

# The Hobby package: code

Andrew Stacey  
[stacey@math.ntnu.no](mailto:stacey@math.ntnu.no)

1.4 from 2014-01-21

## 1 Implementation

### 1.1 Main Code

We use L<sup>A</sup>T<sub>E</sub>X3 syntax so need to load the requisite packages

```
1 \RequirePackage{expl3}
2 \RequirePackage{xparse}
3 \RequirePackage{pml3array}
4 \ExplSyntaxOn

5 \cs_generate_variant:Nn \fp_set:Nn {Nx}
6 \cs_generate_variant:Nn \tl_if_eq:nnTF {VnTF}
7 \cs_generate_variant:Nn \tl_if_eq:nnTF {xnTF}
```

#### 1.1.1 Initialisation

We declare all our variables.

Start with version and date, together with a check to see if we've been loaded twice (fail gracefully if so).

```
8 \tl_clear:N \l_tmpa_tl
9 \tl_if_exist:NT \g__hobby_version
10 {
11   \tl_set:Nn \l_tmpa_tl {
12     \ExplSyntaxOff
13     \tl_clear:N \l_tmpa_tl
14     \endinput
15   }
16 }
17 \tl_use:N \l_tmpa_tl
18
19 \tl_new:N \g__hobby_version
20 \tl_new:N \g__hobby_date
21 \tl_set:Nn \g__hobby_version {1.4}
22 \tl_set:Nn \g__hobby_date {2014-01-21}
23 \DeclareDocumentCommand \hobbyVersion {}
24 {
25   \tl_use:N \g__hobby_version
26 }
27 \DeclareDocumentCommand \hobbyDate {}
28 {
29   \tl_use:N \g__hobby_date
30 }
```

The function for computing the lengths of the control points depends on three parameters. These are set to  $a = \sqrt{2}$ ,  $b = 1/16$ , and  $c = \frac{3-\sqrt{5}}{2}$ .

```

31 \fp_new:N \g_hobby_parama_fp
32 \fp_new:N \g_hobby_paramb_fp
33 \fp_new:N \g_hobby_paramc_fp
34 \fp_gset:Nn \g_hobby_parama_fp {2^.5}
35 \fp_gset:Nn \g_hobby_paramb_fp {1/16}
36 \fp_gset:Nn \g_hobby_paramc_fp {(3-5^.5)/2}

```

Now we define our objects for use in generating the path.

```

\l_hobby_closed_bool \l_hobby_closed_bool is true if the path is closed.
37 \bool_new:N \l_hobby_closed_bool

\l_hobby_disjoint_bool \l_hobby_disjoint_bool is true if the path should start with a moveto command.
38 \bool_new:N \l_hobby_disjoint_bool

\l_hobby_save_aux_bool \l_hobby_save_aux_bool is true if when saving paths then they should be saved to the aux file.
39 \bool_new:N \l_hobby_save_aux_bool
40 \bool_set_true:N \l_hobby_save_aux_bool
41 \DeclareDocumentCommand \HobbyDisableAux {}
42 {
43   \bool_set_false:N \l_hobby_save_aux_bool
44 }

\l_hobby_points_array \l_hobby_points_array is an array holding the specified points on the path. In the LATEX3 code, a
“point” is a token list of the form x = <number>, y = <number>. This gives us the greatest flexibility
in passing points back and forth between the LATEX3 code and any calling code. The array is indexed
by integers beginning with 0. In the documentation, we will use the notation  $z_k$  to refer to the  $k$ th
point.
45 \array_new:N \l_hobby_points_array

\l_hobby_points_x_array \l_hobby_points_x_array is an array holding the  $x$ -coordinates of the specified points.
46 \array_new:N \l_hobby_points_x_array

\l_hobby_points_y_array \l_hobby_points_y_array is an array holding the  $y$ -coordinates of the specified points.
47 \array_new:N \l_hobby_points_y_array

\l_hobby_actions_array \l_hobby_actions_array is an array holding the (encoded) action to be taken out on the segment of
the path ending at that point.
48 \array_new:N \l_hobby_actions_array

\l_hobby_angles_array \l_hobby_angles_array is an array holding the angles of the lines between the points. Specifically,
the angle indexed by  $k$  is the angle in radians of the line from  $z_k$  to  $z_{k+1}$ .
49 \array_new:N \l_hobby_angles_array

\l_hobby_distances_array \l_hobby_distances_array is an array holding the distances between the points. Specifically, the
distance indexed by  $k$ , which we will write as  $d_k$ , is the length of the line from  $z_k$  to  $z_{k+1}$ .
50 \array_new:N \l_hobby_distances_array

\l_hobby_tension_out_array \l_hobby_tension_out_array is an array holding the tension for the path as it leaves each point.
This is a parameter that controls how much the curve “flexes” as it leaves the point. In the following,
this will be written  $\tau_k$ .
51 \array_new:N \l_hobby_tension_out_array

```

`hobby_tension_in_array` `\l_hobby_tension_in_array` is an array holding the tension for the path as it arrives at each point. This is a parameter that controls how much the curve “flexes” as it gets to the point. In the following, this will be written  $\bar{\tau}_k$ .

52 `\array_new:N \l_hobby_tension_in_array`

`l_hobby_matrix_a_array` `\l_hobby_matrix_a_array` is an array holding the subdiagonal of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by  $A_i$ . The first index is 1.

53 `\array_new:N \l_hobby_matrix_a_array`

`l_hobby_matrix_b_array` `\l_hobby_matrix_b_array` is an array holding the diagonal of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by  $B_i$ . The first index is 0.

54 `\array_new:N \l_hobby_matrix_b_array`

`l_hobby_matrix_c_array` `\l_hobby_matrix_c_array` is an array holding the superdiagonal of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by  $C_i$ . The first index is 0.

55 `\array_new:N \l_hobby_matrix_c_array`

`l_hobby_matrix_d_array` `\l_hobby_matrix_d_array` is an array holding the target vector of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by  $D_i$ . The first index is 1.

56 `\array_new:N \l_hobby_matrix_d_array`

`l_hobby_vector_u_array` `\l_hobby_vector_u_array` is an array holding the perturbation of the linear system for closed paths. The coefficient matrix for an *open* path is tridiagonal and that means that Gaussian elimination runs faster than expected ( $O(n)$  instead of  $O(n^3)$ ). The matrix for a closed path is not tridiagonal but is not far off. It can be solved by perturbing it to a tridiagonal matrix and then modifying the result. This array represents a utility vector in that perturbation. In the following, the vector will be denoted by  $u$ . The first index is 1.

57 `\array_new:N \l_hobby_vector_u_array`

`bby_excess_angle_array` `\l_hobby_excess_angle_array` is an array that allows the user to say that the algorithm should add a multiple of  $2\pi$  to the angle differences. This is because these angles are wrapped to the interval  $(-\pi, \pi]$  but the wrapping might go wrong near the end points due to computation accuracy. The first index is 1.

58 `\array_new:N \l_hobby_excess_angle_array`

`\l_hobby_psi_array` `\l_hobby_psi_array` is an array holding the difference of the angles of the lines entering and exiting a point. That is,  $\psi_k$  is the angle between the lines joining  $z_k$  to  $z_{k-1}$  and  $z_{k+1}$ . The first index is 1.

59 `\array_new:N \l_hobby_psi_array`

`\l_hobby_theta_array` `\l_hobby_theta_array` is an array holding the angles of the outgoing control points for the generated path. These are measured relative to the line joining the point to the next point on the path. The first index is 0.

60 `\array_new:N \l_hobby_theta_array`

`\l_hobby_phi_array` `\l_hobby_phi_array` is an array holding the angles of the incoming control points for the generated path. These are measured relative to the line joining the point to the previous point on the path. The first index is 1.

61 `\array_new:N \l_hobby_phi_array`

`\l_hobby_sigma_array` `\l_hobby_sigma_array` is an array holding the lengths of the outgoing control points for the generated path. The units are such that the length of the line to the next specified point is one unit.

```
62 \array_new:N \l_hobby_sigma_array
```

`\l_hobby_rho_array` `\l_hobby_rho_array` is an array holding the lengths of the incoming control points for the generated path. The units are such that the length of the line to the previous specified point is one unit.

```
63 \array_new:N \l_hobby_rho_array
```

`\l_hobby_controla_array` `\l_hobby_controla_array` is an array holding the coordinates of the first control points on the curves. The format is the same as for `\l_hobby_points_array`.

```
64 \array_new:N \l_hobby_controla_array
```

`\l_hobby_controlb_array` `\l_hobby_controlb_array` is an array holding the coordinates of the second control points on the curves. The format is the same as for `\l_hobby_points_array`.

```
65 \array_new:N \l_hobby_controlb_array
```

`\l_hobby_matrix_v_fp` `\l_hobby_matrix_v_fp` is a number which is used when doing the perturbation of the solution of the linear system for a closed curve. There is actually a vector,  $v$ , that this corresponds to but that vector only has one component that needs computation.

```
66 \fp_new:N \l_hobby_matrix_v_fp
```

`\l_hobby_tempa_fp` `\l_hobby_tempa_fp` is a temporary variable of type `fp`.

```
67 \fp_new:N \l_hobby_tempa_fp
```

`\l_hobby_tempb_fp` `\l_hobby_tempb_fp` is a temporary variable of type `fp`.

```
68 \fp_new:N \l_hobby_tempb_fp
```

`\l_hobby_tempc_fp` `\l_hobby_tempc_fp` is a temporary variable of type `fp`.

```
69 \fp_new:N \l_hobby_tempc_fp
```

`\l_hobby_tempd_fp` `\l_hobby_tempd_fp` is a temporary variable of type `fp`.

```
70 \fp_new:N \l_hobby_tempd_fp
```

`\l_hobby_temps_fp` `\l_hobby_temps_fp` is a temporary variable of type `fp`.

```
71 \fp_new:N \l_hobby_temps_fp
```

`\l_hobby_in_curl_fp` `\l_hobby_in_curl_fp` is the “curl” at the end of an open path. This is used if the angle at the end is not specified.

```
72 \fp_new:N \l_hobby_in_curl_fp
73 \fp_set:Nn \l_hobby_in_curl_fp {1}
```

`\l_hobby_out_curl_fp` `\l_hobby_out_curl_fp` is the “curl” at the start of an open path. This is used if the angle at the start is not specified.

```
74 \fp_new:N \l_hobby_out_curl_fp
75 \fp_set:Nn \l_hobby_out_curl_fp {1}
```

`\l_hobby_in_angle_fp` `\l_hobby_in_angle_fp` is the angle at the end of an open path. If this is not specified, it will be computed automatically. It is set to `\c_undefined_fp` to allow easy detection of when it has been specified.

```
76 \fp_new:N \l_hobby_in_angle_fp
77 \fp_set_eq:NN \l_hobby_in_angle_fp \c_undefined_fp
```

`\l_hobby_out_angle_fp` `\l_hobby_out_angle_fp` is the angle at the start of an open path. If this is not specified, it will be computed automatically. It is set to `\c_undefined_fp` to allow easy detection of when it has been specified.

```
78 \fp_new:N \l_hobby_out_angle_fp
79 \fp_set_eq:NN \l_hobby_out_angle_fp \c_undefined_fp
```

`\l_hobby_npoints_int` `\l_hobby_npoints_int` is one less than the number of points on the curve. As our list of points starts at 0, this is the index of the last point. In the algorithm for a closed curve, some points are repeated whereupon this is incremented so that it is always the index of the last point.

```
80 \int_new:N \l_hobby_npoints_int
```

`\l_hobby_draw_int`

```
81 \int_new:N \l_hobby_draw_int
```

A “point” is a key-value list setting the x-value, the y-value, and the tensions at that point. Using keys makes it easier to pass points from the algorithm code to the calling code and vice versa without either knowing too much about the other.

```
82 \keys_define:nn {hobby / read in all} {
83   x .fp_set:N = \l_hobby_tempa_fp,
84   y .fp_set:N = \l_hobby_tempb_fp,
85   tension~out .fp_set:N = \l_hobby_tempc_fp,
86   tension~in .fp_set:N = \l_hobby_tempd_fp,
87   excess~angle .fp_set:N = \l_hobby_temps_fp,
88   break .tl_set:N = \l_tmpb_tl,
89   blank .tl_set:N = \l_tmpa_tl,
90   tension .meta:n = { tension~out=#1, tension~in=#1 },
91   break .default:n = false,
92   blank .default:n = false,
93   invert~soft~blanks .choice:,
94   invert~soft~blanks / true .code:n = {
95     \int_gset:Nn \l_hobby_draw_int {0}
96   },
97   invert~soft~blanks / false .code:n = {
98     \int_gset:Nn \l_hobby_draw_int {1}
99   },
100   invert~soft~blanks .default:n = true,
101   tension~out .default:n = 1,
102   tension~in .default:n = 1,
103   excess~angle .default:n = 0,
104   in~angle .fp_gset:N = \l_hobby_in_angle_fp,
105   out~angle .fp_gset:N = \l_hobby_out_angle_fp,
106   in~curl .fp_gset:N = \l_hobby_in_curl_fp,
107   out~curl .fp_gset:N = \l_hobby_out_curl_fp,
108   closed .bool_gset:N = \l_hobby_closed_bool,
109   closed .default:n = true,
110   disjoint .bool_gset:N = \l_hobby_disjoint_bool,
111   disjoint .default:n = true,
112   break~default .code:n = {
113     \keys_define:nn { hobby / read in all }
114     {
115       break .default:n = #1
116     }
117   },
118   blank~default .code:n = {
119     \keys_define:nn { hobby / read in all }
120     {
```

```

121     blank .default:n = #1
122   }
123 },
124 }

```

There are certain other parameters that can be set for a given curve.

```

125 \keys_define:nn { hobby / read in params } {
126   in~angle .fp_gset:N = \l_hobby_in_angle_fp,
127   out~angle .fp_gset:N = \l_hobby_out_angle_fp,
128   in~curl .fp_gset:N = \l_hobby_in_curl_fp,
129   out~curl .fp_gset:N = \l_hobby_out_curl_fp,
130   closed .bool_gset:N = \l_hobby_closed_bool,
131   closed .default:n = true,
132   disjoint .bool_gset:N = \l_hobby_disjoint_bool,
133   disjoint .default:n = true,
134   break~default .code:n = {
135     \keys_define:nn { hobby / read in all }
136     {
137       break .default:n = #1
138     }
139   },
140   blank~default .code:n = {
141     \keys_define:nn { hobby / read in all }
142     {
143       blank .default:n = #1
144     }
145   },
146   invert~soft~blanks .choice:,
147   invert~soft~blanks / true .code:n = {
148     \int_gset:Nn \l_hobby_draw_int {0}
149   },
150   invert~soft~blanks / false .code:n = {
151     \int_gset:Nn \l_hobby_draw_int {1}
152   },
153   invert~soft~blanks .default:n = true,
154 }

```

**\hobby\_distangle:n** Computes the distance and angle between successive points. The argument given is the index of the current point. Assumptions: the points are in `\l_hobby_points_x_array` and `\l_hobby_points_y_array` and the index of the last point is `\l_hobby_npoints_int`.

```

155 \cs_set:Nn \hobby_distangle:n {
156   \fp_set:Nn \l_hobby_tempa_fp {
157     (\array_get:Nn \l_hobby_points_x_array {#1 + 1})
158     - (\array_get:Nn \l_hobby_points_x_array {#1})}
159
160   \fp_set:Nn \l_hobby_tempb_fp {
161     (\array_get:Nn \l_hobby_points_y_array {#1 + 1})
162     - (\array_get:Nn \l_hobby_points_y_array {#1})}
163
164   \fp_set:Nn \l_hobby_tempc_fp { atan ( \l_hobby_tempb_fp, \l_hobby_tempa_fp ) }
165   \fp_veclen:NVV \l_hobby_tempd_fp \l_hobby_tempa_fp \l_hobby_tempb_fp
166
167   \array_push:Nx \l_hobby_angles_array {\fp_to_tl:N \l_hobby_tempc_fp}
168   \array_push:Nx \l_hobby_distances_array {\fp_to_tl:N \l_hobby_tempd_fp}
169 }

```

`\fp_vecilen:NVV` Computes the length of the vector specified by the latter two arguments, storing the answer in the first.

```

170 \cs_new:Nn \fp_vecilen:Nnn {
171   \fp_set:Nn #1 {((#2)^2 + (#3)^2)^.5}
172 }
173 \cs_generate_variant:Nn \fp_vecilen:Nnn {NVV}

```

`\hobby_ctrllen:Nnn` Computes the length of the control point vector from the two angles, storing the answer in the first argument given.

```

174 \cs_new:Nn \hobby_ctrllen:Nnn {
175   \fp_set:Nn #1 {(2 - \g_hobby_parama_fp
176     * ( sin(#2) - \g_hobby_paramb_fp * sin(#3) )
177     * ( sin(#3) - \g_hobby_paramb_fp * sin(#2) )
178     * ( cos(#2) - cos(#3) ) ) }
179   / ( 1 + (1 - \g_hobby_paramc_fp) * cos(#3) + \g_hobby_paramc_fp * cos(#2))}
180 }
181 \cs_generate_variant:Nn \hobby_ctrllen:Nnn {NVV}

```

`\hobby_append_point_copy:n` This function adds a copy of the point (numbered by its argument) to the end of the list of points, copying all the relevant data (coordinates, tension, etc.).  
Originally from Bruno Le Foch on TeX-SX.

```

182 \cs_new_protected:Npn \hobby_append_point_copy:n #1
183 {
184   \hobby_append_point_copy_aux:Nn \l_hobby_points_array {#1}
185   \hobby_append_point_copy_aux:Nn \l_hobby_points_x_array {#1}
186   \hobby_append_point_copy_aux:Nn \l_hobby_points_y_array {#1}
187   \hobby_append_point_copy_aux:Nn \l_hobby_tension_in_array {#1}
188   \hobby_append_point_copy_aux:Nn \l_hobby_tension_out_array {#1}
189   \hobby_append_point_copy_aux:Nn \l_hobby_excess_angle_array {#1}
190   \hobby_append_point_copy_aux:Nn \l_hobby_actions_array {#1}
191 }
192 \cs_new_protected:Npn \hobby_append_point_copy_aux:Nn #1#2
193 { \array_gpush:Nx #1 { \array_get:Nn #1 {#2} } }

```

`\hobby_gen_path:` This is the curve generation function. We assume at the start that we have an array containing all the points that the curve must go through, and the various curve parameters have been initialised. So these must be set up by a wrapper function which then calls this one. The list of required information is:

1. `\l_hobby_points_x_array`
2. `\l_hobby_points_y_array`
3. `\l_hobby_tension_out_array`
4. `\l_hobby_tension_in_array`
5. `\l_hobby_excess_angle_array`
6. `\l_hobby_in_curl_fp`
7. `\l_hobby_out_curl_fp`
8. `\l_hobby_in_angle_fp`
9. `\l_hobby_out_angle_fp`
10. `\l_hobby_closed_bool`

11. \l\_hobby\_actions\_array

```
194 \cs_new:Nn \hobby_gen_path:
195 {
```

For much of the time, we can pretend that a closed path is the same as an open path. To do this, we need to make the end node an internal node by repeating the  $z_1$  node as the  $z_{n+1}$ th node. We also check that the last ( $z_n$ ) and first ( $z_0$ ) nodes are the same, otherwise we repeat the  $z_0$  node as well.

```
196 \bool_if:NT \l_hobby_closed_bool {
```

Are the  $x$ -values of the first and last points different?

```
197 \fp_compare:nTF {(\array_get:Nn \l_hobby_points_x_array {0})
198 =
199 (\array_top:N \l_hobby_points_x_array)}
200 {
```

No, so compare the  $y$ -values. Are the  $y$ -values of the first and last points different?

```
201 \fp_compare:nF {
202 \array_get:Nn \l_hobby_points_y_array {0}
203 =
204 \array_top:N \l_hobby_points_y_array
205 }
206 {
```

Yes, so we need to duplicate the first point, with all of its data.

```
207 \hobby_append_point_copy:n {0}
208 }
209 }
210 {
```

Yes, so we need to duplicate the first point, with all of its data.

```
211 \hobby_append_point_copy:n {0}
212 }
```

Now that we are sure that the first and last points are identical, we need to duplicate the first-but-one point (and all of its data).

```
213 \hobby_append_point_copy:n {1}
214 }
```

Set \l\_hobby\_npoints\_int to the number of points (minus one).

```
215 \int_gset:Nn \l_hobby_npoints_int {\array_length:N \l_hobby_points_y_array}
```

At this point, we need to decide what to do. This will depend on whether we have any intermediate points.

```
216 \int_compare:nNnTF {\l_hobby_npoints_int} = {0} {
```

Only one point, do nothing

```
217 }
218 {
219 \int_compare:nNnTF {\l_hobby_npoints_int} = {1} {
```

Only two points, skip processing. Just need to set the incoming and outgoing angles

```
220 \hobby_distangle:n {0}
221 \fp_if_undefined:NF \l_hobby_out_angle_fp
222 {
223 \fp_set:Nn \l_hobby_tempa_fp { \l_hobby_out_angle_fp
224 - \array_get:Nn \l_hobby_angles_array {0}}
```



We want to ensure that these angles lie in the range  $(-\pi, \pi]$ . So if the angle is bigger than  $\pi$ , we subtract  $2\pi$ . (It shouldn't be that we can get bigger than  $3\pi$  - check this)

```

225 \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
226 {
227   \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
228 }

```

Similarly, we check to see if the angle is less than  $-\pi$ .

```

229 \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
230 {
231   \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
232 }
233 \array_put:Nnx \l_hobby_theta_array {0} {\fp_to_tl:N \l_hobby_tempa_fp}
234 \fp_if_undefined:NT \l_hobby_in_angle_fp
235 {
236   %%^A \fp_mul:Nn \l_hobby_tempa_fp {-1}
237   \array_put:Nnx \l_hobby_phi_array {1}{ \fp_to_tl:N \l_hobby_tempa_fp}
238 }
239 }
240 \fp_if_undefined:NTF \l_hobby_in_angle_fp
241 {
242   \fp_if_undefined:NT \l_hobby_out_angle_fp
243   {
244     \array_put:Nnx \l_hobby_phi_array {1} {0}
245     \array_put:Nnx \l_hobby_theta_array {0} {0}
246   }
247 }
248 {
249   \fp_set:Nn \l_hobby_tempa_fp { - \l_hobby_in_angle_fp + \c_pi_fp
250 + (\array_get:Nn \l_hobby_angles_array {0})}
251   \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
252   {
253     \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
254   }
255   \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
256   {
257     \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
258   }
259
260   \array_put:Nnx \l_hobby_phi_array {1}
261   {\fp_to_tl:N \l_hobby_tempa_fp}
262   \fp_if_undefined:NT \l_hobby_out_angle_fp
263   {
264     %%^A \fp_mul:Nn \l_hobby_tempa_fp {-1}
265     \array_put:Nnx \l_hobby_theta_array {0}{ \fp_to_tl:N \l_hobby_tempa_fp}
266   }
267 }
268
269 }
270 {

```

Got enough points, go on with processing

```

271 \hobby_compute_path:
272 }
273 \hobby_build_path:
274 }
275 }

```

`\hobby_compute_path:` This is the path builder where we have enough points to run the algorithm.

```

276 \cs_new:Nn \hobby_compute_path:
277 {

```

Our first step is to go through the list of points and compute the distances and angles between successive points. Thus  $d_i$  is the distance from  $z_i$  to  $z_{i+1}$  and the angle is the angle of the line from  $z_i$  to  $z_{i+1}$ .

```

278 \int_step_function:nnnN {0} {1} {\l_hobby_npoints_int - 1} \hobby_distangle:n

```

For the majority of the code, we're only really interested in the differences of the angles. So for each internal point we compute the differences in the angles.

```

279 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 1} {
280   \fp_set:Nx \l_hobby_tempa_fp {
281     \array_get:Nn \l_hobby_angles_array {##1}
282     - \array_get:Nn \l_hobby_angles_array {##1 - 1}}

```

We want to ensure that these angles lie in the range  $(-\pi, \pi]$ . So if the angle is bigger than  $\pi$ , we subtract  $2\pi$ . (It shouldn't be that we can get bigger than  $3\pi$  - check this.)

```

283   \fp_compare:nTF {\l_hobby_tempa_fp > \c_pi_fp }
284   {
285     \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
286   }
287   {}

```

Similarly, we check to see if the angle is less than  $-\pi$ .

```

288   \fp_compare:nTF {\l_hobby_tempa_fp <= -\c_pi_fp }
289   {
290     \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
291   }
292   {}

```

The wrapping routine might not get it right at the edges so we add in the override.

```

293 \array_get:NnNTF \l_hobby_excess_angle_array {##1} \l_tmpa_tl {
294   \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp * \l_tmpa_tl}
295 }{}
296 \array_put:Nnx \l_hobby_psi_array {##1}{\fp_to_tl:N \l_hobby_tempa_fp}
297 }

```

Next, we generate the matrix. We start with the subdiagonal. This is indexed from 1 to  $n - 1$ .

```

298 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 1} {
299   \array_put:Nnx \l_hobby_matrix_a_array {##1} {\fp_to_tl:n {
300     \array_get:Nn \l_hobby_tension_in_array {##1}^2
301     * \array_get:Nn \l_hobby_distances_array {##1}
302     * \array_get:Nn \l_hobby_tension_in_array {##1 + 1}
303   }}
304 }

```

Next, we attack main diagonal. We might need to adjust the first and last terms, but we'll do that in a minute.

```

305 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 1} {
306
307   \array_put:Nnx \l_hobby_matrix_b_array {##1} {\fp_to_tl:n
308 { (3 * (\array_get:Nn \l_hobby_tension_in_array {##1 + 1}) - 1) *
309   (\array_get:Nn \l_hobby_tension_out_array {##1})^2 *
310   (\array_get:Nn \l_hobby_tension_out_array {##1 - 1})
311   * ( \array_get:Nn \l_hobby_distances_array {##1 - 1})
312 +
313   (3 * (\array_get:Nn \l_hobby_tension_out_array {##1 - 1}) - 1)
314   * (\array_get:Nn \l_hobby_tension_in_array {##1})^2

```

```

315 * (\array_get:Nn \l_hobby_tension_in_array {##1 + 1})
316 * (\array_get:Nn \l_hobby_distances_array {##1}))}
317 }
318 }

```

Next, the superdiagonal.

```

319 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 2} {
320
321 \array_put:Nnx \l_hobby_matrix_c_array {##1} {\fp_to_tl:n
322 {(\array_get:Nn \l_hobby_tension_in_array {##1})^2
323 * (\array_get:Nn \l_hobby_tension_in_array {##1 - 1})
324 * (\array_get:Nn \l_hobby_distances_array {##1 - 1})
325 }}
326
327 }

```

Lastly (before the adjustments), the target vector.

```

328 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 2} {
329
330 \array_put:Nnx \l_hobby_matrix_d_array {##1} {\fp_to_tl:n
331 {
332 - (\array_get:Nn \l_hobby_psi_array {##1 + 1})
333 * (\array_get:Nn \l_hobby_tension_out_array {##1})^2
334 * (\array_get:Nn \l_hobby_tension_out_array {##1 - 1})
335 * (\array_get:Nn \l_hobby_distances_array {##1 - 1})
336 - (3 * (\array_get:Nn \l_hobby_tension_out_array {##1 - 1}) - 1)
337 * (\array_get:Nn \l_hobby_psi_array {##1})
338 * (\array_get:Nn \l_hobby_tension_in_array {##1})^2
339 * (\array_get:Nn \l_hobby_tension_in_array {##1 + 1})
340 * (\array_get:Nn \l_hobby_distances_array {##1})
341 }
342 }
343 }

```

Next, there are some adjustments at the ends. These differ depending on whether the path is open or closed.

```

344 \bool_if:NTF \l_hobby_closed_bool {

```

Closed path

```

345 \array_put:Nnx \l_hobby_matrix_c_array {0} {\fp_to_tl:n {
346 - (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 2})
347 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2})
348 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^2
349 }}
350
351 \array_put:Nnn \l_hobby_matrix_b_array {0} {1}
352 \array_put:Nnn \l_hobby_matrix_d_array {0} {0}
353
354 \array_put:Nnx \l_hobby_matrix_b_array {\l_hobby_npoints_int - 1} {\fp_to_tl:n {
355 (\array_get:Nn \l_hobby_matrix_b_array {\l_hobby_npoints_int - 1})
356 + 1
357 }}
358
359 \array_put:Nnx \l_hobby_matrix_d_array {\l_hobby_npoints_int - 1} {\fp_to_tl:n {
360 - (\array_get:Nn \l_hobby_psi_array {1})
361 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^2
362 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2})
363 * (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 2})

```

```

364 - (3 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2}) - 1)
365 * (\array_get:Nn \l_hobby_psi_array {\l_hobby_npoints_int - 1})
366 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int - 1})^2
367 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int})
368 * (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 1})
369 }
370 }

```

We also need to populate the  $u$ -vector

```

371 \array_put:Nnn \l_hobby_vector_u_array {0} {1}
372 \array_put:Nnn \l_hobby_vector_u_array {\l_hobby_npoints_int - 1} {1}
373 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 2} {
374 \array_put:Nnn \l_hobby_vector_u_array {##1} {0}
375 }

```

And define the significant entry in the  $v$ -vector.

```

376 \fp_set:Nn \l_hobby_matrix_v_fp {
377 (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^2
378 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2})
379 * (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 2})
380 }
381 }
382 {

```

Open path. First, we test to see if  $\theta_0$  has been specified.

```

383 \fp_if_undefined:NTF \l_hobby_out_angle_fp
384 {
385 \array_put:Nnn \l_hobby_matrix_b_array {0} {\fp_to_tl:n {
386 (\array_get:Nn \l_hobby_tension_in_array {1})^3
387 * \l_hobby_in_curl_fp
388 +
389 (3 * (\array_get:Nn \l_hobby_tension_in_array {1}) - 1)
390 * (\array_get:Nn \l_hobby_tension_out_array {0})^3
391 }}}
392
393 \array_put:Nnn \l_hobby_matrix_c_array {0} {\fp_to_tl:n {
394 (\array_get:Nn \l_hobby_tension_out_array {0})^3
395 +
396 (3 * (\array_get:Nn \l_hobby_tension_out_array {0}) - 1)
397 * (\array_get:Nn \l_hobby_tension_in_array {1})^3
398 * \l_hobby_in_curl_fp
399 }}}
400
401 \array_put:Nnn \l_hobby_matrix_d_array {0} {\fp_to_tl:n {
402 - ( (\array_get:Nn \l_hobby_tension_out_array {0})^3
403 +
404 (3 * (\array_get:Nn \l_hobby_tension_out_array {0}) - 1)
405 * (\array_get:Nn \l_hobby_tension_in_array {1})^3
406 * \l_hobby_in_curl_fp)
407 * (\array_get:Nn \l_hobby_psi_array {1})
408 }}}
409
410 }
411 {
412 \array_put:Nnn \l_hobby_matrix_b_array {0} {1}
413 \array_put:Nnn \l_hobby_matrix_c_array {0} {0}
414 \fp_set:Nn \l_hobby_tempa_fp { \l_hobby_out_angle_fp
415 - \array_get:Nn \l_hobby_angles_array {0}}

```

We want to ensure that these angles lie in the range  $(-\pi, \pi]$ . So if the angle is bigger than  $\pi$ , we subtract  $2\pi$ . (It shouldn't be that we can get bigger than  $3\pi$  - check this)

```

416   \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
417   {
418     \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
419   }

```

Similarly, we check to see if the angle is less than  $-\pi$ .

```

420   \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
421   {
422     \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
423   }
424   \array_put:Nnx \l_hobby_matrix_d_array {0} {\fp_to_tl:N \l_hobby_tempa_fp}
425 }

```

Next, if  $\phi_n$  has been given.

```

426 \fp_if_undefined:NTF \l_hobby_in_angle_fp
427 {
428
429   \array_put:Nnx \l_hobby_matrix_b_array {\l_hobby_npoints_int - 1} {\fp_to_tl:n {
430 \array_get:Nn \l_hobby_matrix_b_array {\l_hobby_npoints_int - 1}
431 - (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^2
432 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2})
433 * (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 2})
434 *
435 ((3 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int} ) - 1)
436 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^3 \l_tmpa_tl
437 * \l_hobby_out_curl_fp
438 +
439 (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int } )^3)
440 /
441 ((3 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2}) - 1)
442 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int})^3
443 +
444 ( \array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^3
445 * \l_hobby_out_curl_fp)
446 }}
447
448   \array_put:Nnx \l_hobby_matrix_d_array {\l_hobby_npoints_int - 1} {\fp_to_tl:n {
449 - (3 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2}) - 1)
450 * (\array_get:Nn \l_hobby_psi_array {\l_hobby_npoints_int - 1})
451 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int - 1})^2
452 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int})
453 * (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 1})
454 }}
455
456 }
457 {
458   \fp_set:Nn \l_hobby_tempa_fp { - \l_hobby_in_angle_fp + \c_pi_fp
459 + (\array_get:Nn \l_hobby_angles_array {\l_hobby_npoints_int - 1})}
460   \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
461   {
462     \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
463   }
464   \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
465   {
466     \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}

```

```

467 }
468
469 \array_put:Nnx \l_hobby_phi_array {\l_hobby_npoints_int}
470 {\fp_to_tl:N \l_hobby_tempa_fp}
471
472 \array_put:Nnx \l_hobby_matrix_d_array {\l_hobby_npoints_int - 1} {\fp_to_tl:n {
473 \l_hobby_tempa_fp
474 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^2
475 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2})
476 * (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 2})
477 -
478 (3 * ( \array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2}) - 1)
479 * (\array_get:Nn \l_hobby_psi_array {\l_hobby_npoints_int - 1})
480 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int - 1})^2
481 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int})
482 * (\array_get:Nn \l_hobby_distances_array {\l_hobby_npoints_int - 1}) }}
483 }

```

End of adjustments for open paths.

```

484 }

```

Now we have the tridiagonal matrix in place, we implement the solution. We start with the forward eliminations.

```

485 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 1} {
486
487   \array_put:Nnx \l_hobby_matrix_b_array {##1} {\fp_to_tl:n {
488   (\array_get:Nn \l_hobby_matrix_b_array {##1 - 1})
489   * (\array_get:Nn \l_hobby_matrix_b_array {##1})
490   -
491   (\array_get:Nn \l_hobby_matrix_c_array {##1 - 1})
492   * (\array_get:Nn \l_hobby_matrix_a_array {##1})
493   }}

```

The last time, we don't touch the  $C$ -vector.

```

494   \int_compare:nT {##1 < \l_hobby_npoints_int - 1} {
495
496     \array_put:Nnx \l_hobby_matrix_c_array {##1} {\fp_to_tl:n {
497   (\array_get:Nn \l_hobby_matrix_b_array {##1 - 1})
498   * (\array_get:Nn \l_hobby_matrix_c_array {##1})
499   }}
500   }
501
502   \array_put:Nnx \l_hobby_matrix_d_array {##1} {\fp_to_tl:n {
503   (\array_get:Nn \l_hobby_matrix_b_array {##1 - 1})
504   * (\array_get:Nn \l_hobby_matrix_d_array {##1})
505   -
506   (\array_get:Nn \l_hobby_matrix_d_array {##1 - 1})
507   * (\array_get:Nn \l_hobby_matrix_a_array {##1})
508   }}

```

On a closed path, we also want to know  $M^{-1}u$  so need to do the elimination steps on  $u$  as well.

```

509   \bool_if:NT \l_hobby_closed_bool {
510     \array_put:Nnx \l_hobby_vector_u_array {##1} {\fp_to_tl:n {
511   (\array_get:Nn \l_hobby_matrix_b_array {##1 - 1})
512   * (\array_get:Nn \l_hobby_vector_u_array {##1})
513   -
514   (\array_get:Nn \l_hobby_vector_u_array {##1 - 1})
515   * (\array_get:Nn \l_hobby_matrix_a_array {##1})

```

```

516 }}
517 }
518 }

```

Now we start the back substitution. The first step is slightly different to the general step.

```

519 \array_put:Nnx \l_hobby_theta_array {\l_hobby_npoints_int - 1} {\fp_to_tl:n {
520 (\array_get:Nn \l_hobby_matrix_d_array {\l_hobby_npoints_int - 1})
521 / (\array_get:Nn \l_hobby_matrix_b_array {\l_hobby_npoints_int - 1})
522 }}

```

For a closed path, we need to work with  $u$  as well.

```

523 \bool_if:NT \l_hobby_closed_bool {
524 \array_put:Nnx \l_hobby_vector_u_array {\l_hobby_npoints_int - 1} {\fp_to_tl:n {
525 (\array_get:Nn \l_hobby_vector_u_array {\l_hobby_npoints_int - 1})
526 / (\array_get:Nn \l_hobby_matrix_b_array {\l_hobby_npoints_int - 1})
527 }}
528 }

```

Now we iterate over the vectors, doing the remaining back substitutions.

```

529 \int_step_inline:nnnn {\l_hobby_npoints_int - 2} {-1} {0} {
530
531 \array_put:Nnx \l_hobby_theta_array {##1} {\fp_to_tl:n {
532 ( (\array_get:Nn \l_hobby_matrix_d_array {##1})
533 - (\array_get:Nn \l_hobby_theta_array {##1 + 1})
534 * (\array_get:Nn \l_hobby_matrix_c_array {##1})
535 ) / (\array_get:Nn \l_hobby_matrix_b_array {##1})
536 }}
537 }
538 \bool_if:NT \l_hobby_closed_bool {

```

On a closed path, we also need to work out  $M^{-1}u$ .

```

539 \int_step_inline:nnnn {\l_hobby_npoints_int - 2} {-1} {0} {
540 \array_put:Nnx \l_hobby_vector_u_array {##1} {\fp_to_tl:n
541 {
542 ((\array_get:Nn \l_hobby_vector_u_array {##1})
543 - (\array_get:Nn \l_hobby_vector_u_array {##1 + 1})
544 * (\array_get:Nn \l_hobby_matrix_c_array {##1})
545 ) / (\array_get:Nn \l_hobby_matrix_b_array {##1})
546 }}
547 }

```

Then we compute  $v^\top M^{-1}u$  and  $v^\top M^{-1}\theta$ . As  $v$  has a particularly simple form, these inner products are easy to compute.

```

548
549 \fp_set:Nn \l_hobby_tempb_fp {
550 ((\array_get:Nn \l_hobby_theta_array {1})
551 * \l_hobby_matrix_v_fp
552 - (\array_get:Nn \l_hobby_theta_array {\l_hobby_npoints_int - 1})
553 ) / (
554 (\array_get:Nn \l_hobby_vector_u_array {1})
555 * \l_hobby_matrix_v_fp
556 - (\array_get:Nn \l_hobby_vector_u_array {\l_hobby_npoints_int - 1})
557 + 1
558 )}
559
560 \int_step_inline:nnnn {0} {1} {\l_hobby_npoints_int - 1} {
561
562 \array_put:Nnx \l_hobby_theta_array {##1} {\fp_to_tl:n {

```

```

563   (\array_get:Nn \l_hobby_theta_array {##1})
564   - (\array_get:Nn \l_hobby_vector_u_array {##1})
565   * \l_hobby_tempb_fp
566 }}
567 }
568 }

```

Now that we have computed the  $\theta_i$ s, we can quickly compute the  $\phi_i$ s.

```

569 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int - 1} {
570
571   \array_put:Nnx \l_hobby_phi_array {##1} {\fp_to_tl:n {
572     - (\array_get:Nn \l_hobby_psi_array {##1})
573     - (\array_get:Nn \l_hobby_theta_array {##1})
574   }}
575 }

```

If the path is open, this works for all except  $\phi_n$ . If the path is closed, we can drop our added point. Cheaply, of course.

```

576 \bool_if:NTF \l_hobby_closed_bool {
577   \int_gdecr:N \l_hobby_npoints_int
578 }{

```

If  $\phi_n$  was not given, we compute it from  $\theta_{n-1}$ .

```

579 \fp_if_undefined:NT \l_hobby_in_angle_fp
580 {
581   \array_put:Nnx \l_hobby_phi_array {\l_hobby_npoints_int} {\fp_to_tl:n {
582     ((3 * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int}) - 1)
583     * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^3
584     * \l_hobby_out_curl_fp
585     +
586     (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int })^3)
587     /
588     ((3 * (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 2}) - 1)
589     * (\array_get:Nn \l_hobby_tension_in_array {\l_hobby_npoints_int})^3 \l_tmpa_tl
590     +
591     (\array_get:Nn \l_hobby_tension_out_array {\l_hobby_npoints_int - 1})^3
592     * \l_hobby_out_curl_fp)
593     *
594     (\array_get:Nn \l_hobby_theta_array {\l_hobby_npoints_int - 1})
595   }}
596 }
597 }
598 }

```

`\hobby_build_path:` Once we've computed the angles, we build the actual path.

```

599 \cs_new:Nn \hobby_build_path:
600 {

```

Next task is to compute the  $\rho_i$  and  $\sigma_i$ .

```

601 \int_step_inline:nnnn {0} {1} {\l_hobby_npoints_int - 1} {
602
603   \fp_set:Nn \l_hobby_tempa_fp {\array_get:Nn \l_hobby_theta_array {##1}}
604
605   \fp_set:Nn \l_hobby_tempb_fp {\array_get:Nn \l_hobby_phi_array {##1 + 1}}
606
607   \hobby_ctrlllen:NVV \l_hobby_temps_fp \l_hobby_tempa_fp \l_hobby_tempb_fp
608

```



```

609   \array_put:Nnx \l_hobby_sigma_array {##1 + 1} {\fp_to_tl:N \l_hobby_temps_fp}
610
611   \hobby_ctrlilen:NVV \l_hobby_temps_fp \l_hobby_tempb_fp \l_hobby_tempa_fp
612
613   \array_put:Nnx \l_hobby_rho_array {##1} {\fp_to_tl:N \l_hobby_temps_fp}
614
615 }

```

Lastly, we generate the coordinates of the control points.

```

616 \int_step_inline:nnnn {0} {1} {\l_hobby_npoints_int - 1} {
617   \array_gput:Nnx \l_hobby_controla_array {##1 + 1} {x = \fp_eval:n {
618     (\array_get:Nn \l_hobby_points_x_array {##1})
619     +
620     (\array_get:Nn \l_hobby_distances_array {##1}) *
621     (\array_get:Nn \l_hobby_rho_array {##1}) *
622     cos ( (\array_get:Nn \l_hobby_angles_array {##1})
623     +
624     (\array_get:Nn \l_hobby_theta_array {##1}))
625     /3
626   }, y = \fp_eval:n {
627     (\array_get:Nn \l_hobby_points_y_array {##1}) +
628     (\array_get:Nn \l_hobby_distances_array {##1}) *
629     (\array_get:Nn \l_hobby_rho_array {##1}) *
630     sin ( (\array_get:Nn \l_hobby_angles_array {##1})
631     +
632     (\array_get:Nn \l_hobby_theta_array {##1}))
633     /3
634   }
635 }
636 }
637 \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int} {
638   \array_gput:Nnx \l_hobby_controlb_array {##1} {
639     x = \fp_eval:n {\array_get:Nn \l_hobby_points_x_array {##1}
640       - (\array_get:Nn \l_hobby_distances_array {##1 - 1})
641       * (\array_get:Nn \l_hobby_sigma_array {##1})
642       * cos((\array_get:Nn \l_hobby_angles_array {##1 - 1})
643       - (\array_get:Nn \l_hobby_phi_array {##1}))/3
644     }, y = \fp_eval:n {
645       (\array_get:Nn \l_hobby_points_y_array {##1})
646       - (\array_get:Nn \l_hobby_distances_array {##1 - 1})
647       * (\array_get:Nn \l_hobby_sigma_array {##1})
648       * sin((\array_get:Nn \l_hobby_angles_array {##1 - 1})
649       - (\array_get:Nn \l_hobby_phi_array {##1}))/3
650     }
651   }
652 }

```

`\hobbyinit` Initialise the settings for Hobby's algorithm

```

653 \NewDocumentCommand \hobbyinit {m m m} {
654   \hobby_set_cmds:nnn#1#2#3
655   \hobby_clear_path:
656 }

```

`\hobbyaddpoint` This adds a point, possibly with tensions, to the current stack.

```

657 \NewDocumentCommand \hobbyaddpoint { m } {
658   \keys_set:nn { hobby/read in all }
659   {

```

```

660     tension~out,
661     tension~in,
662     excess~angle,
663     blank,
664     break,
665     #1
666 }
667 \tl_if_eq:NnTF {\l_tmpa_tl} {true}
668 {\tl_set:Nn \l_tmpa_tl {2}}
669 {
670     \tl_if_eq:NnTF {\l_tmpa_tl} {soft}
671     {\tl_set:Nn \l_tmpa_tl {0}}
672     {\tl_set:Nn \l_tmpa_tl {1}}
673 }
674 \tl_if_eq:NnTF {\l_tmpb_tl} {true}
675 {\tl_put_right:Nn \l_tmpa_tl {1}}
676 {\tl_put_right:Nn \l_tmpa_tl {0}}
677 \array_gpush:Nx \l_hobby_actions_array {\l_tmpa_tl}
678 \array_gpush:Nx \l_hobby_tension_out_array {\fp_to_tl:N \l_hobby_tempc_fp}
679 \array_gpush:Nx \l_hobby_tension_in_array {\fp_to_tl:N \l_hobby_tempd_fp}
680 \array_gpush:Nx \l_hobby_excess_angle_array {\fp_to_tl:N \l_hobby_temps_fp}
681 \array_gpush:Nx \l_hobby_points_array {
682     x = \fp_use:N \l_hobby_tempa_fp,
683     y = \fp_use:N \l_hobby_tempb_fp }
684 \array_gpush:Nx \l_hobby_points_x_array {\fp_to_tl:N \l_hobby_tempa_fp}
685 \array_gpush:Nx \l_hobby_points_y_array {\fp_to_tl:N \l_hobby_tempb_fp}
686 }

```

`\hobbysetparams` This sets the parameters for the curve.

```

687 \NewDocumentCommand \hobbysetparams { m } {
688     \keys_set:nn { hobby / read in params }
689     {
690         #1
691     }
692 }

```

`\hobby_set_cmds:nnn` The path-generation code doesn't know what to actually do with the path so the initialisation code will set some macros to do that. This is an auxiliary command that sets these macros.

```

693 \cs_new:Npn \hobby_moveto:nnn #1#2#3 {}
694 \cs_new:Npn \hobby_curveto:nnn #1#2#3 {}
695 \cs_new:Npn \hobby_close:n #1 {}
696 \cs_generate_variant:Nn \hobby_moveto:nnn {VVV,nnV}
697 \cs_generate_variant:Nn \hobby_curveto:nnn {VVV}
698 \cs_generate_variant:Nn \hobby_close:n {V}
699 \cs_new:Nn \hobby_set_cmds:nnn {
700     \cs_gset_eq:NN \hobby_moveto:nnn #1
701     \cs_gset_eq:NN \hobby_curveto:nnn #2
702     \cs_gset_eq:NN \hobby_close:n #3
703 }

```

`\hobbygenpath` This is the user (well, sort of) command that generates the curve.

```

704 \NewDocumentCommand \hobbygenpath { } {
705     \array_if_empty:NF \l_hobby_points_array {
706         \hobby_gen_path:
707     }
708 }

```

`\hobbygenifnecpath` If the named path doesn't exist, it is generated and named. If it does exist, we restore it. Either way, we save it to the aux file.

```

709 \NewDocumentCommand \hobbygenifnecpath { m } {
710   \tl_if_exist:cTF {g_hobby_#1_path}
711   {
712     \tl_use:c {g_hobby_#1_path}
713   }
714   {
715     \hobby_gen_path:
716   }
717   \hobby_save_path:n {#1}
718   \hobby_save_path_to_aux:x {#1}
719 }

```

`\hobbygenifnecusepath` If the named path doesn't exist, it is generated and named. If it does exist, we restore it. Either way, we save it to the aux file.

```

720 \NewDocumentCommand \hobbygenuseifnecpath { m } {
721   \tl_if_exist:cTF {g_hobby_#1_path}
722   {
723     \tl_use:c {g_hobby_#1_path}
724   }
725   {
726     \hobby_gen_path:
727   }
728   \hobby_save_path:n {#1}
729   \hobby_save_path_to_aux:x {#1}
730   \hobby_use_path:
731 }

```

`\hobbyusepath` This is the user (well, sort of) command that uses the last generated curve.

```

732 \NewDocumentCommand \hobbyusepath { m } {
733   \hobbysetparams{#1}
734   \hobby_use_path:
735 }

```

`\hobbysavepath` This is the user (well, sort of) command that uses the last generated curve.

```

736 \NewDocumentCommand \hobbysavepath { m } {
737   \hobby_save_path:n {#1}
738 }

```

`\hobbyrestorepath` This is the user (well, sort of) command that uses the last generated curve.

```

739 \NewDocumentCommand \hobbyrestorepath { m } {
740   \tl_if_exist:cT {g_hobby_#1_path} {
741     \tl_use:c {g_hobby_#1_path}
742   }
743 }

```

`\hobbyshowpath` This is the user (well, sort of) command that uses the last generated curve.

```

744 \NewDocumentCommand \hobbyshowpath { m } {
745   \tl_if_exist:cT {g_hobby_#1_path} {
746     \tl_show:c {g_hobby_#1_path}
747   }
748 }

```

`\hobbygenusepath` This is the user (well, sort of) command that generates a curve and uses it.

```

749 \NewDocumentCommand \hobbygenusepath { } {
750   \array_if_empty:NF \l_hobby_points_array {
751     \hobby_gen_path:
752     \hobby_use_path:
753   }
754 }

```

`\hobbyclearpath` This is the user (well, sort of) command that generates a curve and uses it.

```

755 \NewDocumentCommand \hobbyclearpath { } {
756   \hobby_clear_path:
757 }

```

`\hobby_use_path:` This is the command that uses the curve. As the curve data is stored globally, the same data can be reused by calling this function more than once without calling the generating function.

```

758 \tl_new:N \l_tmpc_tl
759 \cs_new:Nn \hobby_use_path: {
760   \bool_if:NT \l_hobby_disjoint_bool {
761     \array_get:NnN \l_hobby_points_array {0} \l_tmpa_tl
762     \hobby_moveto:nnV {} {} \l_tmpa_tl
763   }
764   \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int} {
765     \array_get:NnN \l_hobby_controla_array {##1} \l_tmpa_tl
766     \array_get:NnN \l_hobby_controlb_array {##1} \l_tmpb_tl
767     \array_get:NnN \l_hobby_points_array {##1} \l_tmpc_tl
768     \array_get:NnN \l_hobby_actions_array {##1} \l_tmpd_tl
769     \int_compare:nNnTF {\tl_item:Nn \l_tmpd_tl {1}} = {\l_hobby_draw_int} {
770       \hobby_curveto:VVV \l_tmpa_tl \l_tmpb_tl \l_tmpc_tl
771     }{
772       \bool_gset_false:N \l_hobby_closed_bool
773       \hobby_moveto:VVV \l_tmpa_tl \l_tmpb_tl \l_tmpc_tl
774     }
775     \tl_if_eq:xnTF {\tl_item:Nn \l_tmpd_tl {2}} {1} {
776       \bool_gset_false:N \l_hobby_closed_bool
777       \hobby_moveto:VVV \l_tmpa_tl \l_tmpb_tl \l_tmpc_tl
778     }{}
779   }
780   \bool_if:NT \l_hobby_closed_bool {
781     \array_get:NnN \l_hobby_points_array {0} \l_tmpa_tl
782     \hobby_close:V \l_tmpa_tl
783   }
784 }

```

`\hobby_save_path:n` This command saves all the data needed to reinvoke the curve in a global token list that can be used to restore it afterwards.

```

785 \cs_new:Nn \hobby_save_path:n {
786   \tl_clear:N \l_tmpa_tl
787   \tl_put_right:Nn \l_tmpa_tl {\int_gset:Nn \l_hobby_npoints_int}
788   \tl_put_right:Nx \l_tmpa_tl {\int_use:N \l_hobby_npoints_int}
789   \bool_if:NTF \l_hobby_disjoint_bool {
790     \tl_put_right:Nn \l_tmpa_tl {\bool_gset_true:N}
791   }{
792     \tl_put_right:Nn \l_tmpa_tl {\bool_gset_false:N}
793   }
794   \tl_put_right:Nn \l_tmpa_tl {\l_hobby_disjoint_bool}
795   \bool_if:NTF \l_hobby_closed_bool {

```

```

796 \tl_put_right:Nn \l_tmpa_tl {\bool_gset_true:N}
797 }{
798 \tl_put_right:Nn \l_tmpa_tl {\bool_gset_false:N}
799 }
800 \tl_put_right:Nn \l_tmpa_tl {\l_hobby_closed_bool}
801 \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \l_hobby_points_array}
802 \array_map_inline:Nn \l_hobby_points_array {
803 \tl_put_right:Nn \l_tmpa_tl {
804 \array_gput:Nnn \l_hobby_points_array {##1} {##2}
805 }
806 }
807 \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \l_hobby_actions_array}
808 \array_map_inline:Nn \l_hobby_actions_array {
809 \tl_put_right:Nn \l_tmpa_tl {
810 \array_gput:Nnn \l_hobby_actions_array {##1} {##2}
811 }
812 }
813 \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \l_hobby_controla_array}
814 \array_map_inline:Nn \l_hobby_controla_array {
815 \tl_put_right:Nn \l_tmpa_tl {
816 \array_gput:Nnn \l_hobby_controla_array {##1} {##2}
817 }
818 }
819 \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \l_hobby_controlb_array}
820 \array_map_inline:Nn \l_hobby_controlb_array {
821 \tl_put_right:Nn \l_tmpa_tl {
822 \array_gput:Nnn \l_hobby_controlb_array {##1} {##2}
823 }
824 }
825 \tl_gclear_new:c {g_hobby_#1_path}
826 \tl_gset_eq:cN {g_hobby_#1_path} \l_tmpa_tl
827 }

```

hobby\_save\_path\_to\_aux:n

```

828 \int_set:Nn \l_tmpa_int {\char_value_catcode:n {'@}}
829 \char_set_catcode_letter:N @
830 \cs_new:Npn \hobby_save_path_to_aux:n #1 {
831 \bool_if:nT {
832 \tl_if_exist_p:c {g_hobby_#1_path}
833 &&
834 ! \tl_if_exist_p:c {g_hobby_#1_path_saved}
835 &&
836 \l_hobby_save_aux_bool
837 }
838 {
839 \tl_clear:N \l_tmpa_tl
840 \tl_put_right:Nn \l_tmpa_tl {
841 \ExplSyntaxOn
842 \tl_gclear_new:c {g_hobby_#1_path}
843 \tl_gput_right:cn {g_hobby_#1_path}
844 }
845 \tl_put_right:Nx \l_tmpa_tl {
846 {\tl_to_str:c {g_hobby_#1_path}}
847 }
848 \tl_put_right:Nn \l_tmpa_tl {
849 \ExplSyntaxOff

```

```

850     }
851     \protected@write\@auxout{}\{
852         \tl_to_str:N \l_tmpa_tl
853     }
854     \tl_new:c {g_hobby_#1_path_saved}
855 }
856 }
857 \char_set_catcode:nn {'@} {\l_tmpa_int}
858 \cs_generate_variant:Nn \hobby_save_path_to_aux:n {x}

```

\hobby\_clear\_path:

```

859 \cs_new:Nn \hobby_clear_path:
860 {
861     \array_gclear:N \l_hobby_points_array
862     \array_gclear:N \l_hobby_points_x_array
863     \array_gclear:N \l_hobby_points_y_array
864     \array_gclear:N \l_hobby_angles_array
865     \array_gclear:N \l_hobby_actions_array
866     \array_gclear:N \l_hobby_distances_array
867     \array_gclear:N \l_hobby_tension_out_array
868     \array_gclear:N \l_hobby_tension_in_array
869     \array_gclear:N \l_hobby_excess_angle_array
870     \array_gclear:N \l_hobby_matrix_a_array
871     \array_gclear:N \l_hobby_matrix_b_array
872     \array_gclear:N \l_hobby_matrix_c_array
873     \array_gclear:N \l_hobby_matrix_d_array
874     \array_gclear:N \l_hobby_vector_u_array
875     \array_gclear:N \l_hobby_psi_array
876     \array_gclear:N \l_hobby_theta_array
877     \array_gclear:N \l_hobby_phi_array
878     \array_gclear:N \l_hobby_sigma_array
879     \array_gclear:N \l_hobby_rho_array
880     \array_gclear:N \l_hobby_controla_array
881     \array_gclear:N \l_hobby_controlb_array
882     \bool_gset_false:N \l_hobby_closed_bool
883     \bool_gset_false:N \l_hobby_disjoint_bool
884
885     \int_gset:Nn \l_hobby_npoints_int {-1}
886     \int_gset:Nn \l_hobby_draw_int {1}
887     \fp_gset_eq:NN \l_hobby_in_angle_fp \c_undefined_fp
888     \fp_gset_eq:NN \l_hobby_out_angle_fp \c_undefined_fp
889     \fp_gset_eq:NN \l_hobby_in_curl_fp \c_one_fp
890     \fp_gset_eq:NN \l_hobby_out_curl_fp \c_one_fp
891 }
892 \ExplSyntaxOff

```

## 1.2 PGF Library

The PGF level is very simple. All we do is set up the path-construction commands that get passed to the path-generation function.

```

893 \input{hobby.code.tex}

```

Points are communicated as key-pairs. These keys translate from the L<sup>A</sup>T<sub>E</sub>X3 style points to PGF points.

```

894 \pgfkeys{
895     /pgf/hobby/.is family,

```

```

896 /pgf/hobby/.cd,
897 x/.code={\pgf@x=#1cm},
898 y/.code={\pgf@y=#1cm}
899 }

```

**hobbyatan2** The original PGF version of `atan2` had the arguments the wrong way around. This was fixed in the CVS version in July 2013, but as old versions are likely to be in use for some time, we define a wrapper function that ensures that the arguments are correct.

```

900 \pgfmathparse{atan2(0,1)}
901 \def\hobby@temp{0.0}
902 \ifx\pgfmathresult\hobby@temp
903   \pgfmathdeclarefunction{hobbyatan2}{2}{%
904     \pgfmathatan2@{#1}{#2}%
905   }
906 \else
907   \pgfmathdeclarefunction{hobbyatan2}{2}{%
908     \pgfmathatan2@{#2}{#1}%
909   }
910 \fi

```

**\hobby@curveto** This is passed to the path-generation code to translate the path into a PGF path.

```

911 \def\hobby@curveto#1#2#3{%
912   \pgfpathcurveto{\hobby@topgf{#1}}{\hobby@topgf{#2}}{\hobby@topgf{#3}}%
913 }

```

**\hobby@moveto** This is passed to the path-generation code to translate the path into a PGF path.

```

914 \def\hobby@moveto#1#2#3{%
915   \pgfpathmoveto{\hobby@topgf{#3}}%
916 }

```

**\hobby@topgf** Translates a L<sup>A</sup>T<sub>E</sub>X3 point to a PGF point.

```

917 \def\hobby@topgf#1{%
918   \pgfqkeys{/pgf/hobby}{#1}%
919 }

```

**\hobby@close** Closes a path.

```

920 \def\hobby@close#1{%
921   \pgfpathclose
922 }

```

Plot handlers

**\pgfplothandlerhobby** Basic plot handler; uses full algorithm but therefore expensive

```

923 \def\pgfplothandlerhobby{%
924   \def\pgf@plotstreamstart{%
925     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
926     \global\let\pgf@plotstreampoint=\pgf@plot@hobby@firstpt
927     \global\let\pgf@plotstreamspecial=\pgfutil@gobble
928     \gdef\pgf@plotstreamend{%
929       \ifhobby@externalise
930       \ifx\hobby@path@name\pgfutil@empty
931         \hobbygenusepath
932       \else
933         \hobbygenuseifnecpath{\hobby@path@name}%
934       \fi
935     \else

```

```

936     \hobbygenusepath
937   \fi
938   \ifx\hobby@path@name\pgfutil@empty
939   \else
940     \hobbysavepath{\hobby@path@name}%
941   \fi
942   \global\let\hobby@path@name=\pgfutil@empty
943 }%
944 \let\tikz@scan@point@options=\pgfutil@empty
945 }
946 }

```

`pgfplotshandlerclosedhobby` Same as above but produces a closed curve

```

947 \def\pgfplotshandlerclosedhobby{%
948   \def\pgf@plotstreamstart{%
949     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
950     \hobbysetparams{closed=true,disjoint=true}%
951     \global\let\pgf@plotstreampoint=\pgf@plot@hobby@firstpt
952     \global\let\pgf@plotstreamspecial=\pgfutil@gobble
953   \gdef\pgf@plotstreamend{%
954     \ifhobby@externalise
955       \ifx\hobby@path@name\pgfutil@empty
956         \hobbygenusepath
957       \else
958         \hobbygenuseifnecpath{\hobby@path@name}%
959       \fi
960     \else
961       \hobbygenusepath
962     \fi
963     \ifx\hobby@path@name\pgfutil@empty
964     \else
965       \hobbysavepath{\hobby@path@name}%
966     \fi
967     \global\let\hobby@path@name=\pgfutil@empty
968   }%
969 }
970 }

```

`pgf@plot@hobby@firstpt` First point, move or line as appropriate and then start the algorithm.

```

971 \def\pgf@plot@hobby@firstpt#1{%
972   \pgf@plot@first@action{#1}%
973   \pgf@plot@hobby@handler{#1}%
974   \global\let\pgf@plotstreampoint=\pgf@plot@hobby@handler
975 }

```

`pgf@plot@hobby@handler` Add points to the array for the algorithm to work on.

```

976 \def\pgf@plot@hobby@handler#1{%
977   #1%
978   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
979   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
980   \hobbyaddpoint{x = \hobby@x, y = \hobby@y}%
981 }

```

`pgfplotshandlerquikhobby` Uses the “quick” algorithm.

```

982 \def\pgfplotshandlerquikhobby{%
983   \def\pgf@plotstreamstart{%

```



```

984 \global\let\hobby@quick@curveto=\pgfpathcurveto
985 \global\let\pgf@plotstreampoint=\pgf@plot@qhobby@firstpt
986 \global\let\pgf@plotstreamspecial=\pgfutil@gobble
987 \global\let\pgf@plotstreamend=\pgf@plot@qhobby@end
988 }
989 }

```

`\pgf@plot@qhobby@firstpt` Carry out first action (move or line) and save point.

```

990 \def\pgf@plot@qhobby@firstpt#1{%
991   #1%
992   \edef\hobby@temp{\noexpand\pgf@plot@first@action{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}}\hobby@temp
993   \xdef\hobby@qpoints{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
994   \gdef\hobby@qpointa{%
995     \gdef\hobby@angle{%
996       \global\let\pgf@plotstreampoint=\pgf@plot@qhobby@secondpt
997     }

```

`\pgf@plot@qhobby@secondpt` Also need to save second point.

```

998 \def\pgf@plot@qhobby@secondpt#1{%
999   #1%
1000   \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1001   \global\let\pgf@plotstreampoint=\pgf@plot@qhobby@handler
1002 }

```

`\pgf@plot@qhobby@handler` Wrapper around the computation macro that saves the variables globally.

```

1003 \def\pgf@plot@qhobby@handler#1{%
1004   #1
1005   \edef\hobby@temp{\noexpand\hobby@quick@compute{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}}\hobby@temp
1006   \global\let\hobby@qpointa=\hobby@qpointa
1007   \global\let\hobby@qpoints=\hobby@qpoints
1008   \global\let\hobby@angle=\hobby@angle

```

Also need to save some data for the last point

```

1009 \global\let\hobby@thetaone=\hobby@thetaone
1010 \global\let\hobby@phitwo=\hobby@phitwo
1011 \global\let\hobby@done=\hobby@done
1012 \global\let\hobby@omegaone=\hobby@omegaone
1013 }

```

`\pgf@plot@qhobby@end` Wrapper around the finalisation step.

```

1014 \def\pgf@plot@qhobby@end{%
1015   \hobby@quick@computeend
1016 }

```

`\hobby@sf` Working with points leads to computations out of range so we scale to get them into the computable arena.

```

1017 \pgfmathsetmacro\hobby@sf{10cm}

```

`\hobby@quick@compute` This is the macro that does all the work of computing the control points. The argument is the current point, `\hobby@qpointa` is the middle point, and `\hobby@qpoints` is the first point.

```

1018 \def\hobby@quick@compute#1{%

```

Save the current (second - counting from zero) point in `\pgf@xb` and `\pgf@yb`.

```

1019   #1%
1020   \pgf@xb=\pgf@x
1021   \pgf@yb=\pgf@y

```

Save the previous (first) point in `\pgf@xa` and `\pgf@ya`.

```
1022 \hobby@qpointa
1023 \pgf@xa=\pgf@x
1024 \pgf@ya=\pgf@y
```

Adjust so that `(\pgf@xb,\pgf@yb)` is the vector from second to third. Then compute and store the distance and angle of this vector. We view this as the vector *from* the midpoint and everything to do with that point has the suffix *one*. Note that we divide by the scale factor here.

```
1025 \advance\pgf@xb by -\pgf@xa
1026 \advance\pgf@yb by -\pgf@ya
1027 \pgfmathsetmacro\hobby@done{sqrt((\pgf@xb/\hobby@sf)^2 + (\pgf@yb/\hobby@sf)^2)}%
1028 \pgfmathsetmacro\hobby@omegaone{rad(hobbyatan2(\pgf@yb,\pgf@xb))}%
```

Now we do the same with the vector from the zeroth to the first point.

```
1029 \hobby@qpoints
1030 \advance\pgf@xa by -\pgf@x
1031 \advance\pgf@ya by -\pgf@y
1032 \pgfmathsetmacro\hobby@dzero{sqrt((\pgf@xa/\hobby@sf)^2 + (\pgf@ya/\hobby@sf)^2)}%
1033 \pgfmathsetmacro\hobby@omegazero{rad(hobbyatan2(\pgf@ya,\pgf@xa))}%
```

`\hobby@psi` is the angle subtended at the midpoint. We adjust to ensure that it is in the right range.

```
1034 \pgfmathsetmacro\hobby@psi{\hobby@omegaone - \hobby@omegazero}%
1035 \pgfmathsetmacro\hobby@psi{\hobby@psi > pi ? \hobby@psi - 2*pi : \hobby@psi}%
1036 \pgfmathsetmacro\hobby@psi{\hobby@psi < -pi ? \hobby@psi + 2*pi : \hobby@psi}%
```

Now we test to see if we're on the first run or not. If the first, we have no incoming angle.

```
1037 \ifx\hobby@angle\pgfutil@empty
```

First.

```
1038 \pgfmathsetmacro\hobby@thetaone{-\hobby@psi * \hobby@done%
1039 /(\hobby@done + \hobby@dzero)}%
1040 \pgfmathsetmacro\hobby@thetazero{-\hobby@psi - \hobby@thetaone}%
1041 \let\hobby@phione=\hobby@thetazero
1042 \let\hobby@phitwo=\hobby@thetaone
1043 \else
```

Second or later.

```
1044 \let\hobby@thetazero=\hobby@angle
1045 \pgfmathsetmacro\hobby@thetaone{%
1046 -(2 * \hobby@psi + \hobby@thetazero) * \hobby@done%
1047 / (2 * \hobby@done + \hobby@dzero)}%
1048 \pgfmathsetmacro\hobby@phione{-\hobby@psi - \hobby@thetaone}%
1049 \let\hobby@phitwo=\hobby@thetaone
1050 \fi
```

Save the outgoing angle.

```
1051 \let\hobby@angle=\hobby@thetaone
```

Compute the control points from the angles.

```
1052 \hobby@quick@ctrlpts{\hobby@thetazero}{\hobby@phione}{\hobby@qpoints}{\hobby@qpointa}{\hobby@dzero}{\hobby@thetaone}
```

Now call the call-back function

```
1053 \edef\hobby@temp{\noexpand\hobby@quick@curveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}{\noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}{\noexpand\pgfqpoint{\the\pgf@xc}{\the\pgf@yc}}}%
1054 \hobby@temp
```

Cycle the points round for the next iteration.

```
1055 \global\let\hobby@qpoints=\hobby@qpointa
1056 #1
1057 \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
```

Save needed values in global macros

```

1058 \global\let\hobby@angle=\hobby@angle
1059 \global\let\hobby@phitwo=\hobby@phitwo
1060 \global\let\hobby@thetaone=\hobby@thetaone
1061 \global\let\hobby@done=\hobby@done
1062 \global\let\hobby@omegaone=\hobby@omegaone
1063 }

```

\hobby@quick@computeend This is the additional code for the final run.

```

1064 \def\hobby@quick@computeend{%

```

Compute the control points for the second part of the curve and add that to the path.

```

1065 \hobby@quick@ctrlpts{\hobby@thetaone}{\hobby@phitwo}{\hobby@qpoints}{\hobby@qpinta}{\hobby@done}{\h

```

Now call the call-back function

```

1066 \edef\hobby@temp{\noexpand\hobby@quick@curveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}{\noexp
1067 \hobby@temp
1068 }%

```

\hobby@quick@ctrlpts Compute the control points from the angles and points given.

```

1069 \def\hobby@quick@ctrlpts#1#2#3#4#5#6{%
1070 \pgfmathsetmacro\hobby@alpha{%
1071 sqrt(2) * (sin(#1 r) - 1/16 * sin(#2 r))%
1072 * (sin(#2 r) - 1/16 * sin(#1 r))%
1073 * (cos(#1 r) - cos(#2 r))}%
1074 \pgfmathsetmacro\hobby@rho{%
1075 (2 + \hobby@alpha)/(1 + (1 - (3 - sqrt(5))/2))%
1076 * cos(#1 r) + (3 - sqrt(5))/2 * cos(#2 r))}%
1077 \pgfmathsetmacro\hobby@sigma{%
1078 (2 - \hobby@alpha)/(1 + (1 - (3 - sqrt(5))/2))%
1079 * cos(#2 r) + (3 - sqrt(5))/2 * cos(#1 r))}%
1080 #3%
1081 \pgf@xa=\pgf@x
1082 \pgf@ya=\pgf@y
1083 \pgfmathsetlength\pgf@xa{%
1084 \pgf@xa + #5 * \hobby@rho%
1085 * cos((#1 + #6) r)/3*\hobby@sf}%
1086 \pgfmathsetlength\pgf@ya{%
1087 \pgf@ya + #5 * \hobby@rho%
1088 * sin((#1 + #6) r)/3*\hobby@sf}%
1089 #4%
1090 \pgf@xb=\pgf@x
1091 \pgf@yb=\pgf@y
1092 \pgfmathsetlength\pgf@xb{%
1093 \pgf@xb - #5 * \hobby@sigma%
1094 * cos((-#2 + #6) r)/3*\hobby@sf}%
1095 \pgfmathsetlength\pgf@yb{%
1096 \pgf@yb - #5 * \hobby@sigma%
1097 * sin((-#2 + #6) r)/3*\hobby@sf}%
1098 #4%
1099 }

```

### 1.3 TikZ Library

```

1100 \usepgflibrary{hobby}
1101 \let\hobby@this@opts=\pgfutil@empty
1102 \let\hobby@next@opts=\pgfutil@empty
1103 \let\hobby@action=\pgfutil@empty
1104 \let\hobby@path@name=\pgfutil@empty
1105 \newif\ifhobby@externalise

```

We set various TikZ keys. These include the `to path` constructor and all the various parameters that will eventually get passed to the path-generation code.

```

1106 \def\hobby@point@options{%
1107 \tikzset{
1108   curve through/.style={
1109     to path={
1110       \pgfextra{
1111         \expandafter\curvethrough\expandafter[\hobby@next@opts]{%
1112           (\tikztostart) .. #1 .. (\tikztotarget)%
1113         }
1114       }
1115     }
1116   },
1117   tension in/.code = {%
1118     \expandafter\gdef\expandafter\hobby@point@options\expandafter{%
1119       {\hobby@point@options,tension in=#1}%
1120     },
1121   tension out/.code = {%
1122     \expandafter\gdef\expandafter\hobby@point@options\expandafter{%
1123       {\hobby@point@options,tension out=#1}%
1124     },
1125   tension/.code = {%
1126     \expandafter\gdef\expandafter\hobby@point@options\expandafter{%
1127       {\hobby@point@options,tension=#1}%
1128     },
1129   excess angle/.code = {%
1130     \expandafter\gdef\expandafter\hobby@point@options\expandafter{%
1131       {\hobby@point@options,excess angle=#1}%
1132     },
1133   break/.code = {%
1134     \expandafter\gdef\expandafter\hobby@point@options\expandafter{%
1135       {\hobby@point@options,break=#1}%
1136     },
1137   blank/.code = {%
1138     \expandafter\gdef\expandafter\hobby@point@options\expandafter{%
1139       {\hobby@point@options,blank=#1}%
1140     },
1141   designated Hobby path/.initial={next},
1142   clear next Hobby path options/.code={%
1143     \gdef\hobby@next@opts{%
1144   },
1145   clear this Hobby path options/.code={%
1146     \gdef\hobby@this@opts{%
1147   },
1148   clear Hobby path options/.style={%
1149     clear \pgfkeysvalueof{/tikz/designated Hobby path} Hobby path options
1150   },
1151   add option to this Hobby path/.code={%

```

```

1152     \expandafter\gdef\expandafter\hobby@this@opts\expandafter{\hobby@this@opts#1,}%
1153 },
1154 add option to next Hobby path/.code={%
1155     \expandafter\gdef\expandafter\hobby@next@opts\expandafter{\hobby@next@opts#1,}%
1156 },
1157 add option to Hobby path/.style={%
1158     add option to \pgfkeysvalueof{/tikz/designated Hobby path} Hobby path={#1}%
1159 },
1160 closed/.style = {%
1161     add option to Hobby path={closed=#1,disjoint=#1}%
1162 },
1163 invert blank/.style = {%
1164     add option to Hobby path={invert blank=#1}%
1165 },
1166 closed/.default = true,
1167 blank/.default = true,
1168 break/.default = true,
1169 invert blank/.default = true,
1170 in angle/.code = {%
1171     \pgfmathparse{#1*pi/180}%
1172     \edef\@temp{in angle=\pgfmathresult,}%
1173     \pgfkeysalso{add option to Hobby path/.expand once=\@temp}%
1174 },
1175 out angle/.code = {%
1176     \pgfmathparse{#1*pi/180}%
1177     \edef\@temp{out angle=\pgfmathresult,}%
1178     \pgfkeysalso{add option to Hobby path/.expand once=\@temp}%
1179 },
1180 in curl/.style = {%
1181     add option to Hobby path={in curl=#1}%
1182 },
1183 out curl/.code = {%
1184     add option to Hobby path={out curl=#1}%
1185 },
1186 use Hobby shortcut/.code={%
1187     \let\tikz@curveto@auto=\hobby@curveto@override
1188     \global\let\hobby@curveto@delegate=\hobby@curveto@auto
1189 },
1190 use quick Hobby shortcut/.code={%
1191     \let\tikz@curveto@auto=\hobby@curveto@override
1192     \global\let\hobby@curveto@delegate=\hobby@qcurveto@auto
1193 },
1194 use previous Hobby path/.code={%
1195     \pgfextra{\hobbyusepath{#1}}
1196 },
1197 use previous Hobby path/.default={},%
1198 save Hobby path/.code={%
1199     \xdef\hobby@path@name{#1}%
1200 },
1201 restore Hobby path/.code={%
1202     \pgfextra{%
1203         \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1204         \global\let\hobby@collected@onpath\pgfutil@empty
1205         \hobbyrestorepath{#1}}
1206 },
1207 restore and use Hobby path/.code 2 args={%

```

```

1208 \pgfextra{%
1209 \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1210 \global\let\hobby@collected@onpath\pgfutil@empty
1211 \hobbyrestorepath{#1}%
1212 \hobbyusepath{#2}%
1213 }
1214 },
1215 show Hobby path/.code={%
1216 \pgfextra{\hobbyshowpath{#1}}
1217 },
1218 Hobby action/.code={%
1219 \expandafter\gdef\expandafter\hobby@action\expandafter{\hobby@action#1}%
1220 },
1221 Hobby finish/.style={%
1222 Hobby action=\hobby@finish%
1223 },
1224 Hobby externalise/.is if=hobby@externalise,
1225 Hobby externalize/.is if=hobby@externalise
1226 }

```

\hobby@tikz@curveto This is passed to the path-generation code to translate the path into a PGF path.

```

1227 \def\hobby@tikz@curveto#1#2#3{%
1228 \pgfutil@ifundefined{tikz@timer@start}{%
1229 \expandafter\hobby@topgf\expandafter{\hobby@initial@pt}%
1230 \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1231 }{}%
1232 \hobby@topgf{#1}%
1233 \edef\tikz@timer@cont@one{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1234 \hobby@topgf{#2}%
1235 \edef\tikz@timer@cont@two{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1236 \hobby@topgf{#3}%
1237 \let\tikz@timer=\tikz@timer@curve
1238 \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1239 \ifx\hobby@collected@onpath\pgfutil@empty
1240 \else
1241 \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1242 \fi
1243 \pgfpathcurveto{\hobby@topgf{#1}}{\hobby@topgf{#2}}{\hobby@topgf{#3}}%
1244 \hobby@topgf{#3}%
1245 \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1246 }

```

\hobby@tikz@moveto This is passed to the path-generation code to translate the path into a PGF path.

```

1247 \def\hobby@tikz@moveto#1#2#3{%
1248 \pgfutil@ifundefined{tikz@timer@start}{%
1249 \expandafter\hobby@topgf\expandafter{\hobby@initial@pt}%
1250 \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1251 }{}%
1252 \hobby@topgf{#3}%
1253 \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1254 \def\pgf@temp{#1}%
1255 \ifx\pgf@temp\pgfutil@empty
1256 \let\tikz@timer=\tikz@timer@line
1257 \else
1258 \hobby@topgf{#1}%
1259 \edef\tikz@timer@cont@one{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1260 \hobby@topgf{#2}%

```

```

1261     \edef\tikz@timer@cont@two{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1262     \let\tikz@timer=\tikz@timer@curve
1263 \fi
1264 \ifx\hobby@collected@onpath\pgfutil@empty
1265 \else
1266 \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1267 \fi
1268 \pgfpathmoveto{\hobby@topgf{#3}}%
1269 \hobby@topgf{#3}%
1270 \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1271 }

```

`\hobby@tikz@close` Closes a path.

```

1272 \def\hobby@tikz@close#1{%
1273 \hobby@topgf{#1}%
1274 \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1275 \let\tikz@timer=\tikz@timer@line
1276 \ifx\hobby@collected@onpath\pgfutil@empty
1277 \else
1278 \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1279 \fi
1280 \pgfpathclose
1281 }

```

`\hobby@nodes@onpath`

```

1282 \def\hobby@nodes@onpath#1#2\relax{%
1283 \gdef\hobby@collected@onpath{#2}%
1284 \def\pgf@temp{#1}%
1285 \ifx\pgf@temp\pgfutil@empty
1286 \else
1287 \def\@gtempa{\relax}
1288 \ifx\pgf@temp\@gtempa
1289 \else
1290 \tikz@node@is@a@labeltrue
1291 \tikz@scan@next@command#1\pgf@stop
1292 \tikz@node@is@a@labelfalse
1293 \fi
1294 \fi
1295 }

```

`\curvethrough` This is the parent command. We initialise the path-generation code, set any parameters, and then hand over control to the point processing macro.

```

1296 \newcommand\curvethrough[2][]{%
1297 \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1298 \global\let\hobby@collected@onpath\pgfutil@empty
1299 \let\hobby@initial@pt\pgfutil@empty
1300 \hobbysetparams{#1}%
1301 \tikzset{designated Hobby path=this}%
1302 \global\let\hobby@this@opts=\pgfutil@empty
1303 \global\let\hobby@next@opts=\pgfutil@empty
1304 \let\tikz@scan@point@options=\pgfutil@empty
1305 \def\hobby@point@options{}%
1306 \tikz@scan@one@point\hobby@processpt #2 \relax%
1307 }

```

`\hobby@processpt` This processes a list of points in the format (0,0) [...] (1,1). Each point is scanned by TikZ and then added to the stack to be built into the path. If there are any remaining points, we call ourself again with them. Otherwise, we hand over control to the path-generation code.

```

1308 \newcommand\hobby@processpt[1]{%
1309   #1%
1310   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1311   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1312   \ifx\hobby@initial@pt\pgfutil@empty
1313     \xdef\hobby@initial@pt{x = \hobby@x, y = \hobby@y}%
1314   \fi
1315   \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1316     x = \hobby@x, y = \hobby@y}%
1317   \def\hobby@point@options{}%
1318   \let\tikz@scan@point@options=\pgfutil@empty
1319   \pgfutil@ifnextchar\relax{%
1320     \expandafter\hobbysetparams\expandafter{\hobby@this@opts}%
1321   \ifhobby@externalise
1322     \ifx\hobby@path@name\pgfutil@empty
1323       \hobbygenusepath
1324     \else
1325       \hobbygenuseifnecpath{\hobby@path@name}%
1326     \fi
1327   \else
1328     \hobbygenusepath
1329   \fi
1330   \ifx\hobby@path@name\pgfutil@empty
1331   \else
1332     \hobbysavepath{\hobby@path@name}%
1333   \fi
1334   \global\let\hobby@path@name=\pgfutil@empty
1335 }{%
1336   \pgfutil@ifnextchar.{%
1337     \hobby@swallowdots}{%
1338     \tikz@scan@one@point\hobby@processpt}}}
```

`\hobby@swallowdots` Remove dots from the input stream.

```

1339 \def\hobby@swallowdots.{%
1340   \pgfutil@ifnextchar.{%
1341     \hobby@swallowdots}{%
1342     \tikz@scan@one@point\hobby@processpt}}
```

There is a “spare hook” in the TikZ path processing code. If TikZ encounters a path of the form (0,0) .. (1,1) then it calls a macro `\tikz@curveto@auto`. However, that macro is not defined in the TikZ code. The following code provides a suitable definition. To play nice, we don’t install it by default but define a key (defined above) that installs it.

`hobby@curveto@override`

```

1343 \def\hobby@curveto@override{%
1344   \hobby@curveto@delegate}
```

`\hobby@curveto@auto` When we’re called by TikZ, we initialise the path generation code and start adding points. To ensure that the generation code is called, we add a lot of hooks to lots of TikZ commands.

```

1345 \def\hobby@curveto@auto{%
1346   \hobbyinit\hobby\tikz@moveto\hobby\tikz@curveto\hobby\tikz@close
1347   \expandafter\gdef\expandafter\hobby@collected@onpath\expandafter{\expandafter{\tikz@collected@onpath
1348   \let\tikz@collected@onpath=\pgfutil@empty
```



```

1349 \pgfmathsetmacro\hobby@x{\the\tikz@lastx/1cm}%
1350 \pgfmathsetmacro\hobby@y{\the\tikz@lasty/1cm}%
1351 \xdef\hobby@initial@pt{x = \hobby@x, y = \hobby@y}%
1352 \expandafter\hobbysetparams\expandafter{\hobby@next@opts}%
1353 \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1354     x = \hobby@x, y = \hobby@y}%
1355 \hobby@init@tikz@commands
1356 \tikzset{designated Hobby path=this}%
1357 \let\tikz@scan@point@options=\pgfutil@empty
1358 \global\let\hobby@action=\pgfutil@empty
1359 \global\let\hobby@this@opts=\pgfutil@empty
1360 \global\let\hobby@next@opts=\pgfutil@empty
1361 \global\let\hobby@point@options=\pgfutil@empty
1362 \tikz@scan@one@point\hobby@addfromtikz%
1363 }

```

\hobby@addfromtikz This adds our current point to the stack.

```

1364 \def\hobby@addfromtikz#1{%
1365     #1%
1366     \tikz@make@last@position{#1}%
1367     \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1368     \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1369     \expandafter\hobbysetparams\expandafter{\hobby@this@opts}%
1370     \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1371         x = \hobby@x, y = \hobby@y}%
1372     \hobby@action
1373     \global\let\hobby@this@opts=\pgfutil@empty
1374     \global\let\hobby@action=\pgfutil@empty
1375     \global\let\hobby@point@options=\pgfutil@empty
1376     \tikz@scan@next@command%
1377 }

```

bby@init@tikz@commands

```

1378 \def\hobby@init@tikz@commands{%
1379     \hobby@init@tikz@modcmd\tikz@movetoabs
1380     \hobby@init@tikz@modcmd\tikz@movetorel
1381     \hobby@init@tikz@modcmd\tikz@lineto
1382     \hobby@init@tikz@modcmd\tikz@rect
1383     \hobby@init@tikz@modcmd\tikz@ccchar
1384     \hobby@init@tikz@modcmd\tikz@finish
1385     \hobby@init@tikz@modcmd\tikz@arcA
1386     \hobby@init@tikz@modcmd\tikz@e@char
1387     \hobby@init@tikz@modcmd\tikz@g@char
1388     \hobby@init@tikz@modcmd\tikz@schar
1389     \hobby@init@tikz@modcmd\tikz@vh@lineto
1390     \hobby@init@tikz@modcmd\tikz@pchar
1391     \hobby@init@tikz@modcmd\tikz@to
1392     \hobby@init@tikz@modcmd\pgf@stop
1393     \hobby@init@tikz@modcmd\tikz@decoration
1394     \global\let\hobby@curveto@delegate=\hobby@midcurveto@auto
1395 }

```

@restore@tikz@commands

```

1396 \def\hobby@restore@tikz@commands{%
1397     \hobby@restore@tikz@modcmd\tikz@movetoabs
1398     \hobby@restore@tikz@modcmd\tikz@movetorel

```

```

1399 \hobby@restore@tikz@modcmd\tikz@lineto
1400 \hobby@restore@tikz@modcmd\tikz@rect
1401 \hobby@restore@tikz@modcmd\tikz@ccchar
1402 \hobby@restore@tikz@modcmd\tikz@finish
1403 \hobby@restore@tikz@modcmd\tikz@arcA
1404 \hobby@restore@tikz@modcmd\tikz@e@char
1405 \hobby@restore@tikz@modcmd\tikz@g@char
1406 \hobby@restore@tikz@modcmd\tikz@schar
1407 \hobby@restore@tikz@modcmd\tikz@vh@lineto
1408 \hobby@restore@tikz@modcmd\tikz@pchar
1409 \hobby@restore@tikz@modcmd\tikz@to
1410 \hobby@restore@tikz@modcmd\pgf@stop
1411 \hobby@restore@tikz@modcmd\tikz@decoration
1412 \global\let\hobby@curveto@delegate=\hobby@curveto@auto
1413 }

```

hobby@init@tikz@modcmd

```

1414 \def\hobby@init@tikz@modcmd#1{%
1415     \expandafter\global\expandafter\let\csname hobby@orig@\string#1\endcsname=#1%
1416     \gdef#1{\hobby@finish#1}%
1417 }

```

by@restore@tikz@modcmd

```

1418 \def\hobby@restore@tikz@modcmd#1{%
1419     \expandafter\global\expandafter\let\expandafter#1\csname hobby@orig@\string#1\endcsname%
1420 }

```

\hobby@midcurveto@auto

```

1421 \def\hobby@midcurveto@auto{%
1422     \expandafter\expandafter\expandafter\gdef\expandafter\expandafter\expandafter\hobby@collected@onpath%
1423     \let\tikz@collected@onpath=\pgfutil@empty
1424     \let\tikz@scan@point@options=\pgfutil@empty
1425     \global\let\hobby@action=\pgfutil@empty
1426     \global\let\hobby@this@opts=\pgfutil@empty
1427     \global\let\hobby@point@options=\pgfutil@empty
1428     \tikz@scan@one@point\hobby@addfromtikz%
1429 }

```

\hobby@finish

```

1430 \def\hobby@finish{%
1431     \ifhobby@externalise
1432         \ifx\hobby@path@name\pgfutil@empty
1433             \hobbygenusepath
1434         \else
1435             \hobbygenuseifnecpath{\hobby@path@name}%
1436         \fi
1437     \else
1438         \hobbygenusepath
1439     \fi
1440     \ifx\hobby@path@name\pgfutil@empty
1441     \else
1442         \hobbysavepath{\hobby@path@name}%
1443     \fi
1444     \global\let\hobby@path@name=\pgfutil@empty
1445     \hobby@restore@tikz@commands
1446 }

```

`quickcurvethrough` The quick curve through is a `to path` which does the “quick” version of Hobby’s algorithm. The syntax is as with the `curve through`: to pass the midpoints as the argument to the style. We need to pass three points to the auxiliary macro. These are passed as `\hobby@qpoints`, `\hobby@qpointa`, and the current point. Then these get cycled round for the next triple. The path gets built up and stored as `\hobby@quick@path`. We also have to remember the angle computed for the next round.

```

1447 \tikzset{
1448   quick curve through/.style={%
1449     to path={%
1450       \pgfextra{%

```

Scan the starting point and store the coordinates in `\hobby@qpointa`

```

1451       \let\hobby@next@qbreak=\relax
1452       \let\hobby@next@qblank=\relax
1453       \tikz@scan@one@point\pgfutil@firstofone(\tikztostart)%
1454       \tikz@make@last@position{\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1455       \edef\hobby@qpoints{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%

```

Blank the path and auxiliary macros.

```

1456       \def\hobby@qpointa{}%
1457       \def\hobby@quick@path{}%
1458       \def\hobby@angle{}%
1459       \let\hobby@quick@curveto=\hobby@quick@makepath

```

Now start parsing the rest of the coordinates.

```

1460       \tikz@scan@one@point\hobby@quickfirst #1 (\tikztotarget)\relax
1461     }

```

Invoke the path

```

1462       \hobby@quick@path
1463     }
1464   },
1465   quick hobby/blank curve/.is choice,
1466   quick hobby/blank curve/true/.code={%
1467     \gdef\hobby@next@qblank{%
1468       \qhobby@blanktrue
1469       \global\let\hobby@next@qblank=\relax
1470     }%
1471   },
1472   quick hobby/blank curve/false/.code={%
1473     \gdef\hobby@next@qblank{%
1474       \qhobby@blankfalse
1475       \global\let\hobby@next@qblank=\relax
1476     }%
1477   },
1478   quick hobby/blank curve/once/.code={%
1479     \gdef\hobby@next@qblank{%
1480       \qhobby@blanktrue
1481       \gdef\hobby@next@qblank{%
1482         \qhobby@blankfalse
1483         \global\let\hobby@next@qblank=\relax
1484       }%
1485     }%
1486   },
1487   quick hobby/blank curve/.default=true,
1488   quick hobby/break curve/.is choice,
1489   quick hobby/break curve/true/.code={%
1490     \gdef\hobby@next@qbreak{%

```

```

1491     \qhobby@breaktrue
1492     \global\let\hobby@next@qbreak=\relax
1493 }%
1494 },
1495 quick hobby/break curve/false/.code={%
1496     \gdef\hobby@next@qbreak{%
1497         \qhobby@breakfalse
1498         \global\let\hobby@next@qbreak=\relax
1499     }%
1500 },
1501 quick hobby/break curve/once/.code={%
1502     \gdef\hobby@next@qbreak{%
1503         \qhobby@breaktrue
1504         \gdef\hobby@next@qbreak{%
1505             \qhobby@breakfalse
1506             \global\let\hobby@next@qbreak=\relax
1507         }%
1508     }%
1509 },
1510 quick hobby/break curve/.default=true,
1511 }
1512 \newif\ifqhobby@break
1513 \newif\ifqhobby@blank
1514
1515 Add plot handlers
1516 \tikzoption{hobby}[]{\let\tikz@plot@handler=\pgfplothandlerhobby}
1517 \tikzoption{quick hobby}[]{\let\tikz@plot@handler=\pgfplothandlerquickhobby}
1518 \tikzoption{closed hobby}[]{\let\tikz@plot@handler=\pgfplothandlerclosedhobby}

```

`\hobby@quickfirst` The first time around we just set the next point.

```

1517 \def\hobby@quickfirst#1{%
1518     #1%
1519     \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf{x}}{\the\pgf{y}}}%
1520     \tikz@make@last@position{\hobby@qpointa}%

```

Now a check to ensure that we have more points.

```

1521     \pgfutil@ifnextchar\relax{%

```

Ooops, no more points. That's not good. Bail-out.

```

1522     \xdef\hobby@quick@path{ -- (\the\pgf{x},\the\pgf{y})}%
1523 }{%

```

Okay, have more points. Phew. Call the next round. If we have dots, swallow them.

```

1524     \pgfutil@ifnextchar.{%
1525         \hobby@qswallowdots}{%
1526         \tikz@scan@one@point\hobby@quick}}

```

`\hobby@qswallowdots` Remove dots from the input stream.

```

1527 \def\hobby@qswallowdots.{%
1528     \pgfutil@ifnextchar.{%
1529         \hobby@qswallowdots}{%
1530         \tikz@scan@one@point\hobby@quick}}

```

`\hobby@quick` This is our wrapper function that handles the loop.

```

1531 \def\hobby@quick#1{%
1532     \hobby@quick@compute{#1}%
1533     \tikz@make@last@position{\hobby@qpointa}%
1534     \pgfutil@ifnextchar\relax{%

```

End of loop

```
1535 \hobby@quick@computeend%  
1536 }{%
```

More to go, scan in the next coordinate and off we go again.

```
1537 \pgfutil@ifnextchar.{%  
1538 \hobby@qswallowdots}{%  
1539 \tikz@scan@one@point\hobby@quick}}}
```

`\hobby@quick@makepath` Path constructor for to path use.

```
1540 \def\hobby@quick@makepath#1#2#3{%  
1541 #1%  
1542 \pgf@xa=\pgf@x\relax  
1543 \pgf@ya=\pgf@y\relax  
1544 #2%  
1545 \pgf@xb=\pgf@x\relax  
1546 \pgf@yb=\pgf@y\relax  
1547 #3%  
1548 \ifqhobby@blank  
1549 \xdef\hobby@quick@path{\hobby@quick@path (\the\pgf@x,\the\pgf@y)}%  
1550 \else  
1551 \xdef\hobby@quick@path{\hobby@quick@path .. controls%  
1552 (\the\pgf@xa,\the\pgf@ya) and (\the\pgf@xb,\the\pgf@yb) .. (\the\pgf@x,\the\pgf@y) }%  
1553 \fi  
1554 \ifqhobby@break  
1555 \xdef\hobby@quick@path{\hobby@quick@path +(0,0)}%  
1556 \fi  
1557 \hobby@next@qbreak  
1558 \hobby@next@qblank  
1559 }
```

`\hobby@qcurveto@auto` Uses the “quick” method for the shortcut syntax.

```
1560 \def\hobby@qcurveto@auto{%  
1561 \global\let\hobby@next@qbreak=\relax  
1562 \global\let\hobby@next@qblank=\relax  
1563 \xdef\hobby@qpoints{\noexpand\pgfqpoint{\the\tikz@lastx}{\the\tikz@lasty}}%  
1564 \gdef\hobby@qpointa{%  
1565 \gdef\hobby@quick@path{%  
1566 \gdef\hobby@angle{%  
1567 \global\let\hobby@quick@curveto=\hobby@quick@makepathauto  
1568 \hobby@qinit@tikz@commands  
1569 \let\tikz@scan@point@options=\pgfutil@empty  
1570 \global\let\hobby@action=\pgfutil@empty  
1571 \global\let\hobby@point@options=\pgfutil@empty  
1572 \tikz@scan@one@point\hobby@qfirst@auto}}
```

`\hobby@qmidcurveto@auto`

```
1573 \def\hobby@qmidcurveto@auto{%  
1574 \let\tikz@scan@point@options=\pgfutil@empty  
1575 \global\let\hobby@action=\pgfutil@empty  
1576 \global\let\hobby@point@options=\pgfutil@empty  
1577 \tikz@scan@one@point\hobby@qaddfromtikz}
```

`\hobby@qfirst@auto`

```
1578 \def\hobby@qfirst@auto#1{%  
1579 #1%
```

```

1580 \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1581 \tikz@make@last@position{\hobby@qpointa}%
1582 \tikz@scan@next@command%
1583 }

```

bby@quick@makepathauto Path constructor for shortcut method to use.

```

1584 \def\hobby@quick@makepathauto#1#2#3{%
1585   #1%
1586   \pgf@xa=\pgf@x\relax
1587   \pgf@ya=\pgf@y\relax
1588   #2%
1589   \pgf@xb=\pgf@x\relax
1590   \pgf@yb=\pgf@y\relax
1591   #3%
1592   \ifqhobby@blank
1593   \edef\hobby@temp{%
1594     \noexpand\pgfpathmoveto{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1595   }%
1596   \hobby@temp
1597   \else
1598   \edef\hobby@temp{%
1599     \noexpand\pgfpathcurveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}%
1600     {\noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}%
1601     {\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1602   }%
1603   \hobby@temp
1604   \fi
1605   \ifqhobby@break
1606   #3%
1607   \edef\hobby@temp{%
1608     \noexpand\pgfpathmoveto{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1609   }%
1610   \hobby@temp
1611   \fi
1612   \hobby@next@qbreak
1613   \hobby@next@qblank
1614 }

```

\hobby@qaddfromtikz This adds our current point to the stack.

```

1615 \def\hobby@qaddfromtikz#1{%
1616   \hobby@quick@compute{#1}%
1617   \tikz@make@last@position{\hobby@qpointa}%
1618   \tikz@scan@next@command%
1619 }

```

bby@qinit@tikz@commands

```

1620 \def\hobby@qinit@tikz@commands{%
1621   \hobby@qinit@tikz@modcmd\tikz@movetoabs
1622   \hobby@qinit@tikz@modcmd\tikz@movetorel
1623   \hobby@qinit@tikz@modcmd\tikz@lineto
1624   \hobby@qinit@tikz@modcmd\tikz@rect
1625   \hobby@qinit@tikz@modcmd\tikz@cchar
1626   \hobby@qinit@tikz@modcmd\tikz@finish
1627   \hobby@qinit@tikz@modcmd\tikz@arcA
1628   \hobby@qinit@tikz@modcmd\tikz@echar
1629   \hobby@qinit@tikz@modcmd\tikz@gchar

```

```

1630 \hobby@qinit@tikz@modcmd\tikz@schar
1631 \hobby@qinit@tikz@modcmd\tikz@vh@lineto
1632 \hobby@qinit@tikz@modcmd\tikz@pchar
1633 \hobby@qinit@tikz@modcmd\tikz@to
1634 \hobby@qinit@tikz@modcmd\pgf@stop
1635 \hobby@qinit@tikz@modcmd\tikz@decoration
1636 \hobby@qinit@tikz@modcmd\tikz@@close
1637 \global\let\hobby@curveto@delegate=\hobby@qmidcurveto@auto
1638 }

```

\hobby@qrestore@tikz@commands

```

1639 \def\hobby@qrestore@tikz@commands{%
1640 \hobby@restore@tikz@modcmd\tikz@movetoabs
1641 \hobby@restore@tikz@modcmd\tikz@movetorel
1642 \hobby@restore@tikz@modcmd\tikz@lineto
1643 \hobby@restore@tikz@modcmd\tikz@rect
1644 \hobby@restore@tikz@modcmd\tikz@ccchar
1645 \hobby@restore@tikz@modcmd\tikz@finish
1646 \hobby@restore@tikz@modcmd\tikz@arcA
1647 \hobby@restore@tikz@modcmd\tikz@e@char
1648 \hobby@restore@tikz@modcmd\tikz@g@char
1649 \hobby@restore@tikz@modcmd\tikz@schar
1650 \hobby@restore@tikz@modcmd\tikz@vh@lineto
1651 \hobby@restore@tikz@modcmd\tikz@pchar
1652 \hobby@restore@tikz@modcmd\tikz@to
1653 \hobby@restore@tikz@modcmd\pgf@stop
1654 \hobby@restore@tikz@modcmd\tikz@decoration
1655 \hobby@restore@tikz@modcmd\tikz@@close
1656 \global\let\hobby@curveto@delegate=\hobby@qcurveto@auto
1657 }

```

obby@qinit@tikz@modcmd

```

1658 \def\hobby@qinit@tikz@modcmd#1{%
1659 \expandafter\global\expandafter\let\csname hobby@orig@\string#1\endcsname=#1%
1660 \gdef#1{\hobby@qfinish#1}%
1661 }

```

\hobby@qfinish

```

1662 \def\hobby@qfinish{%
1663 \hobby@quick@computeend
1664 \hobby@qrestore@tikz@commands
1665 }

```

## 1.4 Arrays

A lot of our data structures are really arrays. These are implemented as L<sup>A</sup>T<sub>E</sub>X3 “property lists”. For ease of use, an array is a property list with numeric entries together with entries “base” and “top” which hold the lowest and highest indices that have been set.

```

1666 \RequirePackage{expl3}
1667 \ExplSyntaxOn

```

Some auxiliary variables.

```

1668 \tl_new:N \l_array_tmp_tl
1669 \tl_new:N \l_array_show_tl
1670 \int_new:N \l_array_base_int

```

```

1671 \int_new:N \l_array_top_int
1672 \int_new:N \l_array_tmp_int

```

The global variable `\g_array_base_int` says what index a blank array should start with when pushed or unshifted.

```

1673 \int_new:N \g_array_base_int
1674 \int_set:Nn \g_array_base_int {0}

```

`\array_adjust_ends:Nn` This ensures that the “base” and “top” are big enough to include the given index.

```

1675 \cs_new:Npn \array_adjust_ends:Nn #1#2 {
1676   \prop_get:NnNTF #1 {base} \l_tmpa_tl
1677   {
1678     \int_compare:nNnTF {\l_tmpa_tl} > {#2}
1679     {
1680       \prop_put:Nnx #1 {base} {\int_eval:n {#2}}
1681     }
1682     {}
1683   }
1684   {
1685     \prop_put:Nnx #1 {base} {\int_eval:n {#2}}
1686   }
1687   \prop_get:NnNTF #1 {top} \l_tmpa_tl
1688   {
1689     \int_compare:nNnTF {\l_tmpa_tl} < {#2}
1690     {
1691       \prop_put:Nnx #1 {top} {\int_eval:n {#2}}
1692     }
1693     {}
1694   }
1695   {
1696     \prop_put:Nnx #1 {top} {\int_eval:n {#2}}
1697   }
1698 }

```

`\array_gadjust_ends:Nn` This ensures that the “base” and “top” are big enough to include the given index. (Global version)

```

1699 \cs_new:Npn \array_gadjust_ends:Nn #1#2 {
1700   \prop_get:NnNTF #1 {base} \l_tmpa_tl
1701   {
1702     \int_compare:nNnTF {\l_tmpa_tl} > {#2}
1703     {
1704       \prop_gput:Nnx #1 {base} {\int_eval:n {#2}}
1705     }
1706     {}
1707   }
1708   {
1709     \prop_gput:Nnx #1 {base} {\int_eval:n {#2}}
1710   }
1711   \prop_get:NnNTF #1 {top} \l_tmpa_tl
1712   {
1713     \int_compare:nNnTF {\l_tmpa_tl} < {#2}
1714     {
1715       \prop_gput:Nnx #1 {top} {\int_eval:n {#2}}
1716     }
1717     {}
1718   }
1719   {

```



```

1720     \prop_gput:Nnx #1 {top} {\int_eval:n {#2}}
1721   }
1722 }

```

`\array_put:Nnn` When adding a value to an array we have to adjust the ends.

```

1723 \cs_new:Npn \array_put:Nnn #1#2#3 {
1724   \exp_args:NNx \prop_put:Nnn #1 {\int_eval:n {#2}} {#3}
1725   \array_adjust_ends:Nn #1{#2}
1726 }
1727 \cs_generate_variant:Nn \array_put:Nnn {Nnx}

```

`\array_gput:Nnn` When adding a value to an array we have to adjust the ends. (Global version)

```

1728 \cs_new:Npn \array_gput:Nnn #1#2#3 {
1729   \exp_args:NNx \prop_gput:Nnn #1 {\int_eval:n {#2}} {#3}
1730   \array_gadjust_ends:Nn #1{#2}
1731 }
1732 \cs_generate_variant:Nn \array_gput:Nnn {Nnx}

```

`\array_get:NnN`

```

1733 \cs_new:Npn \array_get:NnN #1#2#3 {
1734   \exp_args:NNx \prop_get:NnN #1 {\int_eval:n {#2}} #3
1735 }

```

`\array_get:Nn`

```

1736 \cs_new:Npn \array_get:Nn #1#2 {
1737   \exp_args:Nnf \prop_get:Nn #1 { \int_eval:n {#2} }
1738 }

```

`\array_get:NnNTF`

```

1739 \cs_new:Npn \array_get:NnNTF #1#2#3#4#5 {
1740   \exp_args:NNx \prop_get:NnNTF #1 {\int_eval:n {#2}} #3 {#4}{#5}
1741 }

```

`\array_if_empty:NTF`

```

1742 \prg_new_conditional:Npnn \array_if_empty:N #1 { p, T, F, TF }
1743 {
1744   \if_meaning:w #1 \c_empty_prop
1745     \prg_return_true:
1746   \else:
1747     \prg_return_false:
1748   \fi:
1749 }

```

`\array_new:N`

```

1750 \cs_new_eq:NN \array_new:N \prop_new:N

```

`\array_clear:N`

```

1751 \cs_new_eq:NN \array_clear:N \prop_clear:N

```

`\array_gclear:N`

```

1752 \cs_new_eq:NN \array_gclear:N \prop_gclear:N

```

`\array_map_function` When stepping through an array, we want to iterate in order so a simple wrapper to `\prop_map_function` is not enough. This maps through every value from the base to the top so the function should be prepared to deal with a `\q_no_value`.

```

1753 \cs_new:Npn \array_map_function:NN #1#2
1754 {
1755   \array_if_empty:NTF #1 {} {
1756     \prop_get:NnNTF #1 {base} \l_array_tmp_tl {
1757       \int_set:Nn \l_array_base_int {\l_array_tmp_tl}
1758     }{
1759       \int_set:Nn \l_array_base_int {0}
1760     }
1761     \prop_get:NnNTF #1 {top} \l_array_tmp_tl {
1762       \int_set:Nn \l_array_top_int {\l_array_tmp_tl}
1763     }{
1764       \int_set:Nn \l_array_top_int {0}
1765     }
1766     \int_step_inline:nmm {\l_array_base_int} {1} {\l_array_top_int} {
1767       \array_get:NnN #1 {##1} \l_array_tmp_tl
1768       \exp_args:NnV #2 {##1} \l_array_tmp_tl
1769     }
1770   } {}
1771 }
1772 \cs_generate_variant:Nn \array_map_function:NN { Nc }
1773 \cs_generate_variant:Nn \array_map_function:NN { c , cc }

```

`\array_reverse_map_function` This steps through the array in reverse order.

```

1774 \cs_new:Npn \array_reverse_map_function:NN #1#2
1775 {
1776   \array_if_empty:NTF #1 {} {
1777     \prop_get:NnNTF #1 {base} \l_array_tmp_tl {
1778       \int_set:Nn \l_array_base_int {\l_array_tmp_tl}
1779     }{
1780       \int_set:Nn \l_array_base_int {0}
1781     }
1782     \prop_get:NnNTF #1 {top} \l_array_tmp_tl {
1783       \int_set:Nn \l_array_top_int {\l_array_tmp_tl}
1784     }{
1785       \int_set:Nn \l_array_top_int {0}
1786     }
1787     \int_step_inline:nmm {\l_array_top_int} {-1} {\l_array_base_int} {
1788       \array_get:NnN #1 {##1} \l_array_tmp_tl
1789       \exp_args:Nno #2 {##1} \l_array_tmp_tl
1790     }
1791   } {}
1792 }
1793 \cs_generate_variant:Nn \array_reverse_map_function:NN { Nc }
1794 \cs_generate_variant:Nn \array_reverse_map_function:NN { c , cc }

```

`\array_map_inline:Nn` Inline version of the above.

```

1795 \cs_new_protected:Npn \array_map_inline:Nn #1#2
1796 {
1797   \int_gincr:N \g__prg_map_int
1798   \cs_gset:cpn { array_map_inline_ \int_use:N \g__prg_map_int :nn }
1799     ##1##2 {#2}
1800   \exp_args:Nnc \array_map_function:NN #1
1801     { array_map_inline_ \int_use:N \g__prg_map_int :nn }

```

```

1802     \__prg_break_point:Nn \array_map_break: { \int_gdecr:N \g__prg_map_int }
1803   }
1804 \cs_generate_variant:Nn \array_map_inline:Nn { c }

```

`\reverse_map_inline:Nn` Inline version of the above.

```

1805 \cs_new_protected:Npn \array_reverse_map_inline:Nn #1#2
1806 {
1807   \int_gincr:N \g__prg_map_int
1808   \cs_gset:cpn { array_map_inline_ \int_use:N \g__prg_map_int :nn }
1809     ##1##2 {#2}
1810   \exp_args:NNc \array_reverse_map_function:NN #1
1811     { array_map_inline_ \int_use:N \g__prg_map_int :nn }
1812   \__prg_break_point:Nn \array_map_break: { \int_gdecr:N \g__prg_map_int }
1813 }
1814 \cs_generate_variant:Nn \array_reverse_map_inline:Nn { c }

```

`\array_map_break:`

```

1815 \cs_new_nopar:Npn \array_map_break:
1816 { \__prg_map_break:Nn \array_map_break: { } }
1817 \cs_new_nopar:Npn \array_map_break:n
1818 { \__prg_map_break:Nn \array_map_break: }

```

For displaying arrays, we need some messages.

```

1819 \__msg_kernel_new:nnn { kernel } { show-array }
1820 {
1821   The~array~\token_to_str:N #1~
1822   \array_if_empty:NTF #1
1823     { is~empty }
1824     { contains~the~items~(without~outer~braces): }
1825 }

```

`\array_show:N` Mapping through an array isn't expandable so we have to set a token list to its contents first before passing it to the message handler.

```

1826 \cs_new_protected:Npn \array_show:N #1
1827 {
1828   \tl_clear:N \l_array_show_tl
1829   \array_map_function:NN #1 \array_show_aux:nn
1830   \__msg_show_variable:Nno
1831     #1
1832     { array }
1833   { \l_array_show_tl }
1834 }
1835 \cs_generate_variant:Nn \__msg_show_variable:Nnn { Nno }
1836
1837 \cs_new_protected:Npn \array_show_aux:nn #1#2
1838 {
1839   \tl_if_eq:nnTF {#2} {\q_no_value} {}
1840   {
1841     \tl_put_right:No \l_array_show_tl {\__msg_show_item:nn {#1}{#2}}
1842   }
1843 }
1844 \cs_generate_variant:Nn \array_show:N { c }

```

`\array_push:Nn`

```

1845 \cs_new_protected:Npn \array_push:Nn #1#2
1846 {

```

```

1847 \prop_get:NnNTF #1 {top} \l_array_tmp_tl
1848 {
1849   \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1850   \int_incr:N \l_array_tmp_int
1851   \array_put:Nnn #1 {\l_array_tmp_int} {#2}
1852 }
1853 {
1854   \array_put:Nnn #1 {\g_array_base_int} {#2}
1855 }
1856 }
1857 \cs_generate_variant:Nn \array_push:Nn {Nx}

```

\array\_gpush:Nn b

```

1858 \cs_new_protected:Npn \array_gpush:Nn #1#2
1859 {
1860   \prop_get:NnNTF #1 {top} \l_array_tmp_tl
1861   {
1862     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1863     \int_incr:N \l_array_tmp_int
1864     \array_gput:Nnn #1 {\l_array_tmp_int} {#2}
1865   }
1866   {
1867     \array_gput:Nnn #1 {\g_array_base_int} {#2}
1868   }
1869 }
1870 \cs_generate_variant:Nn \array_gpush:Nn {Nx}

```

\array\_unshift:Nn

```

1871 \cs_new_protected:Npn \array_unshift:Nn #1#2
1872 {
1873   \prop_get:NnNTF #1 {base} \l_array_tmp_tl
1874   {
1875     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1876     \int_decr:N \l_array_tmp_int
1877     \array_put:Nnn #1 {\l_array_tmp_int} {#2}
1878   }
1879   {
1880     \array_put:Nnn #1 {\g_array_base_int} {#2}
1881   }
1882 }
1883 \cs_generate_variant:Nn \array_unshift:Nn {Nx}

```

\array\_gunshift:Nn

```

1884 \cs_new_protected:Npn \array_gunshift:Nn #1#2
1885 {
1886   \prop_get:NnNTF #1 {base} \l_array_tmp_tl
1887   {
1888     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1889     \int_decr:N \l_array_tmp_int
1890     \array_gput:Nnn #1 {\l_array_tmp_int} {#2}
1891   }
1892   {
1893     \array_gput:Nnn #1 {\g_array_base_int} {#2}
1894   }
1895 }
1896 \cs_generate_variant:Nn \array_gunshift:Nn {Nx}

```

```

\array_pop:NN
1897 \cs_new_protected:Npn \array_pop:NN #1#2
1898 {
1899   \prop_get:NnN #1 {top} \l_array_tmp_tl
1900   \array_get:NnN #1 {\l_array_tmp_tl} #2
1901   \array_del:Nn #1 {\l_array_tmp_tl}
1902 }

\array_gpop:NN
1903 \cs_new_protected:Npn \array_gpop:NN #1#2
1904 {
1905   \prop_get:NnN #1 {top} \l_array_tmp_tl
1906   \array_get:NnN #1 {\l_array_tmp_tl} #2
1907   \array_gdel:Nn #1 {\l_array_tmp_tl}
1908 }

\array_shift:NN
1909 \cs_new_protected:Npn \array_shift:NN #1#2
1910 {
1911   \prop_get:NnN #1 {base} \l_array_tmp_tl
1912   \array_get:NnN #1 {\l_array_tmp_tl} #2
1913   \array_del:Nn #1 {\l_array_tmp_tl}
1914 }

\array_gshift:NN
1915 \cs_new_protected:Npn \array_gshift:NN #1#2
1916 {
1917   \prop_get:NnN #1 {base} \l_array_tmp_tl
1918   \array_get:NnN #1 {\l_array_tmp_tl} #2
1919   \array_gdel:Nn #1 {\l_array_tmp_tl}
1920 }

\array_top:NN
1921 \cs_new_protected:Npn \array_top:NN #1#2
1922 {
1923   \prop_get:NnN #1 {top} \l_array_tmp_tl
1924   \array_get:NnN #1 {\l_array_tmp_tl} #2
1925 }

\array_base:NN
1926 \cs_new_protected:Npn \array_base:NN #1#2
1927 {
1928   \prop_get:NnN #1 {base} \l_array_tmp_tl
1929   \array_get:NnN #1 {\l_array_tmp_tl} #2
1930 }

\array_top:N
1931 \cs_new:Npn \array_top:N #1
1932 {
1933   \array_get:Nn #1 {\prop_get:Nn #1 {top}}
1934 }

```

\array\_base:N

```
1935 \cs_new:Npn \array_base:N #1
1936 {
1937   \array_get:Nn #1 {\prop_get:Nn #1 {base}}
1938 }
```

\array\_del:Nn

```
1939 \cs_new_protected:Npn \array_del:Nn #1#2
1940 {
1941   \exp_args:Nnx \prop_pop:Nn #1 {\int_eval:n {#2}}
1942   \int_set:Nn \l_array_tmp_int {0}
1943   \array_map_inline:Nn #1 {
1944     \tl_if_eq:NNTF {##2} {\q_no_value} {}
1945     {
1946       \int_incr:N \l_array_tmp_int
1947     }
1948   }
1949   \int_compare:nNnTF {\l_array_tmp_int} = {0}
1950   {
1951     \prop_clear:N #1
1952   }
1953   {
1954     \prop_get:NnN #1 {top} \l_array_tmp_tl
1955     \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
1956       \prop_get:NnN #1 {base} \l_array_tmp_tl
1957       \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1958       \array_map_inline:Nn #1 {
1959         \tl_if_eq:NNTF {##2} {\q_no_value} {}
1960         {
1961           \int_compare:nNnTF {\l_array_tmp_int} < {##1} {
1962             \int_set:Nn \l_array_tmp_int {##1}
1963           }{}
1964         }
1965       }
1966       \prop_put:Nnx #1 {top} {\int_use:N \l_array_tmp_int}
1967     }{}
1968     \prop_get:NnN #1 {base} \l_array_tmp_tl
1969     \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
1970       \prop_get:NnN #1 {top} \l_array_tmp_tl
1971       \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1972       \array_map_inline:Nn #1 {
1973         \tl_if_eq:NNTF {##2} {\q_no_value} {}
1974         {
1975           \int_compare:nNnTF {\l_array_tmp_int} > {##1} {
1976             \int_set:Nn \l_array_tmp_int {##1}
1977           }{}
1978         }
1979       }
1980       \prop_put:Nnx #1 {base} {\int_use:N \l_array_tmp_int}
1981     }{}
1982   }
1983 }
```

\array\_gdel:Nn

```
1984 \cs_new_protected:Npn \array_gdel:Nn #1#2
1985 {
```

```

1986 \exp_args:Nn \prop_gpop:Nn #1 {\int_eval:n {#2}}
1987 \int_set:Nn \l_array_tmp_int {0}
1988 \array_map_inline:Nn #1 {
1989   \tl_if_eq:NNTF {##2} {\q_no_value} {}
1990   {
1991     \int_incr:N \l_array_tmp_int
1992   }
1993 }
1994 \int_compare:nNnTF {\l_array_tmp_int} = {0}
1995 {
1996   \prop_gclear:N #1
1997 }
1998 {
1999   \prop_get:NnN #1 {top} \l_array_tmp_tl
2000   \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2001     \prop_get:NnN #1 {base} \l_array_tmp_tl
2002     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2003     \array_map_inline:Nn #1 {
2004       \tl_if_eq:NNTF {##2} {\q_no_value} {}
2005       {
2006         \int_compare:nNnTF {\l_array_tmp_int} < {##1} {
2007           \int_set:Nn \l_array_tmp_int {##1}
2008         }{}
2009       }
2010     }
2011     \prop_gput:Nnx #1 {top} {\int_use:N \l_array_tmp_int}
2012   }{}
2013   \prop_get:NnN #1 {base} \l_array_tmp_tl
2014   \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2015     \prop_get:NnN #1 {top} \l_array_tmp_tl
2016     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2017     \array_map_inline:Nn #1 {
2018       \tl_if_eq:NNTF {##2} {\q_no_value} {}
2019       {
2020         \int_compare:nNnTF {\l_array_tmp_int} > {##1} {
2021           \int_set:Nn \l_array_tmp_int {##1}
2022         }{}
2023       }
2024     }
2025     \prop_gput:Nnx #1 {base} {\int_use:N \l_array_tmp_int}
2026   }{}
2027 }
2028 }

\array_length:N

2029 \cs_new_protected:Npn \array_length:N #1
2030 {
2031   \int_eval:n {\prop_get:Nn #1 {top} - \prop_get:Nn #1 {base}}
2032 }

2033 \ExplSyntaxOff

```