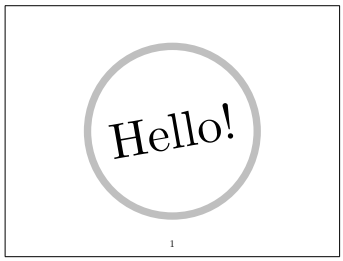
The place package
Absolute placement with coffins

Magnus Lie Hetland

September 20, 2022

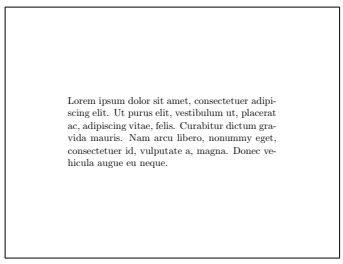
The place package lets you place contents at an absolute position, anchored at some specified part of the content, similar to how tikz nodes work, though without using the two-pass strategy of tikz. It also avoids messing with the order of beamer overlays, which is what happens when one uses the textpos package with the overlay option. The solution used is quite straightforward, combining *coffins* (using l3coffins) with the placement mechanisms of atbegshi.

The main command of the package is `\place`, which is somewhat similar to the `\put` command of the LaTeX `picture` environment. By default, it places its contents in the middle of the page, so `\place{Hello}` will put the word "Hello" front and center. Variations are then possible using an optional argument consisting of keys and key–value pairs, as shown in the following.




```
% In preamble:
% \usepackage{tikz}
\place{\tikz{
    \draw[line width=8pt, lightgray] circle[radius=3.25];
}}
\place[angle=10, scale=5]{Hello!}
\place[pg-b, y=.5cm]{\thepage}
```

Supplying a `width` creates a vertical box. If the surrounding text is set, for example, using `\raggedright` (as is common in beamer), the contents of the `\place` command will be, too. So if you want your text justified, you can use something like the `\justifying` command from ragged2e.




```
% In preamble:
% \usepackage{ragged2e}
% \usepackage{lipsum}
\place[width=8cm]{
    \justifying
    \lipsum[1][1-5] % Example text
}
```

The options can also be set globally (or within a group) using `\placesetup`. For example, if you with to scale everything you place by a certain amount, you could do that as in

the following.

Scaled!

```
\placesetup{scale=5}
\place{Scaled!}
```

By default, the contents are placed *in front* of the current page, but this can be modified by supplying your own "page coffin" (an l3coffins or xcoffins coffin) and typesetting that with, e.g., atbegshi. If, for example, you wish to place something in the background of every slide of a beamer presentation, you could do that by adding the following to the preamble.[1]

```
\setbeamercolor{background canvas}{bg=} % So the coffin isn't obscured

\NewCoffin \MyPageCoffin
\AtBeginShipout{
    \AtBeginShipoutUpperLeft{
        \TypesetCoffin \MyPageCoffin [t,l] (0pt,0pt)
    }
}
```

Now simply add a `\place` command at the beginning of the document, before your first `frame` (after `\begin{document}`), using `pg=\MyPageCoffin` to specify which coffin to use for rendering and positioning contents. This gives you a coordinate system starting at the top left corner of the page. If you wish to anchor your positioned elements elsewhere, you'll need to resize your page coffin (see p. 4).

Reference

`\place` `\place [⟨options⟩] {⟨contents⟩}`

Places ⟨*contents*⟩ at some position (possibly scaled or rotated) as dictated by the optional ⟨*options*⟩, supplied as keys and key–value pairs. The *defaults* are used as values if only the keys are given, and the *initial values* are used if the options are not specified. The available options are as follows.

- `angle=`⟨*angle*⟩: The angle to which the coffin is rotated before being positioned.

  No default. Initial value `0`.

- `h=`⟨*h-anchor*⟩: The horizontal anchor to use for positioning. This corresponds to a l3coffins (or xcoffins) *pole*. The available anchors are `l` for left, `hc` for center and `r` for right. These are also available as separate valueless keys, so if you supply `\place` with the key `l`, that is equivalent to using `h=l`, etc.

[1] Note that because it will end up behind even the slide itself, you need to remove the background color, by setting `bg` to nothing, as in the example code. If you want it behind the content, but still have a background color, you would need to add that yourself, with a colored box or filled tikz rectangle, possibly also using the `\place` command.

There are also two-letter shortcuts defined for setting the horizontal and vertical anchors simultaneously, with the vertical one first. For example, `tl` is equivalent to `v=t,h=l`. In these double shortcuts, the prefixes `h` and `v` have been stripped from the center anchors, so that bottom center is `bc`, for example, and not `bhc`.

No default. Initial value `hc`, which centers the coffin horizontally at ⟨*x-coord*⟩.

- `pg=`⟨*page-coffin*⟩: Supplies the coffin to use as the page, or "canvas," for placing the contents.

  No default. The initial value is an internal canvas that is rendered in front of the current page (on shipout), and then cleared.

- `pg-h=`⟨*page-h-anchor*⟩: Similar to `h`, except that it applies to the canvas (set by `pg`). Shortcuts (like `pg-l` for `pg-h=l`) are defined for the default anchors (`l`, `hc`, `r`).

  Double shortcuts are defined for `pg-h` and `pg-v` in a similar manner to those defined for `h` and `v`, so that `pg-tl` is equivalent to `pg-v=t,pg-h=l`.

  No default. Initial value `hc`.

- `pg-v=`⟨*page-v-anchor*⟩: Similar to `v`, except that it applies to the canvas (set by `pg`). Shortcuts (like `pg-t` for `pg-v=t`) are defined for the default anchors (`t`, `vc`, `b`, `T`, `B`, `H`).

  Combined shortcuts such as `pg-tl` and `pg-br`, etc., are also defined. See the descriptions of `h` and `pg-h` for more.

  No default. Initial value `vc`.

- `scale=`⟨*scale*⟩: The factor by which the coffin is scaled.

  No default. If `scale` is not used, no scaling is performed.

- `v=`⟨*v-anchor*⟩: The vertical anchor to use for positioning (see `h`). The available anchors are `t` for top, `vc` for center, `b` for bottom and `H` for baseline. These are also available as separate valueless keys, so if you supply `\place` with the key `t`, that is equivalent to using `v=t`, etc.

  Note that if the `width` key is set, a vertial coffin is used, which makes the additional anchors (or poles) `T` (baseline of the first line, or other material at the top) and `B` (baseline for the last line, or other material at the bottom).

  Combined shortcuts such as `tl` and `br`, etc., are also defined. See the description of `h` for more.

  No default. Initial value `vc`, which centers the coffin vertically at ⟨*y-coord*⟩.

- `width=`⟨*width*⟩: Set the width constraining the contents, before scaling. If this is set, the contents are set in vertical mode; otherwise, they are set in horizontal mode.

  No default and no initial value.

- `x=`⟨*x-coord*⟩: The x coordinate, measured from the current horizontal page anchor, set by `pg-h`.

  No default. Initial value `0pt`.

- `y=`⟨*y-coord*⟩: The y coordinate, measured from the current vertical page anchor, set by `pg-v`.

  No default. Initial value `0pt`.

`\placesetup` `\placesetup {`⟨*options*⟩`}`

Sets the options, as described under `\place`, to the value they should have within the current group (or globally, if not inside a group).

`\placecoffin` `\placecoffin [`⟨*options*⟩`]` ⟨*coffin*⟩

Places ⟨*coffin*⟩ at some position (possibly scaled or rotated), just like `\place`. The difference is that a coffin is provided directly, rather than constructed by typesetting some supplied contents. This means that the `width` key is not used, though it is permitted for consistency (and ease of reusing option lists).

Implementation

```
⟨*package⟩
⟨@@=plc⟩
```

First, we need to import the necessary packages. We're using expl3 to set up the package, so we get l3coffins for free (and cannot import it directly, anyway). If place is used with beamer, then atbegshi is already included. Elsewhere, though, it will not be, so we require it here.

```
\RequirePackage{atbegshi}
```

We make sure we have the variants we need of some existing commands:

```
\cs_generate_variant:Nn \tl_if_novalue:nTF       { VTF }
\cs_generate_variant:Nn \tl_if_novalue:nF        { VF }
\cs_generate_variant:Nn \coffin_gattach:NnnNnnnn { NooNoonn }
```

`\g_plc_shipout_coffin` This variable is normally used to store the contents to be placed on the current page. It accumulates the contents from every call to `\place` until the page is about to be typeset (shipped out), at which point `\g_plc_shipout_coffin` is typeset in front.

```
\coffin_new:N \g_plc_shipout_coffin
```

`\plc_reset_shipout_coffin:` Used to initialize, and later reinitialize, the shipout coffin, to simply be an empty coffin with the width and height of the current page.

```
\cs_new:Npn \plc_reset_shipout_coffin: {
    \hcoffin_gset:Nn \g_plc_shipout_coffin {
        \phantom{\rule{\paperwidth}{\paperheight}}
    }
}
```

The following hook ensures that that `\g_plc_shipout_coffin` is typeset in front of the page as it is being shipped out (anchored top left), and then cleared, ready to accumulate contents for the next page. The atbegshi command used wraps its contents in a `picture` environment, which we don't really need, but it seems to be more trouble than it's worth to get rid of it.

```
\plc_reset_shipout_coffin:
```

```
\AtBeginShipout{
    \AtBeginShipoutUpperLeftForeground{
        \coffin_typeset:Nnnnn \g_plc_shipout_coffin {t} {l} {0pt} {0pt}
        \plc_reset_shipout_coffin:
    }
}
```

`\g_plc_canvas_coffin` This variable is used as part of the `pg` key, and is initially set equal to `\g_plc_shipout_coffin`, so that whatever is rendered to the canvas (i.e., `\g_plc_canvas_coffin`) is placed on top of the page contents on shipout, and then cleared. If a different behavior is desired, another coffin can can be supplied instead. Rather than constructing a coffin, we just set up a token list, which initially will be aliased to the actual coffin `\g_plc_shipout_coffin` as part of the key handling.

```
\tl_new:N \g_plc_canvas_coffin
```

`\plc_reset_overlay:` This is a utility to reset beamer overlay specifications at the end of our coffin contents. The idea is that if we use, say, `\pause` inside `\place`, then everything after the `\pause` command will be paused, including material *outside* `\place`. To restrict such commands to the contents provided to `\place`, we use `\onslide<1->` after typesetting the contents in our coffin. This is only useful (or possible) when using beamer, so otherwise, we just alias `\plc_reset_overlay:` to `\relax`, i.e., don't do anything.

```
\tl_new:N \plc_reset_overlay:
\@ifclassloaded{beamer}
    { \tl_gset:Nn    \plc_reset_overlay: { \onslide<1-> } }
    { \tl_gset_eq:NN \plc_reset_overlay:   \relax         }
```

We now define the various keys used to configure `\place`, as described in the reference.

```
\keys_define:nn { place } {

    angle .tl_set:N          = \l_plc_angle_tl,
    angle .initial:V         = \c_novalue_tl,
    angle .value_required:n  = true,

    h     .tl_set:N          = \l_plc_hanchor_tl,
    h     .initial:n         = hc,
    h     .value_required:n  = true,

    pg    .code:n            = { \tl_set_eq:NN \g_plc_canvas_coffin #1 },
    pg    .initial:n         = \g_plc_shipout_coffin,
    pg    .value_required:n  = true,

    pg-h  .tl_set:N          = \l_plc_on_hanchor_tl,
    pg-h  .initial:n         = hc,
    pg-h  .value_required:n  = true,

    pg-v  .tl_set:N          = \l_plc_on_vanchor_tl,
    pg-v  .initial:n         = vc,
    pg-v  .value_required:n  = true,

    scale .tl_set:N          = \l_plc_scale_tl,
    scale .initial:V         = \c_novalue_tl,
    scale .value_required:n  = true,

```

```
    v     .tl_set:N          = \l_plc_vanchor_tl,
    v     .initial:n         = vc,
    v     .value_required:n  = true,

    width .tl_set:N          = \l_plc_width_tl,
    width .initial:V         = \c_novalue_tl,
    width .value_required:n  = true,

    x     .tl_set:N          = \l_plc_x_tl,
    x     .initial:n         = { 0pt },
    x     .value_required:n  = true,

    y     .tl_set:N          = \l_plc_y_tl,
    y     .initial:n         = { 0pt },
    y     .value_required:n  = true,

}

\clist_const:Nn \__plc_hanchors_clist {l,hc,r}
\clist_const:Nn \__plc_vanchors_clist {t,vc,b,H,T,B}

\cs_new:Npn \__plc_define_shortcuts:nnn #1 #2 #3 {
    \keys_define:nn { place } {

    #1    .meta:n            = { #2 },
    #1    .value_forbidden:n = true,

    pg-#1 .meta:n            = { #3 },
    pg-#1 .value_forbidden:n = true,

    }
}

\cs_generate_variant:Nn \__plc_define_shortcuts:nnn { Vnn }

\clist_map_inline:Nn \__plc_hanchors_clist {

    \__plc_define_shortcuts:nnn { #1 } { h = #1 } { pg-h = #1 }

}

\clist_map_inline:Nn \__plc_vanchors_clist {

    \__plc_define_shortcuts:nnn { #1 } { v = #1 } { pg-v = #1 }

}

\clist_map_inline:Nn \__plc_vanchors_clist {
    \clist_map_inline:Nn \__plc_hanchors_clist {

        \tl_clear:N \g_tmpa_tl

        \tl_if_eq:nnTF { #1 } { vc }
            { \tl_put_right:Nn \g_tmpa_tl { c } }
```

```
            { \tl_put_right:Nn \g_tmpa_tl { #1 } }

        \tl_if_eq:nnTF { ##1 } { hc }
            { \tl_put_right:Nn \g_tmpa_tl { c } }
            { \tl_put_right:Nn \g_tmpa_tl { ##1 } }

        \__plc_define_shortcuts:Vnn
            \g_tmpa_tl { v = #1, h = ##1 } { pg-v = #1, pg-h = ##1 }

    }
}
```

In order to let the user set options outside the actual use of `\place` and `\placecoffin`, we introduce a wrapper:

```
\NewDocumentCommand \placesetup { +m } {
    \keys_set:nn { place } { #1 }
}
```

`\place` This is the main command of the package. It takes a key–value list as its first optional argument, followed by the contents that are to be placed.

```
\NewDocumentCommand \place { +O{ } +m } {
    \group_begin:
```

First we handle the keys, updating variables holding coordinates, etc.

```
    \keys_set:nn { place } { #1 }
```

Then the contents are typeset, with some additions, into a temporary coffin. The additions are that whitespace is stripped around the contents, and that we reset any overlay counters (in case they are modified by `\onslide` or `\pause`, or the like). If we're not using beamer, this last part is a no-op.

If `width` has been set, the contents is set in vertical mode (using the given width); otherwise, it is set in horizontal mode.

```
    \tl_set:Nn \l_tmpa_tl {
        \ignorespaces #2 \unskip
        \plc_reset_overlay:
    }
    \tl_if_novalue:VTF \l_plc_width_tl {
        \hcoffin_set:Nn \l_tmpa_coffin \l_tmpa_tl
    } {
        \vcoffin_set:Nnn \l_tmpa_coffin \l_plc_width_tl \l_tmpa_tl
    }
```

Finally, actually place the typeset coffin on the canvas coffin.

```
    \__plc_place_coffin:N \l_tmpa_coffin
    \group_end:
}
```

`\placecoffin` A thin wrapper around the internal `\__plc_place_coffin:N`. The only extra work done here is setting the keys.

```
\NewDocumentCommand \placecoffin { +O{ } m } {
    \group_begin:
    \keys_set:nn { place } { #1 }
    \__plc_place_coffin:N #2
    \group_end:
}
```

`\__plc_place_coffin:N` An internal macro for placing a given coffin, using the options currently in effect (i.e., as configured by `\keys_set:nn` in some outer group). Used by `\place` and `\placecoffin`.

```
\cs_new:Npn \__plc_place_coffin:N #1 {
```

If a scale has been provided, we scale the temporary coffin (with the same factor in both directions).

```
    \tl_if_novalue:VF \l_plc_scale_tl {
        \coffin_scale:Nnn #1 \l_plc_scale_tl \l_plc_scale_tl
    }
```

If an angle has been provided, we rotate the temporary coffin.

```
    \tl_if_novalue:VF \l_plc_angle_tl {
        \coffin_rotate:Nn #1 \l_plc_angle_tl
    }
```

We now join the temporary coffin to our canvas coffin, so it will be rendered at the correct position at shipout.

```
    \coffin_gattach:NooNoonn
        \g_plc_canvas_coffin
            {\l_plc_on_hanchor_tl} {\l_plc_on_vanchor_tl}
        #1
            {\l_plc_hanchor_tl} {\l_plc_vanchor_tl}
            {\l_plc_x_tl} {\l_plc_y_tl}
```

Ideally, what we just did should have been enough. However, until October 2021, the `\coffin_gattach:` command was not actually global,[2] so to have our modification take effect outside the current group also for older versions of l3coffins, we perform a final (normally redundant) global self-assignment before ending the `\place` command.

```
    \coffin_gset_eq:NN \g_plc_canvas_coffin \g_plc_canvas_coffin
}
```

⟨/package⟩

[2] https://tex.stackexchange.com/questions/618198