Infinite Precision Document Formats

Author: Samuel Moore[1]
Partners: David Gow[2]
Supervisor: Prof Tim French

THE UNIVERSITY OF
WESTERN AUSTRALIA
Achieve International Excellence

April 29, 2014

Abstract

At the fundamental level, a document is a means to convey information. The limitations on a digital document format therefore restrict the types and quality of information that can be communicated. Whilst modern document formats are now able to include increasingly complex dynamic content, they still suffer from early views of a document as a static page; to be viewed at a fixed scale and position. In this report, we focus on the limitations of modern document formats (including PDF, PostScript, SVG) with regards to the level of detail, or precision at which primatives can be drawn. We propose a research project to investigate whether it is possible to obtain an "infinite precision" document format, capable of including primitives created at an arbitrary level of zoom.

Move to introduction? But it discusses the Introduction :S

In Chapter 1 we give an overview of the current state of the research in document formats, and the motivation for implementing "infinite precision" in a document format. We will outline our approach to research in collaboration with David Gow[]. In Chapter 2 we provide more detailed background examining the literature related to rendering, interpreting, and creating document formats, as well as possible techniques for increased and possibly infinite precision. In Chapter **??** gives the current state of our research and the progress towards the goals outlined in Chapter 1. In Chapter 4 we will conclude with a summary of our findings and goals.

Keywords: *document formats, precision, floating point, graphics, OpenGL, VHDL, PostScript, PDF, bootstraps*

TODO: Make document smaller; currently 16 pages with almost no content; limit is 20 with actual content

Contents

Chapter 1

Introduction

Most of this chapter is copy pasted from the project proposal
⟨http://szmoore.net/ipdf/documents/ProjectProposalSam.pdf⟩

Early electronic document formats such as PostScript were motivated by a need to print documents onto a paper medium. In the PostScript standard, this lead to a model of the document as a program; a series of instructions to be executed by an interpreter which would result in "ink" being placed on "pages" of a fixed size[3]. The ubiquitous Portable Document Format (PDF) standard provides many enhancements to PostScript taking into account desktop publishing requirements[4], but it is still fundamentally based on the same imaging model[5]. This idea of a document as a static "page" has lead to limited precision in these and other traditional document formats.

The emergence of the internet, web browsers, XML/HTML, JavaScript and related technologies has seen a revolution in the ways in which information can be presented digitally, and the PDF standard itself has begun to move beyond static text and figures[6, 7]. However, the popular document formats are still designed with the intention of showing information at either a single, fixed level of detail, or a small range of levels.

As most digital display devices are smaller than physical paper medium, all useful viewers are able to "zoom" to a subset of the document. Vector graphics formats including PostScript and PDF support rasterisation at different zoom levels[3, 5], but the use of fixed precision floating point numbers causes problems due to imprecision either far from the origin, or at a high level of detail[**?**, 8].

We are now seeing a widespread use of mobile computing devices with touch screens, where the display size is typically much smaller than paper pages and traditional computer monitors; it seems that there is much to be gained by breaking free of the restricted precision of traditional document formats.

1.1 Aim

In this project, we will explore the state of the art of current document formats including PDF, PostScript, SVG, HTML, and the limitations of each in terms of precision. We will consider designs for a document format allowing graphics primitives at an arbitrary level of zoom with no loss of detail. We will refer to such a document format as "infinite precision". A viewer and editor will be implemented as a proof of concept; we adopt a low level, ground up approach to designing this viewer so as to not become restricted by any single existing document format.

New bit

It is necessary to clarify what we mean by "infinite" precision. We do not propose to be able to contain an infinite amount of information within a document. The goal is to be able to render a primitive at the same level of detail it is specified by a document format. For example, the precision of coordinates of primitives drawn in a graphical editor will always be limited by the resolution of the display on which they are drawn, but not by the viewer.

There are many possible applications for documents in which precision is unlimited. Several areas of use include: visualisation of extremely large or infinite data sets; visualisation of high precision numerical computations; digital artwork; computer aided design; and maps.

1.2 Methods

Initial research and software development is being conducted in collaboration with David Gow[2]. Once a simple testbed application has been developed, we will individually explore approaches for introducing arbitrary levels of precision; these approaches will be implemented as alternate versions of the same software. The focus will be on drawing simple primitives (lines, polygons, circles). However, if time permits we will explore adding more complicated primitives (font glyphs, bezier curves, embedded bitmaps).

At this stage we have identified two possible areas for individual research:

1. **Arbitrary Precision real valued numbers** — Sam Moore

   We plan to investigate the representation of real values to a high or arbitary degree of precision. Such representations would allow for a document to be implemented using a single global coordinate system. However, we would expect a decrease in performance with increased complexity of the data structure used to represent a real value. Both software and hardware techniques will be explored. We will also consider the limitations imposed by performing calculations on the GPU or CPU.

Starting points for research in this area are Priest's 1991 paper, "Algorithms for Arbitrary Precision Floating Point Arithmetic"[9], and Goldberg's 1992 paper "The design of floating point data types"[8]. A more recent and comprehensive text book, "Handbook of Floating Point Arithmetic"[**?**], published in 2010, has also been identified as highly relevant.

2. **Local coordinate systems** — David Gow [2]

   An alternative approach involves segmenting the document into different regions using fixed precision floats to define primitives within each region. A quadtree or similar data structure could be employed to identify and render those regions currently visible in the document viewer. Say more here?

We aim to compare these and any additional implementations considered using the following metrics:

1. **Performance vs Number of Primitives**

   As it is clearly desirable to include more objects in a document, this is a natural metric for the usefulness of an implementation. We will compare the performance of rendering different implementations, using several "standard" test documents.

2. **Performance vs Visible Primitives**

   There will inevitably be an overhead to all primitives in the document, whether drawn or not. As the structure of the document format and rendering algorithms may be designed independently, we will repeat the above tests considering only the number of visible primitives.

3. **Performance vs Zoom Level**

   We will also consider the performance of rendering at zoom levels that include primitives on both small and large scales, since these are the cases under which floating point precision causes problems in the PostScript and PDF standards.

4. **Performance whilst translation and scaling**

   Whilst changing the view, it is ideal that the document be re-rendered as efficiently as possible, to avoid disorienting and confusing the user. We will therefore compare the speed of rendering as the standard documents are translated or scaled at a constant rate.

5. **Artifacts and Limitations on Precision**

   As we are unlikely to achieve truly "infinite" precision, qualitative comparisons of the accuracy of rendering under different implementations should be made.

1.3 Software and Hardware Requirements

Due to the relative immaturity and inconsistency of graphics drivers on mobile devices, our proof of concept will be developed for a conventional GNU/Linux desktop or laptop computer using OpenGL. However, the techniques explored could easily be extended to other platforms and libraries.

1.4 Timeline

Deadlines enforced by the faculty of Engineering Computing and Mathematics are *italicised*.[1].

Date	Milestone
17th April	Draft Literature Review completed. This sort of didn't happen...
1st May	Testbed Software (basic document format and viewer) completed and approaches for extending to allow infinite precision identified.
26th May	*Progress Report and Revised Literature Review due.*
9th June	Demonstrations of limitations of floating point precision in the Testbed software.
1st July	At least one implementation of infinite precision for basic primitives (lines, polygons, curves) completed. Other implementations, advanced features, and areas for more detailed research identified.
1st August	Experiments and comparison of various infinite precision implementations completed.
1st September	Advanced features implemented and tested, work underway on Final Report.
TBA	*Conference Abstract and Presentation due.*
10th October	*Draft of Final Report due.*
27th October	*Final Report due.*

[1]David Gow is being assessed under the 2014 rules for a BEng (Software) Final Year Project, whilst the author is being assessed under the 2014 rules for a BEng (Mechatronics) Final Year Project; deadlines and requirements as shown in Gow's proposal[2] may differ

Chapter 2

Literature Review

This chapter will review the literature. It will also include some figures created by us from our test programs to aid with conceptual understanding of the literature.

TODO: Decide exactly what figures to make, then go make them; so far I have some ideas for a few about Floating Point operations, but none about the other stuff.

TODO: Actually (re)write this entire chapter. ????: Do I really want to make this go down to `\subsubsection`

A paper by paper summary of the literature is also available at: ⟨http://szmoore.net/ipdf/documents/LiteratureNotes.pdf⟩.

TODO: Actually make that readable or just remove the link.

2.1 Document Formats

2.1.1 History

Since mankind climbed down from the trees... plagiarism alert!

2.1.2 Vector Graphics vs Raster Graphics

Raster Graphics: Stores the exact pixels as they would appear on a device. Causes obvious issues with scaling. Vector Graphics: Stores relative position of primitives - scales better. BUT still can't scale forever.

Figures: Raster and Vector graphics at different scales

2.1.3 Document Format Categories

Main reference: Pixels or Perish[6]

1. DOM - eg: HTML/XMLish - defined in terms of elements that can contain other elements
2. Programming Language - eg: PostScript - programmer (or program) produces a program that is interpreted
3. Combination - eg: Javascript with HTML/XML - Program is interpreted that modifies the DOM.
    - The lines are becomming increasingly blurred between 1. and 2.
    - In the case of Javascript/HTML, a special DOM element `<canvas>` allows even lower level manipulation of graphics.

Can be either human readable[1] or binary[2]

2.1.4 Modern Graphics Formats

PostScript: Not actually widely used now, but PDF is basically the same thing. PDF: A way for adobe to make money SVG: Based on the DOM model, vector format BMP,PNG,GIF: Are these even worth mentioning?

HTML: With the evolution of JavaScript and CSS, this has actually become a competitive document format; it can display much more complex dynamic content than eg: PDF.

Web based documents still suffer from precision issues (JavaScript is notorious for its representation of everything, even integers, as floats, so you get rounding errors even with integer maths), and it also has other limitations (requires reasonably skilled programmer to create Javascript and/or a disgusting GUI tool that auto generates 10000 lines to display a button (I have seen one of these)). Hate on javascript a bit less maybe.

2.1.5 Precision Limitations

2.2 Representation of Numbers

Although this project has been motivated by a desire for more flexible document formats, the fundamental source of limited precision in vector document formats such as PDF and

[1] For some definition of human and some definition of readable
[2] So, our viewer is basically a DOM style but stored in a binary format

PostScript is the use of floating point numbers to represent the coordinates of vertex positions. In particular, implementations of PostScript and PDF must by definition restrict themselves to IEEE binary32 "single precision" floating point number representations in order to conform to the standards[3, 5].

Whilst David Gow will be focusing on how to structure a document format so as to avoid or reduce these limitations[2], the focus of our own research will be NUMBERS.

2.2.1 Floating Point Number Representations

$$x = (-1)^s \times m \times B^e$$

$B = 2$, although IEEE also defines decimal representations for $B = 10$ — these are useful in financial software[10].

Aside: Are decimal representations for a document format eg: CAD also useful because you can then use metric coordinate systems?

Precision

The floats map an infinite set of real numbers onto a discrete set of representations.

Figure: 8 bit "minifloats" (all 255 of them) clearly showing the "precision vs range" issue

The most a result can be rounded in conversion to a floating point number is the units in last place; $m_N \times B^e$.

Even though that paper that claims double is the best you will ever need because the error can be as much as the size of a bacterium relative to the distance to the moon[] there are many cases where increased number of bits will not save you.[11]

2.2.2 Alternate Number Representations

They exist[11].

Chapter 3

Progress Report

This chapter outlines the current state of our research in relation to the aims outlined in Chapter 1. It will serve as an explanation for where the Figures in Chapter 2 came from. It will just be a short summary of the implementation details.

3.1 Development of Testbed Software

We wrote a very simple OpenGL 1.1 program to experiment with, and then David Gow converted it to OpenGL 3.1 and I have no idea how it works anymore.

3.2 Design and Implementation of "Tests"

- Compile by swapping out `main()` for a tester
- There are tests for doing some of the things in Chapter 1 but most still aren't written yet.

3.3 Floating Point Number Representations

I have[1] some figures that I would prefer to include in Chapter 2 when I am talking about the papers that inspired them. This section will probably briefly talk about how they were created and just refer back to them.

- `calculatepi.test`
- `typedef` of `Real`

[1] Ok... "will have"

3.4 Virtual FPU

Techniques for dealing with FP numbers can be implemented in software (CPU) or on dedicated hardware (FPU). We are able to run FP arithmetic on arbitrary simulations of FPUs created using VHDL. Hopefully explore this a bit in Chapter 2.

3.5 Version Control

Git is a distributed version control system widely used in the development of open source software[]. All rescources created for or used by this project have been placed in git repositories on several servers. The repositories are publically accessable at ⟨http://git.ucc.asn.au⟩

Chapter 4

Conclusion

This report has provided motivation for considering approaches to achieving an infinite level of zoom in a document.

4.1 Acheived Milestones

4.2 Areas of further Research

4.3 Witty Conclusion Goes Here

References

[1] Sam Moore. Infinite precision document formats (project proposal). ⟨http://szmoore.net/ipdf/documents/ProjectProposalSam.pdf⟩, 2014.

[2] David Gow. Infinite-precision document formats (project proposal). ⟨http://davidgow.net/stuff/ProjectProposal.pdf⟩, 2014.

[3] Adobe Systems Incorporated. *PostScript Language Reference*. Addison-Wesley Publishing Company, 3rd edition, 1985 - 1999.

[4] Michael A. Wan-Lee Cheng. Portable document format (pdf) – finally, a universal document exchange technology. *Journal of Technology Studies*, 28(1):59 – 63, 2002.

[5] Adobe Systems Incorporated. *PDF Reference*. Adobe Systems Incorporated, 6th edition, 2006.

[6] Brian Hayes. Pixels or perish. *American Scientist*, 100(2):106 – 111, 2012.

[7] David G. Barnes, Michail Vidiassov, Bernhard Ruthensteiner, Christopher J. Fluke, Michelle R. Quayle, and Colin R. McHenry. Embedding and publishing interactive, 3-dimensional, scientific figures in portable document format (pdf) files. *PLoS ONE*, 8(9):1 – 15, 2013.

[8] David Goldberg. The design of floating-point data types. *ACM Lett. Program. Lang. Syst.*, 1(2):138–151, June 1992.

[9] D.M. Priest. Algorithms for arbitrary precision floating point arithmetic. In *Computer Arithmetic, 1991. Proceedings., 10th IEEE Symposium on*, pages 132–143, Jun 1991.

[10] Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

[11] Jean-Michel Muller, Nicolas Brisebarre, Florent de Dinechin, Claude-Pierre Jeannerod, Vincent Lefèvre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, and Serge Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston Inc., Cambridge, MA, USA, 2010.