Metapost improvement proposal

May 18, 2009

In the original Metapost library proposal we wrote in May 2007, one of the big user-side problem points that was mentioned was this:

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

The current proposal aims to resolve that issue once and for all. The goal is to replace the Metapost internal 32bit numeric values with something more useful, and to achieve that goal the plan is to incorporate one of these libraries:

GNU MPFR library	`http://www.mpfr.org/`
IBM decNumber	`http://www.alphaworks.ibm.com/tech/decnumber`

We have not decided yet which one. 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 a little later in the project. The final decision will be based on a discussion to be held on the Metapost mailing list.
There are a number of subsidiary goals in this project, as follows:

1. 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.

2. 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 the arbitrary precision library.

As a bonus, this will make it possible to add even more different numerical engines in the future.

3. 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.

4. The input language needs to be extended in a few ways:

   1. it has to 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 pi) 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 an estimate of the needed development work, expressed in hours:

1	Dynamic memory management	200
2	Numerical engine API	400
3	Backend updates	50
4	Metapost language improvements	150
5	Documentation and testing	200
Total	1000	

This converts into approximately six months of continuous work.
A small part of the work can be done in private time and by other people, but a considerable amount of time will have to be spent by Taco Hoekwater in a commercial setting, there is just too much work to be done. Therefore, on behalf of Taco, we seek funding for 75% of the amount of programming work involved at a low but financially just viable rate, resulting in a total funding need of 15.000 Euro for getting the complete project done in time for the TUG/EuroTEX conferences in 2010.