Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Real time textrendering

Examensarbete utfört i Informationskodning
vid Tekniska högskolan vid Linköpings universitet
av

Gustav Adamsson

LiTH-ISY-EX--YY/NNNN--SE

Linköping 2015

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Real time textrendering

Examensarbete utfört i Informationskodning
vid Tekniska högskolan vid Linköpings universitet
av

Gustav Adamsson

LiTH-ISY-EX--YY/NNNN--SE

Handledare: **Jens Ogniewski**
ISY, Linköpings universitet
Mattias Wingstedt
Visiarc

Examinator: **Ingemar Ragnemalm**
ISY, Linköpings universitet

Linköping, 9 april 2015

Avdelning, Institution
Division, Department

informationskodning
Department of Electrical Engineering
SE-581 83 Linköping

Datum
Date

2015-04-09

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ ____________

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ ____________

URL för elektronisk version

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-XXXXX

ISBN

—

ISRN

LiTH-ISY-EX--YY/NNNN--SE

Serietitel och serienummer
Title of series, numbering

ISSN

—

Titel Realtidstextrendrering
Title Real time textrendering

Författare Gustav Adamsson
Author

Sammanfattning
Abstract

Det här som vi har hållit på med är jätteviktigt faktiskt och det vi gjort blev bara sååå bra. Kanske inte helt otippat, men det glass är såå gott!

Förresten har vi blivit bäst på att skriva rapporter, så nu ska ska vi inte gå in närmare på några detaljer såhär i sammanfattningen.

Nyckelord
Keywords textrendering, distance field, distance map, distance transform

Abstract

If your thesis is written in English, the primary abstract would go here while the Swedish abstract would be optional.

Acknowledgments

We think alla har varit så himla goa hela den här långa och tuffa tiden i våra liv.

Linköping, Januari 2020
N N och M M

Contents

Notation **ix**

1 Introduction **1**

2 Background **3**
2.1 Computer graphics 3
2.2 Basic text rendering 3
2.3 Distance fields 5
2.4 Beziér curves 5

3 Related work **9**
3.1 Early EDT algorithms 9
3.2 Exact EDT algorithms 11
3.3 Improved distance measure for EDT algorithms 11

4 Method **13**
4.1 Distance field generation initial attempt 13

5 Resultat **15**
5.1 Ditten 15
5.2 Framtiden 15
5.A Ett par långa bevis 18

6 Avslutande kommentarer **19**

Bibliography **21**

Index **24**

Notation

Några mängder

Notation	Betydelse
$\mathbb{N}$	Mängden av naturliga tal
$\mathbb{R}$	Mängden av reella tal
$\mathbb{C}$	Mängden av komplexa tal

Förkortningar

Förkortning	Betydelse
ARMA	Auto-regressive moving average
PID	Proportional, integral, differential (regulator)

1 Introduction

Redan de gamla grekerna och rommarna trodde att...

2

Background

This chapter includes relevant background information about the main topics of this report. The purpose of the chapter is to give the reader some basic knowledge and undersstanding about the touched topics to facilitate further reading of the report.

2.1 Computer graphics

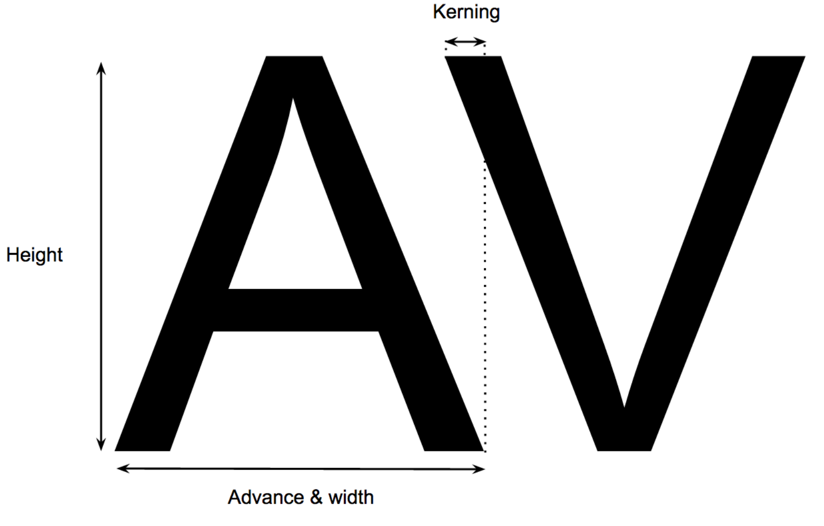
Computer graphics is good.

2.2 Basic text rendering

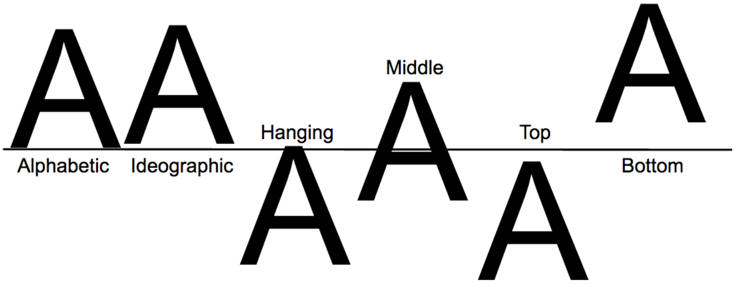
Text rendering is a non trivial subject in computer graphics and has been for a long time. There are many different ways to render text to a screen. A common way is to pre render all the glyphs of a font to an image. Characters of this font can then be rendered by using a part of the image as texture for a polygon. Another way is to use some library to render the text to an image and use it to texture a quad [FreeType, 2014a]. Both of the above mentioned methods has some drawback, for example the text can not be scaled up without losing the smooth edges and the images takes a lot of memory if you want to have high resolution text.

When rendering a text it is really important that every glyph is positioned to eachother as specified in the font file. Every glyph in a font has data stored in the font with information. For example the OpenType and the TrueType™ font format has information about of advance, kerning, height, widht etc[Microsoft Corporation, 2015, Apple Inc., 2015].

Without kerning it might be hard to get a good looking text if the font is not build in a way that kerning is not a factor. Kerning is displacement along the axis of the advancing direction of the text. A kerning value can be positive and negative and is a function of two parameters, the previous letter and the current letter. A negative kerning value is the most common and it means that the current glyph will be moved back in the advancing direction of the text and a positive value means that it will be moved forward. To get a better understanding of what kerning is take the strings “AV” and “AA” as an example. It is obvious that the left part of “V” is hanging above the “A” but that is not the case in the second string where “A” is followed by an “A”. This is because the kerning table in the font has a negative entry for the combination “AV” but not for the combimation “AA”. [FreeType, 2014b]



Another important concepts when rendering text is baseline. A text can be placed on several different baselines on differente heights. The baselines used in this thesis work is top, hanging, middle, alphabetic, ideographic and bottom. The most commonly used baseline is the alphabetic baseline which is used when for example writing english text on a piece of paper.



2.3 Distance fields

Binary images is used frequently in image processing [Ragnemalm, 1993]. A binary image is an image with only one binary value for each pixel. This will limit the image to only contain two values or colors. One example of an image that could be represented as a binary image is a picture of a white letter on a black background. In this case white would be represented as 0 and black as 1 or vice versa.

A distance transform is the transform of an input binary image to an output image called distance field or distance map [Rosenfeld and Pfaltz, 1966]. The values of the distance field pixels represents the closest distance by some distance metric to an arbitrary shape in the input image. The distance metric can differ between different applications but the most commonly used distance metric is the euclidean distance metrics which is also called real distance. If euclidean distance metrics is used when doing the distance transform it is called euclidean distance transform(EDT). Other distance metrics that can be used in distance transforms is the city-block distance metric and the chessboard distance metric. When using city-block distances you are only allowed to travel horizontally or vertically in a grid and with the cost 1. Chessboard distance is an extension of city-block distance where you are also allowed to travel diagonally with the cost of 1.

A distance field can be signed or unsigned. An unsigned distance field only maps the distances either inside the shape or outside the shape while a signed distance field maps the pixels inside the shape with negative distances and the outside of the shape with positive distances or vice versa.

Distance fields has many good advantages compared to other methods for storing shapes like boundary representation. One of the most obvious advantages is that a distance field represents more than just the boundary. For example a signed distance field also represents the environment around the boundary, both the interior and the exterior. This makes it easy to check if a point is inside or outside a shape by checking the value of that point in the distance field. It is also easy to move the boundary of the object by just changing the threshold value determining where the boundary is located. [Jones et al., 2006]

2.4 Beziér curves

A beziér curve is defined in space or the plane as two endpoints and a number of control points which is blended together with one blending functions per point. The number of control points depends on the degree of the beziér curve. The most commonly used beziér curves is the cubic beziér with four points and the quadratic beziér with three points. A cubic beziér is defined as:

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3,$$

where $t \in [0,1]$, P_0 and P_3 are the endpoints of the beziér and P_1 and P_2 are the control points of the beziér. The sum of the blending functions is always equal to 1 for any t in the function domain.[Ragnemalm, 2008]

A bezier curve can not represent arbitrary curves, therefore it is important to be able to connect several beziers to a path to be able to describe more complex shapes. When connecting curves, continuity between the curves is an important concept have in mind. Parametric continuity is a measure that will have grave importance of the appearance of the curve. There are three levels of parametric continuity.

C^0 = The curves meet
C^1 = The derivative for the both curves is equal where the curves meet
C^2 = The second derivative for the both curves is equal where the curves meet

Another measure than can be used for continuity is geometric continuity which is almost the same as parametric continuity. The only difference is that geometric continuity only require the derivatives of G^1 and G^2 to be proportional and not equal. Given two cubic beziér curves p defined by p_1, p_2, p_3 and p_4 and q defined by q_1, q_2, q_3 and q_4. Assume that p and q is placed in a way that $p_4 = q_1$. This will trivially fulfill the requirement for C^0 because they share one endpoint and $p(1) = q(0)$. C^1 continuity, $p'(1) = q'(0)$ will be fulfilled if p_3, p_4 and q_1 is located on the same line. C2 continuity $p''(1) = q''(0)$ will be fulfilled if p_3, p_4 and q_1 is located on the same line and $|p_4 - p_3| = |q_2 - q_1|$ meaning that the distance from p_3 to p_4 must be equal to the distance from p_4 to q_2.

An example of an application of beziérs is fonts. Every glyph in a font is stored as a number of straight lines and bezier curves, either cubic or quadratic.[Phinney, 2001] Even though beziérs is used in fonts it is not a trivial problem to draw a beziér to the screen. Beziér curves is hard to draw because at the lowest level graphics hardware can only draw polygons and line segments. This is solved by approximating smooth curves to line segments before drawing. [Shreiner et al., 2009] Another problem with beziér curves is that it time consuming to find the closest point on the beziér curve given an arbitrary point. To solve this problem $(p - q(t))q'(t) = 0$, where $t \in [0, 1]$, p is a point in space and q is a beziér curve, has to be solved[Chen et al., 2007]. This implies that a quintic polynomial needs to be solved to find the closest point on a cubic beziér curve given a point in space. This is another reason why another representation like line segments is much easier to work with in applications like computer graphics.

De Casteljau's algorithm can be used to subdivide a bezier curve recursively by using the properties of bezier curves to calculate new beziér points for both the left and the right part of the old beziér.[Fischer, 2000] A special case of the algorithm is to divide the curve at $t = 0.5$ which gives a first curve $t \in [0, 0.5]$ and a second curve $t \in [0.5, 1]$. An example of this subdivision is presented in figure XXX below, where the first curve is described by the points p_0, p_{01}, p_{012} and p_{0123} and the second curve is described by p_{0123}, p_{123}, p_{23} and p_3 is derived from the initial curve described by p_0, p_1, p_2 and p_3. The following calculations shows how the points is computed.

$$
\begin{aligned}
P_{01} &= \tfrac{P_0+P_1}{2}\\
P_{23} &= \tfrac{P_2+P_3}{2}\\
P_{12} &= \tfrac{P_1+P_2}{2}\\
P_{012} &= \tfrac{P_{01}+P_{12}}{2}\\
P_{123} &= \tfrac{P_{12}+P_{23}}{2}\\
P_{0123} &= \tfrac{P_{012}+P_{123}}{2}
\end{aligned}
$$

3

Related work

3.1 Early EDT algorithms

In a very often mentioned article Danielsson [1980] proposed an improved way to generate distance maps by representing the output of the distance transform as a vector, separating the distance of the different dimensions. Danielsson also proposed two sequential algorithms 4SED (4-point Sequential Euclidean Distance mapping) and 8SED (8-point Sequential Euclidean Distance mapping) for calculating the EDT using his representation of distance. Both the 4SED algorithm and the 8SED algorithm consists of 2 consecutive picture scans where they incrementally update pixel values depending on nearby pixels. The 8SED algorithm is described in the following pseudocode.

```
Data: M: Width, N: Height, L: Binary image of size MxN
Result: L: Distance field
initialization;
for j = 1 to N − 1 step 1 do
    for i = 0 to M − 1 step 1 do
        L(i, j) = min(L(i, j), L(i-1, j-1)+(1, 1), L(i, j-1)+(0+1), L(i+1,
        j-1)+(1,1));
    end
    for i = 1 to M − 1 step 1 do
        L(i, j) = min(L(i, j), L(i-1, j)+(1, 0));
    end
    for i = M − 2 to 0 step 1 do
        L(i, j) = min(L(i, j), L(i+1, j)+(1, 0));
    end
end
```

Algorithm 1: First scan of the 8SED algorithm

```
Data: M: Width, N: Height, L: Binary image of size MxN
Result: L: Distance field
initialization;
for j = N − 2 to 0 step 1 do
    for i = 0 to M − 1 step 1 do
        L(i, j) = min(L(i, j), L(i-1, j+1)+(1, 1), L(i, j+1)+(0, 1), L(i+1, j+1)+(1,
        1));
    end
    for i = 1 to M − 1 step 1 do
        L(i, j) = min(L(i, j), L(i-1, j)+(1, 0));
    end
    for i = M − 2 to 0 step 1 do
        L(i, j) = min(L(i, j), L(i+1, j)+(1, 0));
    end
end
```

Algorithm 2: Second scan of the 8SED algorithm

The two scans are very similar. The only difference is that the first scan is done top down evaluating the pixels below and on the sides of the current pixel and the second scan is done bottom up evaluating the pixels above and on the sides of the current pixel[Ragnemalm, 1993]. The 4SED algorithm and the 8SED algorithm are not error free as Danielsson [1980] proves in his article but he also claims that the errors are rare and small and is negligible for practical purposes. In 8SED and 4SED every pixel is visited a constant number of times. This makes the time complexity of 8SED and 4SED trivially $\mathcal{O}(mn)$, if m and n are the width respectively the height of the input image.

3.2 Exact EDT algorithms

Distance transforms has been used in different applications in many years. The article from Danielsson [1980] was not the first on the subject but the fact that the algorithms he proposed in his article are some of the most widely used[Fabbri et al., 2008] makes it easy to call Danielsson one of the pionjeers on the subject. As mentioned earlier in this report, 8SED and 4SED is not exact. Exact algorithms for EDT has only been around since about the 1990s. In an article Fabbri et al. [2008] compares the execution time of exact EDT algorithms. The 6 algorithms compared in the test is Meijster et al. [2000], Maurer Jr et al. [2001], Eggers [1998], Lotufo and Zampirolli [2001], Cuisenaire and Macq [1999] and Saito and Toriwaki [1994]. The conclusion of the comparison is that the algorithms from Meijster and Maurer are the fastest but Meijster's algorithm is preferred due to slightly better performance and the fact that it is easier to implement than Maurer's algorithm.

Meijster's algorithm consist of two separate phases. The first phase iterates through each column performing a distance propagation in both directions. This will create a new image $g(i,j)$ which used in the second phase. The second phase iterates through each row left to right and right to left applying a function $DT(x,y)$ to calculate the output value of each pixel. The function depends on what distance metric is used. The function used for euclidean distance transform follows.

$$DT(x,y) = \min_{0...m}((x-i)^2 + g(i)^2)$$

With the same motivation for 4SED and 8SED the time complexity of Meijster's algorithm is $\mathcal{O}(nm)$, if m and n are the width respectively the height of the image.

3.3 Improved distance measure for EDT algorithms

Calculating a distance transform of large size images can be very time consuming. There is a significant difference in computation time between creating a distance map from a 4096x4096 image and creating a distance map from a 64x64 image. For example, the 8SED and the 4SED algorithms both has time complexity O(n) where n is the number of pixels. Time complexity O(n) means that every pixel is visited a constant number of times when transforming the image. Assuming every pixel is visited once and one calculation is done per visit the larger image would require 4096 times more calculations than the smaller image. In an article from Green [2007], distance fields was generated by using a 4096x4096 binary image of a glyph as input to a distance transform. The output from the distance transform was then downsampled to a 64x64 texture. Generating a distance field using a large input image makes the discrete set of possible boundary pixels more dense compared to if a smaller input image is used. The increased density of the possible pixel set helps decrease the calculation error of the distance field assuming subpixel distance measures is not used.

In an article, Gustavson and Strand [2011] showed that the calculation errors of distance transforms can be decreased by using information about the subpixel boundary between the foreground and background pixels. In the article they use an anti-aliased greyscale input image to be able to locate the boundary in the pixels to get more precise distance measures between the pixel and the boundary.

They show that they get approximately the same amount of errors using their method on a 16x16 times smaller input image than by using the method Chris Green proposed in his article. The smaller input image increases the the execution time and decreases the memory consumption by a factor of approximately 30 times according to Gustavson and Strand.

Skriv lite om hur algoritmen funkar....

4 Method

This chapter will give a detailed description on which how the problem real-time text rendering on mobile devices was solved in this thesis. This chapter is divided into two different parts. The first part gives a detailed description about the implementation of the distance transform module. This part has been reimplemented several times due to poor performance. The second part gives a detailed description about the implementation of the distance field rendering module.

4.1 Distance field generation initial attempt

There is many EDT algorithms for creating distance fields. Most of the EDT algorithms used today run in $\mathcal{O}(nm)$, where n and m are the width respectively the height of the image. Example of algorithms running in $\mathcal{O}(nm)$ is Danielsson [1980] and Meijster et al. [2000].

The initial implementation of distance field generation was built around Gustavson and Strand [2011] article on the subject. A version of the 8SED algorithm was used along with the anti-aliased sub-pixel distance measure proposed in the article. A signed distance field has several advantages compared to an unsigned distance field. The most obvious advantages is the increased flexibility it gives and that it is easier to perform proper anti-aliasing on the border due to the increased information in an environment around the border pixels[Gustavson, 2012]. This is more than enough reason to implement the generation to generate a signed distance field. To generate a signed distance field the distance transform was ran twice with inverted input representation. This will make one of the transforms create a distance field on the inside of the glyph and the other transform on the outside of the glyph. The resulting distance field was then calculated with the following formula.

$$result(i,j) = inside(i,j) - outside(i,j), \forall (i,j), i \in \{0,\ldots,m-1\}, j \in \{0,\ldots,n-1\}$$

5 Resultat

Det här är kapitlet där resultaten presenteras.

5.1 Ditten

Liksom [?] har vi kommit fram till att glass smakar bäst på sommaren.

När vi nu går in på hur glass smakar vid olika tidpunkter under dagen hänvisar vi till figur 5.1, och speciellt till figur 5.1b. Jämför sedan med figur 5.2 för att se hur det kan bli när man äter glass vid okontrollerade tidpunkter.

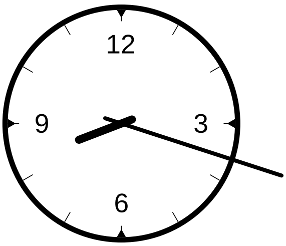
Veselić, Krešimir (Veselić, Krešimir) skrev en gång en artikel med titeln *Bounds for exponentially stable semigroups*.

Kommer att tänka på en liten anekdot...

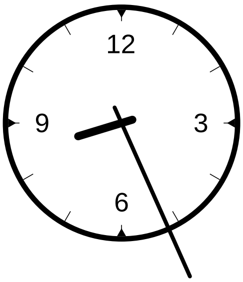
TODO: Ta bort den löjliga anekdoten!

5.2 Framtiden

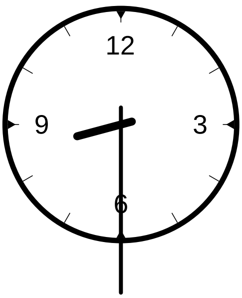
Sen när glassen är uppäten är det bara till att sätta igång och skriva på exjobbet igen!



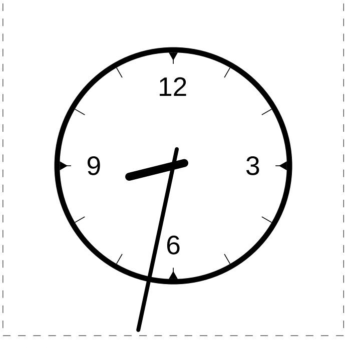
*(**a**) Det här är väl tidigt — din glass hinner smälta innan ditt sällskap dyker upp.*



*(**b**) Kiosken stänger snart, men inte nu — perfekt!*

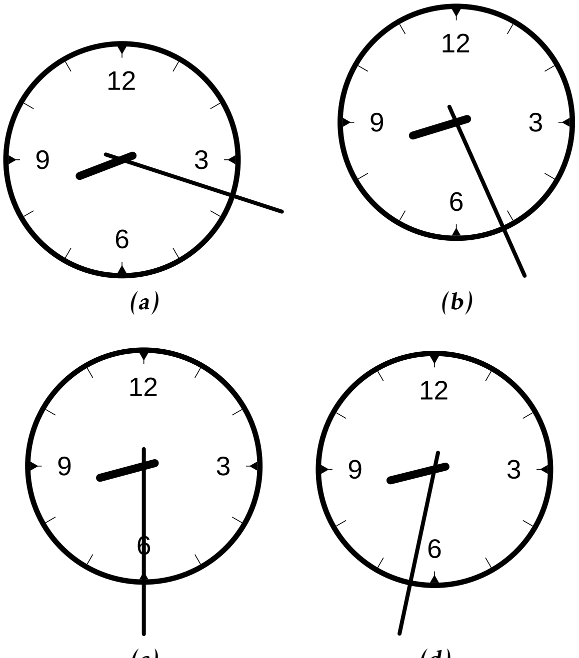


*(**c**) Precis i tid — du får in ett finger i luckan just när kiosken ska stänga. Han som jobbar blir sur, och det blir smolk i bägaren.*



*(**d**) Du är sen — kiosken är stängd.*

Figure 5.1: *Illustration* av subfloats. *Den så kallade* bounding box*en visas i (d). Lägg märke till att bounding boxen har satts så att alla bilder har samma storlek, med enhetlig placering av själva innehållet i förhållande till bounding boxen. Antag att du ska träffa en kompis för att äta glass just när kiosken stänger för dagen vid 08:30. När dyker du upp?*



(a) *(b)*

(c) *(d)*

Figure 5.2: *En andra illustration av* subfloats. *Den här gången har bounding boxen gjorts så liten som möjligt runt själva innehållet. Resultatet är stökiga placeringar på sidan. Samma sak kan hända med vanliga fyrkantiga figurer när man har text som spretar ut åt lite olika håll från själva rutan med kurvor i.*

Appendix

5.A Ett par långa bevis

Det här är en appendix-del av det aktuella kapitlet.

6

Avslutande kommentarer

Sätt av ett kort kapitel sist i rapporten till att avrunda och föreslå rikningar för framtida utveckling av arbetet.

Bibliography

Apple Inc. Truetype™ reference manual, 2015. URL `https://developer.apple.com/fonts/TrueType-Reference-Manual/`. Cited on page 3.

Xiao-Diao Chen, Yin Zhou, Zhenyu Shu, Hua Su, and J.-C. Paul. Improved algebraic algorithm on point projection for beziércurves. In *Computer and Computational Sciences, 2007. IMSCCS 2007. Second International Multi-Symposiums on*, pages 158–163, Aug 2007. doi: 10.1109/IMSCCS.2007.17. Cited on page 6.

Olivier Cuisenaire and Benoît Macq. Fast euclidean distance transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding*, 76(2):163–172, 1999. Cited on page 11.

Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227 – 248, 1980. ISSN 0146-664X. doi: http://dx.doi.org/10.1016/0146-664X(80)90054-4. URL `http://www.sciencedirect.com/science/article/pii/0146664X80900544`. Cited on pages 9, 10, 11, and 13.

Hinnik Eggers. Two fast euclidean distance transformations in z 2 based on sufficient propagation. *Computer Vision and Image Understanding*, 69(1):106–116, 1998. Cited on page 11.

Ricardo Fabbri, Luciano Da F. Costa, Julio C. Torelli, and Odemir M. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.*, 40(1):2:1–2:44, February 2008. ISSN 0360-0300. doi: 10.1145/1322432.1322434. URL `http://doi.acm.org/10.1145/1322432.1322434`. Cited on page 11.

Kaspar Fischer. Piecewise linear approximation of bezier curves. In *HTTP://HCKLBRRFNN. WORDPRESS. COM/2012/08/20/PIECEWISE-LINEAR-APPROXIMATION-OF-BEZIER-CURVES/*. Citeseer, 2000. Cited on page 6.

FreeType. Freetype outlines, 2014a. URL `http://www.freetype.org/freetype2/docs/glyphs/glyphs-6.html`. Cited on page 3.

FreeType. Kerning, 2014b. URL http://www.freetype.org/freetype2/docs/glyphs/glyphs-4.html. Cited on page 4.

Chris Green. Improved alpha-tested magnification for vector textures and special effects. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, pages 9–18, New York, NY, USA, 2007. ACM. ISBN 978-1-4503-1823-5. doi: 10.1145/1281500.1281665. URL http://doi.acm.org/10.1145/1281500.1281665. Cited on page 11.

Stefan Gustavson. 2d shape rendering by distance fields. 2012. Cited on page 13.

Stefan Gustavson and Robin Strand. Anti-aliased euclidean distance transform. *Pattern Recognition Letters*, 32(2):252 – 257, 2011. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2010.08.010. URL http://www.sciencedirect.com/science/article/pii/S0167865510002953. Cited on pages 12 and 13.

M. Jones, J.A. Baerentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *Visualization and Computer Graphics, IEEE Transactions on*, 12(4):581–599, July 2006. ISSN 1077-2626. doi: 10.1109/TVCG.2006.56. Cited on page 5.

Roberto A Lotufo and Francisco A Zampirolli. Fast multidimensional parallel euclidean distance transform based on mathematical morphology. In *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*, pages 100–105. IEEE, 2001. Cited on page 11.

Calvin R Maurer Jr, Vijay Raghavan, and Rensheng Qi. A linear time algorithm for computing the euclidean distance transform in arbitrary dimensions. In *Information Processing in Medical Imaging*, pages 358–364. Springer, 2001. Cited on page 11.

Arnold Meijster, Jos BTM Roerdink, and Wim H Hesselink. A general algorithm for computing distance transforms in linear time. In *Mathematical Morphology and its applications to image and signal processing*, pages 331–340. Springer, 2000. Cited on pages 11 and 13.

Microsoft Corporation. Opentype specification, 2015. URL https://www.microsoft.com/typography/otspec/default.htm. Cited on page 3.

Thomas W Phinney. Truetype, postscript type 1, & opentype: What's the difference, 2001. Cited on page 6.

Ingemar Ragnemalm. *The Euclidean Distance Transform*. Linköping Studies in Science and Technology, first edition, 1993. ISBN 91-7871-083-9. Cited on pages 5 and 10.

Ingemar Ragnemalm. *Polygons feel no pain*. First edition, 2008. Cited on page 6.

Azriel Rosenfeld and John L Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM (JACM)*, 13(4):471–494, 1966. Cited on page 5.

Toyofumi Saito and Jun-Ichiro Toriwaki. New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern recognition*, 27(11):1551–1565, 1994. Cited on page 11.

Dave Shreiner, Bill The Khronos OpenGL ARB Working Group, et al. *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009. Cited on page 6.

Index

ARMA
abbreviation, ix

greker, 1

PID
abbreviation, ix