MetaPost 2 project goals

Abstract
Now that MetaPost 1.200 has been released the time has finally come to focus on the numerical precision extensions that we have been hinting at for some years already. Version 2.000 of MetaPost will have a runtime configurable precision and infinite numeric input range.

Introduction

A few years ago John Hobby transferred MetaPost development to a team of users of which Taco Hoekwater is now responsible for the coding. Some pending bugs were resolved and a few extensions made.

A major step was made when the code base was converted to C and MetaPost became a library component. This made usage in, for instance, luaTEX possible, and we could also get rid of some dependencies of external programs. This project was funded by TEX user groups and the results have been reported at user group meetings and in journals.

Recently an extra backend was added (SVG) which was also partially funded. The version that ships with TEX Live 2009 is the first version that is built on top of the library and it has these extensions on board.

More extensions

However, we are not yet finished as it has been a long standing wish to free MetaPost from one particularly significant limitation. In the original MetaPost library proposal we wrote in May 2007, one of the big user-side problem points mentioned was:

> "All number handling is based on fractions of a 32-bit integer and user input often hits one of the many boundaries that are the result of that. For instance, no numbers can be larger than 16384 and there is a noticeable lack of precision in the intersection point calculations."

It is for this reason that we will start the next stage in the development of the MetaPost library. The aim is to resolve the mentioned limitation once and for all.

In order to do so, we will have to replace the MetaPost internal 32-bit numeric values with something more useful, and to achieve that, the plan is to incorporate one or both of the following external libraries.

GNU MPFR library

From the web site:

> "The *MPFR* library is a C library for multiple-precision floating-point computations with correct rounding. MPFR has continuously been supported by the INRIA and the current main authors come from the CACAO and Arénaire project-teams at Loria (Nancy, France) and LIP (Lyon, France) respectively; see more on the credit page. MPFR is based on the GMP multiple-precision library.
>
> The main goal of MPFR is to provide a library for multiple-precision floating-point computation which is both efficient and has a well-defined semantics. It copies the good ideas from the ANSI/IEEE-754 standard for double-precision floating-point arithmetic (53-bit mantissa)."

See http://www.mpfr.org/ for more details.

IBM decNumber

From the web site:

> "decNumber is a high-performance decimal arithmetic library in ANSI C, especially suitable for commercial and human-oriented applications."

See http://www.alphaworks.ibm.com/tech/decnumber/ for more details.

We have not decided yet which one will be used, and possibly we will include both. MPFR will likely be faster and has a larger development base, but decNumber is more interesting from a user interface point of view because decimal calculus is generally more intuitive. For both libraries the same internal steps need to be taken, so that decision can be safely postponed until later in the project. The final decision will be based on a discussion to be held on the MetaPost mailing list.

The goals of the project

The project can be split up into a number of subsidiary goals.

Dynamic memory allocation

Because values in any numerical calculation library are always expressed as C pointers, it is necessary to move away from the current array-based memory structure with overloaded members to a system using dynamic allocation (using `malloc()`) and named structure components everywhere, so that all internal MetaPost values can be expressed as C pointers internally.

As a bonus, this will remove the last bits of static allocation code from MetaPost so that it will finally be able to use all of the available RAM.

Internal API

An internal application programming interface layer will need to be added for all the internal calculation functions and the numeric parsing and serialization routines. All such functions will have to be stored in an array of function pointers, thus allowing a run-time switch between 32-bit backward-compatible calculation and an arbitrary precision library.

This internal interface will also make it possible to add additional numerical engines in the future.

The external application programming interface layer will be extended to make direct access to the numerical and path handling functions possible.

Backends

The SVG and PostScript backends need to be updated to use double precision float values for exported points instead of the current 32-bit scaled integers.

In the picture export API, doubles are considered to be the best common denominator because there is an acceptable range and precision and they are simple to manipulate in all C code. This way, the actual SVG and PostScript backend implementations and the Lua bindings can remain small and simple.

Input language

The input language needs to be extended in order to fulfil the following project objectives.

1. It must be possible to select which numerical engine to use.
2. In the case of an arbitrary precision engine, it has to be possible to set up the actual precision.
3. It has to be possible to read (and write) numerical values in scientific notation.
4. Some mathematical functions (especially trigonometry) and constants (like π) will have to be added to make sure all the numerical engines present a unified interface while still offering the best possible precision.

Here is a tentative input example that would select the decNumber library with 50 decimal digits of precision:

```
mathengine := "decNumber";
mathprecision := 50;
beginfig(1);
z1 = (1/3,0.22153333455654453226632 2);
v = 23.45678888E512;
x2 = v/1E511;
endfig;
end.
```

Timeline

Thanks to funding received from various local user groups, we hope to be able to have MetaPost 2 ready in time for the TUG and EuroTEX conferences in 2010.

Hans Hagen & Taco Hoekwater
taco (at) metapost dot org