Supporting color and graphics in expl3

Joseph Wright

1 Introduction

The expl3 language has grown over the past decade to cover a wide range of programming tasks [4]. However, at present there are a number of areas where expl3 offers little or no 'core' support and which will need functionality at this level. Here, I'll be focussing on one in particular: color and graphics support.

In the classical LaTeX 2ε setup, the `picture` environment along with the packages graphics [5] and color provide the basis for this area. To allow driver-dependent operations, a set of definition files is loaded by graphics to map user operations to driver-specific instructions. Nowadays, these are managed by the LaTeX team in the bundle graphics-def [2].

In addition to this core support, a number of well-established contributed packages offer significant additional features. Particularly notable here are xcolor [1], which allows user-friendly mixing of colors, and Ti*k*Z/pgf [6], an extremely rich and versatile system for the programmatic creation of graphics.

Here, I will look at recent efforts to begin providing a similar level of overall functionality *via* expl3. Central to these efforts is the availability of a fast, expandable and accurate software floating-point unit (FPU) within expl3. This provides a base on which many graphics-related functions can build: calculations are a core part of many image-related functions.

2 The driver layer

Unlike the LaTeX 2ε situation, where the graphics and color driver code is managed (somewhat) separately from the kernel, the expl3 versions are part of the core distribution. Development of the driver code in expl3 has been informed by recent efforts to standardise the LaTeX 2ε versions, and *vice versa*.

As new features are added to expl3 which require driver support, the driver layer is being adjusted to match. This means that unlike in LaTeX 2ε, for expl3 there should be a single set of definitive driver files, supported by the team and usable by (and documented for) others.

3 Colo(u)r

The LaTeX 2ε (required) package color provides a base interface for using pre-defined colors. However, one of the most common ways to use a color is to describe it as a mix of base colors: red, green and blue, or cyan, magenta, yellow and black. The xcolor package provides a convenient 'expression' interface for creating mixtures: `\color{red!50!blue}`.

Supporting this mixing, conversion between different color models, and other features such as spot colors, are all (largely) covered in the experimental l3color package [3]. Using the LaTeX3 FPU makes much of the core support very easy to implement: the various pieces of mathematics can be expressed directly, rather than requiring complex dimension shuffling.

At present, the nature of input in l3color is limited to the 'simple' color expressions defined by xcolor: feedback on what is helpful to end users would be very welcome.

4 Image inclusion

At present, expl3 support for image inclusion is only ready at the driver level. Implementing a code-level set of `\image_...` functions is on the 'to do' list, and is likely straightforward.

5 Drawing

Whilst the `picture` environment of the LaTeX kernel does provide a way to create simple graphical elements, today perhaps the most powerful tool for this task is Ti*k*Z/pgf. Reimplementing all of the latter may seem excessive, but there are several reasons to explore this. First, a core aim of expl3/LaTeX3 work is to eventually provide a full set of features for supporting document preparation, certainly providing code-level tools for all common tasks. Coupled to this, an expl3 implementation will have API consistency with the rest of the code: mixing Ti*k*Z and expl3 can be tricky. We are also able to use existing expl3 tools in the implementation and usage. Finally, there is the potential offered by the LaTeX3 FPU: this avoids using dimensions for floating point work, and so also avoids the `Dimension too big` issue that comes up from time-to-time using Ti*k*Z.

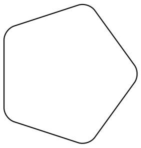
Much like expl3, pgf is divided into different layers: these line up as show in Table 1. There is good alignment, and thus in many ways it is simply a case of re-creating the macros with new names. Of course, there is more to do than that: for example, the use of the LaTeX3 FPU means that co-ordinate expressions are processed expandably by l3draw, with a knock-on effect in usage. However, as far as possible the interfaces in l3draw retain the same arguments as those in pgf.

6 Examples

At the time of writing, l3draw is very much a work in progress. However, the core idea of constructing paths is fully implemented. For example, a simple geometric shape including smoothing joins:

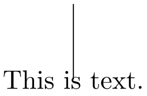
Table 1: Comparison of Ti*k*Z/pgf and l3draw concepts

Layer	Ti*k*Z/pgf	l3draw
System	`\pgfsys@moveto`	`\driver_draw_moveto:nn`
Base	`\pgfpathmoveto`	`\draw_path_moveto:n`
Interface	`\draw`	—



```
\draw_begin:
  \draw_path_corner_arc:nn { 4pt } { 4pt }
  \draw_path_moveto:n
    { \draw_point_polar:nn { 0 } { 1cm } }
  \int_step_inline:nnnn { 72 } { 72 } { 359 }
    {
      \draw_path_lineto:n
        {
          \draw_point_polar:nn { #1 } { 1cm }
        }
    }
  \draw_path_close:
  \draw_path_use_clear:n { stroke }
\draw_end:
```

The new code also integrates with existing ideas such as coffins. Here, we draw a line to the center of typeset text:



```
\draw_begin:
  \draw_path_moveto:n { 0cm , 0cm }
  \draw_path_lineto:n { 0cm , 1cm }
  \draw_path_use_clear:n { stroke }
  \hcoffin_set:Nn \l_tmpa_coffin
    { This~is~text. }
  \draw_coffin_use:Nnn \l_tmpa_coffin
    { hc } { vc }
\draw_end:
```

We can also exploit the expandable nature of the FPU:

```
22.72949518869545pt,-17.11517943480897pt
```

```
\tl_set:Nx \l_tmpa_tl
  {
    \draw_point_intersect_circles:nnnnn
      { (0,0) } { 1cm }
      { (sqrt(2),sqrt(3)) } { 1cm }
      { 1 }
  }
\tl_to_str:N \l_tmpa_tl
```

Thus, l3draw is ready for application in expl3 contexts which require drawing. Over time, we expect to cover essentially the entire API provided by pgf's core, plus probably node handling (loaded by pgf but not technically part of the core of the bundle).

References

[1] U. Kern. Extending LaTeX's color facilities: the xcolor package, 2016. `ctan.org/pkg/xcolor`

[2] LaTeX Project. Color and graphics option files, 2018. `ctan.org/pkg/graphics-def`

[3] LaTeX Project. Experimental LaTeX3 concepts, 2018. `ctan.org/pkg/l3experimental`

[4] LaTeX Project. The expl3 package and LaTeX3 programming, 2018. `ctan.org/pkg/expl3`

[5] LaTeX Project. The graphics bundle, 2018. `ctan.org/pkg/graphics`

[6] T. Tantau and C. Feuersänger. Ti*k*Z and pgf, 2015. `ctan.org/pkg/pgf`

◇ Joseph Wright
LaTeX Project
joseph dot wright (at)
morningstar2.co.uk