# Research Project Proposal

David Gow
20513684

March 12, 2014

---

| | |
|---|---|
| **Title:** | Infinite-precision document formats |
| **Supervisor** | A/Prof. Tim French |
| **Project Group:** | David Gow, Samuel Moore |

## 1 Background

Traditional document formats such as PDF and Postscript were designed for documents to be printed. They therefore are optimised to rasterize an entire page at a given resolution, often on the embedded processor found in (older) printers. This, however, does not fit well with interactive display on screens, particularly on mobile devices such as telephones and tablets.

In particular, when displaying a document on a screen, the user has several additional ways of interacting with the document, such as viewing a subset of the document in high resolution using the "zoom" option.

Existing vector file formats do support rasterization at different resolutions, but are limited by the precision of coordinates and other intermediate values. PostScript and PDF, for example, store many values in a *real* type with a limit of approximately 8 decimal digits of precision (PostScript[2]) or 5 decimal digits of precision (PDF[3]). These floating-point data types, while they provide both range (allowing for large documents) and precision (allowing for fine detail), cannot combine the two[**?**]. It follows that objects placed further from the origin must have less detail. Furthermore, these numerical datatypes have a limited range and therefore have an absolute limit on the size or precision of any document.

Documents with high — or at least consistent — levels of precision have many applications. The building industry uses tools such as Computer-Aided Drafting (CAD) and Building Information Modelling (BIM) systems for managing schematics which require precision. At the moment, this requires special tools, and it is not possible to export these as a single view without loss of precision. Similarly, infinite precision document formats would allow maps covering large areas to be stored contiguously (without requiring plates at different zoom levels) without any loss of precision.

## 2 Aim

We aim to prototype a simple document system which does not have these restrictions on zoom, and which can store data precisely at a given level of

zoom. While the primary goal is to ensure the objects within documents retain the precision at which they were created (typically being the resolution of the display used during document editing), an ideal approach would also ensure stability of the object when magnified beyond its original size.

Should this be successful, we will prototype different methods of implementing this system, using different data structures, to compare their relative merits in terms of — amongst other things — performance. In particular, we hope to avoid some of the performance issues resulting from traditional use of arbitrary-precision data types, particularly on the Graphics Processing Unit (GPU)[1].

As this is a group project each group member will develop and prototype a different method of retaining precision. For *this* individual part of the project, quadtrees will be investigated to provide a form of coordinate system renormalisation as the document view is scaled.

These systems will be compared in terms of their performance, with the aim to identify in which situations each implementation is most performant, consistent and correct.

# 3 Method

In order to make the most efficient use of our time, much of the early implementation work will be done as a collaboration with Samuel Moore. In particular, the work to implement the basic document system will be done jointly, which we will then use as a base to implement our (individual) data structures and algorithms for infinite-precision.

Initially, this document system will support documents consisting only of basic shapes (such as polygons, circles and Bézier curves), but, should time allow, this can be extended to include font glyphs, embedded raster images and other objects.

We will be comparing these methods primarily in terms of performance using a number of metrics:

- Performance per document object.

  As objects are the basis of these documents, we will test against a set of "standardised" documents with to determine how performance scales with the number of objects. We will test this with each of the different types (or shapes) of objects, as they may have different performance characteristics. Measured in $ms/object$

- Performance per visible object.

  As above, but measuring performance against the number of objects visible in a given view. For example, when zoomed in, fewer objects will be onscreen, and so — in some implementations — may not contribute as heavily to the computational load.

  Similarly measured in $ms/object$.

- Performance per zoom level.

  Time taken to render frames with the same number of visible objects will be compared at different zoom levels to determine if there is a relationship between performance and zoom. We will perform this test both with views

centred at the origin and centred at a random coordinate to see which, if any, implementations are affected by this. Measured in $ms/length$, where length is the length of one edge of the view in global document coordinates.

- Stability of performance under translation and scaling.

  We will measure performance during the process of a steady zoom or translation at a predetermined rate. This should allow us to identify "spikes" of low performance on implementations which require a calculation (such as renormalisation) periodically. Measured in $ms/frame$.

To accompany this, we will also be comparing the generated output from each implementation to find any discrepancies or artefacts introduced by the implementation, as well as to demonstrate the improvement in precision compared to the basic implementation. Should time permit, we would also like to demonstrate stability of the document under transformations (zooms and translations).

We intend to rasterize objects on the GPU in order to take advantage of the extra performance this provides. There is copious existing research on rendering gemetric shapes such as Bézier curves[5] on the GPU, and indeed entire implementations of the postscript rendering model[6][4]. We therefore believe that this will allow us to better focus on the performance of the infinite-precision document structures, whose calculations will likely largely reside on the Central Processing Unit (CPU).

# 4 Timeline

| **Date** | **Milestone** |
|---|---|
| 17th April | Draft Literature Review due. |
| 1st May | A simple document format should be designed and implemented, upon which to base further development and experiments. |
| 22nd May | Literature Review and Revised Proposal due. |
| 9th June | Demonstrated and documented artefacts and instability caused by floating point imprecision in the basic implementation. |
| 1st July | Have one or more algorithms/data structures for handling infinite-precision documents implemented. |
| 1st August | Initial performance measurements complete. Identified areas for further optimisation and experimentation. |
| 1st September | Adjustments to existing systems and experiments with additional techniques performed, and performance measured. Work underway on dissertation. |
| 18th September | Draft dissertation due. |
| 23rd October | Final dissertation due. |
| 27th – 31st October | Seminar presentation. |

# 5 Software & Hardware Requirements

We intend to develop this system on a common `x86` compatible Personal Computer running the `Linux`® operating system, using the `C++` programming language and making use of the `OpenGL` graphics library. However, the techniques we develop will likely also be applicable — or be easily extended — to other desktop and mobile systems. We also hope that the prototype developed will be portable to other systems.

# References


[1] Niall Emmart and Charles Weems. High precision integer multiplication with a graphics processing unit. In *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–6. IEEE, 2010.

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

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

[4] Mark J Kilgard and Jeff Bolz. Gpu-accelerated path rendering. *ACM Transactions on Graphics (TOG)*, 31(6):172, 2012.

[5] Charles Loop and Jim Blinn. Resolution independent curve rendering using programmable graphics hardware. *ACM Transactions on Graphics (TOG)*, 24(3):1000–1009, 2005.

[6] Peter Nilsson and David Reveman. Glitz: Hardware accelerated image compositing using OpenGL. In *USENIX Annual Technical Conference, FREENIX Track*, pages 29–40, 2004.