

The Hobby package: code

Andrew Stacey
loopspace@mathforge.org

1.6 from 2014-08-11

1 Implementation

1.1 Main Code

We use L^AT_EX3 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 {1.6}
22 \tl_set:Nn \g__hobby_date {2014-08-11}
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  $x = \langle \text{number} \rangle$ ,  $y = \langle \text{number} \rangle$ . 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

```

`\l_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`

`\l_hobby_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 2π 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, ψ_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_inf_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_inf_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_inf_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_inf_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_veclen:NVV` Computes the length of the vector specified by the latter two arguments, storing the answer in the first.

```

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

```

`\hobby_ctrl1len: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_ctrl1len: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_ctrl1len: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_compare:nF { \l_hobby_out_angle_fp == \c_inf_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 π , we subtract 2π . (It shouldn't be that we can get bigger than 3π - 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_compare:nT { \l_hobby_in_angle_fp == \c_inf_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_compare:nTF { \l_hobby_in_angle_fp == \c_inf_fp }
241 {
242   \fp_compare:nT { \l_hobby_out_angle_fp == \c_inf_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_compare:nT { \l_hobby_out_angle_fp == \c_inf_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 π , we subtract 2π . (It shouldn't be that we can get bigger than 3π - 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 θ_0 has been specified.

```

383 \fp_compare:nTF { \l_hobby_out_angle_fp == \c_inf_fp }
384 {
385 \array_put:Nnx \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:Nnx \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:Nnx \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 π , we subtract 2π . (It shouldn't be that we can get bigger than 3π - 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 ϕ_n has been given.

```

426 \fp_compare:nTF { \l_hobby_in_angle_fp == \c_inf_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:nmmn {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 θ_i s, we can quickly compute the ϕ_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 ϕ_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 ϕ_n was not given, we compute it from θ_{n-1} .

```

579 \fp_compare:nT { \l_hobby_in_angle_fp == \c_inf_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 ρ_i and σ_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:VnTF {\l_tmpa_tl} {true}
668   {\tl_set:Nn \l_tmpa_tl {2}}
669   {
670     \tl_if_eq:VnTF {\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:VnTF {\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_tmpe_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_tmpe_tl  
762     \hobby_moveto:nmV {} {} \l_tmpe_tl  
763   }  
764   \int_step_inline:nnnn {1} {1} {\l_hobby_npoints_int} {  
765     \array_get:NnN \l_hobby_controla_array {##1} \l_tmpe_tl  
766     \array_get:NnN \l_hobby_controlb_array {##1} \l_tmpe_tl  
767     \array_get:NnN \l_hobby_points_array {##1} \l_tmpe_tl  
768     \array_get:NnN \l_hobby_actions_array {##1} \l_tmpe_tl  
769     \int_compare:nNnTF {\tl_item:Nn \l_tmpe_tl {1}} = {\l_hobby_draw_int} {  
770       \hobby_curveto:VVV \l_tmpe_tl \l_tmpe_tl \l_tmpe_tl  
771     }{  
772       \bool_gset_false:N \l_hobby_closed_bool  
773       \hobby_moveto:VVV \l_tmpe_tl \l_tmpe_tl \l_tmpe_tl  
774     }  
775     \tl_if_eq:xnTF {\tl_item:Nn \l_tmpe_tl {2}} {1} {  
776       \bool_gset_false:N \l_hobby_closed_bool  
777       \hobby_moveto:VVV \l_tmpe_tl \l_tmpe_tl \l_tmpe_tl  
778     }{}}  
779   }  
780   \bool_if:NT \l_hobby_closed_bool {  
781     \array_get:NnN \l_hobby_points_array {0} \l_tmpe_tl  
782     \hobby_close:V \l_tmpe_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_tmpe_tl  
787   \tl_put_right:Nn \l_tmpe_tl {\int_gset:Nn \l_hobby_npoints_int}  
788   \tl_put_right:Nx \l_tmpe_tl {\int_use:N \l_hobby_npoints_int}  
789   \bool_if:NTF \l_hobby_disjoint_bool {  
790     \tl_put_right:Nn \l_tmpe_tl {\bool_gset_true:N}  
791   }{  
792     \tl_put_right:Nn \l_tmpe_tl {\bool_gset_false:N}  
793   }  
794   \tl_put_right:Nn \l_tmpe_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_inf_fp
888     \fp_gset_eq:NN \l_hobby_out_angle_fp \c_inf_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^AT_EX3 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     \pgfmathatantwo@{#1}{#2}%
905   }
906 \else
907   \pgfmathdeclarefunction{hobbyatan2}{2}{%
908     \pgfmathatantwo@{#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^AT_EX3 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 }

```

\pgfpathhobby Low-level interface to the hobby construction. This sets up the commands and starts the iterator.

```

923 \def\pgfpathhobby{%
924   \pgfutil@ifnextchar\bgroup{\pgfpath@hobby}{\pgfpath@hobby{}}
925   \def\pgfpath@hobby#1{%
926     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
927     \hobbysetparams{#1}%
928     \pgfmathsetmacro\hobby@x{\the\pgf@path@lastx/1cm}%
929     \pgfmathsetmacro\hobby@y{\the\pgf@path@lasty/1cm}%
930     \hobbyaddpoint{x = \hobby@x, y = \hobby@y}%
931   }

```

`\pgfpathhobbypt` Adds a point to the construction

```
932 \def\pgfpathhobbypt#1{%
933   #1%
934   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
935   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
936   \pgfutil@ifnextchar\bgroup{\pgfpathhobbyptparams}{\pgfpathhobbyptparams{}}%
937 }
```

`\pgfpathhobbyptparams`

```
938 \def\pgfpathhobbyptparams#1{%
939   \hobbyaddpoint{#1,x = \hobby@x, y = \hobby@y}%
940 }
```

`\pgfpathhobbyend`

```
941 \def\pgfpathhobbyend{%
942   \ifhobby@externalise
943     \ifx\hobby@path@name\pgfutil@empty
944       \hobbygenusepath
945     \else
946       \hobbygenuseifnecpath{\hobby@path@name}%
947     \fi
948   \else
949     \hobbygenusepath
950   \fi
951   \ifx\hobby@path@name\pgfutil@empty
952   \else
953     \hobbysavepath{\hobby@path@name}%
954   \fi
955   \global\let\hobby@path@name=\pgfutil@empty
956 }
```

Plot handlers

`\pgfplothanderhobby` Basic plot handler; uses full algorithm but therefore expensive

```
957 \def\pgfplothandlerhobby{%
958   \def\pgf@plotstreamstart{%
959     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
960     \global\let\pgf@plotstreampoint=\pgf@plot@hobby@firstpt
961     \global\let\pgf@plotstreamspecial=\pgfutil@gobble
962     \gdef\pgf@plotstreamend{%
963       \ifhobby@externalise
964         \ifx\hobby@path@name\pgfutil@empty
965           \hobbygenusepath
966         \else
967           \hobbygenuseifnecpath{\hobby@path@name}%
968         \fi
969       \else
970         \hobbygenusepath
971       \fi
972       \ifx\hobby@path@name\pgfutil@empty
973       \else
974         \hobbysavepath{\hobby@path@name}%
975       \fi
976       \global\let\hobby@path@name=\pgfutil@empty
977     }%
978     \let\tikz@scan@point@options=\pgfutil@empty
```



```

979 }
980 }

```

`pgfplots` `closedhobby` Same as above but produces a closed curve

```

981 \def\pgfplotsclosedhobby{%
982   \def\pgfplotsstart{%
983     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
984     \hobbysetparams{closed=true,disjoint=true}%
985     \global\let\pgfplotspoint=\pgfplots@hobby@firstpt
986     \global\let\pgfplotsspecial=\pgfutil@gobble
987     \gdef\pgfplotsend{%
988       \ifhobby@externalise
989         \ifx\hobby@path@name\pgfutil@empty
990           \hobbygenusepath
991         \else
992           \hobbygenuseifnecpath{\hobby@path@name}%
993         \fi
994       \else
995         \hobbygenusepath
996       \fi
997       \ifx\hobby@path@name\pgfutil@empty
998       \else
999         \hobbysavepath{\hobby@path@name}%
1000       \fi
1001       \global\let\hobby@path@name=\pgfutil@empty
1002     }%
1003   }
1004 }

```

`pgfplots` `firstpt` First point, move or line as appropriate and then start the algorithm.

```

1005 \def\pgfplotsfirstpt#1{%
1006   \pgfplots@first@action{#1}%
1007   \pgfplots@hobby@handler{#1}%
1008   \global\let\pgfplotspoint=\pgfplots@hobby@handler
1009 }

```

`pgfplots` `handler` Add points to the array for the algorithm to work on.

```

1010 \def\pgfplotshandler#1{%
1011   #1%
1012   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1013   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1014   \hobbyaddpoint{x = \hobby@x, y = \hobby@y}%
1015 }

```

`pgfplots` `quickhobby` Uses the “quick” algorithm.

```

1016 \def\pgfplotsquickhobby{%
1017   \def\pgfplotsstart{%
1018     \global\let\hobby@quick@curveto=\pgfpathcurveto
1019     \global\let\pgfplotspoint=\pgfplots@qhobby@firstpt
1020     \global\let\pgfplotsspecial=\pgfutil@gobble
1021     \global\let\pgfplotsend=\pgfplots@qhobby@end
1022   }
1023 }

```

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

```

1024 \def\pgf@plot@qhobby@firstpt#1{%
1025   #1%
1026   \edef\hobby@temp{\noexpand\pgf@plot@first@action{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}}\hobby@
1027   \xdef\hobby@qpoints{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1028   \gdef\hobby@qpointa{}%
1029   \gdef\hobby@angle{}%
1030   \global\let\pgf@plotstreampoint=\pgf@plot@qhobby@secondpt
1031 }

```

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

```

1032 \def\pgf@plot@qhobby@secondpt#1{%
1033   #1%
1034   \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1035   \global\let\pgf@plotstreampoint=\pgf@plot@qhobby@handler
1036 }

```

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

```

1037 \def\pgf@plot@qhobby@handler#1{%
1038   #1
1039   \edef\hobby@temp{\noexpand\hobby@quick@compute{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}}\hobby@t
1040   \global\let\hobby@qpointa=\hobby@qpointa
1041   \global\let\hobby@qpoints=\hobby@qpoints
1042   \global\let\hobby@angle=\hobby@angle

```

Also need to save some data for the last point

```

1043   \global\let\hobby@thetaone=\hobby@thetaone
1044   \global\let\hobby@phitwo=\hobby@phitwo
1045   \global\let\hobby@done=\hobby@done
1046   \global\let\hobby@omegaone=\hobby@omegaone
1047 }

```

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

```

1048 \def\pgf@plot@qhobby@end{%
1049   \hobby@quick@computeend
1050 }

```

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

```

1051 \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.

```

1052 \def\hobby@quick@compute#1{%

```

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

```

1053   #1%
1054   \pgf@xb=\pgf@x
1055   \pgf@yb=\pgf@y

```

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

```

1056   \hobby@qpointa
1057   \pgf@xa=\pgf@x
1058   \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.

```

1059 \advance\pgf@xb by -\pgf@xa
1060 \advance\pgf@yb by -\pgf@ya
1061 \pgfmathsetmacro\hobby@done{sqrt((\pgf@xb/\hobby@sf)^2 + (\pgf@yb/\hobby@sf)^2)}%
1062 \pgfmathsetmacro\hobby@omegaone{rad(hobbyatan2(\pgf@yb, \pgf@xb))}%

```

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

```

1063 \hobby@qpoints
1064 \advance\pgf@xa by -\pgf@x
1065 \advance\pgf@ya by -\pgf@y
1066 \pgfmathsetmacro\hobby@dzero{sqrt((\pgf@xa/\hobby@sf)^2 + (\pgf@ya/\hobby@sf)^2)}%
1067 \pgfmathsetmacro\hobby@omegazerow{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.

```

1068 \pgfmathsetmacro\hobby@psi{\hobby@omegaone - \hobby@omegazerow}%
1069 \pgfmathsetmacro\hobby@psi{\hobby@psi > pi ? \hobby@psi - 2*pi : \hobby@psi}%
1070 \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.

```

1071 \ifx\hobby@angle\pgfutil@empty

```

First.

```

1072 \pgfmathsetmacro\hobby@thetaone{-\hobby@psi * \hobby@done%
1073 /(\hobby@done + \hobby@dzero)}%
1074 \pgfmathsetmacro\hobby@thetazero{-\hobby@psi - \hobby@thetaone}%
1075 \let\hobby@phione=\hobby@thetazero
1076 \let\hobby@phitwo=\hobby@thetaone
1077 \else

```

Second or later.

```

1078 \let\hobby@thetazero=\hobby@angle
1079 \pgfmathsetmacro\hobby@thetaone{%
1080 -(2 * \hobby@psi + \hobby@thetazero) * \hobby@done%
1081 / (2 * \hobby@done + \hobby@dzero)}%
1082 \pgfmathsetmacro\hobby@phione{-\hobby@psi - \hobby@thetaone}%
1083 \let\hobby@phitwo=\hobby@thetaone
1084 \fi

```

Save the outgoing angle.

```

1085 \let\hobby@angle=\hobby@thetaone

```

Compute the control points from the angles.

```

1086 \hobby@quick@ctrlpts{\hobby@thetazero}{\hobby@phione}{\hobby@qpoints}{\hobby@qpointa}{\hobby@dzero}{\hobby@done}

```

Now call the call-back function

```

1087 \edef\hobby@temp{\noexpand\hobby@quick@curveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}{\noexpand\hobby@temp
1088 \hobby@temp

```

Cycle the points round for the next iteration.

```

1089 \global\let\hobby@qpoints=\hobby@qpointa
1090 #1
1091 \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%

```

Save needed values in global macros

```
1092 \global\let\hobby@angle=\hobby@angle
1093 \global\let\hobby@phitwo=\hobby@phitwo
1094 \global\let\hobby@thetaone=\hobby@thetaone
1095 \global\let\hobby@done=\hobby@done
1096 \global\let\hobby@omegaone=\hobby@omegaone
1097 }
```

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

```
1098 \def\hobby@quick@computeend{%
```

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

```
1099 \hobby@quick@ctrlpts{\hobby@thetaone}{\hobby@phitwo}{\hobby@qpoints}{\hobby@qpointa}{\hobby@done}{\hobby@omegaone}
```

Now call the call-back function

```
1100 \edef\hobby@temp{\noexpand\hobby@quick@curveto{\noexpand\pgf@point{\the\pgf@xa}{\the\pgf@ya}}{\noexpand\hobby@temp}}
1101 \hobby@temp
1102 }%
```

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

```
1103 \def\hobby@quick@ctrlpts#1#2#3#4#5#6{%
1104 \pgfmathsetmacro\hobby@alpha{%
1105 sqrt(2) * (sin(#1 r) - 1/16 * sin(#2 r))%
1106 * (sin(#2 r) - 1/16 * sin(#1 r))%
1107 * (cos(#1 r) - cos(#2 r))}%
1108 \pgfmathsetmacro\hobby@rho{%
1109 (2 + \hobby@alpha)/(1 + (1 - (3 - sqrt(5))/2))%
1110 * cos(#1 r) + (3 - sqrt(5))/2 * cos(#2 r))}%
1111 \pgfmathsetmacro\hobby@sigma{%
1112 (2 - \hobby@alpha)/(1 + (1 - (3 - sqrt(5))/2))%
1113 * cos(#2 r) + (3 - sqrt(5))/2 * cos(#1 r))}%
1114 #3%
1115 \pgf@xa=\pgf@x
1116 \pgf@ya=\pgf@y
1117 \pgfmathsetlength\pgf@xa{%
1118 \pgf@xa + #5 * \hobby@rho%
1119 * cos((#1 + #6) r)/3*\hobby@sf}%
1120 \pgfmathsetlength\pgf@ya{%
1121 \pgf@ya + #5 * \hobby@rho%
1122 * sin((#1 + #6) r)/3*\hobby@sf}%
1123 #4%
1124 \pgf@xb=\pgf@x
1125 \pgf@yb=\pgf@y
1126 \pgfmathsetlength\pgf@xb{%
1127 \pgf@xb - #5 * \hobby@sigma%
1128 * cos((-#2 + #6) r)/3*\hobby@sf}%
1129 \pgfmathsetlength\pgf@yb{%
1130 \pgf@yb - #5 * \hobby@sigma%
1131 * sin((-#2 + #6) r)/3*\hobby@sf}%
1132 #4%
1133 }
```

1.3 TikZ Library

```
1134 \usepgflibrary{hobby}
1135 \let\hobby@this@opts=\pgfutil@empty
1136 \let\hobby@next@opts=\pgfutil@empty
1137 \let\hobby@action=\pgfutil@empty
1138 \let\hobby@path@name=\pgfutil@empty
1139 \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.

```
1140 \def\hobby@point@options{%
1141 \tikzset{
1142   curve through/.style={
1143     to path={
1144       \pgfextra{
1145         \expandafter\curvethrough\expandafter[\hobby@next@opts]{%
1146           (\tikztostart) .. #1 .. (\tikztotarget)%
1147         }
1148       }
1149     }
1150   },
1151   tension in/.code = {%
1152     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1153     {\hobby@point@options,tension in=#1}%
1154   },
1155   tension out/.code = {%
1156     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1157     {\hobby@point@options,tension out=#1}%
1158   },
1159   tension/.code = {%
1160     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1161     {\hobby@point@options,tension=#1}%
1162   },
1163   excess angle/.code = {%
1164     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1165     {\hobby@point@options,excess angle=#1}%
1166   },
1167   break/.code = {%
1168     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1169     {\hobby@point@options,break=#1}%
1170   },
1171   blank/.code = {%
1172     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1173     {\hobby@point@options,blank=#1}%
1174   },
1175   designated Hobby path/.initial={next},
1176   clear next Hobby path options/.code={%
1177     \gdef\hobby@next@opts{%
1178     }
1179   },
1179   clear this Hobby path options/.code={%
1180     \gdef\hobby@this@opts{%
1181     }
1182   },
1182   clear Hobby path options/.style={%
1183     clear \pgfkeysvalueof{/tikz/designated Hobby path} Hobby path options
1184   },
1185   add option to this Hobby path/.code={%
```

```

1186   \expandafter\gdef\expandafter\hobby@this@opts\expandafter{\hobby@this@opts#1,}%
1187 },
1188 add option to next Hobby path/.code={%
1189   \expandafter\gdef\expandafter\hobby@next@opts\expandafter{\hobby@next@opts#1,}%
1190 },
1191 add option to Hobby path/.style={%
1192   add option to \pgfkeysvalueof{/tikz/designated Hobby path} Hobby path={#1}%
1193 },
1194 closed/.style = {%
1195   add option to Hobby path={closed=#1,disjoint=#1}%
1196 },
1197 invert blank/.style = {%
1198   add option to Hobby path={invert blank=#1}%
1199 },
1200 closed/.default = true,
1201 blank/.default = true,
1202 break/.default = true,
1203 invert blank/.default = true,
1204 in angle/.code = {%
1205   \pgfmathparse{#1*pi/180}%
1206   \edef\@temp{in angle=\pgfmathresult,}%
1207   \pgfkeysalso{add option to Hobby path/.expand once=\@temp}%
1208 },
1209 out angle/.code = {%
1210   \pgfmathparse{#1*pi/180}%
1211   \edef\@temp{out angle=\pgfmathresult,}%
1212   \pgfkeysalso{add option to Hobby path/.expand once=\@temp}%
1213 },
1214 in curl/.style = {%
1215   add option to Hobby path={in curl=#1}%
1216 },
1217 out curl/.code = {%
1218   add option to Hobby path={out curl=#1}%
1219 },
1220 use Hobby shortcut/.code={%
1221   \let\tikz@curveto@auto=\hobby@curveto@override
1222   \global\let\hobby@curveto@delegate=\hobby@curveto@auto
1223 },
1224 use quick Hobby shortcut/.code={%
1225   \let\tikz@curveto@auto=\hobby@curveto@override
1226   \global\let\hobby@curveto@delegate=\hobby@qcurveto@auto
1227 },
1228 use previous Hobby path/.code={%
1229   \pgfextra{\hobbyusepath{#1}}
1230 },
1231 use previous Hobby path/.default={},%
1232 save Hobby path/.code={%
1233   \xdef\hobby@path@name{#1}%
1234 },
1235 restore Hobby path/.code={%
1236   \pgfextra{%
1237     \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1238     \global\let\hobby@collected@onpath\pgfutil@empty
1239     \hobbyrestorepath{#1}}
1240 },
1241 restore and use Hobby path/.code 2 args={%

```

```

1242 \pgfextra{%
1243   \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1244   \global\let\hobby@collected@onpath\pgfutil@empty
1245   \hobbyrestorepath{#1}%
1246   \hobbyusepath{#2}%
1247 }
1248 },
1249 show Hobby path/.code={%
1250   \pgfextra{\hobbyshowpath{#1}}
1251 },
1252 Hobby action/.code={%
1253   \expandafter\gdef\expandafter\hobby@action\expandafter{\hobby@action#1}%
1254 },
1255 Hobby finish/.style={%
1256   Hobby action=\hobby@finish%
1257 },
1258 Hobby externalise/.is if=hobby@externalise,
1259 Hobby externalize/.is if=hobby@externalise
1260 }

```

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

```

1261 \def\hobby@tikz@curveto#1#2#3{%
1262   \pgfutil@ifundefined{tikz@timer@start}{%
1263     \expandafter\hobby@topgf\expandafter{\hobby@initial@pt}%
1264     \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1265   }{%
1266     \hobby@topgf{#1}%
1267     \edef\tikz@timer@cont@one{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1268     \hobby@topgf{#2}%
1269     \edef\tikz@timer@cont@two{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1270     \hobby@topgf{#3}%
1271     \let\tikz@timer=\tikz@timer@curve
1272     \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1273     \ifx\hobby@collected@onpath\pgfutil@empty
1274     \else
1275     \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1276     \fi
1277     \pgfpathcurveto{\hobby@topgf{#1}}{\hobby@topgf{#2}}{\hobby@topgf{#3}}%
1278     \hobby@topgf{#3}%
1279     \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1280   }

```

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

```

1281 \def\hobby@tikz@moveto#1#2#3{%
1282   \pgfutil@ifundefined{tikz@timer@start}{%
1283     \expandafter\hobby@topgf\expandafter{\hobby@initial@pt}%
1284     \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1285   }{%
1286     \hobby@topgf{#3}%
1287     \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1288     \def\pgf@temp{#1}%
1289     \ifx\pgf@temp\pgfutil@empty
1290     \let\tikz@timer=\tikz@timer@line
1291     \else
1292     \hobby@topgf{#1}%
1293     \edef\tikz@timer@cont@one{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1294     \hobby@topgf{#2}%

```

```

1295     \edef\tikz@timer@cont@two{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1296     \let\tikz@timer=\tikz@timer@curve
1297     \fi
1298     \ifx\hobby@collected@onpath\pgfutil@empty
1299     \else
1300     \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1301     \fi
1302     \pgfpathmoveto{\hobby@topgf{#3}}%
1303     \hobby@topgf{#3}%
1304     \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1305 }

```

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

```

1306 \def\hobby@tikz@close#1{%
1307   \hobby@topgf{#1}%
1308   \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1309   \let\tikz@timer=\tikz@timer@line
1310   \ifx\hobby@collected@onpath\pgfutil@empty
1311   \else
1312   \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1313   \fi
1314   \pgfpathclose
1315 }

```

`\hobby@nodes@onpath`

```

1316 \def\hobby@nodes@onpath#1#2\relax{%
1317   \gdef\hobby@collected@onpath{#2}%
1318   \def\pgf@temp{#1}%
1319   \ifx\pgf@temp\pgfutil@empty
1320   \else
1321   \def\@gtempa{\relax}
1322   \ifx\pgf@temp\@gtempa
1323   \else
1324   \tikz@node@is@a@labeltrue
1325   \tikz@scan@next@command#1\pgf@stop
1326   \tikz@node@is@a@labelfalse
1327   \fi
1328   \fi
1329 }

```

`\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.

```

1330 \newcommand\curvethrough[2] []{%
1331   \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1332   \global\let\hobby@collected@onpath\pgfutil@empty
1333   \let\hobby@initial@opt\pgfutil@empty
1334   \hobbysetparams{#1}%
1335   \tikzset{designated Hobby path=this}%
1336   \global\let\hobby@this@opts=\pgfutil@empty
1337   \global\let\hobby@next@opts=\pgfutil@empty
1338   \let\tikz@scan@point@options=\pgfutil@empty
1339   \def\hobby@point@options{}%
1340   \tikz@scan@one@point\hobby@processpt #2 \relax%
1341 }

```


`\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.

```

1342 \newcommand\hobby@processpt[1]{%
1343   #1%
1344   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1345   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1346   \ifx\hobby@initial@pt\pgfutil@empty
1347     \xdef\hobby@initial@pt{x = \hobby@x, y = \hobby@y}%
1348   \fi
1349   \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1350     x = \hobby@x, y = \hobby@y}%
1351   \def\hobby@point@options{}%
1352   \let\tikz@scan@point@options=\pgfutil@empty
1353   \pgfutil@ifnextchar\relax{%
1354     \expandafter\hobbysetparams\expandafter{\hobby@this@opts}%
1355   \ifhobby@externalise
1356     \ifx\hobby@path@name\pgfutil@empty
1357       \hobbygenusepath
1358     \else
1359       \hobbygenuseifnecpath{\hobby@path@name}%
1360     \fi
1361   \else
1362     \hobbygenusepath
1363   \fi
1364   \ifx\hobby@path@name\pgfutil@empty
1365   \else
1366     \hobbysavepath{\hobby@path@name}%
1367   \fi
1368   \global\let\hobby@path@name=\pgfutil@empty
1369 }{%
1370   \pgfutil@ifnextchar.{%
1371     \hobby@swallowdots}{%
1372     \tikz@scan@one@point\hobby@processpt}}}
```

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

```

1373 \def\hobby@swallowdots.{%
1374   \pgfutil@ifnextchar.{%
1375     \hobby@swallowdots}{%
1376     \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`

```

1377 \def\hobby@curveto@override{%
1378   \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.

```

1379 \def\hobby@curveto@auto{%
1380   \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1381   \expandafter\gdef\expandafter\hobby@collected@onpath\expandafter{\expandafter{\tikz@collected@onpath
1382   \let\tikz@collected@onpath=\pgfutil@empty
```

```

1383 \pgfmathsetmacro\hobby@x{\the\tikz@lastx/1cm}%
1384 \pgfmathsetmacro\hobby@y{\the\tikz@lasty/1cm}%
1385 \xdef\hobby@initial@pt{x = \hobby@x, y = \hobby@y}%
1386 \expandafter\hobbysetparams\expandafter{\hobby@next@opts}%
1387 \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1388     x = \hobby@x, y = \hobby@y}%
1389 \hobby@init@tikz@commands
1390 \tikzset{designated Hobby path=this}%
1391 \let\tikz@scan@point@options=\pgfutil@empty
1392 \global\let\hobby@action=\pgfutil@empty
1393 \global\let\hobby@this@opts=\pgfutil@empty
1394 \global\let\hobby@next@opts=\pgfutil@empty
1395 \global\let\hobby@point@options=\pgfutil@empty
1396 \tikz@scan@one@point\hobby@addfromtikz%
1397 }

```

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

```

1398 \def\hobby@addfromtikz#1{%
1399     #1%
1400     \tikz@make@last@position{#1}%
1401     \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1402     \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1403     \expandafter\hobbysetparams\expandafter{\hobby@this@opts}%
1404     \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1405         x = \hobby@x, y = \hobby@y}%
1406     \hobby@action
1407     \global\let\hobby@this@opts=\pgfutil@empty
1408     \global\let\hobby@action=\pgfutil@empty
1409     \global\let\hobby@point@options=\pgfutil@empty
1410     \tikz@scan@next@command%
1411 }

```

`\hobby@init@tikz@commands`

```

1412 \def\hobby@init@tikz@commands{%
1413     \hobby@init@tikz@modcmd\tikz@movetoabs
1414     \hobby@init@tikz@modcmd\tikz@movetorel
1415     \hobby@init@tikz@modcmd\tikz@lineto
1416     \hobby@init@tikz@modcmd\tikz@rect
1417     \hobby@init@tikz@modcmd\tikz@cchar
1418     \hobby@init@tikz@modcmd\tikz@finish
1419     \hobby@init@tikz@modcmd\tikz@arcA
1420     \hobby@init@tikz@modcmd\tikz@e@char
1421     \hobby@init@tikz@modcmd\tikz@g@char
1422     \hobby@init@tikz@modcmd\tikz@schar
1423     \hobby@init@tikz@modcmd\tikz@vh@lineto
1424     \hobby@init@tikz@modcmd\tikz@pchar
1425     \hobby@init@tikz@modcmd\tikz@to
1426     \hobby@init@tikz@modcmd\pgf@stop
1427     \hobby@init@tikz@modcmd\tikz@decoration
1428     \global\let\hobby@curveto@delegate=\hobby@midcurveto@auto
1429 }

```

`\restore@tikz@commands`

```

1430 \def\hobby@restore@tikz@commands{%
1431     \hobby@restore@tikz@modcmd\tikz@movetoabs
1432     \hobby@restore@tikz@modcmd\tikz@movetorel

```

```

1433 \hobby@restore@tikz@modcmd\tikz@lineto
1434 \hobby@restore@tikz@modcmd\tikz@rect
1435 \hobby@restore@tikz@modcmd\tikz@cchar
1436 \hobby@restore@tikz@modcmd\tikz@finish
1437 \hobby@restore@tikz@modcmd\tikz@arcA
1438 \hobby@restore@tikz@modcmd\tikz@e@char
1439 \hobby@restore@tikz@modcmd\tikz@g@char
1440 \hobby@restore@tikz@modcmd\tikz@schar
1441 \hobby@restore@tikz@modcmd\tikz@vh@lineto
1442 \hobby@restore@tikz@modcmd\tikz@pchar
1443 \hobby@restore@tikz@modcmd\tikz@to
1444 \hobby@restore@tikz@modcmd\pgf@stop
1445 \hobby@restore@tikz@modcmd\tikz@decoration
1446 \global\let\hobby@curveto@delegate=\hobby@curveto@auto
1447 }

```

hobby@init@tikz@modcmd

```

1448 \def\hobby@init@tikz@modcmd#1{%
1449     \expandafter\global\expandafter\let\csname hobby@orig@string#1\endcsname=#1%
1450     \gdef#1{\hobby@finish#1}%
1451 }

```

hobby@restore@tikz@modcmd

```

1452 \def\hobby@restore@tikz@modcmd#1{%
1453     \expandafter\global\expandafter\let\expandafter#1\csname hobby@orig@string#1\endcsname%
1454 }

```

\hobby@midcurveto@auto

```

1455 \def\hobby@midcurveto@auto{%
1456     \expandafter\expandafter\expandafter\gdef\expandafter\expandafter\expandafter\hobby@collected@onpath%
1457     \let\tikz@collected@onpath=\pgfutil@empty
1458     \let\tikz@scan@point@options=\pgfutil@empty
1459     \global\let\hobby@action=\pgfutil@empty
1460     \global\let\hobby@this@opts=\pgfutil@empty
1461     \global\let\hobby@point@options=\pgfutil@empty
1462     \tikz@scan@one@point\hobby@addfromtikz%
1463 }

```

\hobby@finish

```

1464 \def\hobby@finish{%
1465     \ifhobby@externalise
1466         \ifx\hobby@path@name\pgfutil@empty
1467             \hobbygenusepath
1468         \else
1469             \hobbygenuseifnecpath{\hobby@path@name}%
1470         \fi
1471     \else
1472         \hobbygenusepath
1473     \fi
1474     \ifx\hobby@path@name\pgfutil@empty
1475     \else
1476         \hobbysavepath{\hobby@path@name}%
1477     \fi
1478     \global\let\hobby@path@name=\pgfutil@empty
1479     \hobby@restore@tikz@commands
1480 }

```

`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.

```

1481 \tikzset{
1482   quick curve through/.style={%
1483     to path={%
1484       \pgfextra{%

```

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

```

1485     \let\hobby@next@qbreak=\relax
1486     \let\hobby@next@qblank=\relax
1487     \tikz@scan@one@point\pgfutil@firstofone(\tikztostart)%
1488     \tikz@make@last@position{\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1489     \edef\hobby@qpoints{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%

```

Blank the path and auxiliary macros.

```

1490     \def\hobby@qpointa{}%
1491     \def\hobby@quick@path{}%
1492     \def\hobby@angle{}%
1493     \let\hobby@quick@curveto=\hobby@quick@makepath

```

Now start parsing the rest of the coordinates.

```

1494     \tikz@scan@one@point\hobby@quickfirst #1 (\tikztotarget)\relax
1495   }

```

Invoke the path

```

1496     \hobby@quick@path
1497   }
1498 },
1499 quick hobby/blank curve/.is choice,
1500 quick hobby/blank curve/true/.code={%
1501   \gdef\hobby@next@qblank{%
1502     \qhobby@blanktrue
1503     \global\let\hobby@next@qblank=\relax
1504   }%
1505 },
1506 quick hobby/blank curve/false/.code={%
1507   \gdef\hobby@next@qblank{%
1508     \qhobby@blankfalse
1509     \global\let\hobby@next@qblank=\relax
1510   }%
1511 },
1512 quick hobby/blank curve/once/.code={%
1513   \gdef\hobby@next@qblank{%
1514     \qhobby@blanktrue
1515     \gdef\hobby@next@qblank{%
1516       \qhobby@blankfalse
1517       \global\let\hobby@next@qblank=\relax
1518     }%
1519   }%
1520 },
1521 quick hobby/blank curve/.default=true,
1522 quick hobby/break curve/.is choice,
1523 quick hobby/break curve/true/.code={%
1524   \gdef\hobby@next@qbreak{%

```

```

1525     \qhobby@breaktrue
1526     \global\let\hobby@next@qbreak=\relax
1527 }%
1528 },
1529 quick hobby/break curve/false/.code={%
1530     \gdef\hobby@next@qbreak{%
1531         \qhobby@breakfalse
1532         \global\let\hobby@next@qbreak=\relax
1533     }%
1534 },
1535 quick hobby/break curve/once/.code={%
1536     \gdef\hobby@next@qbreak{%
1537         \qhobby@breaktrue
1538         \gdef\hobby@next@qbreak{%
1539             \qhobby@breakfalse
1540             \global\let\hobby@next@qbreak=\relax
1541         }%
1542     }%
1543 },
1544 quick hobby/break curve/.default=true,
1545 }
1546 \newif\ifqhobby@break
1547 \newif\ifqhobby@blank

```

Add plot handlers

```

1548 \tikzoption{hobby} [] {\let\tikz@plot@handler=\pgfplothandlerhobby}
1549 \tikzoption{quick hobby} [] {\let\tikz@plot@handler=\pgfplothandlerquickhobby}
1550 \tikzoption{closed hobby} [] {\let\tikz@plot@handler=\pgfplothandlerclosedhobby}

```

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

```

1551 \def\hobby@quickfirst#1{%
1552     #1%
1553     \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1554     \tikz@make@last@position{\hobby@qpointa}%

```

Now a check to ensure that we have more points.

```

1555     \pgfutil@ifnextchar\relax{%

```

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

```

1556         \xdef\hobby@quick@path{ -- (\the\pgf@x,\the\pgf@y)}%
1557     }{%

```

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

```

1558         \pgfutil@ifnextchar.{%
1559             \hobby@qswallowdots}{%
1560             \tikz@scan@one@point\hobby@quick}}

```

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

```

1561 \def\hobby@qswallowdots.{%
1562     \pgfutil@ifnextchar.{%
1563         \hobby@qswallowdots}{%
1564         \tikz@scan@one@point\hobby@quick}}

```

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

```

1565 \def\hobby@quick#1{%
1566     \hobby@quick@compute{#1}%
1567     \tikz@make@last@position{\hobby@qpointa}%
1568     \pgfutil@ifnextchar\relax{%

```

End of loop

```
1569 \hobby@quick@computeend%  
1570 }-%
```

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

```
1571 \pgfutil@ifnextchar.{%  
1572 \hobby@qswallowdots}{%  
1573 \tikz@scan@one@point\hobby@quick}}}
```

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

```
1574 \def\hobby@quick@makepath#1#2#3{%  
1575 #1%  
1576 \pgf@xa=\pgf@x\relax  
1577 \pgf@ya=\pgf@y\relax  
1578 #2%  
1579 \pgf@xb=\pgf@x\relax  
1580 \pgf@yb=\pgf@y\relax  
1581 #3%  
1582 \ifqhobby@blank  
1583 \xdef\hobby@quick@path{\hobby@quick@path (\the\pgf@x,\the\pgf@y)}%  
1584 \else  
1585 \xdef\hobby@quick@path{\hobby@quick@path .. controls%  
1586 (\the\pgf@xa,\the\pgf@ya) and (\the\pgf@xb,\the\pgf@yb) .. (\the\pgf@x,\the\pgf@y) }%  
1587 \fi  
1588 \ifqhobby@break  
1589 \xdef\hobby@quick@path{\hobby@quick@path +(0,0)}%  
1590 \fi  
1591 \hobby@next@qbreak  
1592 \hobby@next@qblank  
1593 }
```

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

```
1594 \def\hobby@qcurveto@auto{%  
1595 \global\let\hobby@next@qbreak=\relax  
1596 \global\let\hobby@next@qblank=\relax  
1597 \xdef\hobby@qpoints{\noexpand\pgfpoint{\the\tikz@lastx}{\the\tikz@lasty}}%  
1598 \gdef\hobby@qpointa{%  
1599 \gdef\hobby@quick@path{}%  
1600 \gdef\hobby@angle{}%  
1601 \global\let\hobby@quick@curveto=\hobby@quick@makepathauto  
1602 \hobby@qinit@tikz@commands  
1603 \let\tikz@scan@point@options=\pgfutil@empty  
1604 \global\let\hobby@action=\pgfutil@empty  
1605 \global\let\hobby@point@options=\pgfutil@empty  
1606 \tikz@scan@one@point\hobby@qfirst@auto}
```

`hobby@qmidcurveto@auto`

```
1607 \def\hobby@qmidcurveto@auto{%  
1608 \let\tikz@scan@point@options=\pgfutil@empty  
1609 \global\let\hobby@action=\pgfutil@empty  
1610 \global\let\hobby@point@options=\pgfutil@empty  
1611 \tikz@scan@one@point\hobby@qaddfromtikz}
```

`\hobby@qfirst@auto`

```
1612 \def\hobby@qfirst@auto#1{%  
1613 #1%
```

```

1614 \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1615 \tikz@make@last@position{\hobby@qpointa}%
1616 \tikz@scan@next@command%
1617 }

```

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

```

1618 \def\hobby@quick@makepathauto#1#2#3{%
1619 #1%
1620 \pgf@xa=\pgf@x\relax
1621 \pgf@ya=\pgf@y\relax
1622 #2%
1623 \pgf@xb=\pgf@x\relax
1624 \pgf@yb=\pgf@y\relax
1625 #3%
1626 \ifqhobby@blank
1627 \edef\hobby@temp{%
1628 \noexpand\pgfpathmoveto{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1629 }%
1630 \hobby@temp
1631 \else
1632 \edef\hobby@temp{%
1633 \noexpand\pgfpathcurveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}%
1634 {\noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}%
1635 {\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1636 }%
1637 \hobby@temp
1638 \fi
1639 \ifqhobby@break
1640 #3%
1641 \edef\hobby@temp{%
1642 \noexpand\pgfpathmoveto{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1643 }%
1644 \hobby@temp
1645 \fi
1646 \hobby@next@qbreak
1647 \hobby@next@qblank
1648 }

```

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

```

1649 \def\hobby@qaddfromtikz#1{%
1650 \hobby@quick@compute{#1}%
1651 \tikz@make@last@position{\hobby@qpointa}%
1652 \tikz@scan@next@command%
1653 }

```

by@qinit@tikz@commands

```

1654 \def\hobby@qinit@tikz@commands{%
1655 \hobby@qinit@tikz@modcmd\tikz@movetoabs
1656 \hobby@qinit@tikz@modcmd\tikz@movetorel
1657 \hobby@qinit@tikz@modcmd\tikz@lineto
1658 \hobby@qinit@tikz@modcmd\tikz@rect
1659 \hobby@qinit@tikz@modcmd\tikz@cchar
1660 \hobby@qinit@tikz@modcmd\tikz@finish
1661 \hobby@qinit@tikz@modcmd\tikz@arcA
1662 \hobby@qinit@tikz@modcmd\tikz@e@char
1663 \hobby@qinit@tikz@modcmd\tikz@g@char

```

```

1664 \hobby@qinit@tikz@modcmd\tikz@schar
1665 \hobby@qinit@tikz@modcmd\tikz@vh@lineto
1666 \hobby@qinit@tikz@modcmd\tikz@pchar
1667 \hobby@qinit@tikz@modcmd\tikz@to
1668 \hobby@qinit@tikz@modcmd\pgf@stop
1669 \hobby@qinit@tikz@modcmd\tikz@decoration
1670 \hobby@qinit@tikz@modcmd\tikz@@close
1671 \global\let\hobby@curveto@delegate=\hobby@qmidcurveto@auto
1672 }

```

\hobby@qrestore@tikz@commands

```

1673 \def\hobby@qrestore@tikz@commands{%
1674 \hobby@qrestore@tikz@modcmd\tikz@movetoabs
1675 \hobby@qrestore@tikz@modcmd\tikz@movetorel
1676 \hobby@qrestore@tikz@modcmd\tikz@lineto
1677 \hobby@qrestore@tikz@modcmd\tikz@rect
1678 \hobby@qrestore@tikz@modcmd\tikz@cchar
1679 \hobby@qrestore@tikz@modcmd\tikz@finish
1680 \hobby@qrestore@tikz@modcmd\tikz@arcA
1681 \hobby@qrestore@tikz@modcmd\tikz@e@char
1682 \hobby@qrestore@tikz@modcmd\tikz@g@char
1683 \hobby@qrestore@tikz@modcmd\tikz@schar
1684 \hobby@qrestore@tikz@modcmd\tikz@vh@lineto
1685 \hobby@qrestore@tikz@modcmd\tikz@pchar
1686 \hobby@qrestore@tikz@modcmd\tikz@to
1687 \hobby@qrestore@tikz@modcmd\pgf@stop
1688 \hobby@qrestore@tikz@modcmd\tikz@decoration
1689 \hobby@qrestore@tikz@modcmd\tikz@@close
1690 \global\let\hobby@curveto@delegate=\hobby@qcurveto@auto
1691 }

```

obby@qinit@tikz@modcmd

```

1692 \def\hobby@qinit@tikz@modcmd#1{%
1693 \expandafter\global\expandafter\let\csname hobby@orig@\string#1\endcsname=#1%
1694 \gdef#1{\hobby@qfinish#1}%
1695 }

```

\hobby@qfinish

```

1696 \def\hobby@qfinish{%
1697 \hobby@quick@computeend
1698 \hobby@qrestore@tikz@commands
1699 }

```

1.4 Arrays

A lot of our data structures are really arrays. These are implemented as L^AT_EX₃ “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.

```

1700 \RequirePackage{expl3}
1701 \ExplSyntaxOn

```

Some auxiliary variables.

```

1702 \tl_new:N \l_array_tmp_tl
1703 \tl_new:N \l_array_show_tl
1704 \int_new:N \l_array_base_int

```



```
1705 \int_new:N \l_array_top_int
```

```
1706 \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.

```
1707 \int_new:N \g_array_base_int
```

```
1708 \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.

```
1709 \cs_new:Npn \array_adjust_ends:Nn #1#2 {
1710   \prop_get:NnNTF #1 {base} \l_tmpa_tl
1711   {
1712     \int_compare:nNnTF {\l_tmpa_tl} > {#2}
1713     {
1714       \prop_put:Nnx #1 {base} {\int_eval:n {#2}}
1715     }
1716     {}
1717   }
1718   {
1719     \prop_put:Nnx #1 {base} {\int_eval:n {#2}}
1720   }
1721   \prop_get:NnNTF #1 {top} \l_tmpa_tl
1722   {
1723     \int_compare:nNnTF {\l_tmpa_tl} < {#2}
1724     {
1725       \prop_put:Nnx #1 {top} {\int_eval:n {#2}}
1726     }
1727     {}
1728   }
1729   {
1730     \prop_put:Nnx #1 {top} {\int_eval:n {#2}}
1731   }
1732 }
```

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

```
1733 \cs_new:Npn \array_gadjust_ends:Nn #1#2 {
1734   \prop_get:NnNTF #1 {base} \l_tmpa_tl
1735   {
1736     \int_compare:nNnTF {\l_tmpa_tl} > {#2}
1737     {
1738       \prop_gput:Nnx #1 {base} {\int_eval:n {#2}}
1739     }
1740     {}
1741   }
1742   {
1743     \prop_gput:Nnx #1 {base} {\int_eval:n {#2}}
1744   }
1745   \prop_get:NnNTF #1 {top} \l_tmpa_tl
1746   {
1747     \int_compare:nNnTF {\l_tmpa_tl} < {#2}
1748     {
1749       \prop_gput:Nnx #1 {top} {\int_eval:n {#2}}
1750     }
1751     {}
1752   }
1753   {
```

```

1754     \prop_gput:Nnx #1 {top} {\int_eval:n {#2}}
1755   }
1756 }

```

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

```

1757 \cs_new:Npn \array_put:Nnn #1#2#3 {
1758   \exp_args:NNx \prop_put:Nnn #1 {\int_eval:n {#2}} {#3}
1759   \array_adjust_ends:Nn #1{#2}
1760 }
1761 \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)

```

1762 \cs_new:Npn \array_gput:Nnn #1#2#3 {
1763   \exp_args:NNx \prop_gput:Nnn #1 {\int_eval:n {#2}} {#3}
1764   \array_gadjust_ends:Nn #1{#2}
1765 }
1766 \cs_generate_variant:Nn \array_gput:Nnn {Nnx}

```

`\array_get:NnN`

```

1767 \cs_new:Npn \array_get:NnN #1#2#3 {
1768   \exp_args:NNx \prop_get:NnN #1 {\int_eval:n {#2}} #3
1769 }

```

`\array_get:Nn`

```

1770 \cs_new:Npn \array_get:Nn #1#2 {
1771   \exp_args:Nnf \prop_get:Nn #1 { \int_eval:n {#2} }
1772 }

```

`\array_get:NnNTF`

```

1773 \cs_new:Npn \array_get:NnNTF #1#2#3#4#5 {
1774   \exp_args:NNx \prop_get:NnNTF #1 {\int_eval:n {#2}} #3 {#4}{#5}
1775 }

```

`\array_if_empty:NTF`

```

1776 \prg_new_conditional:Npnn \array_if_empty:N #1 { p, T, F, TF }
1777 {
1778   \if_meaning:w #1 \c_empty_prop
1779   \prg_return_true:
1780   \else:
1781     \prg_return_false:
1782   \fi:
1783 }

```

`\array_new:N`

```

1784 \cs_new_eq:NN \array_new:N \prop_new:N

```

`\array_clear:N`

```

1785 \cs_new_eq:NN \array_clear:N \prop_clear:N

```

`\array_gclear:N`

```

1786 \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`.

```

1787 \cs_new:Npn \array_map_function:NN #1#2
1788 {
1789   \array_if_empty:NTF #1 {} {
1790     \prop_get:NnNTF #1 {base} \l_array_tmp_tl {
1791       \int_set:Nn \l_array_base_int {\l_array_tmp_tl}
1792     }{
1793       \int_set:Nn \l_array_base_int {0}
1794     }
1795     \prop_get:NnNTF #1 {top} \l_array_tmp_tl {
1796       \int_set:Nn \l_array_top_int {\l_array_tmp_tl}
1797     }{
1798       \int_set:Nn \l_array_top_int {0}
1799     }
1800     \int_step_inline:nnnn {\l_array_base_int} {1} {\l_array_top_int} {
1801       \array_get:NnN #1 {##1} \l_array_tmp_tl
1802       \exp_args:NnV #2 {##1} \l_array_tmp_tl
1803     }
1804   } {}
1805 }
1806 \cs_generate_variant:Nn \array_map_function:NN { Nc }
1807 \cs_generate_variant:Nn \array_map_function:NN { c , cc }

```

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

```

1808 \cs_new:Npn \array_reverse_map_function:NN #1#2
1809 {
1810   \array_if_empty:NTF #1 {} {
1811     \prop_get:NnNTF #1 {base} \l_array_tmp_tl {
1812       \int_set:Nn \l_array_base_int {\l_array_tmp_tl}
1813     }{
1814       \int_set:Nn \l_array_base_int {0}
1815     }
1816     \prop_get:NnNTF #1 {top} \l_array_tmp_tl {
1817       \int_set:Nn \l_array_top_int {\l_array_tmp_tl}
1818     }{
1819       \int_set:Nn \l_array_top_int {0}
1820     }
1821     \int_step_inline:nnnn {\l_array_top_int} {-1} {\l_array_base_int} {
1822       \array_get:NnN #1 {##1} \l_array_tmp_tl
1823       \exp_args:Nno #2 {##1} \l_array_tmp_tl
1824     }
1825   } {}
1826 }
1827 \cs_generate_variant:Nn \array_reverse_map_function:NN { Nc }
1828 \cs_generate_variant:Nn \array_reverse_map_function:NN { c , cc }

```

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

```

1829 \cs_new_protected:Npn \array_map_inline:Nn #1#2
1830 {
1831   \int_gincr:N \g__prg_map_int
1832   \cs_gset:cpn { array_map_inline_ \int_use:N \g__prg_map_int :nn }
1833     ##1##2 {#2}
1834   \exp_args:NNc \array_map_function:NN #1
1835     { array_map_inline_ \int_use:N \g__prg_map_int :nn }

```

```

1836     \__prg_break_point:Nn \array_map_break: { \int_gdecr:N \g__prg_map_int }
1837   }
1838 \cs_generate_variant:Nn \array_map_inline:Nn { c }

```

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

```

1839 \cs_new_protected:Npn \array_reverse_map_inline:Nn #1#2
1840   {
1841     \int_gincr:N \g__prg_map_int
1842     \cs_gset:cpn { array_map_inline_ \int_use:N \g__prg_map_int :nn }
1843       ##1##2 {#2}
1844     \exp_args:NNc \array_reverse_map_function:NN #1
1845       { array_map_inline_ \int_use:N \g__prg_map_int :nn }
1846     \__prg_break_point:Nn \array_map_break: { \int_gdecr:N \g__prg_map_int }
1847   }
1848 \cs_generate_variant:Nn \array_reverse_map_inline:Nn { c }

```

`\array_map_break:`

```

1849 \cs_new_nopar:Npn \array_map_break:
1850   { \__prg_map_break:Nn \array_map_break: { } }
1851 \cs_new_nopar:Npn \array_map_break:n
1852   { \__prg_map_break:Nn \array_map_break: }

```

For displaying arrays, we need some messages.

```

1853 \__msg_kernel_new:nnn { kernel } { show-array }
1854   {
1855     The~array~\token_to_str:N #1~
1856     \array_if_empty:NTF #1
1857       { is~empty }
1858       { contains~the~items~(without~outer~braces): }
1859   }

```

`\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.

```

1860 \cs_new_protected:Npn \array_show:N #1
1861   {
1862     \tl_clear:N \l_array_show_tl
1863     \array_map_function:NN #1 \array_show_aux:nn
1864     \__msg_show_variable:Nno
1865       #1
1866       { array }
1867     { \l_array_show_tl }
1868   }
1869 \cs_generate_variant:Nn \__msg_show_variable:Nnn { Nno }
1870
1871 \cs_new_protected:Npn \array_show_aux:nn #1#2
1872   {
1873     \tl_if_eq:nnTF {#2} {\q_no_value} {}
1874     {
1875       \tl_put_right:No \l_array_show_tl {\__msg_show_item:nn {#1}{#2}}
1876     }
1877   }
1878 \cs_generate_variant:Nn \array_show:N { c }

```

`\array_push:Nn`

```

1879 \cs_new_protected:Npn \array_push:Nn #1#2
1880   {

```

```

1881 \prop_get:NnNTF #1 {top} \l_array_tmp_tl
1882 {
1883   \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1884   \int_incr:N \l_array_tmp_int
1885   \array_put:Nnn #1 {\l_array_tmp_int} {#2}
1886 }
1887 {
1888   \array_put:Nnn #1 {\g_array_base_int} {#2}
1889 }
1890 }
1891 \cs_generate_variant:Nn \array_push:Nn {Nx}

```

\array_gpush:Nn b

```

1892 \cs_new_protected:Npn \array_gpush:Nn #1#2
1893 {
1894   \prop_get:NnNTF #1 {top} \l_array_tmp_tl
1895   {
1896     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1897     \int_incr:N \l_array_tmp_int
1898     \array_gput:Nnn #1 {\l_array_tmp_int} {#2}
1899   }
1900   {
1901     \array_gput:Nnn #1 {\g_array_base_int} {#2}
1902   }
1903 }
1904 \cs_generate_variant:Nn \array_gpush:Nn {Nx}

```

\array_unshift:Nn

```

1905 \cs_new_protected:Npn \array_unshift:Nn #1#2
1906 {
1907   \prop_get:NnNTF #1 {base} \l_array_tmp_tl
1908   {
1909     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1910     \int_decr:N \l_array_tmp_int
1911     \array_put:Nnn #1 {\l_array_tmp_int} {#2}
1912   }
1913   {
1914     \array_put:Nnn #1 {\g_array_base_int} {#2}
1915   }
1916 }
1917 \cs_generate_variant:Nn \array_unshift:Nn {Nx}

```

\array_gunshift:Nn

```

1918 \cs_new_protected:Npn \array_gunshift:Nn #1#2
1919 {
1920   \prop_get:NnNTF #1 {base} \l_array_tmp_tl
1921   {
1922     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1923     \int_decr:N \l_array_tmp_int
1924     \array_gput:Nnn #1 {\l_array_tmp_int} {#2}
1925   }
1926   {
1927     \array_gput:Nnn #1 {\g_array_base_int} {#2}
1928   }
1929 }
1930 \cs_generate_variant:Nn \array_gunshift:Nn {Nx}

```

`\array_pop:NN`

```
1931 \cs_new_protected:Npn \array_pop:NN #1#2
1932 {
1933   \prop_get:NnN #1 {top} \l_array_tmp_tl
1934   \array_get:NnN #1 {\l_array_tmp_tl} #2
1935   \array_del:Nn #1 {\l_array_tmp_tl}
1936 }
```

`\array_gpop:NN`

```
1937 \cs_new_protected:Npn \array_gpop:NN #1#2
1938 {
1939   \prop_get:NnN #1 {top} \l_array_tmp_tl
1940   \array_get:NnN #1 {\l_array_tmp_tl} #2
1941   \array_gdel:Nn #1 {\l_array_tmp_tl}
1942 }
```

`\array_shift:NN`

```
1943 \cs_new_protected:Npn \array_shift:NN #1#2
1944 {
1945   \prop_get:NnN #1 {base} \l_array_tmp_tl
1946   \array_get:NnN #1 {\l_array_tmp_tl} #2
1947   \array_del:Nn #1 {\l_array_tmp_tl}
1948 }
```

`\array_gshift:NN`

```
1949 \cs_new_protected:Npn \array_gshift:NN #1#2
1950 {
1951   \prop_get:NnN #1 {base} \l_array_tmp_tl
1952   \array_get:NnN #1 {\l_array_tmp_tl} #2
1953   \array_gdel:Nn #1 {\l_array_tmp_tl}
1954 }
```

`\array_top:NN`

```
1955 \cs_new_protected:Npn \array_top:NN #1#2
1956 {
1957   \prop_get:NnN #1 {top} \l_array_tmp_tl
1958   \array_get:NnN #1 {\l_array_tmp_tl} #2
1959 }
```

`\array_base:NN`

```
1960 \cs_new_protected:Npn \array_base:NN #1#2
1961 {
1962   \prop_get:NnN #1 {base} \l_array_tmp_tl
1963   \array_get:NnN #1 {\l_array_tmp_tl} #2
1964 }
```

`\array_top:N`

```
1965 \cs_new:Npn \array_top:N #1
1966 {
1967   \array_get:Nn #1 {\prop_get:Nn #1 {top}}
1968 }
```

\array_base:N

```
1969 \cs_new:Npn \array_base:N #1
1970 {
1971   \array_get:Nn #1 {\prop_get:Nn #1 {base}}
1972 }
```

\array_del:Nn

```
1973 \cs_new_protected:Npn \array_del:Nn #1#2
1974 {
1975   \exp_args:NnX \prop_pop:Nn #1 {\int_eval:n {#2}}
1976   \int_set:Nn \l_array_tmp_int {0}
1977   \array_map_inline:Nn #1 {
1978     \tl_if_eq:NNTF {##2} {\q_no_value} {}
1979     {
1980       \int_incr:N \l_array_tmp_int
1981     }
1982   }
1983   \int_compare:nNnTF {\l_array_tmp_int} = {0}
1984   {
1985     \prop_clear:N #1
1986   }
1987   {
1988     \prop_get:NnN #1 {top} \l_array_tmp_tl
1989     \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
1990       \prop_get:NnN #1 {base} \l_array_tmp_tl
1991       \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1992       \array_map_inline:Nn #1 {
1993         \tl_if_eq:NNTF {##2} {\q_no_value} {}
1994         {
1995           \int_compare:nNnTF {\l_array_tmp_int} < {##1} {
1996             \int_set:Nn \l_array_tmp_int {##1}
1997           }{}
1998         }
1999       }
2000       \prop_put:Nnx #1 {top} {\int_use:N \l_array_tmp_int}
2001     }{}
2002     \prop_get:NnN #1 {base} \l_array_tmp_tl
2003     \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2004       \prop_get:NnN #1 {top} \l_array_tmp_tl
2005       \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2006       \array_map_inline:Nn #1 {
2007         \tl_if_eq:NNTF {##2} {\q_no_value} {}
2008         {
2009           \int_compare:nNnTF {\l_array_tmp_int} > {##1} {
2010             \int_set:Nn \l_array_tmp_int {##1}
2011           }{}
2012         }
2013       }
2014       \prop_put:Nnx #1 {base} {\int_use:N \l_array_tmp_int}
2015     }{}
2016   }
2017 }
```

\array_gdel:Nn

```
2018 \cs_new_protected:Npn \array_gdel:Nn #1#2
2019 {
```

```

2020 \exp_args:Nn \prop_gpop:Nn #1 {\int_eval:n {#2}}
2021 \int_set:Nn \l_array_tmp_int {0}
2022 \array_map_inline:Nn #1 {
2023   \tl_if_eq:NNTF {##2} {\q_no_value} {}
2024   {
2025     \int_incr:N \l_array_tmp_int
2026   }
2027 }
2028 \int_compare:nNnTF {\l_array_tmp_int} = {0}
2029 {
2030   \prop_gclear:N #1
2031 }
2032 {
2033   \prop_get:NnN #1 {top} \l_array_tmp_tl
2034   \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2035     \prop_get:NnN #1 {base} \l_array_tmp_tl
2036     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2037     \array_map_inline:Nn #1 {
2038       \tl_if_eq:NNTF {##2} {\q_no_value} {}
2039       {
2040         \int_compare:nNnTF {\l_array_tmp_int} < {##1} {
2041           \int_set:Nn \l_array_tmp_int {##1}
2042         }{}
2043       }
2044     }
2045     \prop_gput:Nnx #1 {top} {\int_use:N \l_array_tmp_int}
2046   }{}
2047   \prop_get:NnN #1 {base} \l_array_tmp_tl
2048   \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2049     \prop_get:NnN #1 {top} \l_array_tmp_tl
2050     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2051     \array_map_inline:Nn #1 {
2052       \tl_if_eq:NNTF {##2} {\q_no_value} {}
2053       {
2054         \int_compare:nNnTF {\l_array_tmp_int} > {##1} {
2055           \int_set:Nn \l_array_tmp_int {##1}
2056         }{}
2057       }
2058     }
2059     \prop_gput:Nnx #1 {base} {\int_use:N \l_array_tmp_int}
2060   }{}
2061 }
2062 }

\array_length:N
2063 \cs_new_protected:Npn \array_length:N #1
2064 {
2065   \int_eval:n {\prop_get:Nn #1 {top} - \prop_get:Nn #1 {base}}
2066 }
2067 \ExplSyntaxOff

```