asypictureB — user-friendly integration of Asymptote into LaTeX*

Charles Staats III†

Released 2025/01/04

Abstract

The asypictureB package allows users to integrate Asymptote code for producing pictures into LaTeX source code using the shell-escape functionality. It is an alternative to the asymptote package that comes with Asymptote. The most important advantage of the asypictureB package is that it provides immediate access to Asymptote errors by repackaging them as LaTeX errors. It also allows limited use of TeX macros such as `\the\linewidth` inside Asymptote code.

Contents

---

*This document corresponds to asypictureB v0.4, dated 2025/01/04.

†E-mail: charles dot staats dot iii at gmail.com

1 Introduction

The Asymptote programming language[1] is a powerful tool for the creation of diagrams, both two- and three-dimensional, that are compatible with TeX documents. The programming language ships with a LaTeX package called (appropriately but confusingly) asymptote that makes it easy to draw pictures by including Asymptote code in the LaTeX source file. The asymptote package also allows the inclusion of interactive three-dimensional images in pdf files.[2]

Unfortunately, the author has encountered the several annoyances using the asymptote package.

1. When my Asymptote code contains errors that prevent it from compiling, I have found it extremely difficult to track down the offending line in the TeX source file. This has been the single biggest annoyance and has led me to compose all but the simplest Asymptote images as separate .asy files, which are then included into the TeX source once they do what I want.

2. The asymptote package does not support PNG files, although the Asymptote language does.[3]

3. The asymptote package does have a mechanism in place so that when it is used with latexmk, Asymptote images that have not changed are not recompiled. Unfortunately, the mechanism for recognizing unchanged images is somewhat fragile: if a single image is inserted or deleted, then all subsequent images will have to be recompiled. In my experience, this can make a compilation last several minutes that would otherwise have lasted several seconds.

4. The asymptote package can rescale an Asymptote-produced image to a given width and/or a given height, but it cannot rescale the image by a given scaling factor.[4]

The most important issue here is the first, which can to some extent be fixed using editor features; see, for instance, this explanation for TeXnicCenter (Windows only)[5].

The asypictureB package is an alternative to the asymptote package that (optionally) uses \write18 to call the Asymptote compiler directly from LaTeX. The package provides some sort of fix for all the above issues:

1. Asymptote errors are repackaged as LaTeX errors and reported immediately. At a minimum, the user is shown the Asymptote error log and a line number that will allow him to locate the correct Asymptote picture within the TeX source file.

---

[1] http://asymptote.sourceforge.net

[2] The asypictureB package does not currently support this feature.

[3] A note on why this is desirable: Asymptote's 3d capabilities are currently much better for rasterized images than for vector graphics. Thus, it is often desirable to produce a rasterized image. Because of a quirk of Asymptote, it is easy to produce high-resolution images in PNG format, but much harder to raise the resolution when the output is in the default PDF format.

[4] This is relevant because a second trick for producing a high-resolution rasterized image with the Asymptote language is to produce a scaled-up PDF image, and then scale it back down when including it into the TeX file.

[5] http://www.artofproblemsolving.com/Wiki/index.php/Asymptote:
_Advanced_Configuration#Showing_Asymptote_error_messages_in_TeXnicCenter

Additionally, asypictureB is usually able to display the five lines of Asymptote code up to and including the first error. This is useful to locate the line of code on which the error occurs, since the line numbers in the Asymptote error log do not correspond to the line numbers in the LaTeX source file.

2. Any file type that is supported by both the Asymptote language and the `\includegraphics` command is supported by asypictureB. Assuming that the document is compiled using `pdflatex`, this includes the PNG, PDF, and (more or less) EPS file formats.

3. Users are permitted and strongly encouraged to specify a distinct name for each Asymptote image in the file. Distinctly named Asymptote images are not recompiled if their code has not changed since the last `latex` run.

4. Any option that works for the `\includegraphics` command can be given as an option to an `asypicture` environment.

There is one other feature worth mentioning: LaTeX macros will be expanded inside Asymptote code if they are prefixed by @ instead of \. This can be used as an incomplete substitute for the `inline` option of the asymptote package. It also provides a way to use macros in Asymptote code, which is not a feature that Asymptote supports otherwise.

Each of these features could stand significant improvements. However, since implementing them, the author has found himself much more willing to compose Asymptote code directly in a TeX source file. This indicates to him that the package might prove useful to others, even in its current form. His hope is that the best ideas of this package would be copied and improved in the official asymptote package, allowing him to deprecate asypictureB.

2 Installation and running

First of all, the asypictureB package is mostly useless unless you have Asymptote installed on your system.

- For a Mac OS X system, this installation is automatic with a standard installation of MacTeX.

- For a Windows system, the official installation instructions are fairly good. As of this writing, the most recent version of the `setup.exe` file can be downloaded from http://sourceforge.net/projects/asymptote/files/2.24/.

- For a Unix-like system, a version of Asymptote is included in TeX Live, but there may be additional dependencies; see, for instance, http://tex.stackexchange.com/a/155284/484. You should also consult these two pages from the official documentation.

To use the asypictureB package, the `.tex` file should be run with shell-escape enabled:

```
pdflatex -shell-escape ⟨filename⟩
```

[Note that `latex`, `lualatex`, etc. can be substituted for `pdflatex`, although you should make sure that the engine you use is compatible with whatever graphics formats your Asymptote pictures are compiled to.]

2.1 Running without shell escape

If you are unwilling to use shell-escape, asypictureB creates a script that makes it easy to execute the necessary commands afterwards. To use it, run the following three commands at the terminal. For convenience, it is assumed that the name of the LaTeX source file is `foo.tex`.

Windows: Mac OS X and Unix-like systems:

```
pdflatex foo                                  pdflatex foo
foo-asy_compile.bat                           sh foo-asy_compile.sh
pdflatex foo                                  pdflatex foo
```

Instead of `pdflatex`, one may use `latex`, `lualatex`, ... Asymptote errors will be visible as TeX errors on the second run of LaTeX.

⚠ **Warning:** This method is not entirely foolproof. In particular, if `pdflatex` is run twice in a row without running the ⟨*filename*⟩-`asy_compile` script in between, then Asymptote pictures which have been compiled at some point in the past, even if they have since been altered, will not be recompiled. Should this happen, you can force every `asypicture` to be recompiled using the `\RequireAsyRecompile` command.

The safer method If you want to compile a TeX source file from someone else (e.g., the internet), you may want neither to enable shell-escape nor to run a script generated by this file. A safer way to compile all Asymptote pictures generated by asypictureB (and for that matter by the asymptote package) is to run `asy -noV` ⟨*filename*⟩-`*.asy` after compiling ⟨*filename*⟩`.tex`. Once this is done, all the Asymptote files should be compiled, will be correctly imported upon a second run of `pdflatex` (or `latex`, ...). Re-running `asy` is not necessary until and unless any of the Asymptote pictures change. This method is "safe" in that you have greater control over which programs are actually being run.

This method will also work for your own files-in-progress, but is not recommended for two reasons:

- Asymptote errors will not be repackaged as TeX errors, negating one of the main features of the asypictureB package.

- All Asymptote pictures will be recompiled, even if they have not changed since the last run. This can add considerably to the compile time.

2.2 Dependencies

As currently implemented, the asypictureB package requires the packages fancyvrb (tested with version 2.8), graphicx (tested with version 2005/11/14), pgfkeys, and ifplatform (tested with version 0.4). In fact, it depends on undocumented internals of the fancyvrb package, so later versions of this package could conceivably break it as well as earlier versions.

If shell-escape is not enabled, it also requires the verbatimcopy package, version 0.06 or later. Since verbatimcopy is not backwards compatible, earlier versions will, in fact, break asypictureB.

3   Usage

asypicture (*env.*) Code for Asymptote pictures should be placed within the `asypicture` environment, which takes one mandatory argument: a list of comma-separated expressions of the form ⟨*key*⟩=⟨*value*⟩. It is strongly recommended that the key `name` always be used, since this will prevent the package from re-compiling pictures whose code has not changed. The time saving can be substantial if you have a document with a significant number of Asymptote pictures.

Keys other than `name` are passed onto an `\includegraphics` command from the `graphicx` package. The keys should be given in the following order:

1. Keys other than `scale` or `angle`.

2. The `scale` key (if it is used).

3. The `angle` key (if it is used).

The way the package is currently implemented, the `scale` and `angle` keys will effectively be evaluated last, regardless of the order in which they are given. However, this behavior is not ideal and may change in future versions of the package; the goal would be that keys should be evaluated in the order in which they are given. For the time being, giving keys in the order specified should ensure compatibility with future versions.

It is possible to expand macros within an `asypicture` environment by using an at symbol `@` in place of a backslash `\`. For instance, the line

```
size(@the@linewidth, 0);
```

will be translated to something like

```
size(345.0pt, 0);
```

before being compiled by Asymptote.

asyheader (*env.*) Code within an `asyheader` environment is appended to the "header," which is inserted at the beginning of every `.asy` file output by the asypictureB package. Initially, the header consists of the two lines

```
defaultpen(fontsize(@getfontsize pt));
settings.prc = false;
```

Again, macros can be expanded inside an `asyheader` environment by prefixing them by @ rather than `\`. The macros are fully expanded when they are added to the header, not when they are written to a file. Thus, for instance, in a 10-point document, the first line of the header will always read

```
defaultpen(fontsize(10pt));
```

even if the font size is later changed to 12 points.

Note that any change to the header will require all subsequent `asypicture` environments to be recompiled. In a document with many Asympotote images, this could take a while.

\getfontsize The `\getfontsize` macro is an alias for `\f@size` that does not require

`\makeatletter`. It expands to the current font size (without the suffix `pt`).

`\asylistingfile` The `\asylistingfile` macro contains the filename of the Asymptote code for the most recent `asypicture`. It can be used with commands from e.g. `fancyvrb` or `listings` to display the Asymptote code for an image.

`\RequireAsyRecompile`
`\AsyCompileIfNecessary` By default, Asymptote images are compiled only if the Asymptote code has changed. The command `\RequireAsyRecompile` changes this setting to make all subsequent images recompile even if the code has not changed. (This can be useful, for instance, if you have just updated Asymptote.) The command `\AsyCompileIfNecessary` restores the default behavior for all subsequent Asymptote pictures.

3.1 Keys for `asypicture`

`name` If the key-value combination `name=`⟨*picturename*⟩ appears in the mandatory argument to an `asypicture` environment, then the contents of that environment will be saved to the file

⟨*filename*⟩-⟨*picturename*⟩`.asy`

where ⟨*filename*⟩ is the name of the current `.tex` file (not including the `.tex` extension). The compiled Asymptote picture is saved to an image file such as ⟨*filename*⟩-⟨*picturename*⟩`.eps`, ⟨*filename*⟩-⟨*picturename*⟩`.pdf`, or ⟨*filename*⟩-⟨*picturename*⟩`.png`.

Alphanumeric characters are allowed in the `name` key, as are the characters `-` (hyphen) and `_` (underscore). Other characters (including spaces, asterisks, etc.) should be avoided. The key may begin with any allowed character; however, names consisting of a single number with three or fewer digits are discouraged, since these may conflict with the `asymptote` package.

Although the `name` key is technically not required, it is strongly recommended that distinct names be given to all the different `asypicture` environments. This allows the asypictureB package to avoid recompiling pictures whose code has not changed—even if the pictures are reordered.

If the name ⟨*picturename*⟩ has been used before, a suffix will be appended to it: `__1` for the first repeat, `__2` for the second repeat, etc. Thus, if the same name (say `foo`) is used for several `asypicture`s, then deleting the first of these will force all the others to recompile; but changing a picture of a different name will not affect any of the pictures named `foo`.

If no `name` key is given, the key-value combination `name=noname` is assumed. This default can be changed using the `\asyset` command: after the line

`\asyset{name = foo}`

nameless pictures will be called `foo` rather than `noname`.

⚠ **Warning:** If asypictureB determines that the contents of an `asypicture` environment named ⟨*picturename*⟩ have changed since the last run, it will delete all of the following files that exist:

⟨*filename*⟩-⟨*picturename*⟩`.eps`
⟨*filename*⟩-⟨*picturename*⟩`.pdf`
⟨*filename*⟩-⟨*picturename*⟩`.png`

3.2 Styles

\asyset Since asypictureB uses pgfkeys for keys, it is possible to create new keys, called "styles," that set several other keys at once. For instance, the following line

```
\asyset{mysize/.style = {width=6cm, height=4cm}}
```

creates a new key called mysize. Any time in the document after this line, the command

```
\begin{asypicture}{name=picturename, mysize}
```

is exactly equivalent to the line

```
\begin{asypicture}{name=picturename, width=6cm, height=4cm}
```

This is precisely the same style mechanism that is used in TikZ; in fact, \asyset{⟨keys⟩} is equivalent to \pgfkeys{/asy/.cd,⟨keys⟩}, much as \tikzset{⟨keys⟩} is equivalent to \pgfkeys{/tikz/.cd,⟨keys⟩}.

The \asyset command may be used in the preamble or anywhere in the body of the document where macros are processed normally. It may not be used in the body of an asypicture or any other verbatim-like environment.

4 Examples

Here is a simple example:


```
\begin{asypicture}{name=sphere_image}
settings.outformat = "png";
settings.render = 16;
size(2.5cm,0);
import three;
draw(unitsphere, white);
\end{asypicture}
```

Here is an example that uses the user-defined macro \asywidth inside the Asymptote code to avoid duplicate code.

```
\noindent\def\asywidth{4cm}
\begin{asypicture}{name=triangle,width=\asywidth}
    settings.outformat="pdf";
    size(@asywidth,0);
    draw((0,0) -- (1,0) -- (1,1) -- cycle);
    dot((1,0), L=Label(
        "\textbf{Bold} \textsc{Smallcaps}",
        align=NE));
\end{asypicture}
\hfill
\begin{minipage}{\dimexpr\linewidth-\asywidth-15pt\relax}
\VerbatimInput[frame=leftline]{\asylistingfile}
\end{minipage}
```

The result:

```
defaultpen(fontsize(10 pt));
settings.prc = false;


    settings.outformat="pdf";
    size(4cm,0);
    draw((0,0) -- (1,0) -- (1,1) -- cycle);
    dot((1,0), L=Label(
        "\textbf{Bold} \textsc{Smallcaps}",
        align=NE));
```

The `\VerbatimInput` command is from the fancyvrb package. The first two lines of the Asymptote code listing come from the default `asyheader`. These, along with some extra whitespace, can be eliminated by playing with the fancyvrb options. More importantly for our purposes, changing the single line

`\noindent\def\asywidth{4cm}`

to

`\noindent\def\asywidth{4.5cm}`

will automatically affect both the width of the `asypicture` and the width of the `minipage`:

```
\noindent\def\asywidth{4.5cm}
\begin{asypicture}{name=triangle2,width=\asywidth}
    settings.outformat="pdf";
    size(@asywidth,0);
    draw((0,0) -- (1,0) -- (1,1) -- cycle);
    dot((1,0), L=Label(
        "\textbf{Bold} \textsc{Smallcaps}",
        align=NE));
\end{asypicture}
\hfill
\begin{minipage}{\dimexpr\linewidth-\asywidth-15pt\relax}
\VerbatimInput[frame=leftline,gobble=4,firstline=5]
    {\asylistingfile}
\end{minipage}
```

produces
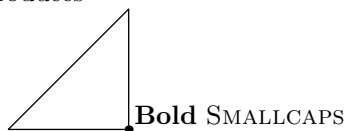
```
settings.outformat="pdf";
size(4.5cm,0);
draw((0,0) -- (1,0) -- (1,1) -- cycle);
dot((1,0), L=Label(
    "\textbf{Bold} \textsc{Smallcaps}",
    align=NE));
```

5 Error handling

It has been stated several places in this document that asypictureB repackages
Asymptote errors as TeX errors (and at least once that this feature, among others,
could stand significant improvements). In this section is described exactly how
these errors are repackaged, at least as of the current implementation. None of
this is guaranteed to remain the same in future versions.

When asypictureB tells Asymptote to compile the file

$$\langle\textit{filename}\rangle\texttt{-}\langle\textit{picturename}\rangle\texttt{.asy,}$$

it reroutes all warning and error messages to the file

$$\langle\textit{filename}\rangle\texttt{-}\langle\textit{picturename}\rangle\texttt{\_errors.txt.}$$

Once the Asymptote run is complete, it checks whether the error file has any lines
of the form

$$\langle\textit{filename}\rangle\texttt{-}\langle\textit{picturename}\rangle\texttt{.asy:}\langle\textit{number}\rangle\texttt{.}\langle\textit{stuff}\rangle$$

If so, it throws a LaTeX package error and displays the contents of the error file,
which include all the errors and warnings issued by Asymptote. Note that if some
kind of asy error occurs that does not match this output form, asypictureB will not
notice the error.

The LaTeX error message will give the line number of the `\end{asypicture}`
command for the `asypicture` that caused the error, which is of limited use in
isolating the error. More precise line numbers are given by the Asymptote error
log; however, these line numbers are for the `asy` file rather than the `tex` file, and
consequently not terribly helpful.

To allow the user to locate the line on which the actual error occurred, the
asypictureB package attempts to parse the Asymptote error log and print out the
five lines leading up to the error. More precisely, it does the following:

1. From the first line of the form

   $$\langle\textit{filename}\rangle\texttt{-}\langle\textit{picturename}\rangle\texttt{.asy:}\langle\textit{number}\rangle\texttt{.}\langle\textit{stuff}\rangle$$

   the $\langle\textit{number}\rangle$ is extracted. Note that if no successful extraction occurs, there
   will be no LaTeX error.

2. Assuming $\langle\textit{number}\rangle$ was extracted successfully, the lines of the `asy` file from
   the inclusive range $\langle\textit{number}\rangle - 5$ to $\langle\textit{number}\rangle$ are displayed as part of the
   LaTeX error message. These lines are usually identical and almost always
   similar to the lines in the actual `tex` file leading up to the error. In the
   author's experience, this is usually enough information to locate without
   difficulty the line on which the error occurred.[6]

Here's an example: Consider LaTeX file

---

[6]I.e., the line that caused Asymptote to choke. The usual rules of debugging apply: the line
on which the compiler identified an error might have been correct if not for an earlier mistake
that was syntactically correct.

```
1  \documentclass{article}
2  \usepackage{asypictureB}
3   \begin{document}
4  \def\asywidth{5cm}
5  \begin{asypicture}{name=error_example}
6      // A comment
7      size(@asywidth, 0);
8      path l1 = (0,0) -- (1,1);
9      // Another comment
10     draw(box((0,0),(1,1)))
11     draw(l1, dotted);
12     draw(l2, dashed);
13 \end{asypicture}
14 \end{document}
```

If this file is saved as `asyerrorexample.tex` and then compiled with the shell-escape option, the following error results:

```
! Package asypictureB Error:
draw(l1, dotted);
^
asyerrorexample-error_example.asy: 10.5: syntax error
error: could not load module 'asyerrorexample-error_example.asy'

6      size(5cm, 0);
7      path l1 = (0,0) -- (1,1);
8      // Another comment
9      draw(box((0,0),(1,1)))
10      draw(l1, dotted);
.


See the asypictureB package documentation for explanation.
Type  H <return>  for immediate help.
      ...

l.13 \end{asypicture}
```

The LATEX error message first repeats the Asymptote error log, which explains that there was a syntax error. Next, the five lines leading up to the error are displayed. Looking at these five lines, one can determine that the difficulty was a comma omitted on line 9 of the Asympote file. (It is diagnosed on line 10 because of how the compiler works.) Looking at the context, this corresponds to line 10 of the LATEX file.

Note that the macro `@asywidth` in the `asypicture` code has been expanded to `5cm` in the error message.

6   Macros in `asypictures`

When the author first conceived of allowing macros in an `asypicture` environment, the goal was to allow expressions like `size(@asywidth,0);` where `\asywidth` was

a user-defined macro also used in the LaTeX code to avoid hard-coding numbers. However, it was almost immediately apparent that this feature has more sophisticated uses, such as allowing user-defined syntax or even creating something similar to templates.

6.1 Limitations

Before discussing the nifty features, let's discuss the fairly severe limitations they will have to work around.

No grouping symbols. If a macro takes a mandatory argument inside braces `{}`, then that macro cannot be used inside an `asypicture`.

Purely expandable macros only. Macros will be expanded, but not executed. In particular, macros that allow optional arguments will usually go horribly wrong.

One line only. As we will discuss, it is possible to use "unconventionally delimited" arguments. However, even in this case, a macro and all its arguments must fit on a single line.

Since the first two points all but forbid the use of macros with conventional arguments, it might be wondered whether macros in Asymptote code can be used for anything more interesting than storing user-defined lengths. They can.

6.2 Single-token arguments

[This is really more a fix than an example, but this seems as good a place as any to discuss it.]

Some macros, such as `\the`, take arguments that consist only of a single character or control sequence. For instance, `\the\textwidth` expands to something like `345.0pt`, whereas `\textwidth` by itself expands only to `\textwidth`. This can be significant if you want to produce an Asymptote picture that takes up a specified fraction of the text width.

6.3 Unconventionally delimited arguments

The TeX primitive `\def` can be used to produce macros that are quite flexible about how they are delimited. For instance, the TeX code

```
\def\draw#1;{draw(#1);}
```

will take as its argument everything between the `\draw` command and the first subsequent semicolon. If this line showed up in TeX code (preferably in the preamble), then subsequent `\asypicture` environments could include a line such as

```
@draw box((0,0), (1,1));
```

as an arguably more aesthetic alternative to the translation

```
draw( box((0,0), (1,1)));
```

A more advanced example is essentially a template for a sorting function[7]. Include the following code in the TeX preamble:

```
\def\definesortfunction #1;{%
#1[] sort(#1[]  a) {
    if (a.length <= 1) return a;
    static #1[] merge(#1[] b, #1[] c) {
        #1[] toreturn;
        int i = 0, j = 0;
        while (i < b.length && j < c.length) {
            if (!(c[j] < b[i])) { toreturn.push(b[i]); ++i; }
            else { toreturn.push(c[j]); ++j; }
        }
        while (i < b.length) {
            toreturn.push(b[i]);
            ++i;
        }
        while (j < c.length) {
            toreturn.push(c[j]);
            ++j;
        }
        return toreturn;
    }
    int halfway = floor(a.length / 2);
    #1[] b = sort(a[0:halfway]);
    #1[] c = sort(a[halfway:a.length]);


    return merge(b, c);
}}
```

Then within any `asypicture` environment, the line `@definesortfunction T;` can be used to define a function `T[] sort(T[])` that returns a sorted version of its argument, for any type `T` for which the less than operator `<` is defined. For instance, within an `asypicture`, the code

```
bool operator <(pair a, pair b) {
   return (a.x < b.x  || (a.x == b.x && a.y < b.y));
}
```

```
@definesortfunction pair;
```

makes available a lexicographic sorting routine for ordered pairs of real numbers.

Unfortunately, while this compiles correctly, the resulting Asymptote file has no line breaks in the entire definition of the sort function. If you want the Asymptote code (as opposed to just the `asypicture` code) to be readable, the following setup, proposed by Enrico Gregorio[8], does the job:

```
\begingroup
\endlinechar=`^^J \obeyspaces% end of lines are newlines
```

---

[7]The algorithm here is a somewhat inefficient mergesort.
[8]http://tex.stackexchange.com/a/160740/484

```
\gdef\definesortfunction #1;{% eat up the space following the macro
#1[] sort(#1[]  a) {
    if (a.length <= 1) return a;
    static #1[] merge(#1[] b, #1[] c) {
        #1[] toreturn;
        int i = 0, j = 0;
        while (i < b.length && j < c.length) {
            if (!(c[j] < b[i])) { toreturn.push(b[i]); ++i; }
            else { toreturn.push(c[j]); ++j; }
        }
        while (i < b.length) {
            toreturn.push(b[i]);
            ++i;
        }
        while (j < c.length) {
            toreturn.push(c[j]);
            ++j;
        }
        return toreturn;
    }
    int halfway = floor(a.length / 2);
    #1[] b = sort(a[0:halfway]);
    #1[] c = sort(a[halfway:a.length]);


    return merge(b, c);
}% this % is necessary
}% this % is necessary
\endgroup% this % is necessary
```

⚠ **Warning:** If two commands use the same unconventional delimiter, then they cannot be nested. In particular, a command cannot appear inside its own argument. This is one of the reasons people usually delimit with grouping symbols, which is not an option here, short of using `@bgroup` and `@egroup` with copious occurrences of `@expandafter`.

⚠ **Warning:** The `\def` command is not expandable. Thus, the code defining the macros must be given *outside* `asypicture` environments, even when the macros are intended for use exclusively inside `asypicture`s.

7   Missing features

The introduction of this documentation touted ways in which asypictureB improves on the behavior of the asymptote package. In this section, I clarify ways in which it falls short and describe how to compensate where possible. My hope is that these differences would narrow in both directions until only one package is necessary—preferably the official asymptote package.

Note that some of these "missing features" cannot be fixed without breaking backwards compatibility. This is largely why the package is named asypictureB,

allowing room for a future, more powerful version named asypicture, in case the asymptote package does not step into the gap.

> **Tip:** To retrieve the documentation for the asymptote LaTeX package, type `texdoc asy-latex` at the command line. To retrieve the documentation for the Asymptote programming language, type `texdoc asymptote`. To retrieve the documentation for asypictureB, type `texdoc asypictureB`.

No 3d interaction One of the most spectacular features of the asymptote package is the embedding of interactive three-dimensional images in PDF files. (Note that this requires the `inline` option when loading the package.) The asypictureB package currently has no such feature. In fact, the default header includes the line `settings.prc = false;` to prevent Asymptote from trying to produce an interactive (`prc`) image, since this might break things.

No `inline` option A second feature that is important, but less spectacular, is the `inline` option itself, which causes the Asymptote labels to be compiled (in LaTeX) with all the same packages loaded and macros defined as in the original document. For instance, font consistency between the Asymptote images and the larger document is ensured by this option. It comes with an important limitations— Asymptote cannot use size information about the labels when this option is in use.

The closest equivalent in the asypictureB package is the immediate expansion of macros introduced by the @ symbol. This does not have the drawback of the `inline` option, and has many additional uses. However, it is less flexible inside labels (since, e.g., macros with arguments cannot be used this way), and does not ensure font consistency.

To compensate for this when using asypictureB, important packages and macro definitions should be echoed in the Asymptote header. Here is an example of code that could be placed in the preamble of a document to make the macro `\RR` (for $\mathbb{R}$) available in both the document text and the labels of its Asymptote pictures:

```
\usepackage{amssymb}
\newcommand{\RR}{\mathbb{R}}
\begin{asyheader}
usepackage("amssymb");
texpreamble("\newcommand{\RR}{\mathbb{R}}");
\end{asyheader}
```

The output format is not automatically set. When using asymptote, the output format is automatically set to either `eps` or `pdf` depending on what TeX engine is being run (and which kind of graphics file it prefers). When using asypictureB, no such provision is made; if any format other than `eps` is desired, it must be selected by including the Asymptote code `settings.outformat="pdf";` (or `"png"`). This is by design, to allow the use of `png` files; but it can be a bit annoying at times. To set all Asymptote files to have `pdf` format, include the code

14

```
\begin{asyheader}
settings.outformat="pdf";
\end{asyheader}
```

in the preamble (or anywhere prior to the first `asypicture` environment).

Keys to asypicture are not conveyed to Asymptote In the `asymptote` package, keys like `width` and `height` are conveyed both to the implicit `\includegraphics` command and to Asymptote. In `asypictureB`, they are conveyed only to the `\includegraphics` command; font size and even resolution (for rasterized images) will be changed in the process of scaling the picture. Thus, the preferred method is to set the width, height, etc. through Asymptote's `size` command.
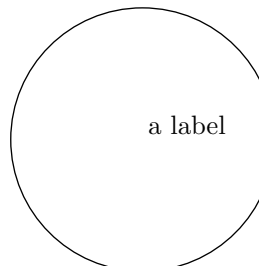
Bad (font size distorted):

a label

```
\begin{asypicture}{name=bad_width,
    width=3.5cm}
settings.outformat="pdf";
draw(unitcircle);
label("a label", position=(0,0), align=NE);
\end{asypicture}
```

Good:

a label

```
\begin{asypicture}{name=good_width, width=3.5cm}
size(3.5cm,0);
settings.outformat="pdf";
draw(unitcircle);
label("a label", position=(0,0), align=NE);
\end{asypicture}
```

Note that in the "good" example, the key `width=3.5cm` is necessary only if you want to make sure the included image has *precisely* the specified width, and probably not even then.

No spaces in file names The `asymptote` package takes measures to accomodate peculiar file names—in particular, file names that include spaces. The `asypictureB` package does not.

8   Implementation

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{asypictureB}
3   [2025/01/04 v0.4 user-friendly integration of Asymptote into LaTeX]
4 \RequirePackage{fancyvrb}
5 \RequirePackage{graphicx}
6 \RequirePackage{pgfkeys}
7
```

```
 8 \makeatletter
 9
10 \def\asy@OutFile{\FV@OutFile}
11
12 \RequirePackage{ifplatform}
13
14 \ifshellescape
15     \def\ASYPIC@shell{18}
16     \newcommand{\shell@execute}{\immediate\write\ASYPIC@shell}
17 \else
18     \newwrite\ASYPIC@shell
19     \ifwindows
20         \openout\ASYPIC@shell=\jobname-compile_asy.bat\relax
21     \else
22         \openout\ASYPIC@shell=\jobname-compile_asy.sh\relax
23     \fi
24     \newcommand{\shell@execute}[1]{{%
25         \edef\temp{#1}%
26         \write\expandafter\ASYPIC@shell\expandafter{\temp}%
27     }}
28 \fi
29
30 \ifshellescape
31     \newcommand{\copyfile}[2]{%
32     \ifwindows%
33         \immediate\write18{copy #1 #2 /y}%
34     \else%
35         \immediate\write18{cp #1 #2}%
36     \fi%
37     }
38 \else
39     \RequirePackage{verbatimcopy}
40     \newcommand{\copyfile}[2]{%
41         \OldVerbatimCopy{#1}{#2}%
42     }
43 \fi
44
45 \newcommand{\deletefile}[1]{%
46     \ifwindows%
47         \shell@execute{del #1}%
48     \else%
49         \shell@execute{rm #1}%
50     \fi%
51 }
52
53 \newcommand{\asyset}[1]{\pgfqkeys{/asy}{#1}}
54 \newcommand{\@asyerrorfilename}{\@asypicturename_errors.txt}
55 \newcounter{@asy@linenumber}
56
57 \asyset{name/.initial=noname, name/.value required}%
58
59 \asyset{graphic options/.code={}}
60 \asyset{set graphic option/.style={graphic options/.append code=#1}}
```

Unrecognized keys should be passed to the \includegraphics command using

\setkeys.
```
61 \asyset{.unknown/.code = %
62     {%
63         \edef\unknownkey{\pgfkeyscurrentname}%
64         \asyset{set graphic option/.expand once =   {%
65             \expandafter\setkeys\expandafter{%
66                 \expandafter G\expandafter i\expandafter n\expandafter%
67                 }\expandafter{\unknownkey=#1}%
68         }}%
69     }%
70 }
```
However, scale and angle must be dealt with separately.

Important note: using this implementation, scale and angle will always be the next-to-last, respectively the last, keys executed, no matter in what order the keys are given. Thus, it is impossible to rotate an asypicture and then set the width or height using these keys. However, this functionality can be provided within the Asymptote code.
```
71 \def\asy@scale{1}
72 \asyset{scale/.style={set graphic option = {\def\asy@scale{#1}}},
73     scale/.value required}
74 \newcommand{\asy@angle}{0}
75 \asyset{angle/.style = {set graphic option = {\def\asy@angle{#1}}},
76     angle/.value required}
77
78 \newcommand{\getfontsize}{\f@size}
79
80 \newif\ifasyfilechanged
81 \newif\ifASYPIC@flush
82 \newif\if@asyrepeat
83 \newread\@asyreadold
84 \newread\@asyreadnew
85 \edef\@tempasyfile{\jobname-temp}
86
87 \newcommand{\RequireAsyRecompile}{\ASYPIC@flushtrue}
88 \newcommand{\AsyCompileIfNecessary}{\ASYPIC@flushfalse}
89 \AsyCompileIfNecessary
90
91 \newcommand\clearasyheader{\def\ASYPIC@header{}}
92
93 \def\ASYPIC@header{}
94
95 \def\asyheader{\FV@Environment{}{asyheader}}
96
97 \def\FVB@asyheader{%
98     \@bsphack
99         \begingroup
100        \FV@UseKeyValues
101        \FV@DefineWhiteSpace
102        \def\FV@Space{\space}%
103        \FV@DefineTabOut
104        \def\FV@ProcessLine##1{\g@addto@macro\ASYPIC@header{##1^^J}}%
105        \let\FV@FontScanPrep\relax
106        %% DG/SR modification begin - May. 18, 1998
```

```
107          %% (to avoid problems with ligatures)
108          \let\@noligs\relax
109          %% DG/SR modification end
110          \FV@Scan}%
111
112 \def\FVE@asyheader{\endgroup\@esphack}
113
114 \DefineVerbatimEnvironment{asyheader}{asyheader}%
115     {codes={\catcode`\@=0},tabsize=4}
```
Set up the default `asyheader`:
```
116 \begin{asyheader}
117 defaultpen(fontsize(@getfontsize pt));
118 settings.prc = false;
119 \end{asyheader}
```
Now, define the `asypicture` environment using `fancyvrb` internals:
```
120 \def\asypicture{\FV@Environment{}{asypicture}}
121
122 \newcommand{\ASYPIC@recordname}[1]{%
123     \edef\tempmacroname{ASYPIC@name@#1}%
124     \ifcsname\tempmacroname\endcsname%% if \ASYPIC@name@... is defined
125         \edef\oldnum{\csname\tempmacroname\endcsname}%
126         \edef\@asypicturename%
127             {\jobname-#1__\oldnum}%
128         \expandafter\xdef\csname\tempmacroname\endcsname%
129             {\the\numexpr\oldnum+1\relax}%
130         \edef\ASYPIC@current@num{\csname\tempmacroname\endcsname}%
131     \else%% This is the first time this name is being used.
132         \edef\@asypicturename%
133             {\jobname-#1}%
134         \expandafter\gdef\csname\tempmacroname\endcsname{1}%
135     \fi%
136     \xdef\asylistingfile{\@asypicturename.asy}
137 }
138
```
Most of the following definition is copied verbatim from the definition of the `VerbatimOut` environment in `fancyvrb.dtx`. I don't actually understand what a lot of it does.
```
139
140 \def\FVB@asypicture#1{%
141     \@bsphack
142     \asyset{graphic options/.code={}}%
143     \asyset{#1, name/.get = \currentname}%
144     \ASYPIC@recordname{\currentname}%
145     \begingroup
146         \FV@UseKeyValues
147         \FV@DefineWhiteSpace
148         \def\FV@Space{\space}%
149         \FV@DefineTabOut
150         \def\FV@ProcessLine{\immediate\write\asy@OutFile}%
151         \immediate\openout\asy@OutFile\@tempasyfile.asy\relax
152         \immediate\write\asy@OutFile{\ASYPIC@header^^J}
153         \let\FV@FontScanPrep\relax
154         %% DG/SR modification begin - May. 18, 1998
```

```
155        %% (to avoid problems with ligatures)
156        \let\@noligs\relax
157        %% DG/SR modification end
158        \FV@Scan}
159
160 \newcommand{\ASYPICcomparefiles}[2]{
161     \IfFileExists{\@asypicturename.asy}%
162        {\openin\@asyreadold=#1.asy\relax%
163          \openin\@asyreadnew=#2.asy\relax%
164          \asyfilechangedfalse%
165          \@asyrepeattrue%
166          \loop%
167             \ifeof\@asyreadold%
168                 \@asyrepeatfalse%
169                 \ifeof\@asyreadnew%
170                 \else%
171                     \asyfilechangedtrue%
172                 \fi%
173             \else%
174                 \ifeof\@asyreadnew%
175                     \@asyrepeatfalse%
176                     \asyfilechangedtrue%
177                 \else% Not at the end of either file in this case
178                     \readline\@asyreadold to \oldfileline%
179                     \readline\@asyreadnew to \newfileline%
180                     \ifx\oldfileline\newfileline%
181                         \@asyrepeattrue%
182                     \else
183                         \asyfilechangedtrue%
184                         \@asyrepeatfalse%
185                     \fi%
186                 \fi%
187             \fi%
188             \if@asyrepeat
189          \repeat%
190          \closein\@asyreadold%
191          \closein\@asyreadnew%
192        }%
193        {\asyfilechangedtrue}%
194     \IfFileExists{#1.pdf}{}{%
195        \IfFileExists{#1.png}{}{%
196           \IfFileExists{#1.eps}{}{%
197              \asyfilechangedtrue%
198           }%
199        }%
200     }%
201 }
202
203 \def\ASYPIC@runasy{%
204     \message{Attempting to run asy on \@asypicturename.asy^^J}%
205     \shell@execute{asy -noV \@asypicturename.asy 2> \@asyerrorfilename}%
206     \openin\@asyreadold=\@asyerrorfilename\relax%
207     \ifeof\@asyreadold%
208         \closein\@asyreadold% Error log does not exist.
```

```
209    \else%
210        \def\@asyerrormessage{^^J}%
211        %Create command to process line of log file to figure
212        %out the Asymptote file line number:
213        \edef\@asy@temp{%
214            \def\noexpand\@asy@processerrorline####1\detokenize\expandafter{%
215                \@asypicturename.asy:%
216            }%
217        }%
218        \@asy@temp##2.##3:\relax{%
219            \xdef\@asy@errorlinenumber{\numexpr##2\relax}}%
220        %
221        \gdef\@asy@errorlinenumber{-5}%
222        {%
223            \endlinechar=-1%
224            \loop\unless\ifeof\@asyreadold%
225                \readline\@asyreadold to \@asyerrorcurrentline%
226                \ifnum\@asy@errorlinenumber=-5%
227                {\expandafter\expandafter\expandafter%
228                 \@asy@processerrorline\expandafter%
229                 % Suffix provides default of -5:
230                 \@asyerrorcurrentline\detokenize\expandafter%
231                    {\@asypicturename.asy:} -5.1:\relax%
232                }%
233                \fi%
234                \expandafter\g@addto@macro\expandafter\@asyerrormessage%
235                \expandafter{\@asyerrorcurrentline^^J}%
236            \repeat%
237        }%
238        \closein\@asyreadold%
239        \ifnum\@asy@errorlinenumber=-5%  No error
240        \else%
241            \openin\@asyreadold\@asypicturename.asy\relax%
242            \edef\numlinesout{5}
243            \setcounter{@asy@linenumber}{\numlinesout}
244            \loop\ifnum\value{@asy@linenumber}<\@asy@errorlinenumber%
245                \readline\@asyreadold to \temp%
246                \stepcounter{@asy@linenumber}%
247            \repeat%
248            %
249            \addtocounter{@asy@linenumber}{-\numlinesout}%
250            %
251            {%
252                \endlinechar=`^^J%
253                \loop\ifnum\value{@asy@linenumber}<\@asy@errorlinenumber%
254                    \stepcounter{@asy@linenumber}%
255                    \edef\temp/{\arabic{@asy@linenumber}}%
256                    \expandafter\g@addto@macro\expandafter%
257                        \@asyerrormessage\expandafter{\temp/ }%
258                    \readline\@asyreadold to \@asytempmessage%
259                    \expandafter\g@addto@macro\expandafter%
260                        \@asyerrormessage\expandafter{\@asytempmessage}%
261                \repeat%
262            }%
```

```
263            \closein\@asyreadold%
264            \PackageError{asypictureB}{\@asyerrormessage}{%
265                The Asymptote run described above
266                gave a non-empty error log. I have^^J%
267                reproduced the error log and attempted
268                to print the five lines leading^^J%
269                up to the error. Press enter or return
270                to continue, and then fix your^^J%
271                Asymptote code when this run is done.
272            }%
273        \fi%
274    \fi%
275 }
276
277 \def\FVE@asypicture{\immediate\closeout\asy@OutFile\endgroup%
278    \@esphack%
279    \ifASYPIC@flush%
280        \asyfilechangedtrue%
281    \else%
282        \ASYPICcomparefiles{\@asypicturename}{\@tempasyfile}%
283    \fi%
284    \ifasyfilechanged%
285        \IfFileExists{\@asypicturename.png}%
286            {\deletefile{\@asypicturename.png}}{}%
287        \IfFileExists{\@asypicturename.pdf}%
288            {\deletefile{\@asypicturename.pdf}}{}%
289        \IfFileExists{\@asypicturename.eps}%
290            {\deletefile{\@asypicturename.eps}}{}%
291        \copyfile{\@tempasyfile.asy}{\@asypicturename.asy}%
292        \ASYPIC@runasy%
293    \fi%
294    \asyset{graphic options}%
295    % Avoid giving "file does not exist" errors.
296    \chardef\previousinteractionmode=\interactionmode%
297    \batchmode%
298    \includegraphics[scale=\asy@scale,angle=\asy@angle]%
299        {\@asypicturename}%
300    \interactionmode=\previousinteractionmode%
301 }
302
303 \DefineVerbatimEnvironment{asypicture}{asypicture}%
304    {codes={\catcode`\@=0},tabsize=4}
305
306 %\AtEndDocument{\deletefile{\@tempasyfile.asy}}
307 \ifshellescape\else
308    \AtEndDocument{\closeout\ASYPIC@shell}
309 \fi
310
311
312 \makeatother
```

9 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

23