

The L^AT_EX3 kernel: coffins^{*}

The L^AT_EX3 Project[†]

Released 2011/03/23

Abstract

A L^AT_EX3 ‘coffin’ is a design-level method for typesetting boxed material. The structure of coffins contains not only the boxed material itself but also information about the size of the box and potential alignment positions. This structure makes it possible to build complex layouts rapidly by assembling coffins. The code here provides the low-level support system for coffins.

Contents

1	Introduction	2
2	Code-level functions	2
3	Implementation	5
3.1	Coffins: data structures and general variables	5
3.2	Basic coffin functions	7
3.3	Coffins: handle and pole management	10
3.4	Coffins: calculation of pole intersections	13
3.5	Aligning and typesetting of coffins	16
3.6	Rotating coffins	21
3.7	Resizing coffins	26
3.8	Coffin diagnostics	29
3.9	Messages	35

^{*}This file describes v2209, last revised 2011/03/23.

[†]Frank Mittelbach, Denys Duchier, Chris Rowley, Rainer Schöpf, Johannes Braams, Michael Downes, David Carlisle, Alan Jeffrey, Morten Høgholm, Thomas Lotze, Javier Bezos, Will Robertson, Joseph Wright

1 Introduction

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.

2 Code-level functions

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

Creates a new `<coffin>` or raises an error if the name is already taken. The declaration is global. The `<coffin>` will initially be empty.

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

Clears the content of the `<coffin>` within the current \TeX 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>	<code>\coffin_set_eq:NN <coffin1> <coffin2></code>
--	--

Sets both the content and poles of `<coffin1>` equal to those of `<coffin2>` within the current \TeX group level.

<code>\hcoffin_set:Nn</code> <code>\hcoffin_set:cn</code>	<code>\hcoffin_set:Nn <coffin> {<material>}</code>
--	--

Typesets the `<material>` in horizontal mode, storing the result in the `<coffin>`. The standard poles for the `<coffin>` are then set up based on the size of the typeset material.

<code>\vcoffin_set:Nnn</code> <code>\vcoffin_set:cnn</code>	<code>\vcoffin_set:Nnn <coffin> {<width>} {<material>}</code>
--	---

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

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

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

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

This function carries out alignment such that the bounding box of $\langle coffin1 \rangle$ 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.

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

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.

$\backslash coffin_typeset:Nnnnn$ $\backslash coffin_typeset:cnnnn$	$\backslash coffin_typeset:Nnnnn \langle coffin \rangle \{ \langle pole_1 \rangle \} \{ \langle pole_2 \rangle \}$ $\{ \langle x-offset \rangle \} \{ \langle y-offset \rangle \}$
--	--

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.

$\backslash coffin_mark_handle:Nnnn$ $\backslash coffin_mark_handle:cnnn$	$\backslash coffin_mark_handle:Nnnn \langle coffin \rangle \{ \langle pole_1 \rangle \} \{ \langle pole_2 \rangle \}$ $\{ \langle colour \rangle \}$
--	---

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.

$\backslash coffin_show_structure:N$ $\backslash coffin_show_structure:c$	$\backslash coffin_show_structure:N \langle coffin \rangle$
--	---

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.

3 Implementation

```

1 (*package)
2 \ProvidesExplPackage
3   {\filename}{\filedate}{\fileversion}{\filedescription}

```

3.1 Coffins: data structures and general variables

`\l_coffin_tmp_box` Scratch variables.

```

\l_coffin_tmp_dim
\l_coffin_tmp_fp
\l_coffin_tmp_tl
4 \box_new:N \l_coffin_tmp_box
5 \dim_new:N \l_coffin_tmp_dim
6 \fp_new:N \l_coffin_tmp_fp
7 \tl_new:N \l_coffin_tmp_tl

```

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

`\c_coffin_corners_prop` The ‘corners,’ of a coffin define the real content, as opposed to the \TeX bounding box. They all start off in the same place, of course.

```

8 \prop_new:N \c_coffin_corners_prop
9 \tl_set:Nn \l_coffin_tmp_tl { { 0 pt } { 0 pt } }
10 \prop_put:Nno \c_coffin_corners_prop { tl } { \l_coffin_tmp_tl }
11 \prop_put:Nno \c_coffin_corners_prop { tr } { \l_coffin_tmp_tl }
12 \prop_put:Nno \c_coffin_corners_prop { bl } { \l_coffin_tmp_tl }
13 \prop_put:Nno \c_coffin_corners_prop { br } { \l_coffin_tmp_tl }

```

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

`\c_coffin_poles_prop` Pole positions are given for horizontal, vertical and reference-point based values.

```

14 \prop_new:N \c_coffin_poles_prop
15 \tl_set:Nn \l_coffin_tmp_tl { { 0 pt } { 0 pt } { 0 pt } { 1000 pt } }
16 \prop_put:Nno \c_coffin_poles_prop { l } { \l_coffin_tmp_tl }
17 \prop_put:Nno \c_coffin_poles_prop { hc } { \l_coffin_tmp_tl }
18 \prop_put:Nno \c_coffin_poles_prop { r } { \l_coffin_tmp_tl }
19 \tl_set:Nn \l_coffin_tmp_tl { { 0 pt } { 0 pt } { 1000 pt } { 0 pt } }
20 \prop_put:Nno \c_coffin_poles_prop { b } { \l_coffin_tmp_tl }
21 \prop_put:Nno \c_coffin_poles_prop { vc } { \l_coffin_tmp_tl }
22 \prop_put:Nno \c_coffin_poles_prop { t } { \l_coffin_tmp_tl }
23 \prop_put:Nno \c_coffin_poles_prop { B } { \l_coffin_tmp_tl }
24 \prop_put:Nno \c_coffin_poles_prop { H } { \l_coffin_tmp_tl }
25 \prop_put:Nno \c_coffin_poles_prop { T } { \l_coffin_tmp_tl }

```

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

`\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
\l_coffin_calc_d_fp
26 \fp_new:N \l_coffin_calc_a_fp
27 \fp_new:N \l_coffin_calc_b_fp
\l_coffin_calc_result_fp

```

```

28 \fp_new:N \l_coffin_calc_c_fp
29 \fp_new:N \l_coffin_calc_d_fp
30 \fp_new:N \l_coffin_calc_result_fp

```

(End definition for \l_coffin_calc_a_fp. This function is documented on page ??.)

`\l_coffin_error_bool` For propagating errors so that parts of the code can work around them.

```

31 \bool_new:N \l_coffin_error_bool

```

(End definition for \l_coffin_error_bool. This function is documented on page ??.)

`\l_coffin_offset_x_dim` The offset between two sets of coffin handles when typesetting. These values are corrected from those requested in an alignment for the positions of the handles.

```

32 \dim_new:N \l_coffin_offset_x_dim
33 \dim_new:N \l_coffin_offset_y_dim

```

(End definition for \l_coffin_offset_x_dim. This function is documented on page ??.)

`\l_coffin_pole_a_tl` Needed for finding the intersection of two poles.

`\l_coffin_pole_b_tl`

```

34 \tl_new:N \l_coffin_pole_a_tl
35 \tl_new:N \l_coffin_pole_b_tl

```

(End definition for \l_coffin_pole_a_tl. This function is documented on page ??.)

`\l_coffin_sin_fp` Used for rotations to get the sine and cosine values.

`\l_coffin_cos_fp`

```

36 \fp_new:N \l_coffin_sin_fp
37 \fp_new:N \l_coffin_cos_fp

```

(End definition for \l_coffin_sin_fp. This function is documented on page ??.)

`\l_coffin_x_dim` For calculating intersections and so forth.

`\l_coffin_y_dim`

`\l_coffin_x_prime_dim`

`\l_coffin_y_prime_dim`

```

38 \dim_new:N \l_coffin_x_dim
39 \dim_new:N \l_coffin_y_dim
40 \dim_new:N \l_coffin_x_prime_dim
41 \dim_new:N \l_coffin_y_prime_dim

```

(End definition for \l_coffin_x_dim. This function is documented on page ??.)

`\l_coffin_x_fp` Used for calculations where there are clear x - and y -components, for example during vector rotation.

`\l_coffin_y_fp`

`\l_coffin_x_prime_fp`

`\l_coffin_y_prime_fp`

```

42 \fp_new:N \l_coffin_x_fp
43 \fp_new:N \l_coffin_y_fp
44 \fp_new:N \l_coffin_x_prime_fp
45 \fp_new:N \l_coffin_y_prime_fp

```

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

<code>\l_coffin_Depth_dim</code>	Dimensions for the various parts of a coffin.
<code>\l_coffin_Height_dim</code>	
<code>\l_coffin_TotalHeight_dim</code>	46 <code>\dim_new:N \l_coffin_Depth_dim</code>
<code>\l_coffin_Width_dim</code>	47 <code>\dim_new:N \l_coffin_Height_dim</code>
	48 <code>\dim_new:N \l_coffin_TotalHeight_dim</code>
	49 <code>\dim_new:N \l_coffin_Width_dim</code>

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

<code>\coffin_saved_Depth:</code>	Used to save the meaning of <code>\Depth</code> , <code>\Height</code> , <code>\TotalHeight</code> and <code>\Width</code> .
<code>\coffin_saved_Height:</code>	
<code>\coffin_saved_TotalHeight:</code>	50 <code>\cs_new_nopar:Npn \coffin_saved_Depth: { }</code>
<code>\coffin_saved_Width:</code>	51 <code>\cs_new_nopar:Npn \coffin_saved_Height: { }</code>
	52 <code>\cs_new_nopar:Npn \coffin_saved_TotalHeight: { }</code>
	53 <code>\cs_new_nopar:Npn \coffin_saved_Width: { }</code>

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

3.2 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.

```

54 \cs_new_protected:Npn \coffin_if_exist_execute:Nn #1#2 {
55   \cs_if_exist:NTF #1
56   {
57     \cs_if_exist:cTF { l_coffin_poles_ \tex_number:D #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` Clearing coffins means emptying the box and resetting all of the structures.

`\coffin_clear:c`

```

69 \cs_new_protected_nopar:Npn \coffin_clear:N #1 {
70   \coffin_if_exist_execute:Nn #1
71   {
72     \box_clear:N #1
73     \coffin_reset_structure:N #1
74   }
75 }
76 \cs_generate_variant:Nn \coffin_clear:N { c }

```

(End definition for `\coffin_clear:N`, `\coffin_clear:c`, and `.` These functions are documented on page 2.)

`\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.

```

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

```

(End definition for `\coffin_new:N`, `\coffin_new:c`, and `.` These functions are documented on page 2.)

`\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   \coffin_if_exist_execute:Nn #1
91   {
92     \hbox_set:Nn #1
93     {
94       \group_begin:
95         \set@color
96         #2
97       \group_end:
98     }
99     \coffin_reset_structure:N #1
100    \coffin_update_poles:N #1
101    \coffin_update_corners:N #1

```



```

102     }
103   }
104   \cs_generate_variant:Nn \hcoffin_set:Nn { c }

```

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

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

`\vcoffin_set:cnn`

```

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

```

(End definition for `\vcoffin_set:Nnn`, `\vcoffin_set:cnn`, and `.` . 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     \coffin_if_exist_execute:Nn #1
131     {
132       \box_set_eq:NN #1 #2
133       \coffin_set_eq_structure:NN #1 #2
134     }
135   }
136   \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 2.)

`\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`

```

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

```

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

3.3 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.

```

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

```

(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.

```

150 \cs_new_protected_nopar:Npn \coffin_reset_structure:N #1 {
151   \prop_set_eq:cn { l_coffin_corners_ \tex_number:D #1 _prop }
152   \c_coffin_corners_prop
153   \prop_set_eq:cn { l_coffin_poles_ \tex_number:D #1 _prop }
154   \c_coffin_poles_prop
155 }

```

(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.

```

156 \cs_new_protected_nopar:Npn \coffin_set_eq_structure:NN #1#2 {
157   \prop_set_eq:cc { l_coffin_corners_ \tex_number:D #1 _prop }
158   { l_coffin_corners_ \tex_number:D #2 _prop }
159   \prop_set_eq:cc { l_coffin_poles_ \tex_number:D #1 _prop }
160   { l_coffin_poles_ \tex_number:D #2 _prop }
161 }

```

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

```

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

`\coffin_set_horizontal_pole:Nnn` 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.

`\coffin_set_horizontal_pole:cnm`

`\coffin_set_vertical_pole:Nnn`

`\coffin_set_vertical_pole:cnm`

`\coffin_set_pole:Nnn`

`\coffin_set_pole:Nnx`

```

182 \cs_new_protected_nopar:Npn \coffin_set_horizontal_pole:Nnn #1#2#3 {
183   \coffin_if_exist_execute:Nn #1
184   {
185     \coffin_set_user_dimensions:N #1
186     \coffin_set_pole:Nnx #1 {#2}
187     {
188       { 0 pt } { \dim_eval:n {#3} }
189       { 1000 pt } { 0 pt }
190     }
191     \coffin_end_user_dimensions:
192   }
193 }
194 \cs_new_protected_nopar:Npn \coffin_set_vertical_pole:Nnn #1#2#3 {
195   \coffin_if_exist_execute:Nn #1
196   {
197     \coffin_set_user_dimensions:N #1
198     \coffin_set_pole:Nnx #1 {#2}
199     {
200       { \dim_eval:n {#3} } { 0 pt }
201       { 0 pt } { 1000 pt }
202     }
203   }

```

```

203     \coffin_end_user_dimensions:
204     }
205 }
206 \cs_new_protected_nopar:Npn \coffin_set_pole:Nnn #1#2#3 {
207   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _prop } {#2} {#3}
208 }
209 \cs_generate_variant:Nn \coffin_set_horizontal_pole:Nnn { c }
210 \cs_generate_variant:Nn \coffin_set_vertical_pole:Nnn { c }
211 \cs_generate_variant:Nn \coffin_set_pole:Nnn { Nnx }

```

(End definition for `\coffin_set_horizontal_pole:Nnn` and others. 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.

```

212 \cs_new_protected_nopar:Npn \coffin_update_corners:N #1 {
213   \prop_put:cnx { l_coffin_corners_ \tex_number:D #1 _prop } { tl }
214   { { 0 pt } { \dim_use:N \box_ht:N #1 } }
215   \prop_put:cnx { l_coffin_corners_ \tex_number:D #1 _prop } { tr }
216   { { \dim_use:N \box_wd:N #1 } { \dim_use:N \box_ht:N #1 } }
217   \dim_set:Nn \l_coffin_tmp_dim { - \dim_use:N \box_dp:N #1 }
218   \prop_put:cnx { l_coffin_corners_ \tex_number:D #1 _prop } { bl }
219   { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } }
220   \prop_put:cnx { l_coffin_corners_ \tex_number:D #1 _prop } { br }
221   { { \dim_use:N \box_wd:N #1 } { \dim_use:N \l_coffin_tmp_dim } }
222 }

```

(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.

```

223 \cs_new_protected_nopar:Npn \coffin_update_poles:N #1 {
224   \dim_set:Nn \l_coffin_tmp_dim { 0.5 \box_wd:N #1 }
225   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _prop } { hc }
226   { { \dim_use:N \l_coffin_tmp_dim } { 0 pt } { 0 pt } { 1000 pt } }
227   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _prop } { r }
228   { { \dim_use:N \box_wd:N #1 } { 0 pt } { 0 pt } { 1000 pt } }
229   \dim_set:Nn \l_coffin_tmp_dim { ( \box_ht:N #1 - \box_dp:N #1 ) / 2 }
230   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _prop } { vc }
231   { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } { 1000 pt } { 0 pt } }
232   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _prop } { t }
233   { { 0 pt } { \dim_use:N \box_ht:N #1 } { 1000 pt } { 0 pt } }
234   \dim_set:Nn \l_coffin_tmp_dim { \box_dp:N #1 }
235   \dim_compare:nNnF { \l_coffin_tmp_dim } = { \c_zero_dim }
236   { \dim_set:Nn \l_coffin_tmp_dim { -\l_coffin_tmp_dim } }
237   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _prop } { b }

```

```

238     { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } { 1000 pt } { 0 pt } }
239 }

```

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

3.4 Coffins: calculation of pole intersections

`\coffin_calculate_intersection:Nnn` 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.

```

240 \cs_new_protected_nopar:Npn \coffin_calculate_intersection:Nnn #1#2#3 {
241   \coffin_get_pole:NnN #1 {#2} \l_coffin_pole_a_tl
242   \coffin_get_pole:NnN #1 {#3} \l_coffin_pole_b_tl
243   \bool_set_false:N \l_coffin_error_bool
244   \exp_last_two_unbraced:Noo
245     \coffin_calculate_intersection:nnnnnnnn
246     \l_coffin_pole_a_tl \l_coffin_pole_b_tl
247   \bool_if:NT \l_coffin_error_bool
248   {
249     \msg_kernel_error:nn { coffin } { no-pole-intersection }
250     \dim_zero:N \l_coffin_x_dim
251     \dim_zero:N \l_coffin_y_dim
252   }
253 }

```

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.

```

254 \cs_new_protected_nopar:Npn
255   \coffin_calculate_intersection:nnnnnnnn #1#2#3#4#5#6#7#8 {
256   \dim_compare:nNnTF {#3} = { \c_zero_dim }

```

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

```

257   {
258     \dim_set:Nn \l_coffin_x_dim {#1}
259     \dim_compare:nNnTF {#7} = { \c_zero_dim }
260     { \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.

```

261     {
262         \dim_compare:nNnTF {#8} = { \c_zero_dim }
263         { \dim_set:Nn \l_coffin_y_dim {#6} }
264         {
265             \coffin_calculate_intersection_aux:nnnnnN
266             {#1} {#5} {#6} {#7} {#8} \l_coffin_y_dim
267         }
268     }
269 }

```

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.

```

270     {
271         \dim_compare:nNnTF {#4} = { \c_zero_dim }
272         {
273             \dim_set:Nn \l_coffin_y_dim {#2}
274             \dim_compare:nNnTF {#8} = { \c_zero_dim }
275             { \bool_set_true:N \l_coffin_error_bool }
276             {
277                 \dim_compare:nNnTF {#7} = { \c_zero_dim }
278                 { \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.

```

279         {
280             \coffin_calculate_intersection_aux:nnnnnN
281             {#2} {#6} {#5} {#8} {#7} \l_coffin_x_dim
282         }
283     }
284 }

```

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.

```

285     {
286         \dim_compare:nNnTF {#7} = { \c_zero_dim }
287         {
288             \dim_set:Nn \l_coffin_x_dim {#5}
289             \coffin_calculate_intersection_aux:nnnnnN
290             {#5} {#1} {#2} {#3} {#4} \l_coffin_y_dim
291         }

```

```

292     {
293         \dim_compare:nNnTF {#8} = { \c_zero_dim }
294         {
295             \dim_set:Nn \l_coffin_x_dim {#6}
296             \coffin_calculate_intersection_aux:nnnnnN
297             {#6} {#2} {#1} {#4} {#3} \l_coffin_x_dim
298         }

```

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.

```

299     {
300         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#3}
301         \fp_set_from_dim:Nn \l_coffin_calc_b_fp {#4}
302         \fp_set_from_dim:Nn \l_coffin_calc_c_fp {#7}
303         \fp_set_from_dim:Nn \l_coffin_calc_d_fp {#8}
304         \fp_div:Nn \l_coffin_calc_b_fp \l_coffin_calc_a_fp
305         \fp_div:Nn \l_coffin_calc_d_fp \l_coffin_calc_c_fp
306         \fp_compare:nNnTF
307             { \l_coffin_calc_b_fp } = { \l_coffin_calc_d_fp }
308             { \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.

```

309     {
310         \fp_set_from_dim:Nn \l_coffin_calc_result_fp {#6}
311         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#2}
312         \fp_sub:Nn \l_coffin_calc_result_fp
313             { \l_coffin_calc_a_fp }
314         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#1}
315         \fp_mul:Nn \l_coffin_calc_a_fp
316             { \l_coffin_calc_b_fp }
317         \fp_add:Nn \l_coffin_calc_result_fp
318             { \l_coffin_calc_a_fp }
319         \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#5}
320         \fp_mul:Nn \l_coffin_calc_a_fp
321             { \l_coffin_calc_d_fp }
322         \fp_sub:Nn \l_coffin_calc_result_fp
323             { \l_coffin_calc_a_fp }
324         \fp_sub:Nn \l_coffin_calc_b_fp
325             { \l_coffin_calc_d_fp }
326         \fp_div:Nn \l_coffin_calc_result_fp

```

```

327         { \l_coffin_calc_b_fp }
328     \dim_set:Nn \l_coffin_x_dim
329     { \fp_to_dim:N \l_coffin_calc_result_fp }
330     \coffin_calculate_intersection_aux:nnnnN
331     { \l_coffin_x_dim }
332     {#5} {#6} {#8} {#7} \l_coffin_y_dim
333 }
334 }
335 }
336 }
337 }
338 }

```

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.

```

339 \cs_new_protected_nopar:Npn
340   \coffin_calculate_intersection_aux:nnnnN #1#2#3#4#5#6 {
341     \fp_set_from_dim:Nn \l_coffin_calc_result_fp {#1}
342     \fp_set_from_dim:Nn \l_coffin_calc_a_fp {#2}
343     \fp_set_from_dim:Nn \l_coffin_calc_b_fp {#3}
344     \fp_set_from_dim:Nn \l_coffin_calc_c_fp {#4}
345     \fp_set_from_dim:Nn \l_coffin_calc_d_fp {#5}
346     \fp_sub:Nn \l_coffin_calc_result_fp { \l_coffin_calc_a_fp }
347     \fp_div:Nn \l_coffin_calc_result_fp { \l_coffin_calc_d_fp }
348     \fp_mul:Nn \l_coffin_calc_result_fp { \l_coffin_calc_c_fp }
349     \fp_add:Nn \l_coffin_calc_result_fp { \l_coffin_calc_b_fp }
350     \dim_set:Nn #6 { \fp_to_dim:N \l_coffin_calc_result_fp }
351 }

```

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

3.5 Aligning and typesetting of coffins

`\coffin_join:NnnNnnnn` 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.

```

352 \cs_new_protected_nopar:Npn \coffin_join:NnnNnnnn #1#2#3#4#5#6#7#8 {
353   \coffin_align:NnnNnnnnN
354   #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.

```

355 \hbox_set:Nn \l_coffin_aligned_coffin
356 {
357   \dim_compare:nNnT { \l_coffin_offset_x_dim } < { \c_zero_dim }
358     { \tex_kern:D -\l_coffin_offset_x_dim }
359   \hbox_unpack:N \l_coffin_aligned_coffin
360   \dim_set:Nn \l_coffin_tmp_dim
361     { \l_coffin_offset_x_dim - \box_wd:N #1 + \box_wd:N #4 }
362   \dim_compare:nNnT { \l_coffin_tmp_dim } < { \c_zero_dim }
363     { \tex_kern:D -\l_coffin_tmp_dim }
364 }

```

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

```

365 \coffin_reset_structure:N \l_coffin_aligned_coffin
366 \prop_clear:c
367 {
368   \l_coffin_corners_ \tex_number:D \l_coffin_aligned_coffin _ prop
369 }
370 \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.

```

371 \dim_compare:nNnTF { \l_coffin_offset_x_dim } < { \c_zero_dim }
372 {
373   \coffin_offset_poles:Nnn #1 { -\l_coffin_offset_x_dim } { 0 pt }
374   \coffin_offset_poles:Nnn #4 { 0 pt } { \l_coffin_offset_y_dim }
375   \coffin_offset_corners:Nnn #1 { -\l_coffin_offset_x_dim } { 0 pt }
376   \coffin_offset_corners:Nnn #4 { 0 pt } { \l_coffin_offset_y_dim }
377 }
378 {
379   \coffin_offset_poles:Nnn #1 { 0 pt } { 0 pt }
380   \coffin_offset_poles:Nnn #4
381     { \l_coffin_offset_x_dim } { \l_coffin_offset_y_dim }
382   \coffin_offset_corners:Nnn #1 { 0 pt } { 0 pt }
383   \coffin_offset_corners:Nnn #4
384     { \l_coffin_offset_x_dim } { \l_coffin_offset_y_dim }
385 }
386 \coffin_update_vertical_poles:NNN #1 #4 \l_coffin_aligned_coffin
387 \coffin_set_eq:NN #1 \l_coffin_aligned_coffin
388 }
389 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { c , Nnnc , cnnc }

```

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

`\coffin_attach:NnnNnnnn`
`\coffin_attach:cnncNnnnn`
`\coffin_attach:Nnncnnnn`
`\coffin_attach:cnncnnnn`
`\coffin_attach_mark: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. The function used when marking a position is hear also as it is similar but without the structure updates.

```

390 \cs_new_protected_nopar:Npn \coffin_attach:NnnNnnnn #1#2#3#4#5#6#7#8 {
391   \coffin_align:NnnNnnnnN
392   #1 {#2} {#3} #4 {#5} {#6} {#7} {#8} \l_coffin_aligned_coffin
393   \box_set_ht:Nn \l_coffin_aligned_coffin { \box_ht:N #1 }
394   \box_set_dp:Nn \l_coffin_aligned_coffin { \box_dp:N #1 }
395   \box_set_wd:Nn \l_coffin_aligned_coffin { \box_wd:N #1 }
396   \coffin_reset_structure:N \l_coffin_aligned_coffin
397   \prop_set_eq:cc
398   { \l_coffin_corners_ \tex_number:D \l_coffin_aligned_coffin _prop }
399   { \l_coffin_corners_ \tex_number:D #1 _prop }
400   \coffin_update_poles:N \l_coffin_aligned_coffin
401   \coffin_offset_poles:Nnn #1 { 0 pt } { 0 pt }
402   \coffin_offset_poles:Nnn #4
403   { \l_coffin_offset_x_dim } { \l_coffin_offset_y_dim }
404   \coffin_update_vertical_poles:NNN #1 #4 \l_coffin_aligned_coffin
405   \coffin_set_eq:NN #1 \l_coffin_aligned_coffin
406 }
407 \cs_new_protected_nopar:Npn
408   \coffin_attach_mark:NnnNnnnn #1#2#3#4#5#6#7#8 {
409   \coffin_align:NnnNnnnnN
410   #1 {#2} {#3} #4 {#5} {#6} {#7} {#8} \l_coffin_aligned_coffin
411   \box_set_ht:Nn \l_coffin_aligned_coffin { \box_ht:N #1 }
412   \box_set_dp:Nn \l_coffin_aligned_coffin { \box_dp:N #1 }
413   \box_set_wd:Nn \l_coffin_aligned_coffin { \box_wd:N #1 }
414   \box_set_eq:NN #1 \l_coffin_aligned_coffin
415 }
416 \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.

```

417 \cs_new_protected_nopar:Npn
418   \coffin_align:NnnNnnnnN #1#2#3#4#5#6#7#8#9 {
419   \coffin_calculate_intersection:Nnn #4 {#5} {#6}
420   \dim_set:Nn \l_coffin_x_prime_dim { \l_coffin_x_dim }
421   \dim_set:Nn \l_coffin_y_prime_dim { \l_coffin_y_dim }
422   \coffin_calculate_intersection:Nnn #1 {#2} {#3}

```

```

423 \dim_set:Nn \l_coffin_offset_x_dim
424   { \l_coffin_x_dim - \l_coffin_x_prime_dim + #7 }
425 \dim_set:Nn \l_coffin_offset_y_dim
426   { \l_coffin_y_dim - \l_coffin_y_prime_dim + #8 }
427 \hbox_set:Nn \l_coffin_aligned_internal_coffin
428   {
429     \box_use:N #1
430     \tex_kern:D -\box_wd:N #1
431     \tex_kern:D \l_coffin_offset_x_dim
432     \box_move_up:nn { \l_coffin_offset_y_dim } { \box_use:N #4 }
433   }
434 \coffin_set_eq:NN #9 \l_coffin_aligned_internal_coffin
435 }

```

(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.

```

436 \cs_new_protected_nopar:Npn \coffin_offset_poles:Nnn #1#2#3 {
437   \prop_map_inline:cn { l_coffin_poles_ \tex_number:D #1 _prop }
438     { \coffin_offset_pole:Nnnnnnn #1 {##1} ##2 {#2} {#3} }
439 }
440 \cs_new_protected_nopar:Npn
441   \coffin_offset_pole:Nnnnnnn #1#2#3#4#5#6#7#8 {
442   \dim_set:Nn \l_coffin_x_dim { #3 + #7 }
443   \dim_set:Nn \l_coffin_y_dim { #4 + #8 }
444   \tl_if_in:nnTF {#2} { - }
445     { \tl_set:Nn \l_coffin_tmp_tl { {#2} } }
446     { \tl_set:Nn \l_coffin_tmp_tl { { #1 - #2 } } }
447   \exp_last_unbraced:NNo \coffin_set_pole:Nnx \l_coffin_aligned_coffin
448     { \l_coffin_tmp_tl }
449     {
450       { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim }
451       {#5} {#6}
452     }
453 }

```

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

`\coffin_offset_corners:Nnn`
`\coffin_offset_corners:Nnnnn`

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

```

454 \cs_new_protected_nopar:Npn \coffin_offset_corners:Nnn #1#2#3 {
455   \prop_map_inline:cn { l_coffin_corners_ \tex_number:D #1 _prop }

```

```

456 { \coffin_offset_corner:Nnnnn #1 {##1} ##2 {#2} {#3} }
457 }
458 \cs_new_protected_nopar:Npn \coffin_offset_corner:Nnnnn #1#2#3#4#5#6 {
459   \prop_put:cnx
460   { l_coffin_corners_ \tex_number:D \l_coffin_aligned_coffin _prop }
461   { #1 - #2 }
462   {
463     { \dim_eval:n { #3 + #5 } }
464     { \dim_eval:n { #4 + #6 } }
465   }
466 }

```

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

```

\coffin_update_vertical_poles:NNN
\coffin_update_T:nnnnnnnnN
\coffin_update_B:nnnnnnnnN

```

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

```

467 \cs_new_protected_nopar:Npn \coffin_update_vertical_poles:NNN #1#2#3 {
468   \coffin_get_pole:NnN #3 { #1 -T } \l_coffin_pole_a_tl
469   \coffin_get_pole:NnN #3 { #2 -T } \l_coffin_pole_b_tl
470   \exp_last_two_unbraced:Noo \coffin_update_T:nnnnnnnnN
471   \l_coffin_pole_a_tl \l_coffin_pole_b_tl #3
472   \coffin_get_pole:NnN #3 { #1 -B } \l_coffin_pole_a_tl
473   \coffin_get_pole:NnN #3 { #2 -B } \l_coffin_pole_b_tl
474   \exp_last_two_unbraced:Noo \coffin_update_B:nnnnnnnnN
475   \l_coffin_pole_a_tl \l_coffin_pole_b_tl #3
476 }
477 \cs_new_protected_nopar:Npn
478   \coffin_update_T:nnnnnnnnN #1#2#3#4#5#6#7#8#9 {
479   \dim_compare:nNnTF {#2} < {#6}
480   {
481     \coffin_set_pole:Nnx #9 { T }
482     { { 0 pt } {#6} { 1000 pt } { 0 pt } }
483   }
484   {
485     \coffin_set_pole:Nnx #9 { T }
486     { { 0 pt } {#2} { 1000 pt } { 0 pt } }
487   }
488 }
489 \cs_new_protected_nopar:Npn
490   \coffin_update_B:nnnnnnnnN #1#2#3#4#5#6#7#8#9 {
491   \dim_compare:nNnTF {#2} < {#6}
492   {
493     \coffin_set_pole:Nnx #9 { B }
494     { { 0 pt } {#2} { 1000 pt } { 0 pt } }
495   }
496   {
497     \coffin_set_pole:Nnx #9 { B }
498     { { 0 pt } {#6} { 1000 pt } { 0 pt } }

```

```

499     }
500 }

```

(End definition for `\coffin_update_vertical_poles:NNN`. 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!

```

501 \cs_new_protected_nopar:Npn \coffin_typeset:Nnnnn #1#2#3#4#5 {
502   \coffin_align:NnnNnnnnN \c_empty_coffin { H } { 1 }
503   #1 {#2} {#3} {#4} {#5} \l_coffin_aligned_coffin
504   \hbox_set:Nn \l_coffin_aligned_coffin
505     {
506       \dim_compare:nNnT { \l_coffin_offset_x_dim } < { \c_zero_dim }
507       { \tex_kern:D -\l_coffin_offset_x_dim }
508       \hbox_unpack:N \l_coffin_aligned_coffin
509       \dim_set:Nn \l_coffin_tmp_dim
510       { \l_coffin_offset_x_dim + \box_wd:N #1 }
511       \dim_compare:nNnT { \l_coffin_tmp_dim } < { \c_zero_dim }
512       { \tex_kern:D -\l_coffin_tmp_dim }
513     }
514   \hbox_unpack:N \c_empty_box
515   \box_use:N \l_coffin_aligned_coffin
516 }
517 \cs_generate_variant:Nn \coffin_typeset:Nnnnn { c }

```

(End definition for `\coffin_typeset:Nnnnn`, `\coffin_typeset:cnnnn`, and `.`. These functions are documented on page 4.)

3.6 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.

```

518 \prop_new:N \l_coffin_bounding_prop

```

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

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

```

519 \dim_new:N \l_coffin_bounding_shift_dim

```

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

`\l_coffin_left_corner_dim`
`\l_coffin_right_corner_dim`
`\l_coffin_bottom_corner_dim`
`\l_coffin_top_corner_dim`

These are used to hold maxima for the various corner values: these thus define the minimum size of the bounding box after rotation.

```

520 \dim_new:N \l_coffin_left_corner_dim
521 \dim_new:N \l_coffin_right_corner_dim
522 \dim_new:N \l_coffin_bottom_corner_dim
523 \dim_new:N \l_coffin_top_corner_dim

```

(End definition for \l_coffin_left_corner_dim. This function is documented on page ??.)

\coffin_rotate:Nn Rotating a coffin requires several steps which can be conveniently run together. The 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.

\coffin_rotate:cn

```

524 \cs_new_protected_nopar:Npn \coffin_rotate:Nn #1#2 {
525   \fp_set:Nn \l_coffin_tmp_fp {#2}
526   \fp_div:Nn \l_coffin_tmp_fp { 180 }
527   \fp_mul:Nn \l_coffin_tmp_fp { \c_pi_fp }
528   \fp_sin:Nn \l_coffin_sin_fp { \l_coffin_tmp_fp }
529   \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.

```

530 \prop_map_inline:cn { l_coffin_corners_ \tex_number:D #1 _prop }
531   { \coffin_rotate_corner:Nnnn #1 {##1} ##2 }
532 \prop_map_inline:cn { l_coffin_poles_ \tex_number:D #1 _prop }
533   { \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.

```

534 \coffin_set_bounding:N #1
535 \prop_map_inline:Nn \l_coffin_bounding_prop
536   { \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.

```

537 \coffin_find_corner_maxima:N #1
538 \coffin_find_bounding_shift:
539 \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.

```

540 \hbox_set:Nn #1
541   {

```

```

542 \tex_kern:D \l_coffin_bounding_shift_dim
543 \tex_kern:D -\l_coffin_left_corner_dim
544 \box_move_down:nn { \l_coffin_bottom_corner_dim }
545 { \box_use:N #1 }
546 }

```

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.

```

547 \box_set_ht:Nn #1
548 { \l_coffin_top_corner_dim - \l_coffin_bottom_corner_dim }
549 \box_set_dp:Nn #1 { 0 pt }
550 \box_set_wd:Nn #1
551 { \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.

```

552 \prop_map_inline:cn { l_coffin_corners_ \tex_number:D #1 _prop }
553 { \coffin_shift_corner:Nnnn #1 {##1} ##2 }
554 \prop_map_inline:cn { l_coffin_poles_ \tex_number:D #1 _prop }
555 { \coffin_shift_pole:Nnnnnn #1 {##1} ##2 }
556 }
557 \cs_generate_variant:Nn \coffin_rotate:Nn { c }

```

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

`\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.

```

558 \cs_new_protected_nopar:Npn \coffin_set_bounding:N #1 {
559 \prop_put:Nnx \l_coffin_bounding_prop { tl }
560 { { 0 pt } { \dim_use:N \box_ht:N #1 } }
561 \prop_put:Nnx \l_coffin_bounding_prop { tr }
562 { { \dim_use:N \box_wd:N #1 } { \dim_use:N \box_ht:N #1 } }
563 \dim_set:Nn \l_coffin_tmp_dim { - \dim_use:N \box_dp:N #1 }
564 \prop_put:Nnx \l_coffin_bounding_prop { bl }
565 { { 0 pt } { \dim_use:N \l_coffin_tmp_dim } }
566 \prop_put:Nnx \l_coffin_bounding_prop { br }
567 { { \dim_use:N \box_wd:N #1 } { \dim_use:N \l_coffin_tmp_dim } }
568 }

```

(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.

```

569 \cs_new_protected_nopar:Npn \coffin_rotate_bounding:nnn #1#2#3 {
570   \coffin_rotate_vector:nnNN {#2} {#3} \l_coffin_x_dim \l_coffin_y_dim
571   \prop_put:Nnx \l_coffin_bounding_prop {#1}
572   { { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim } }
573 }
574 \cs_new_protected_nopar:Npn \coffin_rotate_corner:Nnnn #1#2#3#4 {
575   \coffin_rotate_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
576   \prop_put:cnx { \l_coffin_corners_ \tex_number:D #1 _prop } {#2}
577   { { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim } }
578 }

```

(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.

```

579 \cs_new_protected_nopar:Npn \coffin_rotate_pole:Nnnnnn #1#2#3#4#5#6 {
580   \coffin_rotate_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
581   \coffin_rotate_vector:nnNN {#5} {#6}
582   \l_coffin_x_prime_dim \l_coffin_y_prime_dim
583   \coffin_set_pole:Nnx #1 {#2}
584   {
585     { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim }
586     { \dim_use:N \l_coffin_x_prime_dim }
587     { \dim_use:N \l_coffin_y_prime_dim }
588   }
589 }

```

(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.

```

590 \cs_new_protected_nopar:Npn \coffin_rotate_vector:nnNN #1#2#3#4 {
591   \fp_set_from_dim:Nn \l_coffin_x_fp {#1}
592   \fp_set_from_dim:Nn \l_coffin_y_fp {#2}
593   \fp_set_eq:NN \l_coffin_x_prime_fp \l_coffin_x_fp
594   \fp_set_eq:NN \l_coffin_y_prime_fp \l_coffin_y_fp
595   \fp_mul:Nn \l_coffin_x_prime_fp { \l_coffin_cos_fp }
596   \fp_mul:Nn \l_coffin_y_prime_fp { \l_coffin_sin_fp }
597   \fp_sub:Nn \l_coffin_x_prime_fp { \l_coffin_tmp_fp }
598   \fp_set_eq:NN \l_coffin_y_prime_fp \l_coffin_y_fp
599   \fp_set_eq:NN \l_coffin_x_prime_fp \l_coffin_x_fp
600   \fp_mul:Nn \l_coffin_y_prime_fp { \l_coffin_cos_fp }
601   \fp_mul:Nn \l_coffin_x_prime_fp { \l_coffin_sin_fp }
602   \fp_add:Nn \l_coffin_y_prime_fp { \l_coffin_tmp_fp }
603   \dim_set:Nn #3 { \fp_to_dim:N \l_coffin_x_prime_fp }
604   \dim_set:Nn #4 { \fp_to_dim:N \l_coffin_y_prime_fp }
605 }

```


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

`\coffin_find_corner_maxima:N` The idea here is to find the extremities of the content of the coffin. This is done by
`\coffin_find_corner_maxima_aux:nn` 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.

```

606 \cs_new_protected_nopar:Npn \coffin_find_corner_maxima:N #1 {
607   \dim_set:Nn \l_coffin_top_corner_dim { -\c_max_dim }
608   \dim_set:Nn \l_coffin_right_corner_dim { -\c_max_dim }
609   \dim_set:Nn \l_coffin_bottom_corner_dim { \c_max_dim }
610   \dim_set:Nn \l_coffin_left_corner_dim { \c_max_dim }
611   \prop_map_inline:cn { l_coffin_corners_ \tex_number:D #1 _prop }
612     { \coffin_find_corner_maxima_aux:nn ##2 }
613 }
614 \cs_new_protected_nopar:Npn \coffin_find_corner_maxima_aux:nn #1#2 {
615   \dim_set_min:Nn \l_coffin_left_corner_dim {#1}
616   \dim_set_max:Nn \l_coffin_right_corner_dim {#1}
617   \dim_set_min:Nn \l_coffin_bottom_corner_dim {#2}
618   \dim_set_max:Nn \l_coffin_top_corner_dim {#2}
619 }

```

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

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

```

620 \cs_new_protected_nopar:Npn \coffin_find_bounding_shift: {
621   \dim_set:Nn \l_coffin_bounding_shift_dim { \c_max_dim }
622   \prop_map_inline:Nn \l_coffin_bounding_prop
623     { \coffin_find_bounding_shift_aux:nn ##2 }
624 }
625 \cs_new_protected_nopar:Npn \coffin_find_bounding_shift_aux:nn #1#2 {
626   \dim_set_min:Nn \l_coffin_bounding_shift_dim {#1}
627 }

```

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

`\coffin_shift_corner:Nnnn` Shifting the corners and poles of a coffin means subtracting the appropriate values from
`\coffin_shift_pole:Nnnnnn` the x - and y -components. For the poles, this means that the direction vector is un-
changed.

```

628 \cs_new_protected_nopar:Npn \coffin_shift_corner:Nnnn #1#2#3#4 {
629   \prop_put:cnx { l_coffin_corners_ \tex_number:D #1 _prop } {#2}
630   {
631     { \dim_eval:n { #3 - \l_coffin_left_corner_dim } }
632     { \dim_eval:n { #4 - \l_coffin_bottom_corner_dim } }
633   }

```

```

634 }
635 \cs_new_protected_nopar:Npn \coffin_shift_pole:Nnnnnn #1#2#3#4#5#6 {
636   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _ prop } {#2}
637   {
638     { \dim_eval:n { #3 - \l_coffin_left_corner_dim } }
639     { \dim_eval:n { #4 - \l_coffin_bottom_corner_dim } }
640     {#5} {#6}
641   }
642 }

```

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

3.7 Resizing coffins

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

```

643 \fp_new:N \l_coffin_scale_x_fp
644 \fp_new:N \l_coffin_scale_y_fp

```

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

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

```

645 \dim_new:N \l_coffin_scaled_total_height_dim
646 \dim_new:N \l_coffin_scaled_width_dim

```

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

`\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.

```

647 \cs_new_protected_nopar:Npn \coffin_resize:Nnn #1#2#3 {
648   \coffin_set_user_dimensions:N #1
649   \fp_set_from_dim:Nn \l_coffin_scale_x_fp {#2}
650   \fp_set_from_dim:Nn \l_coffin_tmp_fp { \Width }
651   \fp_div:Nn \l_coffin_scale_x_fp { \l_coffin_tmp_fp }
652   \fp_set_from_dim:Nn \l_coffin_scale_y_fp {#3}
653   \fp_set_from_dim:Nn \l_coffin_tmp_fp { \TotalHeight }
654   \fp_div:Nn \l_coffin_scale_y_fp { \l_coffin_tmp_fp }
655   \hbox_set:Nn #1 { \resizebox * {#2} {#3} { \box_use:N #1 } }
656   \coffin_resize_common:Nnn #1 {#2} {#3}
657 }
658 \cs_generate_variant:Nn \coffin_resize:Nnn { c }

```

(End definition for `\coffin_resize:Nnn`, `\coffin_resize:cnn`, and `.`. 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.

```

659 \cs_new_protected_nopar:Npn \coffin_resize_common:Nnn #1#2#3 {
660   \prop_map_inline:cn { l_coffin_corners_ \tex_number:D #1 _prop }
661     { \coffin_scale_corner:Nnnn #1 {##1} ##2 }
662   \prop_map_inline:cn { l_coffin_poles_ \tex_number:D #1 _prop }
663     { \coffin_scale_pole:Nnnnnn #1 {##1} ##2 }

```

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

```

664   \fp_compare:NNNT \l_coffin_scale_x_fp < \c_zero_fp
665   {
666     \prop_map_inline:cn { l_coffin_corners_ \tex_number:D #1 _prop }
667       { \coffin_x_shift_corner:Nnnn #1 {##1} ##2 }
668     \prop_map_inline:cn { l_coffin_poles_ \tex_number:D #1 _prop }
669       { \coffin_x_shift_pole:Nnnnnn #1 {##1} ##2 }
670   }
671   \coffin_end_user_dimensions:
672 }

```

(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 TeX way as this works properly with floating point values without needing to use the `fp` module.

```

673 \cs_new_protected_nopar:Npn \coffin_scale:Nnn #1#2#3 {
674   \coffin_set_user_dimensions:N #1
675   \fp_set:Nn \l_coffin_scale_x_fp {#2}
676   \fp_set:Nn \l_coffin_scale_y_fp {#3}
677   \fp_compare:NNNTF \l_coffin_scale_y_fp > \c_zero_fp
678     { \l_coffin_scaled_total_height_dim #3 \TotalHeight }
679     { \l_coffin_scaled_total_height_dim -#3 \TotalHeight }
680   \fp_compare:NNNTF \l_coffin_scale_x_fp > \c_zero_fp
681     { \l_coffin_scaled_width_dim -#2 \Width }
682     { \l_coffin_scaled_width_dim #2 \Width }
683   \hbox_set:Nn #1 { \scalebox {#2} [#3] { \box_use:N #1 } }
684   \coffin_resize_common:Nnn #1
685     { \l_coffin_scaled_width_dim } { \l_coffin_scaled_total_height_dim }
686 }
687 \cs_generate_variant:Nn \coffin_scale:Nnn { c }

```

(End definition for `\coffin_scale:Nnn`, `\coffin_scale:cnn`, and `.`. 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.

```

688 \cs_new_protected_nopar:Npn \coffin_scale_vector:nnNN #1#2#3#4 {
689   \fp_set_from_dim:Nn \l_coffin_tmp_fp {#1}
690   \fp_mul:Nn \l_coffin_tmp_fp { \l_coffin_scale_x_fp }
691   \dim_set:Nn #3 { \fp_to_dim:N \l_coffin_tmp_fp }
692   \fp_set_from_dim:Nn \l_coffin_tmp_fp {#2}
693   \fp_mul:Nn \l_coffin_tmp_fp { \l_coffin_scale_y_fp }
694   \dim_set:Nn #4 { \fp_to_dim:N \l_coffin_tmp_fp }
695 }

```

(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.

```

696 \cs_new_protected_nopar:Npn \coffin_scale_corner:Nnnn #1#2#3#4 {
697   \coffin_scale_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
698   \prop_put:cnx { l_coffin_corners_ \tex_number:D #1 _prop } {#2}
699   { { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim } }
700 }
701 \cs_new_protected_nopar:Npn \coffin_scale_pole:Nnnnnn #1#2#3#4#5#6 {
702   \coffin_scale_vector:nnNN {#3} {#4} \l_coffin_x_dim \l_coffin_y_dim
703   \coffin_set_pole:Nnx #1 {#2}
704   {
705     { \dim_use:N \l_coffin_x_dim } { \dim_use:N \l_coffin_y_dim }
706     {#5} {#6}
707   }
708 }

```

(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.

```

709 \cs_new_protected_nopar:Npn \coffin_x_shift_corner:Nnnn #1#2#3#4 {
710   \prop_put:cnx { l_coffin_corners_ \tex_number:D #1 _prop } {#2}
711   {
712     { \the_tex:D \etex_dimexpr:D #3 + \box_wd:N #1 \scan_stop: } {#4}
713   }
714 }
715 \cs_new_protected_nopar:Npn \coffin_x_shift_pole:Nnnnnn #1#2#3#4#5#6 {
716   \prop_put:cnx { l_coffin_poles_ \tex_number:D #1 _prop } {#2}
717   {
718     { \the_tex:D \etex_dimexpr:D #3 + \box_wd:N #1 \scan_stop: } {#4}
719     {#5} {#6}
720   }
721 }

```

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

3.8 Coffin diagnostics

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

```
\l_coffin_display_coord_coffin
\l_coffin_display_pole_coffin
722 \coffin_new:N \l_coffin_display_coffin
723 \coffin_new:N \l_coffin_display_coord_coffin
724 \coffin_new:N \l_coffin_display_pole_coffin
```

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

`\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.

```
725 \prop_new:N \l_coffin_display_handles_prop
726 \prop_put:Nnn \l_coffin_display_handles_prop { tl }
727 { { b } { r } { -1 } { 1 } }
728 \prop_put:Nnn \l_coffin_display_handles_prop { thc }
729 { { b } { hc } { 0 } { 1 } }
730 \prop_put:Nnn \l_coffin_display_handles_prop { tr }
731 { { b } { l } { 1 } { 1 } }
732 \prop_put:Nnn \l_coffin_display_handles_prop { vcl }
733 { { vc } { r } { -1 } { 0 } }
734 \prop_put:Nnn \l_coffin_display_handles_prop { vhc }
735 { { vc } { hc } { 0 } { 0 } }
736 \prop_put:Nnn \l_coffin_display_handles_prop { vcr }
737 { { vc } { l } { 1 } { 0 } }
738 \prop_put:Nnn \l_coffin_display_handles_prop { bl }
739 { { t } { r } { -1 } { -1 } }
740 \prop_put:Nnn \l_coffin_display_handles_prop { bhc }
741 { { t } { hc } { 0 } { -1 } }
742 \prop_put:Nnn \l_coffin_display_handles_prop { br }
743 { { t } { l } { 1 } { -1 } }
744 \prop_put:Nnn \l_coffin_display_handles_prop { Tl }
745 { { t } { r } { -1 } { -1 } }
746 \prop_put:Nnn \l_coffin_display_handles_prop { Thc }
747 { { t } { hc } { 0 } { -1 } }
748 \prop_put:Nnn \l_coffin_display_handles_prop { Tr }
749 { { t } { l } { 1 } { -1 } }
750 \prop_put:Nnn \l_coffin_display_handles_prop { Hl }
751 { { vc } { r } { -1 } { 1 } }
752 \prop_put:Nnn \l_coffin_display_handles_prop { Hhc }
753 { { vc } { hc } { 0 } { 1 } }
754 \prop_put:Nnn \l_coffin_display_handles_prop { Hr }
755 { { vc } { l } { 1 } { 1 } }
756 \prop_put:Nnn \l_coffin_display_handles_prop { Bl }
757 { { b } { r } { -1 } { -1 } }
758 \prop_put:Nnn \l_coffin_display_handles_prop { Bhc }
759 { { b } { hc } { 0 } { -1 } }
760 \prop_put:Nnn \l_coffin_display_handles_prop { Br }
761 { { b } { l } { 1 } { -1 } }
```

(End definition for \l_coffin_display_handles_prop. This function is documented on page ??.)

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

```
762 \dim_new:N \l_coffin_display_offset_dim
763 \dim_set:Nn \l_coffin_display_offset_dim { 2 pt }
```

(End definition for \l_coffin_display_offset_dim. This function is documented on page ??.)

`\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.

```
764 \dim_new:N \l_coffin_display_x_dim
765 \dim_new:N \l_coffin_display_y_dim
```

(End definition for \l_coffin_display_x_dim. This function is documented on page ??.)

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

```
766 \prop_new:N \l_coffin_display_poles_prop
```

(End definition for \l_coffin_display_poles_prop. This function is documented on page ??.)

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

```
767 \tl_new:N \l_coffin_display_font_tl
768 \tl_set:Nn \l_coffin_display_font_tl { \sffamily \tiny }
```

(End definition for \l_coffin_display_font_tl. This function is documented on page ??.)

`\l_coffin_show_toks` For the package the L^AT_EX 2_ε code can be used.

```
769 \cs_new_eq:NN \l_coffin_show_toks \toks0
```

(End definition for \l_coffin_show_toks. This function is documented on page ??.)

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

```
770 \prop_new:N \l_coffin_handles_tmp_prop
```

(End definition for \l_coffin_handles_tmp_prop. This function is documented on page ??.)

`\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.

```

771 \cs_new_protected_nopar:Npn \coffin_mark_handle:Nnnn #1#2#3#4 {
772   \hcoffin_set:Nn \l_coffin_display_pole_coffin
773   {
774     \color {#4}
775     \rule { 1 pt } { 1 pt }
776     \hbox:n { \tex_vrule:D width 1 pt height 1 pt \scan_stop: } % TEMP
777   }
778   \coffin_attach_mark:NnnNnnnn #1 {#2} {#3}
779   \l_coffin_display_pole_coffin { hc } { vc } { 0 pt } { 0 pt }
780   \hcoffin_set:Nn \l_coffin_display_coord_coffin
781   {
782     \color {#4}
783     \l_coffin_display_font_tl
784     ( \tl_to_str:n { #2 , #3 } )
785   }
786   \prop_get:NnN \l_coffin_display_handles_prop
787   { #2 #3 } \l_coffin_tmp_tl
788   \quark_if_no_value:NTF \l_coffin_tmp_tl
789   {
790     \prop_get:NnN \l_coffin_display_handles_prop
791     { #3 #2 } \l_coffin_tmp_tl
792     \quark_if_no_value:NTF \l_coffin_tmp_tl
793     {
794       \coffin_attach_mark:NnnNnnnn #1 {#2} {#3}
795       \l_coffin_display_coord_coffin { l } { vc }
796       { 1 pt } { 0 pt }
797     }
798     {
799       \exp_last_unbraced:No \coffin_mark_handle_aux:nnnnNnn
800       \l_coffin_tmp_tl #1 {#2} {#3}
801     }
802   }
803   {
804     \exp_last_unbraced:No \coffin_mark_handle_aux:nnnnNnn
805     \l_coffin_tmp_tl #1 {#2} {#3}
806   }
807 }
808 \cs_new_protected_nopar:Npn
809   \coffin_mark_handle_aux:nnnnNnn #1#2#3#4#5#6#7 {
810   \coffin_attach_mark:NnnNnnnn #5 {#6} {#7}
811   \l_coffin_display_coord_coffin {#1} {#2}
812   { #3 \l_coffin_display_offset_dim }
813   { #4 \l_coffin_display_offset_dim }
814 }
815 \cs_generate_variant:Nn \coffin_mark_handle:Nnnn { c }

```

(End definition for `\coffin_mark_handle:Nnnn`, `\coffin_mark_handle:cnnn`, and `.` These functions are documented on page 4.)

`\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.

```

816 \cs_new_protected_nopar:Npn \coffin_display_handles:Nn #1#2 {
817   \hcoffin_set:Nn \l_coffin_display_pole_coffin
818   {
819     \color {#2}
820     \rule { 1 pt } { 1 pt }
821   }
822   \prop_set_eq:Nc \l_coffin_display_poles_prop
823   { l_coffin_poles_ \tex_number:D #1 _prop }
824   \coffin_get_pole:NnN #1 { H } \l_coffin_pole_a_tl
825   \coffin_get_pole:NnN #1 { T } \l_coffin_pole_b_tl
826   \tl_if_eq:NNT \l_coffin_pole_a_tl \l_coffin_pole_b_tl
827   { \prop_del:Nn \l_coffin_display_poles_prop { T } }
828   \coffin_get_pole:NnN #1 { B } \l_coffin_pole_b_tl
829   \tl_if_eq:NNT \l_coffin_pole_a_tl \l_coffin_pole_b_tl
830   { \prop_del:Nn \l_coffin_display_poles_prop { B } }
831   \coffin_set_eq:NN \l_coffin_display_coffin #1
832   \prop_clear:N \l_coffin_handles_tmp_prop
833   \prop_map_inline:Nn \l_coffin_display_poles_prop
834   {
835     \prop_del:Nn \l_coffin_display_poles_prop {##1}
836     \coffin_display_handles_aux:nnnnnn {##1} ##2 {#2}
837   }
838   \box_use:N \l_coffin_display_coffin
839 }

```

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.

```

840 \cs_new_protected_nopar:Npn
841   \coffin_display_handles_aux:nnnnnn #1#2#3#4#5#6 {
842   \prop_map_inline:Nn \l_coffin_display_poles_prop
843   {
844     \bool_set_false:N \l_coffin_error_bool
845     \coffin_calculate_intersection:nnnnnnnn {#2} {#3} {#4} {#5} ##2
846     \bool_if:NF \l_coffin_error_bool
847     {
848       \dim_set:Nn \l_coffin_display_x_dim { \l_coffin_x_dim }
849       \dim_set:Nn \l_coffin_display_y_dim { \l_coffin_y_dim }
850       \coffin_display_attach:Nnnnn
851       \l_coffin_display_pole_coffin { hc } { vc }
852       { 0 pt } { 0 pt }
853       \hcoffin_set:Nn \l_coffin_display_coord_coffin
854       {
855         \color {#6}
856         \l_coffin_display_font_tl

```



```

857         ( \tl_to_str:n { #1 , ##1 } )
858     }
859     \prop_get:NnN \l_coffin_display_handles_prop
860     { #1 ##1 } \l_coffin_tmp_tl
861     \quark_if_no_value:NTF \l_coffin_tmp_tl
862     {
863         \prop_get:NnN \l_coffin_display_handles_prop
864         { ##1 #1 } \l_coffin_tmp_tl
865         \quark_if_no_value:NTF \l_coffin_tmp_tl
866         {
867             \coffin_display_attach:Nnnnn
868             \l_coffin_display_coord_coffin { 1 } { vc }
869             { 1 pt } { 0 pt }
870         }
871         {
872             \exp_last_unbraced:No
873             \coffin_display_handles_aux:nnnn
874             \l_coffin_tmp_tl
875         }
876     }
877     {
878         \exp_last_unbraced:No \coffin_display_handles_aux:nnnn
879         \l_coffin_tmp_tl
880     }
881 }
882 }
883 }
884 \cs_new_protected_nopar:Npn \coffin_display_handles_aux:nnnn #1#2#3#4 {
885     \coffin_display_attach:Nnnnn
886     \l_coffin_display_coord_coffin {#1} {#2}
887     { #3 \l_coffin_display_offset_dim }
888     { #4 \l_coffin_display_offset_dim }
889 }
890 \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.

```

891 \cs_new_protected_nopar:Npn \coffin_display_attach:Nnnnn #1#2#3#4#5 {
892     \coffin_calculate_intersection:Nnn #1 {#2} {#3}
893     \dim_set:Nn \l_coffin_x_prime_dim { \l_coffin_x_dim }
894     \dim_set:Nn \l_coffin_y_prime_dim { \l_coffin_y_dim }
895     \dim_set:Nn \l_coffin_offset_x_dim
896     { \l_coffin_display_x_dim - \l_coffin_x_prime_dim + #4 }
897     \dim_set:Nn \l_coffin_offset_y_dim
898     { \l_coffin_display_y_dim - \l_coffin_y_prime_dim + #5 }
899     \hbox_set:Nn \l_coffin_aligned_coffin
900     {
901         \box_use:N \l_coffin_display_coffin

```

```

902 \tex_kern:D -\box_wd:N \l_coffin_display_coffin
903 \tex_kern:D \l_coffin_offset_x_dim
904 \box_move_up:nn { \l_coffin_offset_y_dim } { \box_use:N #1 }
905 }
906 \box_set_ht:Nn \l_coffin_aligned_coffin
907 { \box_ht:N \l_coffin_display_coffin }
908 \box_set_dp:Nn \l_coffin_aligned_coffin
909 { \box_dp:N \l_coffin_display_coffin }
910 \box_set_wd:Nn \l_coffin_aligned_coffin
911 { \box_wd:N \l_coffin_display_coffin }
912 \box_set_eq:NN \l_coffin_display_coffin \l_coffin_aligned_coffin
913 }

```

(End definition for `\coffin_display_handles:Nn`, `\coffin_display_handles:cn`, and `.` These functions are documented on page 4.)

`\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`

```

914 \cs_new_protected_nopar:Npn \coffin_show_structure:N #1 {
915   \toks_clear:N \l_coffin_show_toks
916   \cs_if_exist:cTF { l_coffin_poles_ \tex_number:D #1 _prop }
917   {
918     \iow_term:x
919     {
920       \iow_newline:
921       Size~of~coffin~\token_to_str:N #1 : \iow_newline:
922       > ~ ht~==~\dim_use:N \box_ht:N #1 \iow_newline:
923       > ~ dp~==~\dim_use:N \box_dp:N #1 \iow_newline:
924       > ~ wd~==~\dim_use:N \box_wd:N #1 \iow_newline:
925     }
926     \iow_term:x { Poles-of~coffin~\token_to_str:N #1 : }
927     \prop_map_inline:cn { l_coffin_poles_ \tex_number:D #1 _prop }
928     {
929       \toks_if_empty:NF \l_coffin_show_toks
930       {
931         \toks_put_right:Nx \l_coffin_show_toks
932         { \iow_newline: > ~ }
933       }
934       \toks_put_right:Nx \l_coffin_show_toks
935       { ~ \exp_not:n {##1} \c_space_tl => ~ \exp_not:n {##2} }
936     }
937   }
938   {
939     \iow_term:x { ---~No~poles~found~--- }
940     \toks_put_right:Nn \l_coffin_show_toks
941     { Is~this~really~a~coffin? }
942   }
943   \toks_show:N \l_coffin_show_toks
944 }

```

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

(End definition for `\coffin_show_structure:N`. This function is documented on page 4.)

3.9 Messages

```
946 \msg_kernel_new:nnnn { coffin } { no-pole-intersection }
947 { No~intersection~between~coffin~poles. }
948 {
949   \l_msg_coding_error_text_tl
950   LaTeX~was~asked~to~find~the~intersection~between~two~poles,~
951   but~they~do~not~have~a~unique~meeting~point:~
952   the~value~(0~pt,~0~pt)~will~be~used.
953 }
954 \msg_kernel_new:nnnn { coffin } { unknown-coffin }
955 { Unknown~coffin~'#1'. }
956 { The~coffin~'#1'~was~never~defined. }
957 \msg_kernel_new:nnnn { coffin } { unknown-coffin-pole }
958 { Pole~'#1'~unknown~for~coffin~'#2'. }
959 {
960   \l_msg_coding_error_text_tl
961   LaTeX~was~asked~to~find~a~typesetting~pole~for~a~coffin,~
962   but~either~the~coffin~does~not~exist~or~the~pole~name~is~wrong.
963 }
```

```
\exp_two_last_unbraced:Noo
```

```
964 \cs_new:Npn \exp_last_two_unbraced:Noo #1 #2 #3{
965   \exp_after:wN \exp_after:wN \exp_after:wN #1 \exp_after:wN #2 #3
966 }
```

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

```
967 \endpackage
```

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>\box_clear:N</code>	72
<code>\bool_if:NF</code>	846	<code>\box_dp:N</code>	172, 173,
<code>\bool_if:NT</code>	247		217, 229, 234, 394, 412, 563, 909, 923
<code>\bool_new:N</code>	31	<code>\box_ht:N</code>	121, 171, 173, 214, 216,
<code>\bool_set_false:N</code>	243, 844		229, 233, 393, 411, 560, 562, 907, 922
<code>\bool_set_true:N</code>	260, 275, 308	<code>\box_move_down:nn</code>	544

\box_move_up:nn	432, 904	\coffin_find_bounding_shift:	538, 620, 620
\box_new:N	4, 78	\coffin_find_bounding_shift_aux:nn	620, 623, 625
\box_set_dp:Nn	394, 412, 549, 908	\coffin_find_corner_maxima:N	537, 606, 606
\box_set_eq:NN	132, 414, 912	\coffin_find_corner_maxima_aux:nn	606, 612, 614
\box_set_ht:Nn	393, 411, 547, 906	\coffin_get_pole:NnN	141, 141, 241, 242, 468, 469, 472, 473, 824, 825, 828
\box_set_wd:Nn	395, 413, 550, 910	\coffin_if_exist_execute:Nn	54, 54, 70, 90, 106, 130, 183, 195
\box_use:N	429, 432, 515, 539, 545, 655, 683, 838, 901, 904	\coffin_join:cnncnnnn	3, 352
\box_wd:N	174, 216, 221, 224, 228, 361, 395, 413, 430, 510, 562, 567, 712, 718, 902, 911, 924	\coffin_join:cnnNnnnn	3, 352
C			
\c_coffin_corners_prop	8, 8, 10–13, 84, 152	\coffin_join:Nnncnnnn	3, 352
\c_coffin_poles_prop	14, 14, 16–18, 20–25, 86, 154	\coffin_join:NnnNnnnn	3, 352, 352, 389
\c_empty_box	514	\coffin_mark_handle:cnncnn	4, 771
\c_empty_coffin	137, 137, 138, 502	\coffin_mark_handle:Nnnn	4, 771, 771, 815
\c_max_dim	607–610, 621	\coffin_mark_handle_aux:nnnnNnn	771, 799, 804, 809
\c_pi_fp	527	\coffin_new:c	1, 77
\c_space_tl	935	\coffin_new:N	1, 77, 77, 88, 137, 139, 140, 722–724
\c_zero_dim	235, 256, 259, 262, 271, 274, 277, 286, 293, 357, 362, 371, 506, 511	\coffin_offset_corner:Nnnnn	456, 458
\c_zero_fp	664, 677, 680	\coffin_offset_corners:Nnn	375, 376, 382, 383, 454, 454
\coffin_align:NnnNnnnnN	353, 391, 409, 417, 418, 502	\coffin_offset_corners:Nnnnn	454
\coffin_attach:cnncnnnn	3, 390	\coffin_offset_pole:Nnnnnnn	436, 438, 441
\coffin_attach:cnnNnnnn	3, 390	\coffin_offset_poles:Nnn	373, 374, 379, 380, 401, 402, 436, 436
\coffin_attach:Nnncnnnn	3, 390	\coffin_reset_structure:N	73, 99, 116, 150, 150, 365, 396
\coffin_attach:NnnNnnnn	3, 390, 390, 416	\coffin_resize:cnncnn	647
\coffin_attach_mark:NnnNnnnn	390, 408, 778, 794, 810	\coffin_resize:Nnn	647, 647, 658
\coffin_calculate_intersection:Nnn	240, 240, 419, 422, 892	\coffin_resize_common:Nnn	656, 659, 659, 684
\coffin_calculate_intersection:nnnnnnnn	240, 245, 255, 845	\coffin_rotate:cn	3, 524
\coffin_calculate_intersection_aux:nnnnnnN	240, 265, 280, 289, 296, 330, 340	\coffin_rotate:Nn	3, 524, 524, 557
\coffin_clear:c	2, 69	\coffin_rotate_bounding:nnn	536, 569, 569
\coffin_clear:N	2, 69, 69, 76	\coffin_rotate_corner:Nnnn	531, 569, 574
\coffin_display_attach:Nnnnn	816, 850, 867, 885, 891	\coffin_rotate_pole:Nnnnnn	533, 579, 579
\coffin_display_handles:cn	4, 816	\coffin_rotate_vector:nnNN	570, 575, 580, 581, 590, 590
\coffin_display_handles:Nn	816, 816, 890	\coffin_saved_Depth:	50, 50, 164, 178
\coffin_display_handles_aux:nnnn	816, 873, 878, 884	\coffin_saved_Height:	50, 51, 163, 177
\coffin_display_handles_aux:nnnnnn	816, 836, 841	\coffin_saved_TotalHeight:	50, 52, 165, 179
\coffin_end_user_dimensions:	162, 176, 191, 203, 671	\coffin_saved_Width:	50, 53, 166, 180
		\coffin_scale:cnncnn	673
		\coffin_scale:Nnn	673, 673, 687

\coffin_scale_corner:Nnnn .	661, 696, 696	\cs_new_protected_nopar:Npn	69, 77, 129, 150,
\coffin_scale_pole:Nnnnnn .	663, 696, 701	156, 162, 176, 182, 194, 206, 212,	
\coffin_scale_vector:nnNN		223, 240, 254, 339, 352, 390, 407,	
.....	688, 688, 697, 702	417, 436, 440, 454, 458, 467, 477,	
\coffin_set_bounding:N . . .	534, 558, 558	489, 501, 524, 558, 569, 574, 579,	
\coffin_set_eq:cc	2, 129	590, 606, 614, 620, 625, 628, 635,	
\coffin_set_eq:cN	2, 129	647, 659, 673, 688, 696, 701, 709,	
\coffin_set_eq:Nc	2, 129	715, 771, 808, 816, 840, 884, 891, 914	
\coffin_set_eq:NN		\cs_set_eq:NN	163–170, 177–180
..	2, 129, 129, 136, 387, 405, 434, 831	\cs_set_protected_nopar:Npn	141
\coffin_set_eq_structure:NN	133, 156, 156		
\coffin_set_horizontal_pole:cnn .	2, 182		
\coffin_set_horizontal_pole:Nnn			
.....	2, 182, 182, 209		
\coffin_set_pole:Nnn	182, 206, 211		
\coffin_set_pole:Nnx . .	122, 182, 186,		
	198, 447, 481, 485, 493, 497, 583, 703		
\coffin_set_user_dimensions:N			
.....	162, 162, 185, 197, 648, 674		
\coffin_set_vertical_pole:cnn . .	2, 182		
\coffin_set_vertical_pole:Nnn			
.....	2, 182, 194, 210		
\coffin_shift_corner:Nnnn .	553, 628, 628		
\coffin_shift_pole:Nnnnnn .	555, 628, 635		
\coffin_show_structure:c	4, 914		
\coffin_show_structure:N	4, 914, 914, 945		
\coffin_typeset:cnnnn	3, 501		
\coffin_typeset:Nnnnn . .	3, 501, 501, 517		
\coffin_update_B:nnnnnnnnN	467, 474, 490		
\coffin_update_corners:N			
.....	101, 118, 212, 212		
\coffin_update_poles:N			
.....	100, 117, 223, 223, 370, 400		
\coffin_update_T:nnnnnnnnN	467, 470, 478		
\coffin_update_vertical_poles:NNN . .			
.....	386, 404, 467, 467		
\coffin_x_shift_corner:Nnnn	667, 709, 709		
\coffin_x_shift_pole:Nnnnnn	669, 709, 715		
\color	774, 782, 819, 855		
\cs_generate_variant:Nn	76,		
	88, 104, 128, 136, 209–211, 389,		
	416, 517, 557, 658, 687, 815, 890, 945		
\cs_gundefine:c	79, 80		
\cs_if_exist:cTF	57, 916		
\cs_if_exist:NTF	55		
\cs_new:Npn	964		
\cs_new_eq:NN	769		
\cs_new_nopar:Npn	50–53		
\cs_new_protected:Npn	54, 89, 105		

D

\Depth	162, 164, 168, 172, 178
\dim_compare:nNnF	235
\dim_compare:nNnT	357, 362, 506, 511
\dim_compare:nNnTF	256, 259, 262,
	271, 274, 277, 286, 293, 371, 479, 491
\dim_eval:n	
	188, 200, 463, 464, 631, 632, 638, 639
\dim_new:N	5, 32, 33, 38–41,
	46–49, 519–523, 645, 646, 762, 764, 765
\dim_set:Nn	110, 120, 171–
	174, 217, 224, 229, 234, 236, 258,
	263, 273, 278, 288, 295, 328, 350,
	360, 420, 421, 423, 425, 442, 443,
	509, 563, 603, 604, 607–610, 621,
	691, 694, 763, 848, 849, 893–895, 897
\dim_set_max:Nn	616, 618
\dim_set_min:Nn	615, 617, 626
\dim_use:N	124, 214, 216,
	217, 219, 221, 226, 228, 231, 233,
	238, 450, 560, 562, 563, 565, 567,
	572, 577, 585–587, 699, 705, 922–924
\dim_zero:N	250, 251

E

\etex_dimexpr:D	712, 718
\exp_after:wN	965
\exp_last_two_unbraced:Noo	
	244, 470, 474, 964
\exp_last_unbraced:NNo	447
\exp_last_unbraced:No	799, 804, 872, 878
\exp_not:n	935
\exp_two_last_unbraced:Noo	964

F

\filedate	3
\filedescription	3
\filename	3

\fileversion	3	\l_coffin_bounding_shift_dim	519, 519, 542, 621, 626
\fp_add:Nn	317, 349, 602	\l_coffin_calc_a_fp .	26, 26, 300, 304,
\fp_compare:NNNT	664	311, 313–315, 318–320, 323, 342, 346	
\fp_compare:NNNTF	677, 680	\l_coffin_calc_b_fp	26, 27,
\fp_compare:nNnTF	306	301, 304, 307, 316, 324, 327, 343, 349	
\fp_cos:Nn	529	\l_coffin_calc_c_fp	26, 28, 302, 305, 344, 348
\fp_div:Nn 304, 305, 326, 347, 526, 651, 654		\l_coffin_calc_d_fp	26,
\fp_mul:Nn	315, 320,	29, 303, 305, 307, 321, 325, 345, 347	
348, 527, 595, 596, 600, 601, 690, 693		\l_coffin_calc_result_fp .	26, 30, 310,
\fp_new:N 6, 26–30, 36, 37, 42–45, 643, 644		312, 317, 322, 326, 329, 341, 346–350	
\fp_set:Nn	525, 675, 676	\l_coffin_cos_fp ...	36, 37, 529, 595, 600
\fp_set_eq:NN	593, 594, 598, 599	\l_coffin_Depth_dim	46, 46, 168
\fp_set_from_dim:Nn	300–	\l_coffin_display_coffin ...	722, 722,
303, 310, 311, 314, 319, 341–345,		831, 838, 901, 902, 907, 909, 911, 912	
591, 592, 649, 650, 652, 653, 689, 692		\l_coffin_display_coord_coffin	722, 723, 780, 795, 811, 853, 868, 886
\fp_sin:Nn	528	\l_coffin_display_font_tl	767, 767, 768, 783, 856
\fp_sub:Nn	312, 322, 324, 346, 597	\l_coffin_display_handles_prop	725, 725,
\fp_to_dim:N ..	329, 350, 603, 604, 691, 694	726, 728, 730, 732, 734, 736, 738,	
		740, 742, 744, 746, 748, 750, 752,	
		754, 756, 758, 760, 786, 790, 859, 863	
		\l_coffin_display_offset_dim	762, 762, 763, 812, 813, 887, 888
		\l_coffin_display_pole_coffin	722, 724, 772, 779, 817, 851
		\l_coffin_display_poles_prop	766, 766, 822, 827, 830, 833, 835, 842
		\l_coffin_display_x_dim	764, 764, 848, 896
		\l_coffin_display_y_dim	764, 765, 849, 898
		\l_coffin_error_bool	31,
		31, 243, 247, 260, 275, 308, 844, 846	
		\l_coffin_handles_tmp_prop	770, 770, 832
		\l_coffin_Height_dim	46, 47, 167
		\l_coffin_left_corner_dim	520, 520, 543, 551, 610, 615, 631, 638
		\l_coffin_offset_x_dim ..	32, 32, 357,
		358, 361, 371, 373, 375, 381, 384,	
		403, 423, 431, 506, 507, 510, 895, 903	
		\l_coffin_offset_y_dim ..	32, 33, 374,
		376, 381, 384, 403, 425, 432, 897, 904	
		\l_coffin_pole_a_tl	34, 34, 241,
		246, 468, 471, 472, 475, 824, 826, 829	
		\l_coffin_pole_b_tl .	34, 35, 242, 246,
		469, 471, 473, 475, 825, 826, 828, 829	
		\l_coffin_right_corner_dim	520, 521, 551, 608, 616
G			
\group_begin:	94, 111		
\group_end:	97, 114		
H			
\hbox:n	776		
\hbox_set:Nn	92, 138,		
355, 427, 504, 539, 540, 655, 683, 899			
\hbox_unpack:N	359, 508, 514		
\hcoffin_set:cn	2, 89		
\hcoffin_set:Nn	2, 89, 89, 104, 772, 780, 817, 853		
\Height	162, 163, 167, 171, 177		
I			
\iow_newline:	920–924, 932		
\iow_term:x	918, 926, 939		
L			
\l_coffin_aligned_coffin	137, 139, 354, 355, 359, 365, 368,		
370, 386, 387, 392–396, 398, 400,			
404, 405, 410–414, 447, 460, 503,			
504, 508, 515, 899, 906, 908, 910, 912			
\l_coffin_aligned_internal_coffin ..	137, 140, 427, 434		
\l_coffin_bottom_corner_dim	520, 522, 544, 548, 609, 617, 632, 639		
\l_coffin_bounding_prop	518,		
518, 535, 559, 561, 564, 566, 571, 622			

\l_coffin_scale_x_fp	\msg_kernel_new:nnnn	946, 954, 957
643, 643, 649, 651, 664, 675, 680, 690		
\l_coffin_scale_y_fp	P	
643, 644, 652, 654, 676, 677, 693	\prop_clear:c	366
\l_coffin_scaled_total_height_dim ..	\prop_clear:N	832
645, 645, 678, 679, 685	\prop_del:Nn	827, 830, 835
\l_coffin_scaled_width_dim	\prop_get:cnN	142
645, 646, 681, 682, 685	\prop_get:NnN	786, 790, 859, 863
\l_coffin_show_toks	\prop_gset_eq:cN	83, 85
769, 769, 915, 929, 931, 934, 940, 943	\prop_map_inline:cn	437, 455, 530, 532,
\l_coffin_sin_fp ... 36, 36, 528, 596, 601	552, 554, 611, 660, 662, 666, 668, 927	
\l_coffin_tmp_box	\prop_map_inline:Nn ...	535, 622, 833, 842
4, 4, 119, 121	\prop_new:c	81, 82
\l_coffin_tmp_dim	\prop_new:N	8, 14, 518, 725, 766, 770
4,	\prop_put:cnn	207
5, 120, 124, 217, 219, 221, 224,	\prop_put:cnx	213,
226, 229, 231, 234–236, 238, 360,	215, 218, 220, 225, 227, 230, 232,	
362, 363, 509, 511, 512, 563, 565, 567	237, 459, 576, 629, 636, 698, 710, 716	
\l_coffin_tmp_fp	\prop_put:Nnn	726, 728, 730,
4, 6, 525–529, 594, 596, 597, 599,	732, 734, 736, 738, 740, 742, 744,	
601, 602, 650, 651, 653, 654, 689–694	746, 748, 750, 752, 754, 756, 758, 760	
\l_coffin_tmp_tl .. 4, 7, 9–13, 15–25,	\prop_put:Nno	10–13, 16–18, 20–25
445, 446, 448, 787, 788, 791, 792,	\prop_put:Nnx	559, 561, 564, 566, 571
800, 805, 860, 861, 864, 865, 874, 879	\prop_set_eq:cc	157, 159, 397
\l_coffin_top_corner_dim	\prop_set_eq:cN	151, 153
520, 523, 548, 607, 618	\prop_set_eq:Nc	822
\l_coffin_TotalHeight_dim ... 46, 48, 169	\ProvidesExplPackage	2
\l_coffin_Width_dim		
46, 49, 170	Q	
\l_coffin_x_dim . 38, 38, 250, 258, 278,	\quark_if_no_value:NT	143
281, 288, 295, 297, 328, 331, 420,	\quark_if_no_value:NTF	788, 792, 861, 865
424, 442, 450, 570, 572, 575, 577,		
580, 585, 697, 699, 702, 705, 848, 893	R	
\l_coffin_x_fp	\resizebox	655
42, 42, 591, 593, 599	\rotatebox	539
\l_coffin_x_prime_dim	\rule	775, 820
38, 40, 420, 424, 582, 586, 893, 896		
\l_coffin_x_prime_fp	S	
42, 44, 593, 595, 597, 603	\scalebox	683
\l_coffin_y_dim	\scan_stop:	712, 718, 776
38, 39,	\set@color	95, 112
251, 263, 266, 273, 290, 332, 421,	\sffamily	768
426, 443, 450, 570, 572, 575, 577,		
580, 585, 697, 699, 702, 705, 849, 894	T	
\l_coffin_y_fp	\tex_hsize:D	110
42, 43, 592, 594, 598	\tex_kern:D	358, 363,
\l_coffin_y_prime_dim	430, 431, 507, 512, 542, 543, 902, 903	
38, 41, 421, 426, 582, 587, 894, 898	\tex_number:D	
\l_coffin_y_prime_fp	57, 79–83, 85, 142, 151, 153,	
42, 45, 598, 600, 602, 604	157–160, 207, 213, 215, 218, 220,	
\l_msg_coding_error_text_tl ... 949, 960		
M		
\msg_kernel_error:nn		249
\msg_kernel_error:nnx		60, 65
\msg_kernel_error:nnxx		145

225, 227, 230, 232, 237, 368, 398,	\toks_if_empty:NF	929
399, 437, 455, 460, 530, 532, 552,	\toks_put_right:Nn	940
554, 576, 611, 629, 636, 660, 662,	\toks_put_right:Nx	931, 934
666, 668, 698, 710, 716, 823, 916, 927	\toks_show:N	943
\tex_vrule:D	\TotalHeight	
\the_tex:D	162, 165, 169, 173, 179, 653, 678, 679	
\tiny		
\tl_if_eq:NNT		V
\tl_if_in:nnTF	\vbox_set:Nn	108
\tl_new:N	\vbox_set_top:Nn	119
\tl_set:Nn	\vbox_unpack:N	119
\tl_to_str:n	\vcoffin_set:cnn	2, 105
\token_to_str:N	\vcoffin_set:Nnn	2, 105, 105, 128
\toks@		W
\toks_clear:N	\Width 162, 166, 170, 174, 180, 650, 681, 682	