

The l3coffins package

Coffins*

The L^AT_EX3 Project[†]

Released 2011/06/30

The material in this module provides the low-level support system for coffins. For details about the design concept of a coffin, see the xcoffins module.

1 Code-level coffin functions

<code>\coffin_new:N</code>
<code>\coffin_new:c</code>

`\coffin_new:N <coffin>`

Creates a new $\langle coffin \rangle$ or raises an error if the name is already taken. The declaration is global. The $\langle coffin \rangle$ will initially be empty.

<code>\coffin_clear:N</code>
<code>\coffin_clear:c</code>

`\coffin_clear:N <coffin>`

Clears the content of the $\langle coffin \rangle$ within the current T_EX group level.

<code>\coffin_set_eq:NN</code>
<code>\coffin_set_eq:Nc</code>
<code>\coffin_set_eq:cN</code>
<code>\coffin_set_eq:cc</code>

`\coffin_set_eq:NN <coffin1> <coffin2>`

Sets both the content and poles of $\langle coffin1 \rangle$ equal to those of $\langle coffin2 \rangle$ within the current T_EX group level.

<code>\hcoffin_set:Nn</code>
<code>\hcoffin_set:cn</code>

`\hcoffin_set:Nn <coffin> {<material>}`

*This file describes v2488, last revised 2011/06/30.

[†]E-mail: latex-team@latex-project.org

Typesets the $\langle material \rangle$ in horizontal mode, storing the result in the $\langle coffin \rangle$. The standard poles for the $\langle coffin \rangle$ are then set up based on the size of the typeset material.

<code>\vcoffin_set:Nnn</code>	<code>\vcoffin_set:Nnn $\langle coffin \rangle$ {$\langle width \rangle$} {$\langle material \rangle$}</code>
<code>\vcoffin_set:cnn</code>	

Typesets the $\langle material \rangle$ in vertical mode constrained to the given $\langle width \rangle$ and stores the result in the $\langle coffin \rangle$. The standard poles for the $\langle coffin \rangle$ are then set up based on the size of the typeset material.

<code>\coffin_set_horizontal_pole:Nnn</code>	<code>\coffin_set_horizontal_pole:Nnn $\langle coffin \rangle$ {$\langle pole \rangle$} {$\langle offset \rangle$}</code>
<code>\coffin_set_horizontal_pole:cnn</code>	

Sets the $\langle pole \rangle$ to run horizontally through the $\langle coffin \rangle$. The $\langle pole \rangle$ will be located at the $\langle offset \rangle$ from the bottom edge of the bounding box of the $\langle coffin \rangle$. The $\langle offset \rangle$ should be given as a dimension expression; this may include the terms `csTotalHeight`, `\Height`, `\Depth` and `\Width`, which will evaluate to the appropriate dimensions of the $\langle coffin \rangle$.

<code>\coffin_set_vertical_pole:Nnn</code>	<code>\coffin_set_vertical_pole:Nnn $\langle coffin \rangle$ {$\langle pole \rangle$} {$\langle offset \rangle$}</code>
<code>\coffin_set_vertical_pole:cnn</code>	

Sets the $\langle pole \rangle$ to run vertically through the $\langle coffin \rangle$. The $\langle pole \rangle$ will be located at the $\langle offset \rangle$ from the left-hand edge of the bounding box of the $\langle coffin \rangle$. The $\langle offset \rangle$ should be given as a dimension expression; this may include the terms `\TotalHeight`, `\Height`, `\Depth` and `\Width`, which will evaluate to the appropriate dimensions of the $\langle coffin \rangle$.

<code>\coffin_rotate:Nn</code>	<code>\coffin_rotate:Nn $\langle coffin \rangle$ {$\langle angle \rangle$}</code>
<code>\coffin_rotate:cn</code>	

Rotates the $\langle coffin \rangle$ by the given $\langle angle \rangle$ (given in degrees counter-clockwise). This process will rotate both the coffin content and poles. Multiple rotations will not result in the bounding box of the coffin growing unnecessarily.

<code>\coffin_attach:NnnNnnnn</code>	<code>\coffin_attach:NnnNnnnn $\langle coffin1 \rangle$ {$\langle coffin1-pole1 \rangle$} {$\langle coffin1-pole2 \rangle$} $\langle coffin2 \rangle$ {$\langle coffin2-pole1 \rangle$} {$\langle coffin2-pole2 \rangle$} {$\langle x-offset \rangle$} {$\langle y-offset \rangle$}</code>
<code>\coffin_attach:cnnNnnnn</code>	
<code>\coffin_attach:Nnncnnnn</code>	
<code>\coffin_attach:cnnccnnnn</code>	

This function carries out alignment such that the bounding box of $\langle coffin1 \rangle$ is not altered, *i.e.* $\langle coffin2 \rangle$ can protrude outside of the bounding box of the coffin. The alignment is carried out by first calculating $\langle handle1 \rangle$, the point of intersection of $\langle coffin1-pole1 \rangle$ and $\langle coffin1-pole2 \rangle$, and $\langle handle2 \rangle$, the point of intersection of $\langle coffin2-pole1 \rangle$ and

$\langle coffin2-pole2 \rangle$. $\langle coffin2 \rangle$ is then attached to $\langle coffin1 \rangle$ such that the relationship between $\langle handle1 \rangle$ and $\langle handle2 \rangle$ is described by the $\langle x-offset \rangle$ and $\langle y-offset \rangle$. The two offsets should be given as dimension expressions.

$\backslash coffin_join:NnnNnnnn$ $\backslash coffin_join:cnnNnnnn$ $\backslash coffin_join:Nnncnnnn$ $\backslash coffin_join:cnnccnnnn$	$\backslash coffin_join:NnnNnnnn$ $\langle coffin1 \rangle \{ \langle coffin1-pole1 \rangle \} \{ \langle coffin1-pole2 \rangle \}$ $\langle coffin2 \rangle \{ \langle coffin2-pole1 \rangle \} \{ \langle coffin2-pole2 \rangle \}$ $\{ \langle x-offset \rangle \} \{ \langle y-offset \rangle \}$
---	---

This function carries out alignment such that the bounding box of $\langle coffin1 \rangle$ after the process will expand. The new bounding box will cover the area contain the bounding boxes of the two original coffins. The alignment is carried out by first calculating $\langle handle1 \rangle$, the point of intersection of $\langle coffin1-pole1 \rangle$ and $\langle coffin1-pole2 \rangle$, and $\langle handle2 \rangle$, the point of intersection of $\langle coffin2-pole1 \rangle$ and $\langle coffin2-pole2 \rangle$. $\langle coffin2 \rangle$ is then attached to $\langle coffin1 \rangle$ such that the relationship between $\langle handle1 \rangle$ and $\langle handle2 \rangle$ is described by the $\langle x-offset \rangle$ and $\langle y-offset \rangle$. The two offsets should be given as dimension expressions.

$\backslash coffin_typeset:Nnnnn$ $\backslash coffin_typeset:cnnnn$	$\backslash coffin_typeset:Nnnnn \langle coffin \rangle \{ \langle pole1 \rangle \} \{ \langle pole2 \rangle \}$ $\{ \langle x-offset \rangle \} \{ \langle y-offset \rangle \}$
--	--

Typesetting is carried out by first calculating $\langle handle \rangle$, the point of intersection of $\langle pole1 \rangle$ and $\langle pole2 \rangle$. The coffin is then typeset such that the relationship between the current reference point in the document and the $\langle handle \rangle$ is described by the $\langle x-offset \rangle$ and $\langle y-offset \rangle$. The two offsets should be given as dimension expressions. Typesetting a coffin is therefore analogous to carrying out an alignment where the “parent” coffin is the current insertion point.

$\backslash coffin_display_handles:cn$ $\backslash coffin_display_handles:cn$	$\backslash coffin_display_handles:Nn \langle coffin \rangle \{ \langle colour \rangle \}$
--	--

This function first calculates the intersections between all of the $\langle poles \rangle$ of the $\langle coffin \rangle$ to give a set of $\langle handles \rangle$. It then prints the $\langle coffin \rangle$ at the current location in the source, with the position of the $\langle handles \rangle$ marked on the coffin. The $\langle handles \rangle$ will be labelled as part of this process: the locations of the $\langle handles \rangle$ and the labels are both printed in the $\langle colour \rangle$ specified.

$\backslash coffin_mark_handle:Nnnn$ $\backslash coffin_mark_handle:cnnn$	$\backslash coffin_mark_handle:Nnnn \langle coffin \rangle \{ \langle pole1 \rangle \} \{ \langle pole2 \rangle \}$ $\{ \langle colour \rangle \}$
--	---

This function first calculates the $\langle handle \rangle$ for the $\langle coffin \rangle$ as defined by the intersection of $\langle pole1 \rangle$ and $\langle pole2 \rangle$. It then marks the position of the $\langle handle \rangle$ on the $\langle coffin \rangle$.

The $\langle handle \rangle$ will be labelled as part of this process: the location of the $\langle handle \rangle$ and the label are both printed in the $\langle colour \rangle$ specified.

$\backslash\text{coffin_show_structure:N}$ $\backslash\text{coffin_show_structure:c}$	$\backslash\text{coffin_show_structure:N } \langle\text{coffin}\rangle$
--	---

This function shows the structural information about the $\langle coffin \rangle$ in the terminal. The width, height and depth of the typeset material are given, along with the location of all of the poles of the coffin.

Notice that the poles of a coffin are defined by four values: the x and y co-ordinates of a point that the pole passes through and the x - and y -components of a vector denoting the direction of the pole. It is the ratio between the later, rather than the absolute values, which determines the direction of the pole.

2 l3coffins Implementation

```

1  $\langle *package \rangle$ 
2  $\backslash\text{ProvidesExplPackage}$ 
3    $\{\backslash\text{ExplFileName}\}\{\backslash\text{ExplFileDate}\}\{\backslash\text{ExplFileVersion}\}\{\backslash\text{ExplFileDescription}\}$ 

```

2.1 Coffins: data structures and general variables

$\backslash\text{l_coffin_tmp_box}$	Scratch variables.
$\backslash\text{l_coffin_tmp_dim}$	
$\backslash\text{l_coffin_tmp_fp}$	4 $\backslash\text{box_new:N } \backslash\text{l_coffin_tmp_box}$
$\backslash\text{l_coffin_tmp_tl}$	5 $\backslash\text{dim_new:N } \backslash\text{l_coffin_tmp_dim}$
	6 $\backslash\text{fp_new:N } \backslash\text{l_coffin_tmp_fp}$
	7 $\backslash\text{tl_new:N } \backslash\text{l_coffin_tmp_tl}$
$\backslash\text{c_coffin_corners_prop}$	The “corners”; of a coffin define the real content, as opposed to the $\text{T}_{\text{E}}\text{X}$ bounding box. They all start off in the same place, of course.
	8 $\backslash\text{prop_new:N } \backslash\text{c_coffin_corners_prop}$
	9 $\backslash\text{prop_put:Nnn } \backslash\text{c_coffin_corners_prop } \{ \text{tl} \} \{ \{ 0 \text{ pt} \} \{ 0 \text{ pt} \} \}$
	10 $\backslash\text{prop_put:Nnn } \backslash\text{c_coffin_corners_prop } \{ \text{tr} \} \{ \{ 0 \text{ pt} \} \{ 0 \text{ pt} \} \}$
	11 $\backslash\text{prop_put:Nnn } \backslash\text{c_coffin_corners_prop } \{ \text{bl} \} \{ \{ 0 \text{ pt} \} \{ 0 \text{ pt} \} \}$
	12 $\backslash\text{prop_put:Nnn } \backslash\text{c_coffin_corners_prop } \{ \text{br} \} \{ \{ 0 \text{ pt} \} \{ 0 \text{ pt} \} \}$
$\backslash\text{c_coffin_poles_prop}$	Pole positions are given for horizontal, vertical and reference-point based values.
	13 $\backslash\text{prop_new:N } \backslash\text{c_coffin_poles_prop}$
	14 $\backslash\text{tl_set:Nn } \backslash\text{l_coffin_tmp_tl } \{ \{ 0 \text{ pt} \} \{ 0 \text{ pt} \} \{ 0 \text{ pt} \} \{ 1000 \text{ pt} \} \}$
	15 $\backslash\text{prop_put:Nno } \backslash\text{c_coffin_poles_prop } \{ \text{l} \} \{ \backslash\text{l_coffin_tmp_tl} \}$
	16 $\backslash\text{prop_put:Nno } \backslash\text{c_coffin_poles_prop } \{ \text{hc} \} \{ \backslash\text{l_coffin_tmp_tl} \}$
	17 $\backslash\text{prop_put:Nno } \backslash\text{c_coffin_poles_prop } \{ \text{r} \} \{ \backslash\text{l_coffin_tmp_tl} \}$

```

18 \tl_set:Nn \l_coffin_tmp_tl { { 0 pt } { 0 pt } { 1000 pt } { 0 pt } }
19 \prop_put:Nno \c_coffin_poles_prop { b } { \l_coffin_tmp_tl }
20 \prop_put:Nno \c_coffin_poles_prop { vc } { \l_coffin_tmp_tl }
21 \prop_put:Nno \c_coffin_poles_prop { t } { \l_coffin_tmp_tl }
22 \prop_put:Nno \c_coffin_poles_prop { B } { \l_coffin_tmp_tl }
23 \prop_put:Nno \c_coffin_poles_prop { H } { \l_coffin_tmp_tl }
24 \prop_put:Nno \c_coffin_poles_prop { T } { \l_coffin_tmp_tl }

```

\l_coffin_calc_a_fp Used for calculations of intersections and in other internal places.
\l_coffin_calc_b_fp
\l_coffin_calc_c_fp 25 \fp_new:N \l_coffin_calc_a_fp
\l_coffin_calc_d_fp 26 \fp_new:N \l_coffin_calc_b_fp
\l_coffin_calc_result_fp 27 \fp_new:N \l_coffin_calc_c_fp
 28 \fp_new:N \l_coffin_calc_d_fp
 29 \fp_new:N \l_coffin_calc_result_fp

\l_coffin_error_bool For propagating errors so that parts of the code can work around them.
 30 \bool_new:N \l_coffin_error_bool

\l_coffin_offset_x_dim The offset between two sets of coffin handles when typesetting. These values are corrected
\l_coffin_offset_y_dim from those requested in an alignment for the positions of the handles.
 31 \dim_new:N \l_coffin_offset_x_dim
 32 \dim_new:N \l_coffin_offset_y_dim

\l_coffin_pole_a_tl Needed for finding the intersection of two poles.
\l_coffin_pole_b_tl
 33 \tl_new:N \l_coffin_pole_a_tl
 34 \tl_new:N \l_coffin_pole_b_tl

\l_coffin_sin_fp Used for rotations to get the sine and cosine values.
\l_coffin_cos_fp
 35 \fp_new:N \l_coffin_sin_fp
 36 \fp_new:N \l_coffin_cos_fp

\l_coffin_x_dim For calculating intersections and so forth.
\l_coffin_y_dim
\l_coffin_x_prime_dim 37 \dim_new:N \l_coffin_x_dim
\l_coffin_y_prime_dim 38 \dim_new:N \l_coffin_y_dim
 39 \dim_new:N \l_coffin_x_prime_dim
 40 \dim_new:N \l_coffin_y_prime_dim

\l_coffin_x_fp Used for calculations where there are clear *x*- and *y*-components, for example during
\l_coffin_y_fp vector rotation.
\l_coffin_x_prime_fp 41 \fp_new:N \l_coffin_x_fp
\l_coffin_y_prime_fp 42 \fp_new:N \l_coffin_y_fp
 43 \fp_new:N \l_coffin_x_prime_fp
 44 \fp_new:N \l_coffin_y_prime_fp

```

\l_coffin_Depth_dim
\l_coffin_Height_dim
\l_coffin_TotalHeight_dim
\l_coffin_Width_dim

```

Dimensions for the various parts of a coffin.

```

45 \dim_new:N \l_coffin_Depth_dim
46 \dim_new:N \l_coffin_Height_dim
47 \dim_new:N \l_coffin_TotalHeight_dim
48 \dim_new:N \l_coffin_Width_dim

```

```

\coffin_saved_Depth:
\coffin_saved_Height:
\coffin_saved_TotalHeight:
\coffin_saved_Width:

```

Used to save the meaning of `\Depth`, `\Height`, `\TotalHeight` and `\Width`.

```

49 \cs_new_nopar:Npn \coffin_saved_Depth: { }
50 \cs_new_nopar:Npn \coffin_saved_Height: { }
51 \cs_new_nopar:Npn \coffin_saved_TotalHeight: { }
52 \cs_new_nopar:Npn \coffin_saved_Width: { }

```

(End definition for `\coffin_saved_Depth`:. This function is documented on page ??.)

2.2 Basic coffin functions

There are a number of basic functions needed for creating coffins and placing material in them. This all relies on the following data structures.

`\coffin_if_exist_execute:Nn` Several of the higher-level coffin functions will give multiple errors if the coffin does not exist. A cleaner way to handle this is provided here: both the box and the coffin structure are checked.

```

53 \cs_new_protected:Npn \coffin_if_exist_execute:Nn #1#2
54 {
55   \cs_if_exist:NTF #1
56   {
57     \cs_if_exist:cTF { l_coffin_poles_ \int_value:w #1 _prop }
58     { #2 }
59     {
60       \msg_kernel_error:nxx { coffin } { unknown-coffin }
61       { \token_to_str:N #1 }
62     }
63   }
64   {
65     \msg_kernel_error:nxx { coffin } { unknown-coffin }
66     { \token_to_str:N #1 }
67   }
68 }

```

(End definition for `\coffin_if_exist_execute:Nn`. This function is documented on page ??.)

`\coffin_clear:N` `\coffin_clear:c` Clearing coffins means emptying the box and resetting all of the structures.

```

69 \cs_new_protected_nopar:Npn \coffin_clear:N #1
70 {
71   \coffin_if_exist_execute:Nn #1

```

```

72     {
73         \box_clear:N #1
74         \coffin_reset_structure:N #1
75     }
76 }
77 \cs_generate_variant:Nn \coffin_clear:N { c }

```

(End definition for `\coffin_clear:N` and `\coffin_clear:c`. These functions are documented on page [1](#).)

`\coffin_new:N` Creating a new coffin means making the underlying box and adding the data structures.
`\coffin_new:c` These are created globally, as there is a need to avoid any strange effects if the coffin is created inside a group. This means that the usual rule about `\l...` variables has to be broken.

```

78 \cs_new_protected_nopar:Npn \coffin_new:N #1
79 {
80     \box_new:N #1
81     \prop_clear_new:c { l_coffin_corners_ \int_value:w #1 _prop }
82     \prop_clear_new:c { l_coffin_poles_ \int_value:w #1 _prop }
83     \prop_gset_eq:cN { l_coffin_corners_ \int_value:w #1 _prop }
84         \c_coffin_corners_prop
85     \prop_gset_eq:cN { l_coffin_poles_ \int_value:w #1 _prop }
86         \c_coffin_poles_prop
87 }
88 \cs_generate_variant:Nn \coffin_new:N { c }

```

(End definition for `\coffin_new:N` and `\coffin_new:c`. These functions are documented on page [1](#).)

`\hcoffin_set:Nn` Horizontal coffins are relatively easy: set the appropriate box, reset the structures then
`\hcoffin_set:cn` update the handle positions.

```

89 \cs_new_protected:Npn \hcoffin_set:Nn #1#2
90 {
91     \coffin_if_exist_execute:Nn #1
92     {
93         \hbox_set:Nn #1
94         {
95             \group_begin:
96             \set@color
97             #2
98             \group_end:
99         }
100         \coffin_reset_structure:N #1
101         \coffin_update_poles:N #1
102         \coffin_update_corners:N #1
103     }
104 }
105 \cs_generate_variant:Nn \hcoffin_set:Nn { c }

```

(End definition for `\hcoffin_set:Nn` and `\hcoffin_set:cn`. These functions are documented on page 1.)

`\vcoffin_set:Nnn` Setting vertical coffins is more complex. First, the material is typeset with a given width.
`\vcoffin_set:cnm` The default handles and poles are set as for a horizontal coffin, before finding the top baseline using a temporary box.

```

106 \cs_new_protected:Npn \vcoffin_set:Nnn #1#2#3
107 {
108   \coffin_if_exist_execute:Nn #1
109   {
110     \vbox_set:Nn #1
111     {
112       \dim_set:Nn \tex_hsize:D {#2}
113       \group_begin:
114       \set@color
115       #3
116       \group_end:
117     }
118     \coffin_reset_structure:N #1
119     \coffin_update_poles:N #1
120     \coffin_update_corners:N #1
121     \vbox_set_top:Nn \l_coffin_tmp_box { \vbox_unpack:N #1 }
122     \dim_set:Nn \l_coffin_tmp_dim
123     { \box_ht:N #1 - \box_ht:N \l_coffin_tmp_box }
124     \coffin_set_pole:Nnx #1 { T }
125     { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } { 1000 pt } { 0 pt } }
126   }
127 }
128 \cs_generate_variant:Nn \vcoffin_set:Nnn { c }

```

(End definition for `\vcoffin_set:Nnn` and `\vcoffin_set:cnm`. These functions are documented on page 2.)

`\coffin_set_eq:NN` Setting two coffins equal is just a wrapper around other functions.

```

\coffin_set_eq:Nc
\coffin_set_eq:cN
\coffin_set_eq:cc
129 \cs_new_protected_nopar:Npn \coffin_set_eq:NN #1#2
130 {
131   \coffin_if_exist_execute:Nn #1
132   {
133     \box_set_eq:NN #1 #2
134     \coffin_set_eq_structure:NN #1 #2
135   }
136 }
137 \cs_generate_variant:Nn \coffin_set_eq:NN { c , Nc , cc }

```

(End definition for `\coffin_set_eq:NN` and others. These functions are documented on page 1.)

`\c_empty_coffin` Special coffins: these cannot be set up earlier as they need `\coffin_new:N`. The empty coffin is set as a box as the full coffin-setting system needs some material which is not yet available.
`\l_coffin_aligned_coffin`
`\l_coffin_aligned_internal_coffin`


```

138 \coffin_new:N \c_empty_coffin
139 \hbox_set:Nn \c_empty_coffin { }
140 \coffin_new:N \l_coffin_aligned_coffin
141 \coffin_new:N \l_coffin_aligned_internal_coffin

```

2.3 Coffins: handle and pole management

`\coffin_get_pole:NnN` A simple wrapper around the recovery of a coffin pole, with some error checking and recovery built-in.

```

142 \cs_new_protected_nopar:Npn \coffin_get_pole:NnN #1#2#3
143 {
144   \prop_get:cnN { l_coffin_poles_ \int_value:w #1 _prop } {#2} #3
145   \quark_if_no_value:NT #3
146   {
147     \msg_kernel_error:nxxx { coffin } { unknown-coffin-pole }
148     {#2} { \token_to_str:N #1 }
149     \tl_set:Nn #3 { { 0 pt } { 0 pt } { 0 pt } { 0 pt } }
150   }
151 }

```

(End definition for \coffin_get_pole:NnN. This function is documented on page ??.)

`\coffin_reset_structure:N` Resetting the structure is a simple copy job.

```

152 \cs_new_protected_nopar:Npn \coffin_reset_structure:N #1
153 {
154   \prop_set_eq:cn { l_coffin_corners_ \int_value:w #1 _prop }
155   \c_coffin_corners_prop
156   \prop_set_eq:cn { l_coffin_poles_ \int_value:w #1 _prop }
157   \c_coffin_poles_prop
158 }

```

(End definition for \coffin_reset_structure:N. This function is documented on page ??.)

`\coffin_set_eq_structure:NN` Setting coffin structures equal simply means copying the property list.

```

159 \cs_new_protected_nopar:Npn \coffin_set_eq_structure:NN #1#2
160 {
161   \prop_set_eq:cc { l_coffin_corners_ \int_value:w #1 _prop }
162   { l_coffin_corners_ \int_value:w #2 _prop }
163   \prop_set_eq:cc { l_coffin_poles_ \int_value:w #1 _prop }
164   { l_coffin_poles_ \int_value:w #2 _prop }
165 }

```

(End definition for \coffin_set_eq_structure:NN. This function is documented on page ??.)

`\coffin_set_user_dimensions:N` These make design-level names for the dimensions of a coffin easy to get at.

`\coffin_end_user_dimensions:`

- `\Depth`
- `\Height`
- `\TotalHeight`
- `\Width`

```

166 \cs_new_protected_nopar:Npn \coffin_set_user_dimensions:N #1
167 {
168   \cs_set_eq:NN \coffin_saved_Height: \Height
169   \cs_set_eq:NN \coffin_saved_Depth: \Depth
170   \cs_set_eq:NN \coffin_saved_TotalHeight: \TotalHeight
171   \cs_set_eq:NN \coffin_saved_Width: \Width
172   \cs_set_eq:NN \Height \l_coffin_Height_dim
173   \cs_set_eq:NN \Depth \l_coffin_Depth_dim
174   \cs_set_eq:NN \TotalHeight \l_coffin_TotalHeight_dim
175   \cs_set_eq:NN \Width \l_coffin_Width_dim
176   \dim_set:Nn \Height { \box_ht:N #1 }
177   \dim_set:Nn \Depth { \box_dp:N #1 }
178   \dim_set:Nn \TotalHeight { \box_ht:N #1 - \box_dp:N #1 }
179   \dim_set:Nn \Width { \box_wd:N #1 }
180 }
181 \cs_new_protected_nopar:Npn \coffin_end_user_dimensions:
182 {
183   \cs_set_eq:NN \Height \coffin_saved_Height:
184   \cs_set_eq:NN \Depth \coffin_saved_Depth:
185   \cs_set_eq:NN \TotalHeight \coffin_saved_TotalHeight:
186   \cs_set_eq:NN \Width \coffin_saved_Width:
187 }

```

(End definition for `\coffin_set_user_dimensions:N`. This function is documented on page ??.)

`\coffin_set_horizontal_pole:Nnn`
`\coffin_set_horizontal_pole:cnm`
`\coffin_set_vertical_pole:Nnn`
`\coffin_set_vertical_pole:cnm`
`\coffin_set_pole:Nnn`
`\coffin_set_pole:Nnx`

Setting the pole of a coffin at the user/designer level requires a bit more care. The idea here is to provide a reasonable interface to the system, then to do the setting with full expansion. The three-argument version is used internally to do a direct setting.

```

188 \cs_new_protected_nopar:Npn \coffin_set_horizontal_pole:Nnn #1#2#3
189 {
190   \coffin_if_exist_execute:Nn #1
191   {
192     \coffin_set_user_dimensions:N #1
193     \coffin_set_pole:Nnx #1 {#2}
194     {
195       { 0 pt } { \dim_eval:n {#3} }
196       { 1000 pt } { 0 pt }
197     }
198     \coffin_end_user_dimensions:
199   }
200 }
201 \cs_new_protected_nopar:Npn \coffin_set_vertical_pole:Nnn #1#2#3
202 {
203   \coffin_if_exist_execute:Nn #1
204   {
205     \coffin_set_user_dimensions:N #1
206     \coffin_set_pole:Nnx #1 {#2}
207     {
208       { \dim_eval:n {#3} } { 0 pt }

```

```

209         { 0 pt } { 1000 pt }
210     }
211     \coffin_end_user_dimensions:
212 }
213 }
214 \cs_new_protected_nopar:Npn \coffin_set_pole:Nnn #1#2#3
215 { \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _prop } {#2} {#3} }
216 \cs_generate_variant:Nn \coffin_set_horizontal_pole:Nnn { c }
217 \cs_generate_variant:Nn \coffin_set_vertical_pole:Nnn { c }
218 \cs_generate_variant:Nn \coffin_set_pole:Nnn { Nnx }

```

(End definition for `\coffin_set_horizontal_pole:Nnn` and `\coffin_set_horizontal_pole:cnx`. These functions are documented on page ??.)

`\coffin_update_corners:N` Updating the corners of a coffin is straight-forward as at this stage there can be no rotation. So the corners of the content are just those of the underlying TeX box.

```

219 \cs_new_protected_nopar:Npn \coffin_update_corners:N #1
220 {
221     \prop_put:cnx { l_coffin_corners_ \int_value:w #1 _prop } { tl }
222     { { 0 pt } { \dim_use:N \box_ht:N #1 } }
223     \prop_put:cnx { l_coffin_corners_ \int_value:w #1 _prop } { tr }
224     { { \dim_use:N \box_wd:N #1 } { \dim_use:N \box_ht:N #1 } }
225     \dim_set:Nn \l_coffin_tmp_dim { - \dim_use:N \box_dp:N #1 }
226     \prop_put:cnx { l_coffin_corners_ \int_value:w #1 _prop } { bl }
227     { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } }
228     \prop_put:cnx { l_coffin_corners_ \int_value:w #1 _prop } { br }
229     { { \dim_use:N \box_wd:N #1 } { \dim_use:N \l_coffin_tmp_dim } }
230 }

```

(End definition for `\coffin_update_corners:N`. This function is documented on page ??.)

`\coffin_update_poles:N` This function is called when a coffin is set, and updates the poles to reflect the nature of size of the box. Thus this function only alters poles where the default position is dependent on the size of the box. It also does not set poles which are relevant only to vertical coffins.

```

231 \cs_new_protected_nopar:Npn \coffin_update_poles:N #1
232 {
233     \dim_set:Nn \l_coffin_tmp_dim { 0.5 \box_wd:N #1 }
234     \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _prop } { hc }
235     { { \dim_use:N \l_coffin_tmp_dim } { 0 pt } { 0 pt } { 1000 pt } }
236     \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _prop } { r }
237     { { \dim_use:N \box_wd:N #1 } { 0 pt } { 0 pt } { 1000 pt } }
238     \dim_set:Nn \l_coffin_tmp_dim { ( \box_ht:N #1 - \box_dp:N #1 ) / 2 }
239     \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _prop } { vc }
240     { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } { 1000 pt } { 0 pt } }
241     \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _prop } { t }
242     { { 0 pt } { \dim_use:N \box_ht:N #1 } { 1000 pt } { 0 pt } }
243     \dim_set:Nn \l_coffin_tmp_dim { \box_dp:N #1 }

```

```

244 \dim_compare:nNnF { \l_coffin_tmp_dim } = { \c_zero_dim }
245 { \dim_set:Nn \l_coffin_tmp_dim { -\l_coffin_tmp_dim } }
246 \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _prop } { b }
247 { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } { 1000 pt } { 0 pt } }
248 }

```

(End definition for `\coffin_update_poles:N`. This function is documented on page ??.)

2.4 Coffins: calculation of pole intersections

```

\coffin_calculate_intersection:Nnn
\coffin_calculate_intersection:nnnnnnnn
\coffin_calculate_intersection_aux:nnnnnn

```

The lead off in finding intersections is to recover the two poles and then hand off to the auxiliary for the actual calculation. There may of course not be an intersection, for which an error trap is needed.

```

249 \cs_new_protected_nopar:Npn \coffin_calculate_intersection:Nnn #1#2#3
250 {
251   \coffin_get_pole:NnN #1 {#2} \l_coffin_pole_a_tl
252   \coffin_get_pole:NnN #1 {#3} \l_coffin_pole_b_tl
253   \bool_set_false:N \l_coffin_error_bool
254   \exp_last_two_unbraced:Noo
255   \coffin_calculate_intersection:nnnnnnnn
256   \l_coffin_pole_a_tl \l_coffin_pole_b_tl
257   \bool_if:NT \l_coffin_error_bool
258   {
259     \msg_kernel_error:nn { coffin } { no-pole-intersection }
260     \dim_zero:N \l_coffin_x_dim
261     \dim_zero:N \l_coffin_y_dim
262   }
263 }

```

The two poles passed here each have four values (as dimensions), (a, b, c, d) and (a', b', c', d') . These are arguments 1–4 and 5–8, respectively. In both cases a and b are the co-ordinates of a point on the pole and c and d define the direction of the pole. Finding the intersection depends on the directions of the poles, which are given by d/c and d'/c' . However, if one of the poles is either horizontal or vertical then one or more of c , d , c' and d' will be zero and a special case is needed.

```

264 \cs_new_protected_nopar:Npn \coffin_calculate_intersection:nnnnnnnn
265   #1#2#3#4#5#6#7#8
266 {
267   \dim_compare:nNnTF {#3} = { \c_zero_dim }

```

The case where the first pole is vertical. So the x -component of the interaction will be at a . There is then a test on the second pole: if it is also vertical then there is an error.

```

268 {
269   \dim_set:Nn \l_coffin_x_dim {#1}
270   \dim_compare:nNnTF {#7} = { \c_zero_dim
271     { \bool_set_true:N \l_coffin_error_bool }

```

The second pole may still be horizontal, in which case the y -component of the intersection will be b' . If not,

$$y = \frac{d'}{c'} (x - a') + b'$$

with the x -component already known to be #1. This calculation is done as a generalised auxiliary.

```

272         {
273         \dim_compare:nNnTF {#8} = \c_zero_dim
274         { \dim_set:Nn \l_coffin_y_dim {#6} }
275         {
276         \coffin_calculate_intersection_aux:nnnnnN
277         {#1} {#5} {#6} {#7} {#8} \l_coffin_y_dim
278         }
279     }
280 }
```

If the first pole is not vertical then it may be horizontal. If so, then the procedure is essentially the same as that already done but with the x - and y -components interchanged.

```

281     {
282     \dim_compare:nNnTF {#4} = \c_zero_dim
283     {
284     \dim_set:Nn \l_coffin_y_dim {#2}
285     \dim_compare:nNnTF {#8} = { \c_zero_dim }
286     { \bool_set_true:N \l_coffin_error_bool }
287     {
288     \dim_compare:nNnTF {#7} = \c_zero_dim
289     { \dim_set:Nn \l_coffin_x_dim {#5} }

```

The formula for the case where the second pole is neither horizontal nor vertical is

$$x = \frac{c'}{d'} (y - b') + a'$$

which is again handled by the same auxiliary.

```

290         {
291         \coffin_calculate_intersection_aux:nnnnnN
292         {#2} {#6} {#5} {#8} {#7} \l_coffin_x_dim
293         }
294     }
295 }
```

The first pole is neither horizontal nor vertical. This still leaves the second pole, which may be a special case. For those possibilities, the calculations are the same as above with the first and second poles interchanged.

```

296     {
297     \dim_compare:nNnTF {#7} = \c_zero_dim

```

```

298     {
299         \dim_set:Nn \l_coffin_x_dim {#5}
300         \coffin_calculate_intersection_aux:nnnnnN
301         {#5} {#1} {#2} {#3} {#4} \l_coffin_y_dim
302     }
303     {
304         \dim_compare:nNnTF {#8} = \c_zero_dim
305         {
306             \dim_set:Nn \l_coffin_x_dim {#6}
307             \coffin_calculate_intersection_aux:nnnnnN
308             {#6} {#2} {#1} {#4} {#3} \l_coffin_x_dim
309         }

```

If none of the special cases apply then there is still a need to check that there is a unique intersection between the two pole. This is the case if they have different slopes.

```

310     {
311         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#3}
312         \fp_set_from_dim:Nn \l_coffin_calc_b_fp {#4}
313         \fp_set_from_dim:Nn \l_coffin_calc_c_fp {#7}
314         \fp_set_from_dim:Nn \l_coffin_calc_d_fp {#8}
315         \fp_div:Nn \l_coffin_calc_b_fp \l_coffin_calc_a_fp
316         \fp_div:Nn \l_coffin_calc_d_fp \l_coffin_calc_c_fp
317         \fp_compare:nNnTF
318         \l_coffin_calc_b_fp = \l_coffin_calc_d_fp
319         { \bool_set_true:N \l_coffin_error_bool }

```

All of the tests pass, so there is the full complexity of the calculation:

$$x = \frac{a(d/c) - a'(d'/c') - b + b'}{(d/c) - (d'/c')}$$

and noting that the two ratios are already worked out from the test just performed. There is quite a bit of shuffling from dimensions to floating points in order to do the work. The y -values is then worked out using the standard auxiliary starting from the x -position.

```

320     {
321         \fp_set_from_dim:Nn \l_coffin_calc_result_fp {#6}
322         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#2}
323         \fp_sub:Nn \l_coffin_calc_result_fp
324         { \l_coffin_calc_a_fp }
325         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#1}
326         \fp_mul:Nn \l_coffin_calc_a_fp
327         { \l_coffin_calc_b_fp }
328         \fp_add:Nn \l_coffin_calc_result_fp
329         { \l_coffin_calc_a_fp }
330         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#5}
331         \fp_mul:Nn \l_coffin_calc_a_fp
332         { \l_coffin_calc_d_fp }

```

```

333         \fp_sub:Nn \l_coffin_calc_result_fp
334         { \l_coffin_calc_a_fp }
335         \fp_sub:Nn \l_coffin_calc_b_fp
336         { \l_coffin_calc_d_fp }
337         \fp_div:Nn \l_coffin_calc_result_fp
338         { \l_coffin_calc_b_fp }
339         \dim_set:Nn \l_coffin_x_dim
340         { \fp_to_dim:N \l_coffin_calc_result_fp }
341         \coffin_calculate_intersection_aux:nnnnnN
342         { \l_coffin_x_dim }
343         {#5} {#6} {#8} {#7} \l_coffin_y_dim
344     }
345 }
346 }
347 }
348 }
349 }

```

The formula for finding the intersection point is in most cases the same. The formula here is

$$\#6 = \frac{\#5}{\#4} (\#1 - \#2) + \#3$$

Thus #4 and #5 should be the directions of the pole while #2 and #3 are co-ordinates.

```

350 \cs_new_protected_nopar:Npn \coffin_calculate_intersection_aux:nnnnnN
351   #1#2#3#4#5#6
352   {
353     \fp_set_from_dim:Nn \l_coffin_calc_result_fp {#1}
354     \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#2}
355     \fp_set_from_dim:Nn \l_coffin_calc_b_fp {#3}
356     \fp_set_from_dim:Nn \l_coffin_calc_c_fp {#4}
357     \fp_set_from_dim:Nn \l_coffin_calc_d_fp {#5}
358     \fp_sub:Nn \l_coffin_calc_result_fp { \l_coffin_calc_a_fp }
359     \fp_div:Nn \l_coffin_calc_result_fp { \l_coffin_calc_d_fp }
360     \fp_mul:Nn \l_coffin_calc_result_fp { \l_coffin_calc_c_fp }
361     \fp_add:Nn \l_coffin_calc_result_fp { \l_coffin_calc_b_fp }
362     \dim_set:Nn #6 { \fp_to_dim:N \l_coffin_calc_result_fp }
363   }

```

(End definition for \coffin_calculate_intersection:Nnn. This function is documented on page ??.)

2.5 Aligning and typesetting of coffins

`\coffin_join:NnnNnnnn`
`\coffin_join:cnnNnnnn`
`\coffin_join:Nnncnnnn`
`\coffin_join:cnncnnnn`

This command joins two coffins, using a horizontal and vertical pole from each coffin and making an offset between the two. The result is stored as the as a third coffin, which will have all of its handles reset to standard values. First, the more basic alignment function is used to get things started.

```

364 \cs_new_protected_nopar:Npn \coffin_join:NnnNnnnn #1#2#3#4#5#6#7#8

```

```

365 {
366   \coffin_align:NnnNnnnnN
367   #1 {#2} {#3} #4 {#5} {#6} {#7} {#8} \l_coffin_aligned_coffin

```

Correct the placement of the reference point. If the x -offset is negative then the reference point of the second box is to the left of that of the first, which is corrected using a kern. On the right side the first box might stick out, which will show up if it is wider than the sum of the x -offset and the width of the second box. So a second kern may be needed.

```

368   \hbox_set:Nn \l_coffin_aligned_coffin
369   {
370     \dim_compare:nNnT { \l_coffin_offset_x_dim } < \c_zero_dim
371     { \tex_kern:D -\l_coffin_offset_x_dim }
372     \hbox_unpack:N \l_coffin_aligned_coffin
373     \dim_set:Nn \l_coffin_tmp_dim
374     { \l_coffin_offset_x_dim - \box_wd:N #1 + \box_wd:N #4 }
375     \dim_compare:nNnT \l_coffin_tmp_dim < \c_zero_dim
376     { \tex_kern:D -\l_coffin_tmp_dim }
377   }

```

The coffin structure is reset, and the corners are cleared: only those from the two parent coffins are needed.

```

378   \coffin_reset_structure:N \l_coffin_aligned_coffin
379   \prop_clear:c
380   { \l_coffin_corners_ \int_value:w \l_coffin_aligned_coffin _ prop }
381   \coffin_update_poles:N \l_coffin_aligned_coffin

```

The structures of the parent coffins are now transferred to the new coffin, which requires that the appropriate offsets are applied. That will then depend on whether any shift was needed.

```

382   \dim_compare:nNnTF \l_coffin_offset_x_dim < \c_zero_dim
383   {
384     \coffin_offset_poles:Nnn #1 { -\l_coffin_offset_x_dim } { 0 pt }
385     \coffin_offset_poles:Nnn #4 { 0 pt } { \l_coffin_offset_y_dim }
386     \coffin_offset_corners:Nnn #1 { -\l_coffin_offset_x_dim } { 0 pt }
387     \coffin_offset_corners:Nnn #4 { 0 pt } { \l_coffin_offset_y_dim }
388   }
389   {
390     \coffin_offset_poles:Nnn #1 { 0 pt } { 0 pt }
391     \coffin_offset_poles:Nnn #4
392     { \l_coffin_offset_x_dim } { \l_coffin_offset_y_dim }
393     \coffin_offset_corners:Nnn #1 { 0 pt } { 0 pt }
394     \coffin_offset_corners:Nnn #4
395     { \l_coffin_offset_x_dim } { \l_coffin_offset_y_dim }
396   }
397   \coffin_update_vertical_poles:NNN #1 #4 \l_coffin_aligned_coffin
398   \coffin_set_eq:NN #1 \l_coffin_aligned_coffin
399 }
400 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { c , Nnnc , cnc }

```


(End definition for `\coffin_join:NnnNnnnn` and others. These functions are documented on page 3.)

`\coffin_attach:NnnNnnnn` A more simple version of the above, as it simply uses the size of the first coffin for the new one. This means that the work here is rather simplified compared to the above code.
`\coffin_attach:cnNnnnnn` The function used when marking a position is hear also as it is similar but without the structure updates.
`\coffin_attach:Nnncnnnn`
`\coffin_attach_mark:NnnNnnnn`

```

401 \cs_new_protected_nopar:Npn \coffin_attach:NnnNnnnn #1#2#3#4#5#6#7#8
402 {
403   \coffin_align:NnnNnnnnN
404   #1 {#2} {#3} #4 {#5} {#6} {#7} {#8} \l_coffin_aligned_coffin
405   \box_set_ht:Nn \l_coffin_aligned_coffin { \box_ht:N #1 }
406   \box_set_dp:Nn \l_coffin_aligned_coffin { \box_dp:N #1 }
407   \box_set_wd:Nn \l_coffin_aligned_coffin { \box_wd:N #1 }
408   \coffin_reset_structure:N \l_coffin_aligned_coffin
409   \prop_set_eq:cc
410   { \l_coffin_corners_ \int_value:w \l_coffin_aligned_coffin _prop }
411   { \l_coffin_corners_ \int_value:w #1 _prop }
412   \coffin_update_poles:N \l_coffin_aligned_coffin
413   \coffin_offset_poles:Nnn #1 { 0 pt } { 0 pt }
414   \coffin_offset_poles:Nnn #4
415   { \l_coffin_offset_x_dim } { \l_coffin_offset_y_dim }
416   \coffin_update_vertical_poles:NNN #1 #4 \l_coffin_aligned_coffin
417   \coffin_set_eq:NN #1 \l_coffin_aligned_coffin
418 }
419 \cs_new_protected_nopar:Npn \coffin_attach_mark:NnnNnnnn #1#2#3#4#5#6#7#8
420 {
421   \coffin_align:NnnNnnnnN
422   #1 {#2} {#3} #4 {#5} {#6} {#7} {#8} \l_coffin_aligned_coffin
423   \box_set_ht:Nn \l_coffin_aligned_coffin { \box_ht:N #1 }
424   \box_set_dp:Nn \l_coffin_aligned_coffin { \box_dp:N #1 }
425   \box_set_wd:Nn \l_coffin_aligned_coffin { \box_wd:N #1 }
426   \box_set_eq:NN #1 \l_coffin_aligned_coffin
427 }
428 \cs_generate_variant:Nn \coffin_attach:NnnNnnnn { c , Nnnc , cnnc }

```

(End definition for `\coffin_attach:NnnNnnnn` and others. These functions are documented on page ??.)

`\coffin_align:NnnNnnnnN` The internal function aligns the two coffins into a third one, but performs no corrections on the resulting coffin poles. The process begins by finding the points of intersection for the poles for each of the input coffins. Those for the first coffin are worked out after those for the second coffin, as this allows the ‘primed’ storage area to be used for the second coffin. The ‘real’ box offsets are then calculated, before using these to re-box the input coffins. The default poles are then set up, but the final result will depend on how the bounding box is being handled.

```

429 \cs_new_protected_nopar:Npn \coffin_align:NnnNnnnnN #1#2#3#4#5#6#7#8#9
430 {
431   \coffin_calculate_intersection:Nnn #4 {#5} {#6}

```

```

432 \dim_set:Nn \l_coffin_x_prime_dim { \l_coffin_x_dim }
433 \dim_set:Nn \l_coffin_y_prime_dim { \l_coffin_y_dim }
434 \coffin_calculate_intersection:Nnn #1 {#2} {#3}
435 \dim_set:Nn \l_coffin_offset_x_dim
436 { \l_coffin_x_dim - \l_coffin_x_prime_dim + #7 }
437 \dim_set:Nn \l_coffin_offset_y_dim
438 { \l_coffin_y_dim - \l_coffin_y_prime_dim + #8 }
439 \hbox_set:Nn \l_coffin_aligned_internal_coffin
440 {
441   \box_use:N #1
442   \tex_kern:D -\box_wd:N #1
443   \tex_kern:D \l_coffin_offset_x_dim
444   \box_move_up:nn { \l_coffin_offset_y_dim } { \box_use:N #4 }
445 }
446 \coffin_set_eq:NN #9 \l_coffin_aligned_internal_coffin
447 }

```

(End definition for `\coffin_align:NnnNnnnnN`. This function is documented on page ??.)

`\coffin_offset_poles:Nnn`
`\coffin_offset_pole:Nnnnnnn`

Transferring structures from one coffin to another requires that the positions are updated by the offset between the two coffins. This is done by mapping to the property list of the source coffins, moving as appropriate and saving to the new coffin data structures. The test for a - means that the structures from the parent coffins are uniquely labelled and do not depend on the order of alignment. The pay off for this is that - should not be used in coffin pole or handle names, and that multiple alignments do not result in a whole set of values.

```

448 \cs_new_protected_nopar:Npn \coffin_offset_poles:Nnn #1#2#3
449 {
450   \prop_map_inline:cn { l_coffin_poles_ \int_value:w #1 _prop }
451   { \coffin_offset_pole:Nnnnnnn #1 {##1} ##2 {#2} {#3} }
452 }
453 \cs_new_protected_nopar:Npn \coffin_offset_pole:Nnnnnnn #1#2#3#4#5#6#7#8
454 {
455   \dim_set:Nn \l_coffin_x_dim { #3 + #7 }
456   \dim_set:Nn \l_coffin_y_dim { #4 + #8 }
457   \tl_if_in:nnTF {#2} { - }
458   { \tl_set:Nn \l_coffin_tmp_tl { {#2} } }
459   { \tl_set:Nn \l_coffin_tmp_tl { { #1 - #2 } } }
460   \exp_last_unbraced:NNo \coffin_set_pole:Nnx \l_coffin_aligned_coffin
461   { \l_coffin_tmp_tl }
462   {
463     { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim }
464     {#5} {#6}
465   }
466 }

```

(End definition for `\coffin_offset_poles:Nnn`. This function is documented on page ??.)

`\coffin_offset_corners:Nnn` Saving the offset corners of a coffin is very similar, except that there is no need to worry
`\coffin_offset_corners:Nnnnn` about naming: every corner can be saved here as order is unimportant.

```

467 \cs_new_protected_nopar:Npn \coffin_offset_corners:Nnn #1#2#3
468 {
469   \prop_map_inline:cn { l_coffin_corners_ \int_value:w #1 _prop }
470   { \coffin_offset_corner:Nnnnn #1 {##1} ##2 {#2} {#3} }
471 }
472 \cs_new_protected_nopar:Npn \coffin_offset_corner:Nnnnn #1#2#3#4#5#6
473 {
474   \prop_put:cnx
475   { l_coffin_corners_ \int_value:w \l_coffin_aligned_coffin _prop }
476   { #1 - #2 }
477   {
478     { \dim_eval:n { #3 + #5 } }
479     { \dim_eval:n { #4 + #6 } }
480   }
481 }

```

(End definition for \coffin_offset_corners:Nnn. This function is documented on page ??.)

`\coffin_update_vertical_poles:NNN` The T and B poles will need to be recalculated after alignment. These functions find the
`\coffin_update_T:nnnnnnnnN` larger absolute value for the poles, but this is of course only logical when the poles are
`\coffin_update_B:nnnnnnnnN` horizontal.

```

482 \cs_new_protected_nopar:Npn \coffin_update_vertical_poles:NNN #1#2#3
483 {
484   \coffin_get_pole:NnN #3 { #1 -T } \l_coffin_pole_a_tl
485   \coffin_get_pole:NnN #3 { #2 -T } \l_coffin_pole_b_tl
486   \exp_last_two_unbraced:Noo \coffin_update_T:nnnnnnnnN
487   \l_coffin_pole_a_tl \l_coffin_pole_b_tl #3
488   \coffin_get_pole:NnN #3 { #1 -B } \l_coffin_pole_a_tl
489   \coffin_get_pole:NnN #3 { #2 -B } \l_coffin_pole_b_tl
490   \exp_last_two_unbraced:Noo \coffin_update_B:nnnnnnnnN
491   \l_coffin_pole_a_tl \l_coffin_pole_b_tl #3
492 }
493 \cs_new_protected_nopar:Npn \coffin_update_T:nnnnnnnnN #1#2#3#4#5#6#7#8#9
494 {
495   \dim_compare:nNnTF {#2} < {#6}
496   {
497     \coffin_set_pole:Nnx #9 { T }
498     { { 0 pt } {#6} { 1000 pt } { 0 pt } }
499   }
500   {
501     \coffin_set_pole:Nnx #9 { T }
502     { { 0 pt } {#2} { 1000 pt } { 0 pt } }
503   }
504 }
505 \cs_new_protected_nopar:Npn \coffin_update_B:nnnnnnnnN #1#2#3#4#5#6#7#8#9
506 {

```

```

507 \dim_compare:nNnTF {#2} < {#6}
508 {
509   \coffin_set_pole:Nnx #9 { B }
510   { { 0 pt } {#2} { 1000 pt } { 0 pt } }
511 }
512 {
513   \coffin_set_pole:Nnx #9 { B }
514   { { 0 pt } {#6} { 1000 pt } { 0 pt } }
515 }
516 }

```

(End definition for `\coffin_update_vertical_poles:NMN`. This function is documented on page ??.)

`\coffin_typeset:Nnnnn` Typesetting a coffin means aligning it with the current position, which is done using a coffin with no content at all. This is done using the same approach as `\coffin_align:NnnNnnnnN` but without the offset corrections (which would be thrown away). The same is true for overlaying coffins, which uses the known size of an empty box!

```

517 \cs_new_protected_nopar:Npn \coffin_typeset:Nnnnn #1#2#3#4#5
518 {
519   \coffin_align:NnnNnnnnN \c_empty_coffin { H } { 1 }
520   #1 {#2} {#3} {#4} {#5} \l_coffin_aligned_coffin
521   \hbox_set:Nn \l_coffin_aligned_coffin
522   {
523     \dim_compare:nNnT { \l_coffin_offset_x_dim } < \c_zero_dim
524     { \tex_kern:D -\l_coffin_offset_x_dim }
525     \hbox_unpack:N \l_coffin_aligned_coffin
526     \dim_set:Nn \l_coffin_tmp_dim
527     { \l_coffin_offset_x_dim + \box_wd:N #1 }
528     \dim_compare:nNnT { \l_coffin_tmp_dim } < \c_zero_dim
529     { \tex_kern:D -\l_coffin_tmp_dim }
530   }
531   \hbox_unpack:N \c_empty_box
532   \box_use:N \l_coffin_aligned_coffin
533 }
534 \cs_generate_variant:Nn \coffin_typeset:Nnnnn { c }

```

(End definition for `\coffin_typeset:Nnnnn` and `\coffin_typeset:cnnnn`. These functions are documented on page 3.)

2.6 Rotating coffins

`\l_coffin_bounding_prop` A property list for the bounding box of a coffin. This is only needed during the rotation, so there is just the one.

```

535 \prop_new:N \l_coffin_bounding_prop

```

`\l_coffin_bounding_shift_dim` The shift of the bounding box of a coffin from the real content.

```

536 \dim_new:N \l_coffin_bounding_shift_dim

```

`\l_coffin_left_corner_dim` These are used to hold maxima for the various corner values: these thus define the
`\l_coffin_right_corner_dim` minimum size of the bounding box after rotation.
`\l_coffin_bottom_corner_dim`
`\l_coffin_top_corner_dim`

```

537 \dim_new:N \l_coffin_left_corner_dim
538 \dim_new:N \l_coffin_right_corner_dim
539 \dim_new:N \l_coffin_bottom_corner_dim
540 \dim_new:N \l_coffin_top_corner_dim

```

`\coffin_rotate:Nn` Rotating a coffin requires several steps which can be conveniently run together. The
`\coffin_rotate:cn` first step is to convert the angle given in degrees to one in radians. This is then used
to set `\l_coffin_sin_fp` and `\l_coffin_cos_fp`, which are carried through unchanged
for the rest of the procedure.

```

541 \cs_new_protected_nopar:Npn \coffin_rotate:Nn #1#2
542 {
543   \fp_set:Nn \l_coffin_tmp_fp {#2}
544   \fp_div:Nn \l_coffin_tmp_fp { 180 }
545   \fp_mul:Nn \l_coffin_tmp_fp { \c_pi_fp }
546   \fp_sin:Nn \l_coffin_sin_fp { \l_coffin_tmp_fp }
547   \fp_cos:Nn \l_coffin_cos_fp { \l_coffin_tmp_fp }

```

The corners and poles of the coffin can now be rotated around the origin. This is best
achieved using mapping functions.

```

548 \prop_map_inline:cn { l_coffin_corners_ \int_value:w #1 _prop }
549 { \coffin_rotate_corner:Nnnn #1 {##1} ##2 }
550 \prop_map_inline:cn { l_coffin_poles_ \int_value:w #1 _prop }
551 { \coffin_rotate_pole:Nnnnnn #1 {##1} ##2 }

```

The bounding box of the coffin needs to be rotated, and to do this the corners have to be
found first. They are then rotated in the same way as the corners of the coffin material
itself.

```

552 \coffin_set_bounding:N #1
553 \prop_map_inline:Nn \l_coffin_bounding_prop
554 { \coffin_rotate_bounding:nnn {##1} ##2 }

```

At this stage, there needs to be a calculation to find where the corners of the content
and the box itself will end up.

```

555 \coffin_find_corner_maxima:N #1
556 \coffin_find_bounding_shift:
557 \hbox_set:Nn #1 { \rotatebox {#2} { \box_use:N #1 } }

```

The correction of the box position itself takes place here. The idea is that the bounding
box for a coffin is tight up to the content, and has the reference point at the bottom-left.
The x -direction is handled by moving the content by the difference in the positions of
the bounding box and the content left edge. The y -direction is dealt with by moving the
box down by any depth it has acquired.

```

558 \hbox_set:Nn #1
559 {
560   \tex_kern:D \l_coffin_bounding_shift_dim
561   \tex_kern:D -\l_coffin_left_corner_dim
562   \box_move_down:nn { \l_coffin_bottom_corner_dim }
563   { \box_use:N #1 }
564 }

```

If there have been any previous rotations then the size of the bounding box will be bigger than the contents. This can be corrected easily by setting the size of the box to the height and width of the content.

```

565 \box_set_ht:Nn #1
566 { \l_coffin_top_corner_dim - \l_coffin_bottom_corner_dim }
567 \box_set_dp:Nn #1 { 0 pt }
568 \box_set_wd:Nn #1
569 { \l_coffin_right_corner_dim - \l_coffin_left_corner_dim }

```

The final task is to move the poles and corners such that they are back in alignment with the box reference point.

```

570 \prop_map_inline:cn { l_coffin_corners_ \int_value:w #1 _prop }
571 { \coffin_shift_corner:Nnnn #1 {##1} ##2 }
572 \prop_map_inline:cn { l_coffin_poles_ \int_value:w #1 _prop }
573 { \coffin_shift_pole:Nnnnnn #1 {##1} ##2 }
574 }
575 \cs_generate_variant:Nn \coffin_rotate:Nn { c }

```

(End definition for `\coffin_rotate:Nn` and `\coffin_rotate:cn`. These functions are documented on page 2.)

`\coffin_set_bounding:N` The bounding box corners for a coffin are easy enough to find: this is the same code as for the corners of the material itself, but using a dedicated property list.

```

576 \cs_new_protected_nopar:Npn \coffin_set_bounding:N #1
577 {
578   \prop_put:Nnx \l_coffin_bounding_prop { tl }
579   { { 0 pt } { \dim_use:N \box_ht:N #1 } }
580   \prop_put:Nnx \l_coffin_bounding_prop { tr }
581   { { \dim_use:N \box_wd:N #1 } { \dim_use:N \box_ht:N #1 } }
582   \dim_set:Nn \l_coffin_tmp_dim { - \dim_use:N \box_dp:N #1 }
583   \prop_put:Nnx \l_coffin_bounding_prop { bl }
584   { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } }
585   \prop_put:Nnx \l_coffin_bounding_prop { br }
586   { { \dim_use:N \box_wd:N #1 } { \dim_use:N \l_coffin_tmp_dim } }
587 }

```

(End definition for `\coffin_set_bounding:N`. This function is documented on page ??.)

`\coffin_rotate_bounding:nnn` Rotating the position of the corner of the coffin is just a case of treating this as a vector
`\coffin_rotate_corner:Nnnn` from the reference point. The same treatment is used for the corners of the material itself and the bounding box.

```

588 \cs_new_protected_nopar:Npn \coffin_rotate_bounding:nnn #1#2#3
589 {
590   \coffin_rotate_vector:nnNN {#2} {#3} \l_coffin_x_dim \l_coffin_y_dim
591   \prop_put:Nnx \l_coffin_bounding_prop {#1}
592   { { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim } }
593 }
594 \cs_new_protected_nopar:Npn \coffin_rotate_corner:Nnnn #1#2#3#4
595 {
596   \coffin_rotate_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
597   \prop_put:cnx { \l_coffin_corners_ \int_value:w #1 _prop } {#2}
598   { { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim } }
599 }

```

(End definition for `\coffin_rotate_bounding:nnn`. This function is documented on page ??.)

`\coffin_rotate_pole:Nnnnnn` Rotating a single pole simply means shifting the co-ordinate of the pole and its direction. The rotation here is about the bottom-left corner of the coffin.

```

600 \cs_new_protected_nopar:Npn \coffin_rotate_pole:Nnnnnn #1#2#3#4#5#6
601 {
602   \coffin_rotate_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
603   \coffin_rotate_vector:nnNN {#5} {#6}
604   \l_coffin_x_prime_dim \l_coffin_y_prime_dim
605   \coffin_set_pole:Nnx #1 {#2}
606   {
607     { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim }
608     { \dim_use:N \l_coffin_x_prime_dim }
609     { \dim_use:N \l_coffin_y_prime_dim }
610   }
611 }

```

(End definition for `\coffin_rotate_pole:Nnnnnn`. This function is documented on page ??.)

`\coffin_rotate_vector:nnNN` A rotation function, which needs only an input vector (as dimensions) and an output space. The values `\l_coffin_cos_fp` and `\l_coffin_sin_fp` should previously have been set up correctly. Working this way means that the floating point work is kept to a minimum: for any given rotation the sin and cosine values do no change, after all.

```

612 \cs_new_protected_nopar:Npn \coffin_rotate_vector:nnNN #1#2#3#4
613 {
614   \fp_set_from_dim:Nn \l_coffin_x_fp {#1}
615   \fp_set_from_dim:Nn \l_coffin_y_fp {#2}
616   \fp_set_eq:NN \l_coffin_x_prime_fp \l_coffin_x_fp
617   \fp_set_eq:NN \l_coffin_y_prime_fp \l_coffin_y_fp
618   \fp_mul:Nn \l_coffin_x_prime_fp { \l_coffin_cos_fp }
619   \fp_mul:Nn \l_coffin_y_prime_fp { \l_coffin_sin_fp }
620   \fp_sub:Nn \l_coffin_x_prime_fp { \l_coffin_tmp_fp }
621   \fp_set_eq:NN \l_coffin_y_prime_fp \l_coffin_y_fp
622   \fp_set_eq:NN \l_coffin_tmp_fp \l_coffin_x_fp
623   \fp_mul:Nn \l_coffin_y_prime_fp { \l_coffin_cos_fp }

```

```

624 \fp_mul:Nn \l_coffin_tmp_fp { \l_coffin_sin_fp }
625 \fp_add:Nn \l_coffin_y_prime_fp { \l_coffin_tmp_fp }
626 \dim_set:Nn #3 { \fp_to_dim:N \l_coffin_x_prime_fp }
627 \dim_set:Nn #4 { \fp_to_dim:N \l_coffin_y_prime_fp }
628 }

```

(End definition for `\coffin_rotate_vector:nnNN`. This function is documented on page ??.)

`\coffin_find_corner_maxima:N`
`\coffin_find_corner_maxima_aux:nn`

The idea here is to find the extremities of the content of the coffin. This is done by looking for the smallest values for the bottom and left corners, and the largest values for the top and right corners. The values start at the maximum dimensions so that the case where all are positive or all are negative works out correctly.

```

629 \cs_new_protected_nopar:Npn \coffin_find_corner_maxima:N #1
630 {
631   \dim_set:Nn \l_coffin_top_corner_dim { -\c_max_dim }
632   \dim_set:Nn \l_coffin_right_corner_dim { -\c_max_dim }
633   \dim_set:Nn \l_coffin_bottom_corner_dim { \c_max_dim }
634   \dim_set:Nn \l_coffin_left_corner_dim { \c_max_dim }
635   \prop_map_inline:cn { l_coffin_corners_ \int_value:w #1 _prop }
636     { \coffin_find_corner_maxima_aux:nn ##2 }
637 }
638 \cs_new_protected_nopar:Npn \coffin_find_corner_maxima_aux:nn #1#2
639 {
640   \dim_set_min:Nn \l_coffin_left_corner_dim {#1}
641   \dim_set_max:Nn \l_coffin_right_corner_dim {#1}
642   \dim_set_min:Nn \l_coffin_bottom_corner_dim {#2}
643   \dim_set_max:Nn \l_coffin_top_corner_dim {#2}
644 }

```

(End definition for `\coffin_find_corner_maxima:N`. This function is documented on page ??.)

`\coffin_find_bounding_shift:`
`\coffin_find_bounding_shift_aux:nn`

The approach to finding the shift for the bounding box is similar to that for the corners. However, there is only one value needed here and a fixed input property list, so things are a bit clearer.

```

645 \cs_new_protected_nopar:Npn \coffin_find_bounding_shift:
646 {
647   \dim_set:Nn \l_coffin_bounding_shift_dim { \c_max_dim }
648   \prop_map_inline:Nn \l_coffin_bounding_prop
649     { \coffin_find_bounding_shift_aux:nn ##2 }
650 }
651 \cs_new_protected_nopar:Npn \coffin_find_bounding_shift_aux:nn #1#2
652 { \dim_set_min:Nn \l_coffin_bounding_shift_dim {#1} }

```

(End definition for `\coffin_find_bounding_shift:.` This function is documented on page ??.)

`\coffin_shift_corner:Nnnn`
`\coffin_shift_pole:Nnnnnn`

Shifting the corners and poles of a coffin means subtracting the appropriate values from the x - and y -components. For the poles, this means that the direction vector is unchanged.


```

653 \cs_new_protected_nopar:Npn \coffin_shift_corner:Nnnn #1#2#3#4
654 {
655   \prop_put:cnx { l_coffin_corners_ \int_value:w #1 _ prop } {#2}
656   {
657     { \dim_eval:n { #3 - \l_coffin_left_corner_dim } }
658     { \dim_eval:n { #4 - \l_coffin_bottom_corner_dim } }
659   }
660 }
661 \cs_new_protected_nopar:Npn \coffin_shift_pole:Nnnnnn #1#2#3#4#5#6
662 {
663   \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _ prop } {#2}
664   {
665     { \dim_eval:n { #3 - \l_coffin_left_corner_dim } }
666     { \dim_eval:n { #4 - \l_coffin_bottom_corner_dim } }
667     {#5} {#6}
668   }
669 }

```

(End definition for `\coffin_shift_corner:Nnnn`. This function is documented on page ??.)

2.7 Resizing coffins

`\l_coffin_scale_x_fp` Storage for the scaling factors in x and y , respectively.
`\l_coffin_scale_y_fp`

```

670 \fp_new:N \l_coffin_scale_x_fp
671 \fp_new:N \l_coffin_scale_y_fp

```

`\l_coffin_scaled_total_height_dim` When scaling, the values given have to be turned into absolute values.
`\l_coffin_scaled_width_dim`

```

672 \dim_new:N \l_coffin_scaled_total_height_dim
673 \dim_new:N \l_coffin_scaled_width_dim

```

`\coffin_resize:Nnn` Resizing a coffin begins by setting up the user-friendly names for the dimensions of the
`\coffin_resize:cnn` coffin box. The new sizes are then turned into scale factor. This is the same operation as takes place for the underlying box, but that operation is grouped and so the same calculation is done here.

```

674 \cs_new_protected_nopar:Npn \coffin_resize:Nnn #1#2#3
675 {
676   \coffin_set_user_dimensions:N #1
677   \fp_set_from_dim:Nn \l_coffin_scale_x_fp {#2}
678   \fp_set_from_dim:Nn \l_coffin_tmp_fp { \Width }
679   \fp_div:Nn \l_coffin_scale_x_fp { \l_coffin_tmp_fp }
680   \fp_set_from_dim:Nn \l_coffin_scale_y_fp {#3}
681   \fp_set_from_dim:Nn \l_coffin_tmp_fp { \TotalHeight }
682   \fp_div:Nn \l_coffin_scale_y_fp { \l_coffin_tmp_fp }
683   \hbox_set:Nn #1 { \resizebox * {#2} {#3} { \box_use:N #1 } }
684   \coffin_resize_common:Nnn #1 {#2} {#3}
685 }
686 \cs_generate_variant:Nn \coffin_resize:Nnn { c }

```

(End definition for `\coffin_resize:Nnn` and `\coffin_resize:cnn`. These functions are documented on page ??.)

`\coffin_resize_common:Nnn` The poles and corners of the coffin are scaled to the appropriate places before actually resizing the underlying box.

```

687 \cs_new_protected_nopar:Npn \coffin_resize_common:Nnn #1#2#3
688 {
689   \prop_map_inline:cn { l_coffin_corners_ \int_value:w #1 _prop }
690   { \coffin_scale_corner:Nnnn #1 {##1} ##2 }
691   \prop_map_inline:cn { l_coffin_poles_ \int_value:w #1 _prop }
692   { \coffin_scale_pole:Nnnnnn #1 {##1} ##2 }

```

Negative x -scaling values will place the poles in the wrong location: this is corrected here.

```

693   \fp_compare:NNNT \l_coffin_scale_x_fp < \c_zero_fp
694   {
695     \prop_map_inline:cn { l_coffin_corners_ \int_value:w #1 _prop }
696     { \coffin_x_shift_corner:Nnnn #1 {##1} ##2 }
697     \prop_map_inline:cn { l_coffin_poles_ \int_value:w #1 _prop }
698     { \coffin_x_shift_pole:Nnnnnn #1 {##1} ##2 }
699   }
700   \coffin_end_user_dimensions:
701 }

```

(End definition for `\coffin_resize_common:Nnn`. This function is documented on page ??.)

`\coffin_scale:Nnn` For scaling, the opposite calculation is done to find the new dimensions for the coffin.

`\coffin_scale:cnn` Only the total height is needed, as this is the shift required for corners and poles. The scaling is done the T_EX way as this works properly with floating point values without needing to use the `fp` module.

```

702 \cs_new_protected_nopar:Npn \coffin_scale:Nnn #1#2#3
703 {
704   \coffin_set_user_dimensions:N #1
705   \fp_set:Nn \l_coffin_scale_x_fp {#2}
706   \fp_set:Nn \l_coffin_scale_y_fp {#3}
707   \fp_compare:NNNTF \l_coffin_scale_y_fp > \c_zero_fp
708   { \l_coffin_scaled_total_height_dim #3 \TotalHeight }
709   { \l_coffin_scaled_total_height_dim -#3 \TotalHeight }
710   \fp_compare:NNNTF \l_coffin_scale_x_fp > \c_zero_fp
711   { \l_coffin_scaled_width_dim -#2 \Width }
712   { \l_coffin_scaled_width_dim #2 \Width }
713   \hbox_set:Nn #1 { \scalebox {#2} [#3] { \box_use:N #1 } }
714   \coffin_resize_common:Nnn #1
715   { \l_coffin_scaled_width_dim } { \l_coffin_scaled_total_height_dim }
716 }
717 \cs_generate_variant:Nn \coffin_scale:Nnn { c }

```

(End definition for `\coffin_scale:Nnn` and `\coffin_scale:cnm`. These functions are documented on page ??.)

`\coffin_scale_vector:nnNN` This function scales a vector from the origin using the pre-set scale factors in x and y . This is a much less complex operation than rotation, and as a result the code is a lot clearer.

```

718 \cs_new_protected_nopar:Npn \coffin_scale_vector:nnNN #1#2#3#4
719 {
720   \fp_set_from_dim:Nn \l_coffin_tmp_fp {#1}
721   \fp_mul:Nn \l_coffin_tmp_fp { \l_coffin_scale_x_fp }
722   \dim_set:Nn #3 { \fp_to_dim:N \l_coffin_tmp_fp }
723   \fp_set_from_dim:Nn \l_coffin_tmp_fp {#2}
724   \fp_mul:Nn \l_coffin_tmp_fp { \l_coffin_scale_y_fp }
725   \dim_set:Nn #4 { \fp_to_dim:N \l_coffin_tmp_fp }
726 }

```

(End definition for `\coffin_scale_vector:nnNN`. This function is documented on page ??.)

`\coffin_scale_corner:Nnnn` `\coffin_scale_pole:Nnnnnn` Scaling both corners and poles is a simple calculation using the preceding vector scaling.

```

727 \cs_new_protected_nopar:Npn \coffin_scale_corner:Nnnn #1#2#3#4
728 {
729   \coffin_scale_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
730   \prop_put:cnx { l_coffin_corners_ \int_value:w #1 _prop } {#2}
731   { { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim } }
732 }
733 \cs_new_protected_nopar:Npn \coffin_scale_pole:Nnnnnn #1#2#3#4#5#6
734 {
735   \coffin_scale_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
736   \coffin_set_pole:Nnx #1 {#2}
737   {
738     { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim }
739     {#5} {#6}
740   }
741 }

```

(End definition for `\coffin_scale_corner:Nnnn`. This function is documented on page ??.)

`\coffin_x_shift_corner:Nnnn` `\coffin_x_shift_pole:Nnnnnn` These functions correct for the x displacement that takes place with a negative horizontal scaling.

```

742 \cs_new_protected_nopar:Npn \coffin_x_shift_corner:Nnnn #1#2#3#4
743 {
744   \prop_put:cnx { l_coffin_corners_ \int_value:w #1 _prop } {#2}
745   {
746     { \dim_eval:n { #3 + \box_wd:N #1 } } {#4}
747   }
748 }
749 \cs_new_protected_nopar:Npn \coffin_x_shift_pole:Nnnnnn #1#2#3#4#5#6

```

```

750 {
751   \prop_put:cnx { l_coffin_poles_ \int_value:w #1 _prop } {#2}
752   {
753     { \dim_eval:n #3 + \box_wd:N #1 } {#4}
754     {#5} {#6}
755   }
756 }

```

(End definition for `\coffin_x_shift_corner:Nnnn`. This function is documented on page ??.)

2.8 Coffin diagnostics

`\l_coffin_display_coffin` Used for printing coffins with data structures attached.

```

\l_coffin_display_coord_coffin 757 \coffin_new:N \l_coffin_display_coffin
\l_coffin_display_pole_coffin   758 \coffin_new:N \l_coffin_display_coord_coffin
                                759 \coffin_new:N \l_coffin_display_pole_coffin

```

`\l_coffin_display_handles_prop` This property list is used to print coffin handles at suitable positions. The offsets are expressed as multiples of the basic offset value, which therefore acts as a scale-factor.

```

760 \prop_new:N \l_coffin_display_handles_prop
761 \prop_put:Nnn \l_coffin_display_handles_prop { tl }
762   { { b } { r } { -1 } { 1 } }
763 \prop_put:Nnn \l_coffin_display_handles_prop { thc }
764   { { b } { hc } { 0 } { 1 } }
765 \prop_put:Nnn \l_coffin_display_handles_prop { tr }
766   { { b } { l } { 1 } { 1 } }
767 \prop_put:Nnn \l_coffin_display_handles_prop { vcl }
768   { { vc } { r } { -1 } { 0 } }
769 \prop_put:Nnn \l_coffin_display_handles_prop { vhc }
770   { { vc } { hc } { 0 } { 0 } }
771 \prop_put:Nnn \l_coffin_display_handles_prop { vcr }
772   { { vc } { l } { 1 } { 0 } }
773 \prop_put:Nnn \l_coffin_display_handles_prop { bl }
774   { { t } { r } { -1 } { -1 } }
775 \prop_put:Nnn \l_coffin_display_handles_prop { bhc }
776   { { t } { hc } { 0 } { -1 } }
777 \prop_put:Nnn \l_coffin_display_handles_prop { br }
778   { { t } { l } { 1 } { -1 } }
779 \prop_put:Nnn \l_coffin_display_handles_prop { Tl }
780   { { t } { r } { -1 } { -1 } }
781 \prop_put:Nnn \l_coffin_display_handles_prop { Thc }
782   { { t } { hc } { 0 } { -1 } }
783 \prop_put:Nnn \l_coffin_display_handles_prop { Tr }
784   { { t } { l } { 1 } { -1 } }
785 \prop_put:Nnn \l_coffin_display_handles_prop { Hl }
786   { { vc } { r } { -1 } { 1 } }
787 \prop_put:Nnn \l_coffin_display_handles_prop { Hhc }

```

```

788 { { vc } { hc } { 0 } { 1 } }
789 \prop_put:Nnn \l_coffin_display_handles_prop { Hr }
790 { { vc } { l } { 1 } { 1 } }
791 \prop_put:Nnn \l_coffin_display_handles_prop { Bl }
792 { { b } { r } { -1 } { -1 } }
793 \prop_put:Nnn \l_coffin_display_handles_prop { Bhc }
794 { { b } { hc } { 0 } { -1 } }
795 \prop_put:Nnn \l_coffin_display_handles_prop { Br }
796 { { b } { l } { 1 } { -1 } }

```

`\l_coffin_display_offset_dim` The standard offset for the label from the handle position when displaying handles.

```

797 \dim_new:N \l_coffin_display_offset_dim
798 \dim_set:Nn \l_coffin_display_offset_dim { 2 pt }

```

`\l_coffin_display_x_dim` As the intersections of poles have to be calculated to find which ones to print, there is
`\l_coffin_display_y_dim` a need to avoid repetition. This is done by saving the intersection into two dedicated values.

```

799 \dim_new:N \l_coffin_display_x_dim
800 \dim_new:N \l_coffin_display_y_dim

```

`\l_coffin_display_poles_prop` A property list for printing poles: various things need to be deleted from this to get a “nice” output.

```

801 \prop_new:N \l_coffin_display_poles_prop

```

`\l_coffin_display_font_tl` Stores the settings used to print coffin data: this keeps things flexible.

```

802 \tl_new:N \l_coffin_display_font_tl
803 \tl_set:Nn \l_coffin_display_font_tl { \sfamily \tiny }

```

`\l_coffin_handles_tmp_prop` Used for displaying coffins, as the handles need to be stored in this case, at least temporarily.

```

804 \prop_new:N \l_coffin_handles_tmp_prop

```

`\coffin_mark_handle:Nnnn` Marking a single handle is relatively easy. The standard attachment function is used,
`\coffin_mark_handle:cnnn` meaning that there are two calculations for the location. However, this is likely to be
`\coffin_mark_handle_aux:nnnnNnn` okay given the load expected. Contrast with the more optimised version for showing all handles which comes next.

```

805 \cs_new_protected_nopar:Npn \coffin_mark_handle:Nnnn #1#2#3#4
806 {
807   \hcoffin_set:Nn \l_coffin_display_pole_coffin
808   {
809     \color {#4}
810     \rule { 1 pt } { 1 pt }

```

```

811     \hbox:n { \tex_vrule:D width 1 pt height 1 pt \scan_stop: }
812   }
813   \coffin_attach_mark:NnnNnnnn #1 {#2} {#3}
814   \l_coffin_display_pole_coffin { hc } { vc } { 0 pt } { 0 pt }
815   \hcoffin_set:Nn \l_coffin_display_coord_coffin
816   {
817     \color {#4}
818     \l_coffin_display_font_tl
819     ( \tl_to_str:n { #2 , #3 } )
820   }
821   \prop_get:NnN \l_coffin_display_handles_prop
822   { #2 #3 } \l_coffin_tmp_tl
823   \quark_if_no_value:NTF \l_coffin_tmp_tl
824   {
825     \prop_get:NnN \l_coffin_display_handles_prop
826     { #3 #2 } \l_coffin_tmp_tl
827     \quark_if_no_value:NTF \l_coffin_tmp_tl
828     {
829       \coffin_attach_mark:NnnNnnnn #1 {#2} {#3}
830       \l_coffin_display_coord_coffin { l } { vc }
831       { 1 pt } { 0 pt }
832     }
833     {
834       \exp_last_unbraced:No \coffin_mark_handle_aux:nnnnNnn
835       \l_coffin_tmp_tl #1 {#2} {#3}
836     }
837   }
838   {
839     \exp_last_unbraced:No \coffin_mark_handle_aux:nnnnNnn
840     \l_coffin_tmp_tl #1 {#2} {#3}
841   }
842 }
843 \cs_new_protected_nopar:Npn \coffin_mark_handle_aux:nnnnNnn #1#2#3#4#5#6#7
844 {
845   \coffin_attach_mark:NnnNnnnn #5 {#6} {#7}
846   \l_coffin_display_coord_coffin {#1} {#2}
847   { #3 \l_coffin_display_offset_dim }
848   { #4 \l_coffin_display_offset_dim }
849 }
850 \cs_generate_variant:Nn \coffin_mark_handle:Nnnn { c }

```

(End definition for `\coffin_mark_handle:Nnnn` and `\coffin_mark_handle:cnnn`. These functions are documented on page 3.)

```

\coffin_display_handles:Nn
\coffin_display_handles:cn
\coffin_display_handles_aux:nnnnnn
\coffin_display_handles_aux:nnnn
\coffin_display_attach:Nnnnn

```

Printing the poles starts by removing any duplicates, for which the H poles is used as the definitive version for the baseline and bottom. Two loops are then used to find the combinations of handles for all of these poles. This is done such that poles are removed during the loops to avoid duplication.

```

851 \cs_new_protected_nopar:Npn \coffin_display_handles:Nn #1#2

```

```

852 {
853   \hcoffin_set:Nn \l_coffin_display_pole_coffin
854   {
855     \color {#2}
856     \rule { 1 pt } { 1 pt }
857   }
858   \prop_set_eq:Nc \l_coffin_display_poles_prop
859   { l_coffin_poles_ \int_value:w #1 _prop }
860   \coffin_get_pole:NnN #1 { H } \l_coffin_pole_a_tl
861   \coffin_get_pole:NnN #1 { T } \l_coffin_pole_b_tl
862   \tl_if_eq:NNT \l_coffin_pole_a_tl \l_coffin_pole_b_tl
863   { \prop_del:Nn \l_coffin_display_poles_prop { T } }
864   \coffin_get_pole:NnN #1 { B } \l_coffin_pole_b_tl
865   \tl_if_eq:NNT \l_coffin_pole_a_tl \l_coffin_pole_b_tl
866   { \prop_del:Nn \l_coffin_display_poles_prop { B } }
867   \coffin_set_eq:NN \l_coffin_display_coffin #1
868   \prop_clear:N \l_coffin_handles_tmp_prop
869   \prop_map_inline:Nn \l_coffin_display_poles_prop
870   {
871     \prop_del:Nn \l_coffin_display_poles_prop {##1}
872     \coffin_display_handles_aux:nnnnnn {##1} ##2 {##2}
873   }
874   \box_use:N \l_coffin_display_coffin
875 }

```

For each pole there is a check for an intersection, which here does not give an error if none is found. The successful values are stored and used to align the pole coffin with the main coffin for output. The positions are recovered from the preset list if available.

```

876 \cs_new_protected_nopar:Npn \coffin_display_handles_aux:nnnnnn #1#2#3#4#5#6
877 {
878   \prop_map_inline:Nn \l_coffin_display_poles_prop
879   {
880     \bool_set_false:N \l_coffin_error_bool
881     \coffin_calculate_intersection:nnnnnnnn {#2} {#3} {#4} {#5} ##2
882     \bool_if:NF \l_coffin_error_bool
883     {
884       \dim_set:Nn \l_coffin_display_x_dim { \l_coffin_x_dim }
885       \dim_set:Nn \l_coffin_display_y_dim { \l_coffin_y_dim }
886       \coffin_display_attach:Nnnnn
887       \l_coffin_display_pole_coffin { hc } { vc }
888       { 0 pt } { 0 pt }
889       \hcoffin_set:Nn \l_coffin_display_coord_coffin
890       {
891         \color {#6}
892         \l_coffin_display_font_tl
893         ( \tl_to_str:n { #1 , ##1 } )
894       }
895       \prop_get:NnN \l_coffin_display_handles_prop
896       { #1 ##1 } \l_coffin_tmp_tl

```

```

897 \quark_if_no_value:NTF \l_coffin_tmp_tl
898 {
899   \prop_get:NnN \l_coffin_display_handles_prop
900   { ##1 #1 } \l_coffin_tmp_tl
901   \quark_if_no_value:NTF \l_coffin_tmp_tl
902   {
903     \coffin_display_attach:Nnnnn
904     \l_coffin_display_coord_coffin { l } { vc }
905     { 1 pt } { 0 pt }
906   }
907   {
908     \exp_last_unbraced:No
909     \coffin_display_handles_aux:nnnn
910     \l_coffin_tmp_tl
911   }
912 }
913 {
914   \exp_last_unbraced:No \coffin_display_handles_aux:nnnn
915   \l_coffin_tmp_tl
916 }
917 }
918 }
919 }
920 \cs_new_protected_nopar:Npn \coffin_display_handles_aux:nnnn #1#2#3#4
921 {
922   \coffin_display_attach:Nnnnn
923   \l_coffin_display_coord_coffin {#1} {#2}
924   { #3 \l_coffin_display_offset_dim }
925   { #4 \l_coffin_display_offset_dim }
926 }
927 \cs_generate_variant:Nn \coffin_display_handles:Nn { c }

```

This is a dedicated version of `\coffin_attach:NnnNnnnn` with a hard-wired first coffin. As the intersection is already known and stored for the display coffin the code simply uses it directly, with no calculation.

```

928 \cs_new_protected_nopar:Npn \coffin_display_attach:Nnnnn #1#2#3#4#5
929 {
930   \coffin_calculate_intersection:Nnn #1 {#2} {#3}
931   \dim_set:Nn \l_coffin_x_prime_dim { \l_coffin_x_dim }
932   \dim_set:Nn \l_coffin_y_prime_dim { \l_coffin_y_dim }
933   \dim_set:Nn \l_coffin_offset_x_dim
934   { \l_coffin_display_x_dim - \l_coffin_x_prime_dim + #4 }
935   \dim_set:Nn \l_coffin_offset_y_dim
936   { \l_coffin_display_y_dim - \l_coffin_y_prime_dim + #5 }
937   \hbox_set:Nn \l_coffin_aligned_coffin
938   {
939     \box_use:N \l_coffin_display_coffin
940     \tex_kern:D -\box_wd:N \l_coffin_display_coffin
941     \tex_kern:D \l_coffin_offset_x_dim

```



```

942     \box_move_up:nn { \l_coffin_offset_y_dim } { \box_use:N #1 }
943   }
944   \box_set_ht:Nn \l_coffin_aligned_coffin
945   { \box_ht:N \l_coffin_display_coffin }
946   \box_set_dp:Nn \l_coffin_aligned_coffin
947   { \box_dp:N \l_coffin_display_coffin }
948   \box_set_wd:Nn \l_coffin_aligned_coffin
949   { \box_wd:N \l_coffin_display_coffin }
950   \box_set_eq:NN \l_coffin_display_coffin \l_coffin_aligned_coffin
951 }

```

(End definition for `\coffin_display_handles:Nn` and `\coffin_display_handles:cn`. These functions are documented on page 3.)

`\coffin_show_structure:N` For showing the various internal structures attached to a coffin in a way that keeps things relatively readable. If there is no apparent structure then the code complains.

```

\coffin_show_structure:c
\coffin_show_aux:n
\coffin_show_aux:w
952 \cs_new_protected_nopar:Npn \coffin_show_structure:N #1
953 {
954   \cs_if_exist:cTF { l_coffin_poles_ \int_value:w #1 _prop }
955   {
956     \iow_term:x
957     {
958       \iow_newline:
959       Size-of~coffin~\token_to_str:N #1 : \iow_newline:
960       > ~ ht=~\dim_use:N \box_ht:N #1 \iow_newline:
961       > ~ dp=~\dim_use:N \box_dp:N #1 \iow_newline:
962       > ~ wd=~\dim_use:N \box_wd:N #1 \iow_newline:
963     }
964     \iow_term:x { Poles-of~coffin~\token_to_str:N #1 : }
965     \tl_set:Nx \l_coffin_tmp_tl
966     {
967       \prop_map_function:cn
968       { l_coffin_poles_ \int_value:w #1 _prop }
969       \coffin_show_aux:nn
970     }
971     \etex_showtokens:D \exp_after:wN \exp_after:wN \exp_after:wN
972     { \exp_after:wN \coffin_show_aux:w \l_coffin_tmp_tl }
973   }
974   {
975     \iow_term:x { ----No~poles~found---- }
976     \tl_show:n { Is~this~really~a~coffin? }
977   }
978 }
979 \cs_new:Npn \coffin_show_aux:nn #1#2
980 {
981   \iow_newline: > \c_space_tl \c_space_tl
982   #1 \c_space_tl \c_space_tl => \c_space_tl \c_space_tl \exp_not:n {#2}
983 }
984 \cs_new_nopar:Npn \coffin_show_aux:w #1 > ~ { }

```

```
985 \cs_generate_variant:Nn \coffin_show_structure:N { c }
```

(End definition for `\coffin_show_structure:N` and `\coffin_show_structure:c`. These functions are documented on page 4.)

2.9 Messages

```
986 \msg_kernel_new:nnnn { coffin } { no-pole-intersection }
987 { No~intersection~between~coffin~poles. }
988 {
989   \c_msg_coding_error_text_tl
990   LaTeX~was~asked~to~find~the~intersection~between~two~poles,~
991   but~they~do~not~have~a~unique~meeting~point:~
992   the~value~(0~pt,~0~pt)~will~be~used.
993 }
994 \msg_kernel_new:nnnn { coffin } { unknown-coffin }
995 { Unknown~coffin~'#1'. }
996 { The~coffin~'#1'~was~never~defined. }
997 \msg_kernel_new:nnnn { coffin } { unknown-coffin-pole }
998 { Pole~'#1'~unknown~for~coffin~'#2'. }
999 {
1000   \c_msg_coding_error_text_tl
1001   LaTeX~was~asked~to~find~a~typesetting~pole~for~a~coffin,~
1002   but~either~the~coffin~does~not~exist~or~the~pole~name~is~wrong.
1003 }
1004 </package>
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

B			
<code>\bool_if:NF</code>	882	<code>\box_new:N</code>	4, 80
<code>\bool_if:NT</code>	257	<code>\box_set_dp:Nn</code>	406, 424, 567, 946
<code>\bool_new:N</code>	30	<code>\box_set_eq:NN</code>	133, 426, 950
<code>\bool_set_false:N</code>	253, 880	<code>\box_set_ht:Nn</code>	405, 423, 565, 944
<code>\bool_set_true:N</code>	271, 286, 319	<code>\box_set_wd:Nn</code>	407, 425, 568, 948
<code>\box_clear:N</code>	73	<code>\box_use:N</code>	441, 444, 532, 557, 563, 683, 713, 874, 939, 942
<code>\box_dp:N</code>	177, 178, 225, 238, 243, 406, 424, 582, 947, 961	<code>\box_wd:N</code>	179, 224, 229, 233, 237, 374, 407, 425, 442, 527, 581, 586, 746, 753, 940, 949, 962
<code>\box_ht:N</code>	123, 176, 178, 222, 224, 238, 242, 405, 423, 579, 581, 945, 960		
<code>\box_move_down:nn</code>	562	C	
<code>\box_move_up:nn</code>	444, 942	<code>\c_coffin_corners_prop</code> .	8, 8–12, 84, 155

\c_coffin_poles_prop	\coffin_join:cnnNnnnn	3, 364
..... 13 , 13 , 15–17 , 19–24 , 86 , 157	\coffin_join:Nnncnnnn	3, 364
\c_empty_box	\coffin_join:NnnNnnnn	3, 364 , 364 , 400
\c_empty_coffin	\coffin_mark_handle:cnnn	3, 805
\c_max_dim	\coffin_mark_handle:Nnnn	3, 805 , 805 , 850
\c_msg_coding_error_text_tl ..	\coffin_mark_handle_aux:nnnnNnn 805 , 834 , 839 , 843
\c_pi_fp	\coffin_new:c	1, 78
\c_space_tl	\coffin_new:N 1, 78 , 78 , 88 , 138 , 140 , 141 , 757–759
\c_zero_dim 244 , 267 , 270 , 273 , 282 , 285 ,	\coffin_offset_corner:Nnnnn	470, 472
288 , 297 , 304 , 370 , 375 , 382 , 523 , 528	\coffin_offset_corners:Nnn 386, 387 , 393 , 394 , 467 , 467
\c_zero_fp	\coffin_offset_corners:Nnnnn	467
\coffin_align:NnnNnnnnN	\coffin_offset_pole:Nnnnnnn	448, 451 , 453
..... 366 , 403 , 421 , 429 , 429 , 519	\coffin_offset_poles:Nnn	384, 385 , 390 , 391 , 413 , 414 , 448 , 448
\coffin_attach:cnncnnnn	\coffin_reset_structure:N 74, 100 , 118 , 152 , 152 , 378 , 408
\coffin_attach:cnnNnnnn	\coffin_resize:cnn	674
\coffin_attach:Nnncnnnn	\coffin_resize:Nnn	674, 674 , 686
\coffin_attach:NnnNnnnn	\coffin_resize_common:Nnn 684, 687 , 687 , 714
\coffin_attach_mark:NnnNnnnn	\coffin_rotate:cn	2, 541
..... 401 , 419 , 813 , 829 , 845	\coffin_rotate:Nn	2, 541 , 541 , 575
\coffin_calculate_intersection:Nnn ..	\coffin_rotate_bounding:nnn	554, 588 , 588
..... 249 , 249 , 431 , 434 , 930	\coffin_rotate_corner:Nnnn	549, 588 , 594
\coffin_calculate_intersection:nnnnnnnn	\coffin_rotate_pole:Nnnnnnn	551, 600 , 600
..... 249 , 255 , 264 , 881	\coffin_rotate_vector:nnNN 590, 596 , 602 , 603 , 612 , 612
\coffin_calculate_intersection_aux:nnnnnn	\coffin_saved_Depth:	49, 49 , 169 , 184
..... 249 , 276 , 291 , 300 , 307 , 341 , 350	\coffin_saved_Height:	49, 50 , 168 , 183
\coffin_clear:c	\coffin_saved_TotalHeight: 49, 51 , 170 , 185
\coffin_clear:N	\coffin_saved_Width:	49, 52 , 171 , 186
\coffin_display_attach:Nnnnn	\coffin_scale:cnn	702
..... 851 , 886 , 903 , 922 , 928	\coffin_scale:Nnn	702, 702 , 717
\coffin_display_handles:cn	\coffin_scale_corner:Nnnn	690, 727 , 727
\coffin_display_handles:Nn	\coffin_scale_pole:Nnnnnnn	692, 727 , 733
\coffin_display_handles_aux:nnnn	\coffin_scale_vector:nnNN 718, 718 , 729 , 735
..... 851 , 909 , 914 , 920	\coffin_set_bounding:N	552, 576 , 576
\coffin_display_handles_aux:nnnnnn	\coffin_set_eq:cc	1, 129
..... 851 , 872 , 876	\coffin_set_eq:cN	1, 129
\coffin_end_user_dimensions:	\coffin_set_eq:Nc	1, 129
..... 166 , 181 , 198 , 211 , 700	\coffin_set_eq:NN 1, 129 , 129 , 137 , 398 , 417 , 446 , 867
\coffin_find_bounding_shift:	\coffin_set_eq_structure:NN	134, 159 , 159
..... 556 , 645 , 645	\coffin_set_horizontal_pole:cnn ..	2, 188
\coffin_find_bounding_shift_aux:nn ..		
..... 645 , 649 , 651		
\coffin_find_corner_maxima:N		
..... 555 , 629 , 629		
\coffin_find_corner_maxima_aux:nn ..		
..... 629 , 636 , 638		
\coffin_get_pole:NnN ..		
142 , 142 , 251 ,		
252 , 484 , 485 , 488 , 489 , 860 , 861 , 864		
\coffin_if_exist_execute:Nn		
..... 53 , 53 , 71 , 91 , 108 , 131 , 190 , 203		
\coffin_join:cnncnnnn		
..... 3, 364		

\coffin_set_horizontal_pole:Nnn	
. 2, 188, 188, 216	
\coffin_set_pole:Nnn 188, 214, 218	
\coffin_set_pole:Nnx . . 124, 188, 193,	
206, 460, 497, 501, 509, 513, 605, 736	
\coffin_set_user_dimensions:N	
. 166, 166, 192, 205, 676, 704	
\coffin_set_vertical_pole:cnn . . 2, 188	
\coffin_set_vertical_pole:Nnn	
. 2, 188, 201, 217	
\coffin_shift_corner:Nnnn . 571, 653, 653	
\coffin_shift_pole:Nnnnnn . 573, 653, 661	
\coffin_show_aux:n 952	
\coffin_show_aux:nn 969, 979	
\coffin_show_aux:w 952, 972, 984	
\coffin_show_structure:c 4, 952	
\coffin_show_structure:N 4, 952, 952, 985	
\coffin_typeset:cnnnn 3, 517	
\coffin_typeset:Nnnnn . . 3, 517, 517, 534	
\coffin_update_B:nnnnnnnnN 482, 490, 505	
\coffin_update_corners:N	
. 102, 120, 219, 219	
\coffin_update_poles:N	
. 101, 119, 231, 231, 381, 412	
\coffin_update_T:nnnnnnnnN 482, 486, 493	
\coffin_update_vertical_poles:NNN . .	
. 397, 416, 482, 482	
\coffin_x_shift_corner:Nnnn 696, 742, 742	
\coffin_x_shift_pole:Nnnnnn 698, 742, 749	
\color 809, 817, 855, 891	
\cs_generate_variant:Nn 77,	
88, 105, 128, 137, 216–218, 400,	
428, 534, 575, 686, 717, 850, 927, 985	
\cs_if_exist:cTF 57, 954	
\cs_if_exist:NTF 55	
\cs_new:Npn 979	
\cs_new_nopar:Npn 49–52, 984	
\cs_new_protected:Npn 53, 89, 106	
\cs_new_protected_nopar:Npn	
. 69, 78, 129, 142, 152,	
159, 166, 181, 188, 201, 214, 219,	
231, 249, 264, 350, 364, 401, 419,	
429, 448, 453, 467, 472, 482, 493,	
505, 517, 541, 576, 588, 594, 600,	
612, 629, 638, 645, 651, 653, 661,	
674, 687, 702, 718, 727, 733, 742,	
749, 805, 843, 851, 876, 920, 928, 952	
\cs_set_eq:NN 168–175, 183–186	
	D
\Depth 166, 169, 173, 177, 184	
\dim_compare:nNnF 244	
\dim_compare:nNnT 370, 375, 523, 528	
\dim_compare:nNnTF 267, 270, 273,	
282, 285, 288, 297, 304, 382, 495, 507	
\dim_eval:n 195, 208,	
478, 479, 657, 658, 665, 666, 746, 753	
\dim_new:N 5, 31, 32, 37–40,	
45–48, 536–540, 672, 673, 797, 799, 800	
\dim_set:Nn 112, 122, 176–	
179, 225, 233, 238, 243, 245, 269,	
274, 284, 289, 299, 306, 339, 362,	
373, 432, 433, 435, 437, 455, 456,	
526, 582, 626, 627, 631–634, 647,	
722, 725, 798, 884, 885, 931–933, 935	
\dim_set_max:Nn 641, 643	
\dim_set_min:Nn 640, 642, 652	
\dim_use:N 125, 222, 224,	
225, 227, 229, 235, 237, 240, 242,	
247, 463, 579, 581, 582, 584, 586,	
592, 598, 607–609, 731, 738, 960–962	
\dim_zero:N 260, 261	
	E
\etex_showtokens:D 971	
\exp_after:wN 971, 972	
\exp_last_two_unbraced:Noo 254, 486, 490	
\exp_last_unbraced:NNo 460	
\exp_last_unbraced:No . 834, 839, 908, 914	
\exp_not:n 982	
\ExplFileName 3	
\ExplFileDescription 3	
\ExplFileName 3	
\ExplFileVersion 3	
	F
\fp_add:Nn 328, 361, 625	
\fp_compare:NNNT 693	
\fp_compare:NNNTF 707, 710	
\fp_compare:nNnTF 317	
\fp_cos:Nn 547	
\fp_div:Nn 315, 316, 337, 359, 544, 679, 682	
\fp_mul:Nn 326, 331,	
360, 545, 618, 619, 623, 624, 721, 724	
\fp_new:N 6, 25–29, 35, 36, 41–44, 670, 671	
\fp_set:Nn 543, 705, 706	
\fp_set_eq:NN 616, 617, 621, 622	

\fp_set_from_dim:Nn	311–314, 321, 322, 325, 330, 353–357, 614, 615, 677, 678, 680, 681, 720, 723
\fp_sin:Nn	546
\fp_sub:Nn	323, 333, 335, 358, 620
\fp_to_dim:N	340, 362, 626, 627, 722, 725
G	
\group_begin:	95, 113
\group_end:	98, 116
H	
\hbox:n	811
\hbox_set:Nn	93, 139, 368, 439, 521, 557, 558, 683, 713, 937
\hbox_unpack:N	372, 525, 531
\hcoffin_set:cn	1, 89
\hcoffin_set:Nn	1, 89, 89, 105, 807, 815, 853, 889
\Height	166, 168, 172, 176, 183
I	
\int_value:w	57, 81–83, 85, 144, 154, 156, 161–164, 215, 221, 223, 226, 228, 234, 236, 239, 241, 246, 380, 410, 411, 450, 469, 475, 548, 550, 570, 572, 597, 635, 655, 663, 689, 691, 695, 697, 730, 744, 751, 859, 954, 968
\iow_newline:	958–962, 981
\iow_term:x	956, 964, 975
L	
\l_coffin_aligned_coffin	138, 140, 367, 368, 372, 378, 380, 381, 397, 398, 404–408, 410, 412, 416, 417, 422–426, 460, 475, 520, 521, 525, 532, 937, 944, 946, 948, 950
\l_coffin_aligned_internal_coffin	138, 141, 439, 446
\l_coffin_bottom_corner_dim	537, 539, 562, 566, 633, 642, 658, 666
\l_coffin_bounding_prop	535, 553, 578, 580, 583, 585, 591, 648
\l_coffin_bounding_shift_dim	536, 536, 560, 647, 652
\l_coffin_calc_a_fp	25, 25, 311, 315, 322, 324–326, 329–331, 334, 354, 358
\l_coffin_calc_b_fp	25, 26, 312, 315, 318, 327, 335, 338, 355, 361
\l_coffin_calc_c_fp	25, 27, 313, 316, 356, 360
\l_coffin_calc_d_fp	25, 28, 314, 316, 318, 332, 336, 357, 359
\l_coffin_calc_result_fp	25, 29, 321, 323, 328, 333, 337, 340, 353, 358–362
\l_coffin_cos_fp	35, 36, 547, 618, 623
\l_coffin_Depth_dim	45, 45, 173
\l_coffin_display_coffin	757, 757, 867, 874, 939, 940, 945, 947, 949, 950
\l_coffin_display_coord_coffin	757, 758, 815, 830, 846, 889, 904, 923
\l_coffin_display_font_tl	802, 802, 803, 818, 892
\l_coffin_display_handles_prop	760, 760, 761, 763, 765, 767, 769, 771, 773, 775, 777, 779, 781, 783, 785, 787, 789, 791, 793, 795, 821, 825, 895, 899
\l_coffin_display_offset_dim	797, 797, 798, 847, 848, 924, 925
\l_coffin_display_pole_coffin	757, 759, 807, 814, 853, 887
\l_coffin_display_poles_prop	801, 801, 858, 863, 866, 869, 871, 878
\l_coffin_display_x_dim	799, 799, 884, 934
\l_coffin_display_y_dim	799, 800, 885, 936
\l_coffin_error_bool	30, 30, 253, 257, 271, 286, 319, 880, 882
\l_coffin_handles_tmp_prop	804, 804, 868
\l_coffin_Height_dim	45, 46, 172
\l_coffin_left_corner_dim	537, 537, 561, 569, 634, 640, 657, 665
\l_coffin_offset_x_dim	31, 31, 370, 371, 374, 382, 384, 386, 392, 395, 415, 435, 443, 523, 524, 527, 933, 941
\l_coffin_offset_y_dim	31, 32, 385, 387, 392, 395, 415, 437, 444, 935, 942
\l_coffin_pole_a_tl	33, 33, 251, 256, 484, 487, 488, 491, 860, 862, 865
\l_coffin_pole_b_tl	33, 34, 252, 256, 485, 487, 489, 491, 861, 862, 864, 865
\l_coffin_right_corner_dim	537, 538, 569, 632, 641
\l_coffin_scale_x_fp	670, 670, 677, 679, 693, 705, 710, 721
\l_coffin_scale_y_fp	670, 671, 680, 682, 706, 707, 724
\l_coffin_scaled_total_height_dim	672, 672, 708, 709, 715

V			
		<code>\vcoffin_set:cnn</code> 2, 106
<code>\vbox_set:Nn</code> 110	<code>\vcoffin_set:Nnn</code> 2, 106 , 106 , 128
<code>\vbox_set_top:Nn</code> 121	W	
<code>\vbox_unpack:N</code> 121	<code>\Width</code>	166 , 171 , 175 , 179 , 186 , 678 , 711 , 712