconstruction and adding detail to be done in a consistent interface, which improves efficiency. Moreover, an eraser tool can erase edges, a cut tool can remove parts of a face, and the stiffness of the material can be changed.

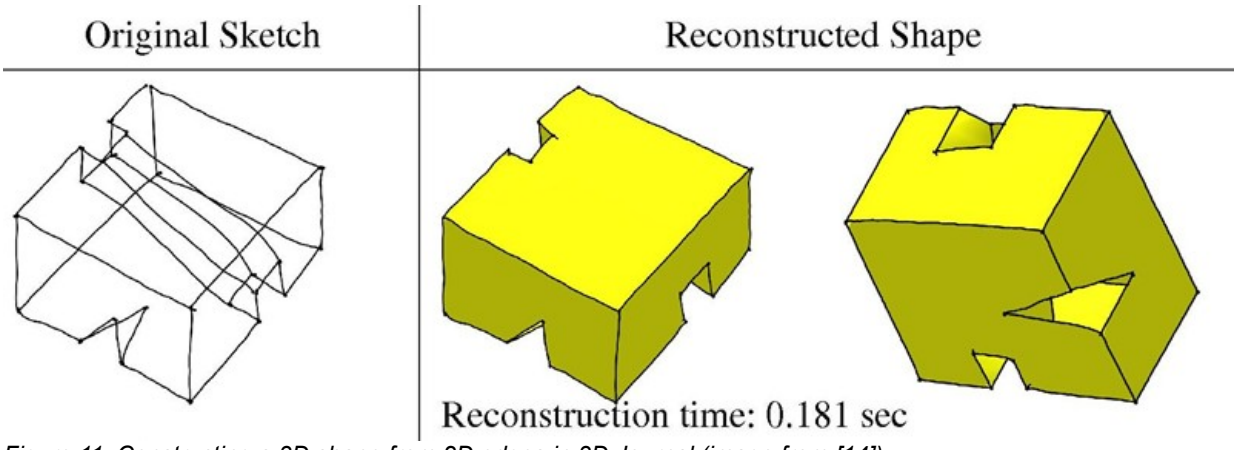

Figure 11: Constructing a 3D shape from 2D edges in 3D Journal (image from [14])

Advantages

- The system is extremely robust as small objects with about 20 strokes take only 0.2 second to construct
- It combines sketching and analysis which benefits the efficiency of CAD process
- It uses the average angular distribution of edges to determine the orientation of the object, which is generally accurate

Disadvantages

- The system can only detect shapes with perpendicular strokes and cannot be used for drawing spheres and cones
- Curves may be misinterpreted as one 2D view may represent 2 different orientations in space [14]
- For more complex objects, the amount of intersecting and overlapping edges may confuse the user
- For a small object with limited number of strokes, the output of the object may be affected by human errors

2.3.3 Google SketchUp

Google SketchUp is a powerful 3D modelling software which can be used for architecture and design, engineering, construction and digital entertainment. It is the pioneer of the patented push/pull technology, which will be incorporated into Speedy Sketching. To create a circle, you simply need to place the centre and drag the length of the radius. The snapping tool is also extremely powerful as it makes suggestions to snap lines to the nearest plane. It allows sharing of models with users worldwide and can be linked to Google Earth.

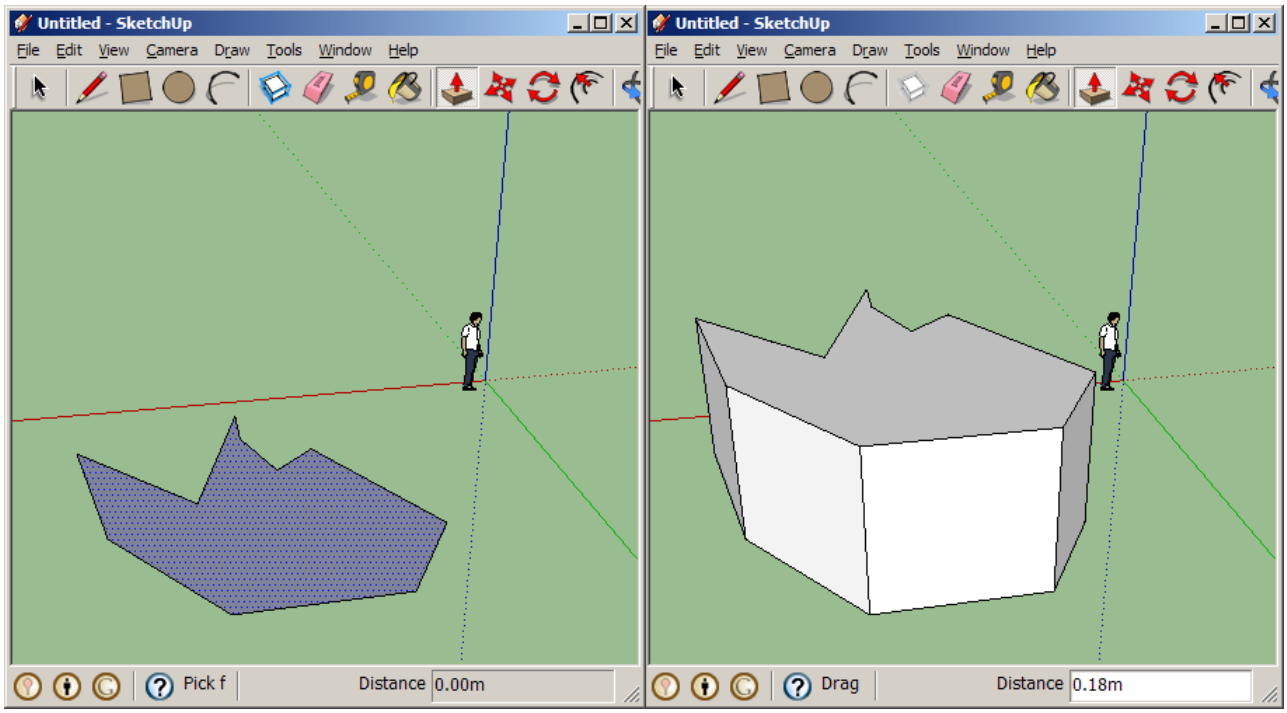

Figure 12: Pulling of an arbitrary polygon

Advantages

- Ease of use. According to Mortenson Construction, cost and speed of construction of 3D models have been vastly reduced [15] because it does not require experienced CAD designers to operate
- The innovative push/pull technology saves time and effort and removes the need to draw all vertices in 3D space

Disadvantages

- It is difficult to draw polygons on an arbitrary plane. As shown in Figure 13, a seemingly planar polygon is snapped wrongly to the main planes (x/y/z)

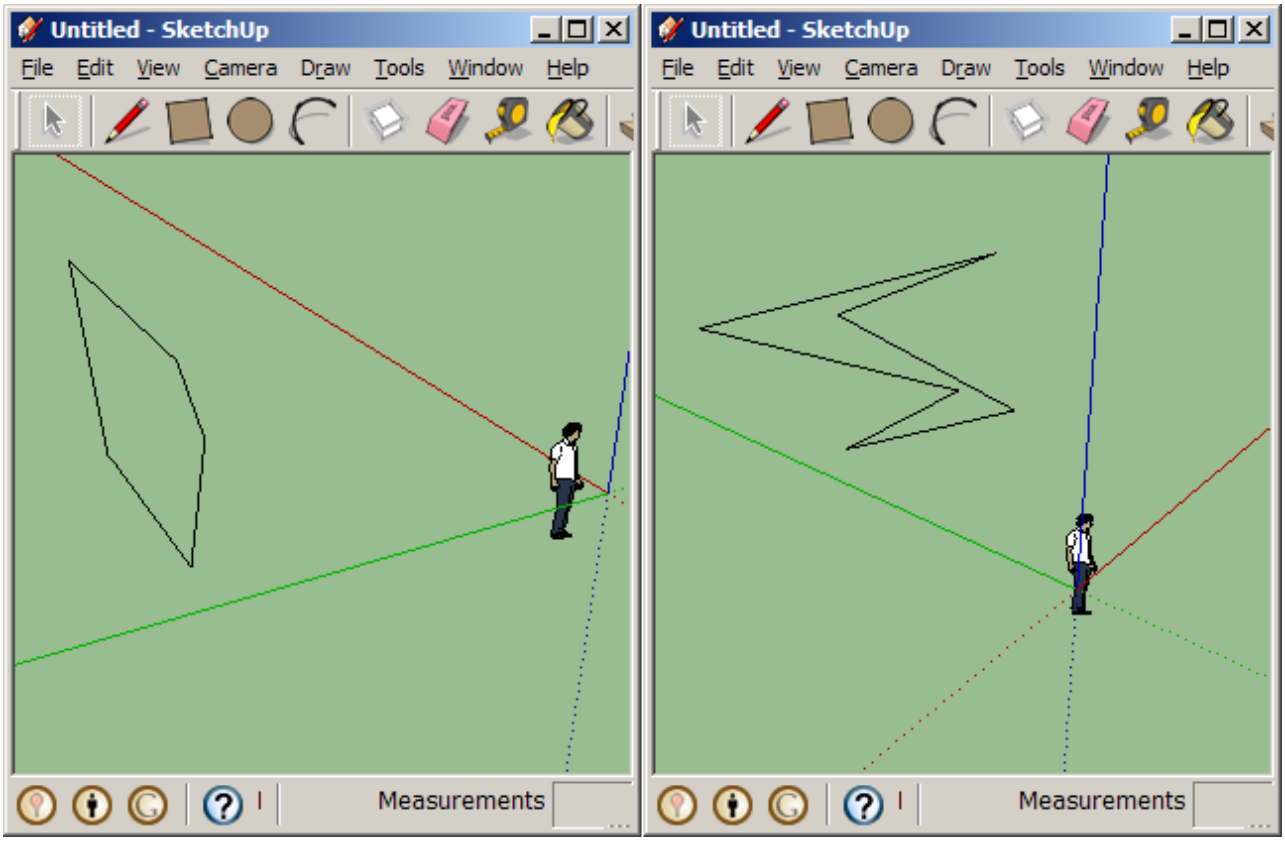

Figure 13: A supposedly planar polygon (left) and the scene rotated (right)

- Drawing of cones and spheres requires rotation of a 2D object or downloading of existing models from the web, which are slow
- Freehand drawing requires clicking the mouse button to create points. This reduces redundant points, but it is unnatural and users do not get the feeling that they are drawing with pen and paper

3 Research Methods

3.1 Choice of Platforms

C++/OpenGL have been the choice of computer graphics and gaming industries for years. C++ is known to be extremely efficient and has a speed advantage over the increasingly popular Java. On the other hand, OpenGL is designed to be full featured and runs efficiently on a wide range of graphics architectures. The package is available for free and is compatible with Apple Macintosh, Microsoft Windows as well as Linux and embedded devices. The scalability and availability make OpenGL an industry standard for graphics applications [17]. The OpenGL API is also commonly known as the easiest graphics API for computer scientists to learn and use. For these reasons, the C++/OpenGL combination is chosen for the implementation of Speedy Sketching.

However, for a GUI-based application like Speedy Sketching, a powerful GUI toolkit is required for maximum usability. The GLUT API only provides minimal interaction support, i.e. callback methods for keyboard and mouse key presses and a simple pop-up menu. Hence, the author has turned to QT for the production of a powerful GUI. QT is a cross-platform integrated development environment (IDE) for GUI design. It provides a widget to render graphics with the OpenGL API [18] so the time required for embedding an OpenGL application in a GUI will be reduced.

3.2 Data Structure

The data structure will be object-oriented which is the common approach of modern software engineering. The following objects will be stored and manipulated in the application:

Point – 3 coordinate values (x, y, z)

Line/Stroke – must contain at least 2 points

Face – must contain at least 3 lines, all lines must lie in the same plane, stores normal vector

Shape (2D) – must contain only 1 face

Shape (3D) – must contain at least 4 faces

Group (2D and/or 3D) – user defined groups of shapes

When shapes are being created, a bounding volume will be actively maintained by storing the maximum and minimum coordinates of the x, y and z axles (Figure 14).

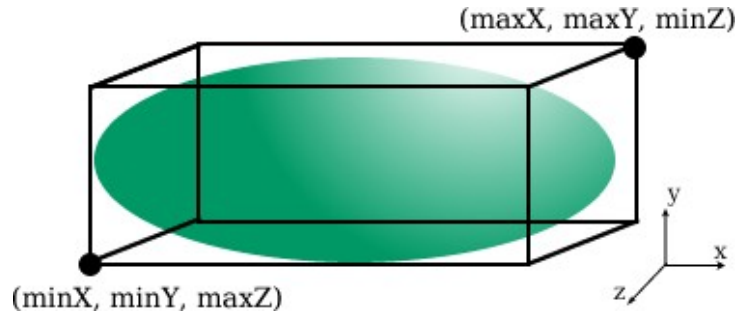

Figure 14: A bounding volume for a round object

3.3 Line Generalization

As mentioned in Section 2.2.2, line generalization techniques are needed for reducing data to be processed by an application, and to provide an aesthetically appealing appearance of shapes. Given the real time nature of line generalization, the choice of line generalization algorithms is bounded by their time and space complexities.

3.3.1 Line Simplification

In digital cartography, line simplification techniques are often used to remove points from a line that represents a feature on a map, for example, a stream. However, in Speedy Sketching, we can use a different approach to limit data size. Since user input can be tracked, data can be reduced as soon as the user draws on the Drawing Panel and before storage or displaying. If data is pruned early, the later stages of any type of processing or storage will be sped up. To determine whether to create a new point when mouse movement is detected, the following pseudo code will be used:

```
if previous point exists in the same line OR previous stroke for the same shape{
    find distance between current point and previous point
    if distance is smaller than threshold
      discard current point
    else create point
}
else create point
```

Angular Tolerance Algorithm [9]

Although the above method is capable of reducing the amount of data created, it does not remove points which are collinear or supposedly so. Angular tolerance algorithm solves this problem by connecting point n with point n+2, and finding the angle between lines n/n+2 and n/n+1. If the angle is smaller than a tolerance value, then point n+1 does not make an impact on the shape of the line and it will be removed. Otherwise, the point will be retained. By setting a high tolerance value, we can eliminate unwanted fluctuations from the line and hence the amount of data is effectively reduced. A special use of this algorithm will be introduced in Section 3.6.2 – Sketching a Cylinder.

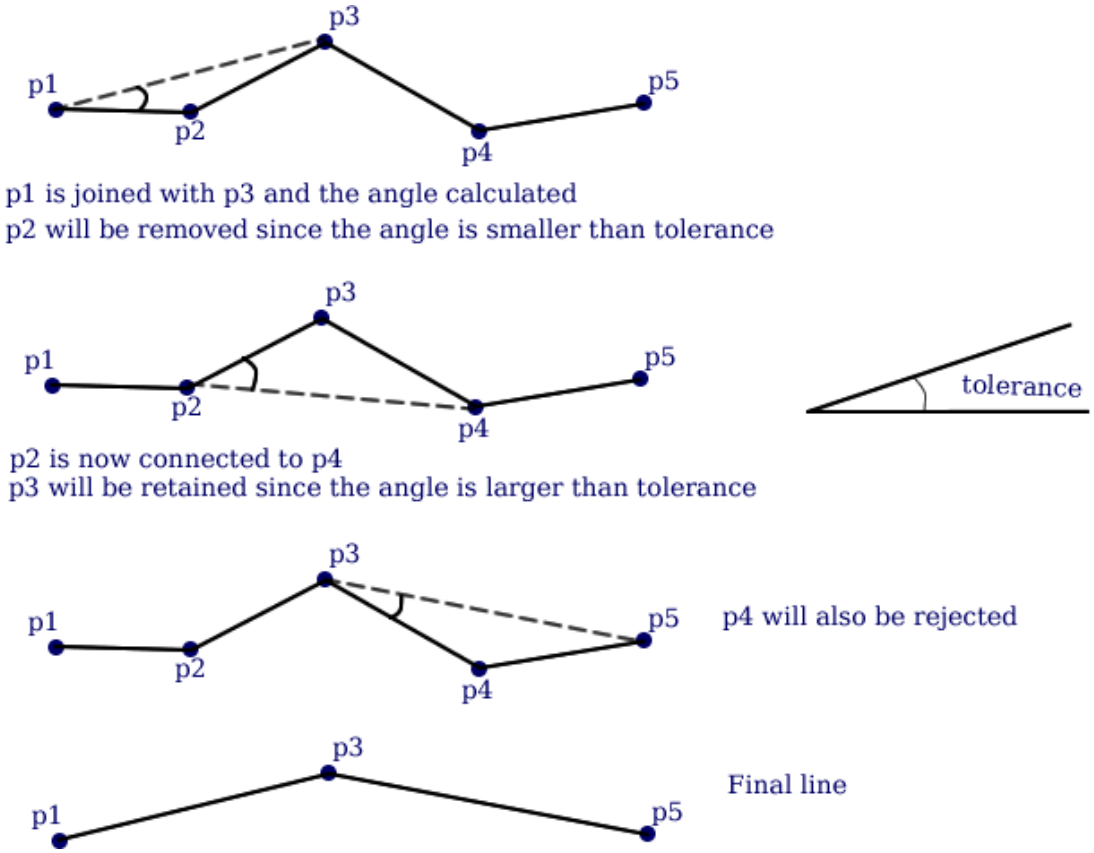

Figure 15: Angular tolerance algorithm explained

3.3.2 Line Smoothing

McMaster's Slide Averaging Algorithm [9]

This algorithm has low time and space complexities as it is only taking the average of every 5 points and sliding the third point halfway towards it. The new value of the third point is saved until all points are considered. Figure 16 explains the Slide Averaging Algorithm in detail. Using this algorithm, the number of memory locations required will be twice the length of the line – O(N). Likewise, the number of divisions will be equal to the number of points minus the 4 end points – O(N).

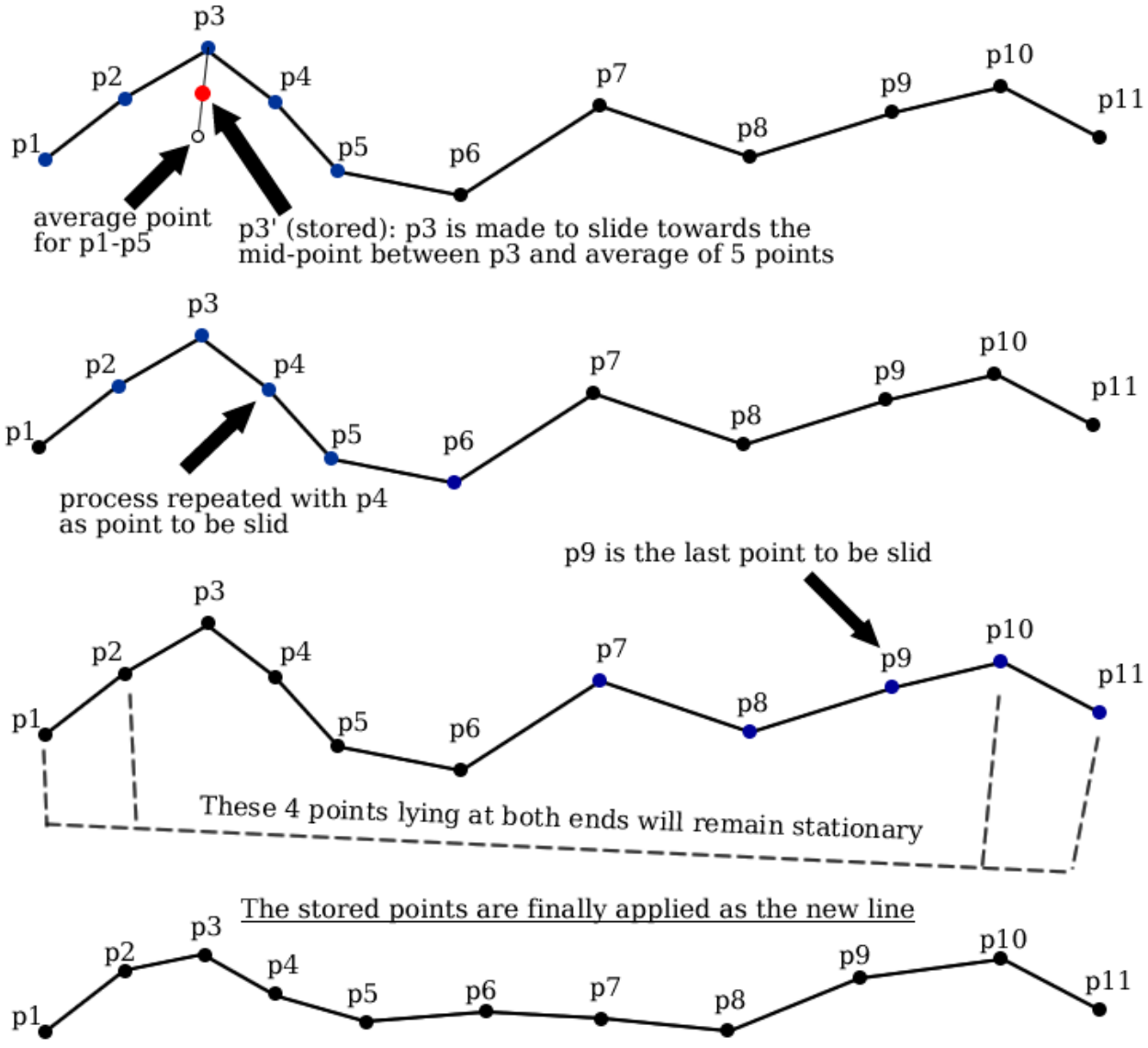

Figure 16: McMaster's Slide Averaging Algorithm [9]

Test program

To test the competence of the McMaster algorithm, a sample application has been built. As seen in the left screenshot in Figure 17, the number 6 drawn with a mouse looks very wobbly and unnatural. The smoothed number is shown in the image on the right, which looks better than before. A number of tests have been done and they show that about 5 iterations of McMaster's algorithm has to be applied for a line to be smoothed adequately. However, this may change in the actual Speedy Sketching application. Also, users may be allowed to adjust the number of iterations done.

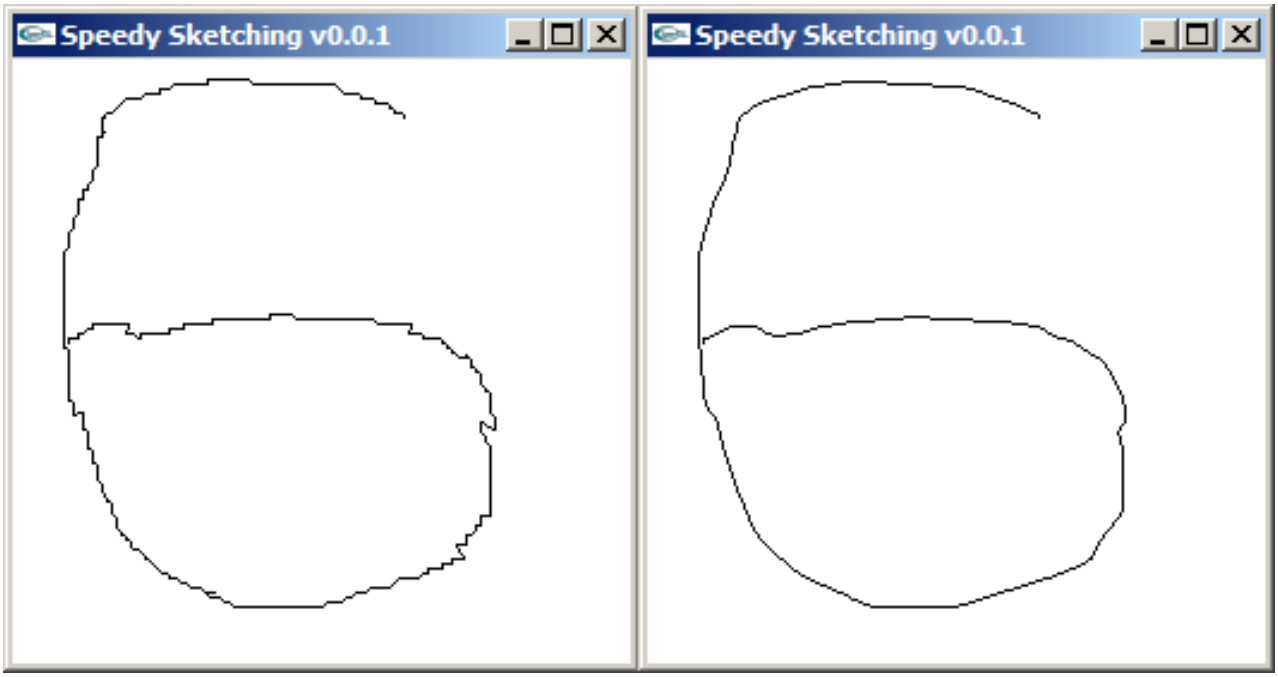

Figure 17: 5 iterations of McMaster's Slide Averaging Algorithm have been applied to the left image to obtain the right one

3.4 Shape Selection

In Speedy Sketching, users will frequently need to select shapes and objects for editing and pushing/pulling. This requires a projection of 2D coordinates on the screen to the 3D space for intersection tests, so the application can determine the user's choice. This is done by shooting a "ray" perpendicularly from the screen pixel and determining if the ray intersects with faces on either a 2D or 3D object.

As mentioned before, each 2D or 3D shape should maintain a bounding area/volume, which is crucial for the first stage of collision detection – broad phase detection – needed for pruning most of the objects in the scene quickly [19]. More research will be needed to determine the best method for shape selection.

3.5 Pulling/Pushing of Polygons

When pulling/pushing a shape, the face polygons on two ends are identical. To push/pull properly, we need to know the normal of the original face, which should have been stored during shape creation. Then, when the user creates a height for the 2D shape, we will be able to project the points of the face to the pulled location. However, the operation is not complete, as for a solid shape, we need walls on its sides as well as the "roof" and "floor". For an n-sided polygon, the number of walls to be created will be n. See Figure 18 for an example.

In this way, any arbitrary 2D shape drawn by freehand draw can be pulled into a 3D prism. Care will be taken because OpenGL only guarantees correct rendering of convex polygons [20]. Tessellation may be required by calling the built-in method in OpenGL or using an external API. Samples of convex and concave polygons are provided in Figure 19. For a convex polygon, straight lines can be drawn from any vertex to any other vertex without leaving the polygon [21].

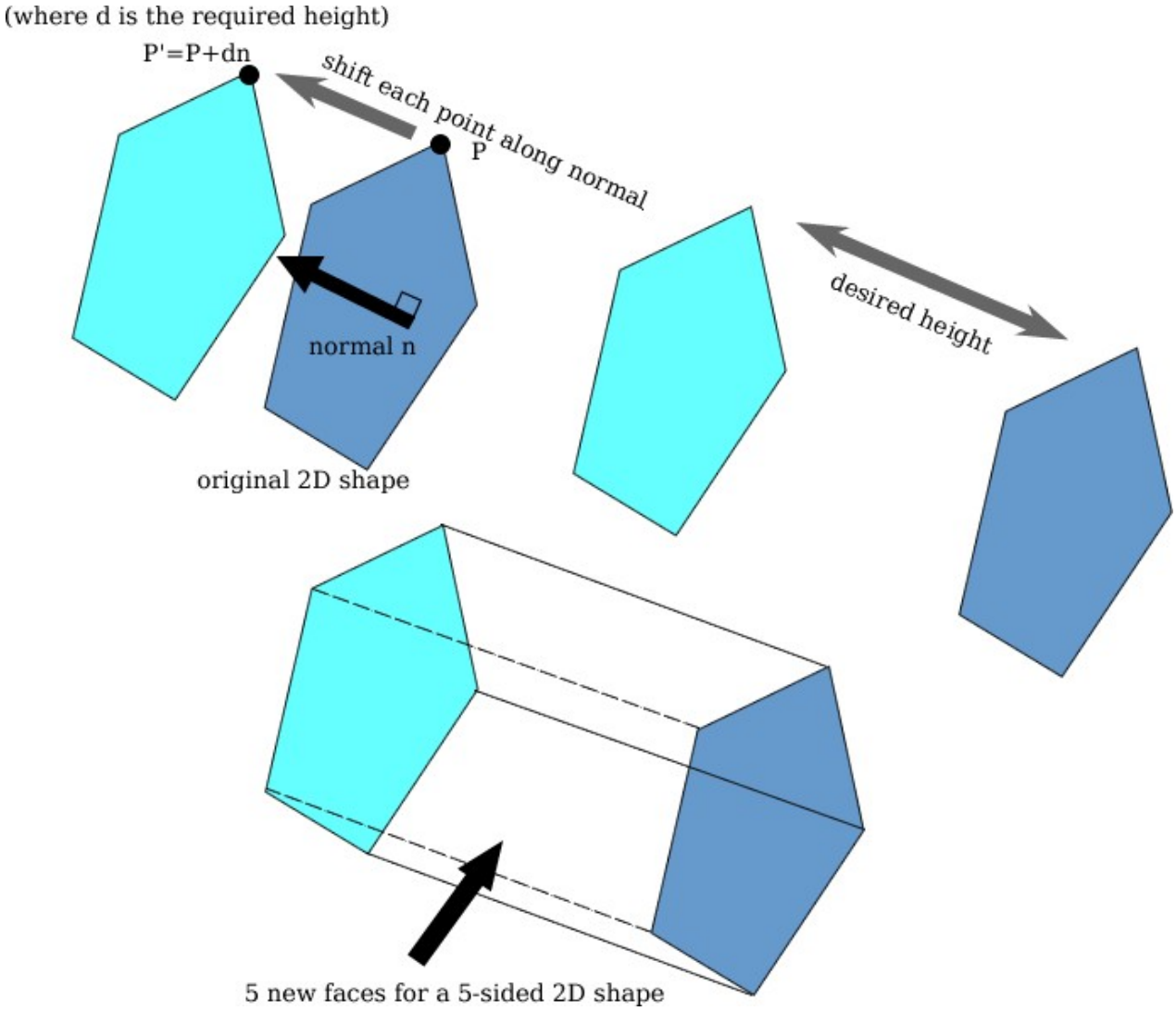

Figure 18: Pulling/pushing operation explained

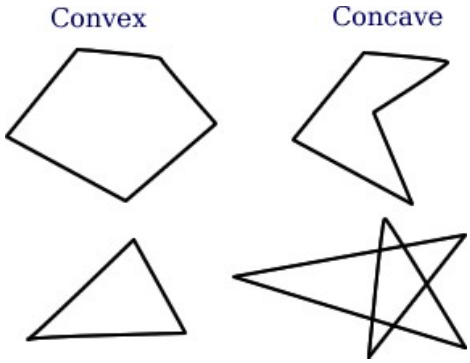

Figure 19: Convex and concave polygons

3.6 Drawing Common Shapes

The introduction of default shapes will speed up the creation process and improve accuracy, as users can rely less on freehand draw. To simplify the design, Speedy Sketching will firstly be equipped with icons for default shapes. When user clicks on them, the recognition engine will be expecting that particular shape. If time allows, Speedy Sketching may be improved to detect any

default shape without the user specifying.

3.6.1 Sketching a Circle/Sphere

To determine the parameters of a default shape, we need to analyse the points data drawn on the Drawing Panel. Take the example of sketching a circular disk or sphere, which can be done by drawing a circle. Firstly, we can average all points to find the centre of the circle. With the centre found, we can estimate the radius by calculating the average distance from all points to the centre (Figure 20). If efficiency is found to be poor, we can reduce the number of samples taken. For n points in the circle, we can take the average for 1st, n/4th, n/2th, 3n/4th and nth points.

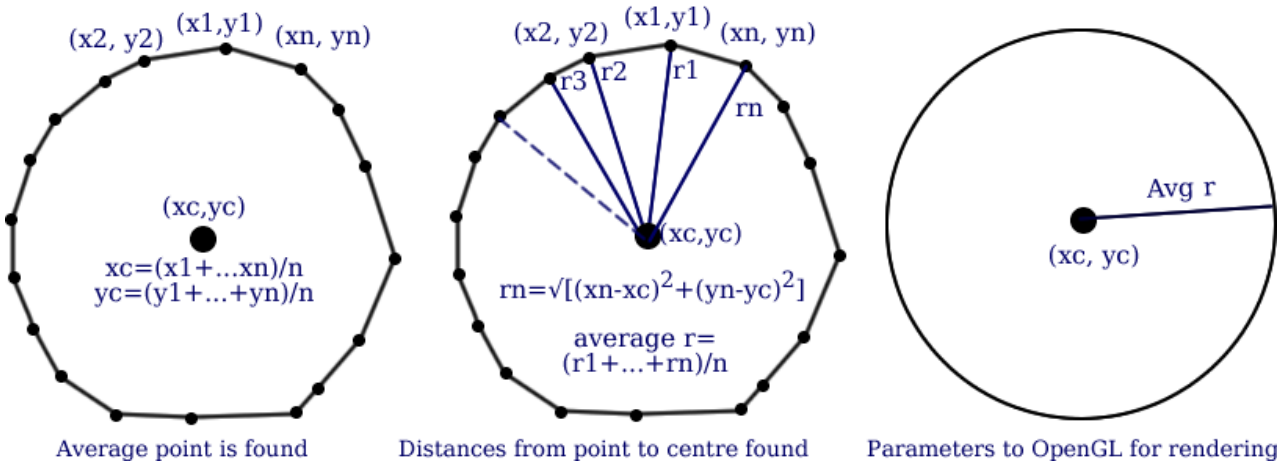

Figure 20: Finding the parameters for a circular object. In the second picture, Pythagoras' Theorem is used.

This approach is different from that in Google SketchUp where users select the centre point and then drag out the radius of the circle. We believe this is a better method because it does not require users to estimate the centre of the circle (which is difficult) and allows them to draw naturally and freely like they are given a pen and a piece of paper.

3.6.2 Sketching a Cylinder

As aforementioned, sketching a cylinder can be achieved by sketching its side view – a rectangle. As with circles, we can find the centre of the shape by averaging all points. However, we cannot determine the radius of the cylinder or the height by simple averaging, as a cylinder is not symmetric. Despite so, we can simplify the data points to obtain the 4 vertices of the rectangle. This is achieved by the angular tolerance algorithm (Section 3.3.1), where insignificant points are removed and corner vertices will remain. With only (about) 4 points left, we can easily determine the width and length of the rectangle, which are the radius and height of the cylinder respectively. Figure 21 illustrates the detection of a cylinder.

Other sketching operations include triangle to cone and square to cube. As they will be done by similar methods, they are left out from this initial report.

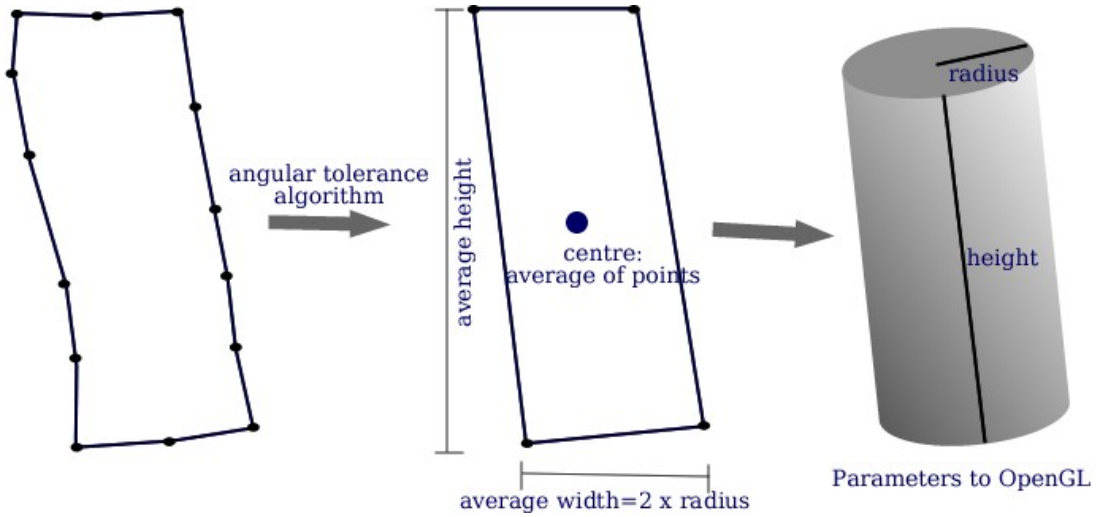

Figure 21: Determining the parameters for a cylinder

3.7 Evaluation of Work

3.7.1 Survey

Since Speedy Sketching aims at providing a quicker interface for generating 3D models, the survey will be conducted with designers who have used 3D design packages before. They will be required to compare Speedy Sketching with traditional software packages and comment on the improvements made. Also, after familiarising with the interface, they may be required to draw a 3D scene in both software they are used to and Speedy Sketching. The processes will be timed and the statistics will be used to evaluate the effectiveness of the application. A questionnaire will also raise similar questions like the following:

- How long did you take to understand Speedy Sketching's interface?
- Do you think Speedy Sketching has enhanced your creative design process? If so, how?

A formal questionnaire will be designed when the application is completed.

3.7.2 Criteria of Success

If the majority of people surveyed agree that Speedy Sketching is faster than traditional software packages, and that it stimulates creative design, the project will be deemed to be a success. If so, it will form a stepping stone for future 3D modelling software.

3.8 Project Plan

The project will begin with GUI design because it is the only possible way for data input. Also, to test various algorithms, the designer will be required to draw diagrams for evaluation and viewing the results. Moreover, it is best for the designer to use the GUI as a user as much as possible, so it can be improved continuously.

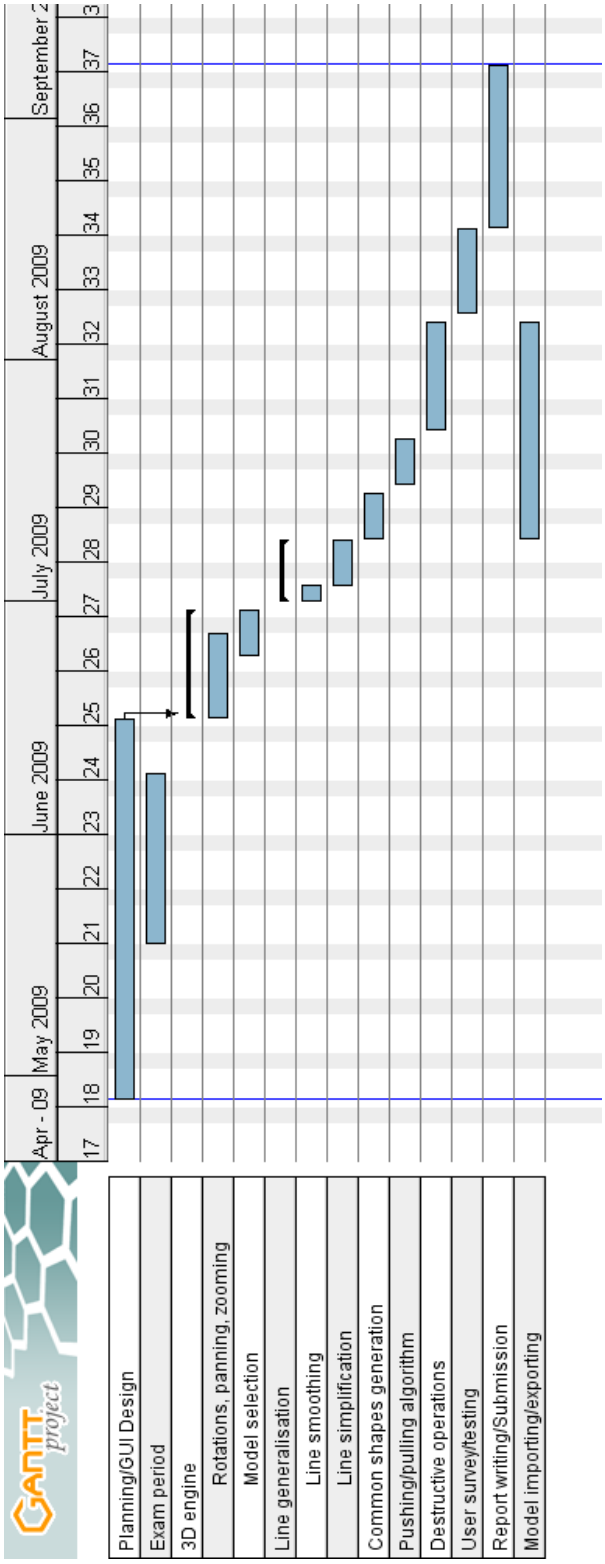

Figure 22: Gantt chart for the development of Speedy Sketching

4 References

[1] Launday, J. A., University of California, Myers, B. A., Carnegie Mellon University (2001). *IEEE Computer*, vol 34, no. 3 (March 2001), pp 56-64

[2] *Excerpts from A Conversation with Gordon Moore: Moore's Law* (Video Transcript), Intel Corporation (2005). From ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf Accessed April 11, 2009 16:45

[3] Gross, M. D. and Do, E. Y. (1996). *Ambiguous Intentions: A Paper-like Interface for Creative Design,* Proc. ACM Symp. User Interface Software and Technology, ACM Press, New York, pp. 183-192. Cited in [1]

[4] Goel, V. (1995). *Sketches of Thought,* MIT Press, Cambridge, Mass. Cited in [1]

[5] LaViola, J. J. (n.d.). *MathPad^2: A System for the Creation and Exploration of Mathematical Sketches,* from http://www.cs.brown.edu/~jjl/mathpad/ Accessed April 12, 2009 15:30

[6] LaViola, J. J. (2007). University of Florida "*An Initial Evalution of MathPad^2: A Tool for Creating Dynamic Mathematical Illustrations", Computers & Graphics: An International Journal of Systems & Applications in Computer Graphics: Algorithms & Techniques for Interaction, Multimedia, Modelling and Visualization,* Elsevier Ltd., vol 31, Issue 4 (Aug 2007), pp 540-553

[7] Buttenfield, B. P. and McMaster, R. B. (1991). *Map Generalization: Making rules for knowledge representation,* Longman Group UK Limited, pp 3-10

[8] Jenks, G. F. (1981). *Lines, Computers and Human Frailities,* Annals of the Association of American Geographers, vol 71(1), pp 1-10. Cited in [9]

[9] McMaster, R. B., and Shea, K.S. (1992). *Generalization in Digital Cartography,* Association of American Geographers, pp 27-58 Cited in [16]

[10] Igarashi, T., Matsuoka, S., Tanaka, H. (1999). *Teddy: A Sketching Interface for 3D Freeform Design,* ACM Siggraph, University of Tokyo, Tokyo Institute of Technology

[11] Igarashi, T. (2003). *SmoothTeddy: Quick 3D Modelling and Painting*, from http://www-ui.is.s.u-tokyo.ac.jp/~takeo/java/smoothteddy/index.html Accessed April 14, 2009 17:30

[12] Igarashi, T., Hughes, J.F. (2003). *Smooth Meshes for Sketch-based Freeform Modelling,* University of Tokyo and Brown University, pp 1-2

[13] Masry, M., Kang, D., Lipson, H. (2005). *A freehand sketching interface for progressive construction of 3D objects*, *Computers & Graphics: An International Journal of Systems & Applications in Computer Graphics: Algorithms & Techniques for Interaction, Multimedia, Modelling and Visualization*, Elsevier Ltd., vol 29, Issue 4 (Aug 2005), pp 563-575

[14] Lipson, H. (n.d.). *Cornell CCSL - Research - 3D Sketching,* from http://216.139.212.17/mae/ccsl/research/sketch/index.html Accessed April 20, 2009

20:00

[15] Google Inc. (2009). *SketchUp Blog,* from http://sketchupdate.blogspot.com/search/label/User%20Stories Accessed April 21, 2009 18:20

[16] Ellis, F. (n.d.). *Line Generalisation Algorithms – Lang Visualisation,* from http://www.sli.unimelb.edu.au/gisweb/LGmodule/LGMcMastersVisualisation.htm, University of Melbourne, Accessed March 23, 2009 13:47

[17] McReynolds T. and Blythe D. (2005). *Advanced Graphics Programming Using OpenGL,* Elsevier Ltd, Preface, pp xxiv

[18] Nokia (n.d.), *Qt – A cross-platform application and UI framework,* from *http://www.qtsoftware.com/products/library/modular-class-library#advanced-3d-graphics-opengl* Accessed April 23, 2009 23:10

[19] Erleben, K., Sporring, J., Henriksen, K., Dohlmann, H. (2005). *Physics-Based Animation*, Charles River Media, Inc.

[20] OpenGL Architecture Review Board (n.d.). *OpenGL Programming Guide: The Official Guide to Learning OpenGL (The Red Book) v1.1,* from http://glprogramming.com/red/chapter11.html Ch. 11. Accessed April 25, 2009 21:45

[21] Howard T. (2009). *An Introduction to Graphics Programming with OpenGL: Tutorial and Reference Manual,* University of Manchester, pp 41