1 Introduction

This is a package for drawing TQFT diagrams using PGF/TikZ. Its inspiration was a question and answer on the website http://tex.stackexchange.com.

2 Implementation

2.1 Old Version: Node Shapes

We can view the cobordisms from the input or output ends, the implementation of the choice is to draw an arc from 0 to 180 or from 0 to -180 so we just need to track minus signs. These macros are for that.

\begin{verbatim}
\def\pgf@tqft@minus{-}
\let\pgf@tqft@upper\@empty
\let\pgf@tqft@lower\pgf@tqft@minus
\end{verbatim}

Some helpful extra functions.
\tqftset is our equivalent of \tikzset.
\begin{verbatim}
\def\tqftset#1{\pgfkeys{/pgf/tqft/.cd,#1}}
\end{verbatim}
\tqft@process This macro applies our flow transformation to the given coordinates, with the result stored in \pgfx and \pgfy.

\begin{verbatim}
\def\tqft@process#1#2{% \\
\edef\tqft@px{#1}
\edef\tqft@py{#2}
\pgfprocess{
\end{verbatim}
\begin{verbatim}
11 \pgftransformreset
12 \let\tikz@transform=\pgfutil@empty
13 \expandafter\tikzset\expandafter{\tqft@transformation}
14 \tikz@transform
15 \pgfpointtransformed{\pgfqpoint{\tqft@px}{\tqft@py}}
16 \}
17 }

 Declare some dimension registers to hold the specifications of the cobordism.
18 \newdimen\tqft@xa
19 \newdimen\tqft@xb
20 \newdimen\tqft@c
21 \newdimen\tqft@ch
22 \newdimen\tqft@h
23 \newdimen\tqft@s
24 \newdimen\tqft@w
25 newif\iftqft@within@node

 Now we set up all the keys that we’ll need in the course of this shape
26 \pgfkeys{

 Add a key to switch between the two versions.
27 /tikz/tqft/use nodes/.is choice,
28 /tikz/tqft/use nodes/true/.code={
29 \tikzset{
30 tqft/.style={%
31 /tikz/shape=tqft cobordism,
32 /pgf/tqft,
33 /tikz/every tqft/.try
34 },
35
36 /pgf/tqft
37 tqft/unknown/.code={%
38 \let\tqft@searchname=\pgfkeyscurrentname%
39 \pgfkeys{%
40 /pgf/tqft/\tqft@searchname={##1}
41 }
42 },%
43 },
44 }

 If not using nodes, set the defaults for the library
45 /tikz/tqft/use nodes/false/.code={%
46 \tikzset{
47 tqft/.style={%
48 pic type=cobordism,
49 tqft/.cd,
50 every tqft/.try,
51 },
52
53 /pgf/tqft
54 tqft/unknown/.code={%
55 \let\tqft@searchname=\pgfkeyscurrentname%
56 \pgfkeys{%
57 }
58%
59 }
60 }

2
\end{verbatim}
This deals with unknown keys, passing them on to TikZ.

Let’s play happy families!

This sets our shape to be the boundary circle

These set our number of boundary components

This is the “horizontal” offset of the first outgoing component from the first incoming one.

This is the “vertical” separation between boundary components.

This is the “horizontal” separation between boundary components.

These are the “horizontal” and “vertical” radii, respectively, of the boundary components (perhaps poorly named!).

These control the separation between the node and its anchors.

This is our flow control. The flow key installs a transformation to be applied to our node shape. The possible transformations are stored in the following keys. They aren’t just rotations so that the numbering is always “top to bottom” or “left to right”.

%}
flow transformation south/.initial={},
flow transformation north/.initial={% xscale=-1,rotate=180%
},
flow transformation east/.initial={% rotate=90,xscale=-1%
},
flow transformation west/.initial={% rotate=270%
},
flow transformation/.initial={%}

These control the direction from which we view the cobordism.

view from/.is choice,
view from/incoming/.code={%
  \let\pgf@tqft@upper\pgf@tqft@minus
  \let\pgf@tqft@lower\@empty
},
view from/outgoing/.code={%
  \let\pgf@tqft@lower\pgf@tqft@minus
  \let\pgf@tqft@upper\@empty
},

The next set of keys are for styling the different pieces of a cobordism.

boundary lower style contents/.initial={},
boundary lower style/.code={%
  \pgfkeys{/pgf/tqft/boundary lower style contents/.style={%
    /tikz/.cd,#1%
  }}
},
boundary style contents/.initial={},
boundary style/.code={%
  \pgfkeys{/pgf/tqft/boundary style contents/.style={%
    /tikz/.cd,#1%
  }}
},
boundary upper style contents/.initial={},
boundary upper style/.code={%
  \pgfkeys{/pgf/tqft/boundary upper style contents/.style={%
    /tikz/.cd,#1%
  }}
},

The next set of keys define some default shapes.

cobordism style contents/.initial={},
cobordism style/.code={%
  \pgfkeys{/pgf/tqft/cobordism style contents/.style={%
    /tikz/.cd,#1%
  }}
},
pair of pants/.style={
/tikz/tqft,
incoming boundary components=1,
outgoing boundary components=2,
offset=-.5
},
/tikz/tqft pair of pants/.style={
/pgf/tqft/pair of pants,
},
reverse pair of pants/.style=
/tikz/tqft,
incoming boundary components=2,
outgoing boundary components=1,
offset=.5
},
/tikz/tqft reverse pair of pants/.style=
/pgf/tqft/reverse pair of pants,
},
cylinder to prior/.style=
/tikz/tqft,
incoming boundary components=1,
outgoing boundary components=1,
offset=-.5
},
/tikz/tqft cylinder to prior/.style=
/pgf/tqft/cylinder to prior,
},
cylinder to next/.style=
/tikz/tqft,
incoming boundary components=1,
outgoing boundary components=1,
offset=.5
},
/tikz/tqft cylinder to next/.style=
/pgf/tqft/cylinder to next,
},
cylinder/.style=
/tikz/tqft,
incoming boundary components=1,
outgoing boundary components=1
},
/tikz/tqft cylinder/.style=
/pgf/tqft/cylinder,
},
cup/.style=
/tikz/tqft,
incoming boundary components=1,
outgoing boundary components=0
},
/tikz/tqft cup/.style=
/pgf/tqft/cup,
},
cap/.style=
/tikz/tqft,
incoming boundary components=0,
outgoing boundary components=1
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
}\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
},
\tikz/tqft cap/.style={
/pgf/tqft/cap,
These are our externally available anchors

\anchor{centre}{\pgfpointorigin}
\anchor{center}{\pgfpointorigin}
\anchor{text}{\tqft@textsize}
\anchor{north}{\pgf@ya=\tqft@height\relax
  \pgf@yb=.5\pgf@ya
  \advance\pgf@yb by \tqft@outerysep\relax
  \tqft@process{0pt}{\the\pgf@yb}
}
\anchor{south}{\pgf@yb=\tqft@height\relax
  \pgf@ya=.5\pgf@yb
  \advance\pgf@ya by \tqft@outerysep\relax
  \pgf@yb=-\pgf@ya
  \tqft@process{0pt}{\the\pgf@yb}
}
\anchor{west}{\tqft@start@incoming
  \pgf@xa=\pgf@x
  \advance\pgf@xa by -\tqft@width
  \pgf@ya=\pgf@y
  \tqft@start@outgoing
  \pgf@xb=\pgf@x
  \pgf@yc=\pgf@y
  \advance\pgf@xc by .5\pgf@xa
  \advance\pgf@yc by .5\pgf@ya
  \advance\pgf@xc by \tqft@outerxsep\relax
  \tqft@process{\the\pgf@xc}{\the\pgf@yc}
}
\anchor{east}{\tqft@start@incoming
  \pgf@xa=\pgf@x
  \pgfmathsetlength{\pgf@xa}{\pgf@xa + (\tqft@incoming - 1) * \tqft@separation}
  \advance\pgf@xa by \tqft@width\relax
  \pgf@ya=\pgf@y
  \tqft@start@outgoing
  \pgf@xb=\pgf@x
  \pgfmathsetlength{\pgf@xb}{\pgf@xb + (\tqft@outgoing - 1) * \tqft@separation}
  \advance\pgf@xb by \tqft@width\relax
  \pgf@yb=\pgf@y
  \pgf@xc=\pgf@xa
  \pgf@yc=\pgf@ya
  \advance\pgf@xc by .5\pgf@xb
  \advance\pgf@yc by .5\pgf@yb
  \advance\pgf@xc by \tqft@outerxsep\relax
  \tqft@process{\the\pgf@xc}{\the\pgf@yc}
To define anchors at the boundary components requires a bit of trickery borrowed from the “regular polygon” shape.
Now we define the background path. This is the upper part of the cobordism.

\begin{tikzpicture}
\backgroundpath{

Apply the internal transformation
\let\tikztransform=\pgfutil@empty
\expandafter\tikzset\expandafter{\tqft@transformation}
\tikztransform

Convert the boundary separation and width to lengths
\pgfmathsetlength{\tqft@s}{\tqft@separation}
\pgfmathsetlength{\tqft@w}{2*\tqft@width}

Compute the starting position of the incoming boundary components so that we get the centre anchor on the centre of the cobordism
\tqft@start@incoming
\tqft@xa=\pgf@x
\advance\tqft@xa by -.5\tqft@w\relax
\tqft@h=\pgf@y
\advance\tqft@xa by \tqft@w\relax
\tqft@c=\tqft@control\relax

Do we have any incoming boundary components at all?
\ifnum\tqft@incoming>0
Yes, so move to the position of the first and draw it
\pgfpathmoveto{\pgfqpoint{\tqft@xa}{\tqft@h}}
\pgfpatharc{\pgf@tqft@upper180}{0}{\tqft@width and \tqft@depth}
\fi

Do we have any more incoming boundary components?
\ifnum\tqft@incoming>1
Yes, so iterate over the remaining incoming boundary components
\foreach \tqft@k in {2,...,\tqft@incoming}{
\advance\tqft@xa by \tqft@k\tqft@s
\advance\tqft@xb by \tqft@k\tqft@s
\advance\tqft@xb by -2\tqft@s
\advance\tqft@xa by -\tqft@s
\pgfpathcurveto{\pgfqpoint{\tqft@xb}{\tqft@c}}{\pgfqpoint{\tqft@xa}{\tqft@c}}{\pgfqpoint{\tqft@xa}{\tqft@h}}
\pgfpatharc{\pgf@tqft@upper180}{0}{\tqft@width and \tqft@depth}
}
\fi

If we don’t have any outgoing boundary components, may as well close up now.
\ifnum\tqft@outgoing=0
\advance\tqft@xb by \tqft@incoming\tqft@s
\advance\tqft@xb by -\tqft@s
\pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\tqft@xb - \tqft@xa) - \tqft@w)/\tqft@s + 1)))}
\pgfpathcurveto{\pgfqpoint{\tqft@xb}{\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@h}}
\fi
\fi

Shift down to the outgoing components, if we have any
\ifnum\tqft@outgoing>0
\advance\tqft@xb by \tqft@incoming\tqft@s
\advance\tqft@xb by -\tqft@s
\pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\tqft@xa + \tqft@outgoing - 1 + \tqft@offset) * \tqft@separation))}}
\fi
\fi
Now draw the lower components

\begin{verbatim}
pgfpatharc{0}{\pgf@tqft@upper180}{\tqft@width and \tqft@depth}
\end{verbatim}

Now iterate over the remaining outgoing boundary components

\begin{verbatim}
ifnum\tqft@outgoing>1
  \foreach \tqft@k in {2,...,\tqft@outgoing} {
    \advance\tqft@xa by -\tqft@k\tqft@s
    \advance\tqft@xb by -\tqft@k\tqft@s
    \advance\tqft@xb by 2\tqft@s
    \advance\tqft@xa by \tqft@s
    \pgfpathcurveto{\pgfqpoint{\tqft@xb}{-\tqft@c}}{\pgfqpoint{\tqft@xa}{-\tqft@c}}{\pgfqpoint{\tqft@xa}{-\tqft@h}}
    \pgfpatharc{0}{\pgf@tqft@upper180}{\tqft@width and \tqft@depth}
  }
fi
\end{verbatim}

Shift back up to the incoming components, if we had any, otherwise arc back to our starting point

\begin{verbatim}
\begin{verbatim}
\else
  \pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\tqft@xb - \tqft@xa) - \tqft@w)/\tqft@s + 1))))
  \pgfpathcurveto{\pgfqpoint{\tqft@xb}{-\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@h}}
\fi
\end{verbatim}
\end{verbatim}

Close the path

\begin{verbatim}
\pgfpathclose
\end{verbatim}

End of background path Now we define the behind background path. This is the lower part of the boundary circles.

\begin{verbatim}
\behindbackgroundpath{
Apply the internal transformation

\begin{verbatim}
let\tikz@transform=\pgfutil@empty
\expandafter\tikzset\expandafter{\tqft@transformation}
\tikz@transform
\end{verbatim}

Convert the boundary separation and width to lengths

\begin{verbatim}
\pgfmathsetlength{\tqft@w}{2*\tqft@width}
\pgfmathsetlength{\tqft@0s}{\tqft@separation}
\end{verbatim}

\end{verbatim}
Compute the starting position of the incoming boundary components so that we get the centre anchor on the centre of the cobordism

\begin{align*}
\text{pgfmathsetlength(\tqft@xa)} &= -(\max(\text{\tqft@incoming} - 1, \text{\tqft@outgoing} - 1 + \text{\tqft@offset})) \\
\text{pgfmathsetlength(\tqft@h)} &= .5 * \text{\tqft@height}
\end{align*}

This section draws the boundary circles

Do we have any incoming boundary components at all?

Yes, so iterate over them

Now iterate over the outgoing boundary components, if we have any

This section draws the lower parts of the boundary circles

Initialise the TikZ path settings and read in the style options for the boundary

Initialise the TikZ path settings and read in the style options for the boundary lower
Do we have any incoming boundary components at all?

Yes, so iterate over them

\foreach \tqft@k in {1,...,\tqft@incoming} {
  \advance\tqft@xa by \tqft@k\tqft@s
  \pgfpathmoveto{\pgfqpoint{\tqft@xa}{\tqft@h}}
  \pgfpatharc{0}{\pgf@tqft@lower180}{\tqft@width and \tqft@depth}
}

Now iterate over the outgoing boundary components, if we have any

\ifnum\tqft@outgoing>0
  \pgfmathsetlength{\tqft@xa}{\tqft@xa + (\tqft@outgoing + \tqft@offset + 1) * \tqft@separation}
  \foreach \tqft@k in {1,...,\tqft@outgoing} {
    \advance\tqft@xa by -\tqft@k\tqft@s
    \pgfpathmoveto{\pgfqpoint{\tqft@xa}{-\tqft@h}}
    \pgfpatharc{0}{\pgf@tqft@lower180}{\tqft@width and \tqft@depth}
  }
\fi

\edef\tikz@temp{\noexpand\pgfusepath{\iftikz@mode@fill fill,\fi% \iftikz@mode@draw draw\fi%}}%
\tikz@temp

End of behind background path.

Now we define the before background path. This is the upper part of the boundary circles and the cobordism edge.

\beforebackgroundpath{

We don’t apply the internal transformation as it is already in place from the \backgroundpath. Convert the boundary separation and width to lengths

\pgfmathsetlength{\tqft@wa}{\tqft@separation}
\pgfmathsetlength{\tqft@w}{2*\tqft@width}

Compute the starting position of the incoming boundary components so that we get the centre anchor on the centre of the cobordism

\pgfmathsetlength{\tqft@xa}{-(max(\tqft@incoming - 1,\tqft@outgoing - 1 + \tqft@offset)\tqft@wa}
\advance\tqft@xb by \tqft@w
\tqft@c=\tqft@control\relax
\pgfmathsetlength{\tqft@h}{.5 * \tqft@height}

This section draws the non-boundary part of the cobordism.

\{ %

Initialise the TikZ path settings and read in the style options for the boundary

\tikz@mode=\pgfutil@empty
\let\tikz@options=\pgfutil@empty
\tqftset{cobordism style contents}
Do we have any incoming boundary components at all?
\begin{macrocode}
@ifnum\tqft@incoming>0

Do we have more than one?
\begin{macrocode}
% \tikz@mode
% \tikz@options
% \begin{macrocode}
\ifnum\tqft@incoming>1

Yes, so iterate over the remaining incoming boundary components
\begin{macrocode}
\foreach \tqft@k in {2,...,\tqft@incoming} {
\begin{macrocode}

If we don’t have any outgoing boundary components, may as well close up now.
\begin{macrocode}
\ifnum\tqft@outgoing=0

Shift down to the outgoing components, if we have any
\begin{macrocode}
\ifnum\tqft@outgoing>0

If we had incoming boundaries, this is a curveto, otherwise it’s a moveto
\begin{macrocode}
\ifnum\tqft@incoming>0

Now draw the lower components
\begin{macrocode}
\begin{macrocode}

Now iterate over the remaining outgoing boundary components
\begin{macrocode}
\foreach \tqft@k in {2,...,\tqft@outgoing} {
\begin{macrocode}

\end{document}
Shift back up to the incoming components, if we had any, otherwise arc back to our starting point

This section draws the upper parts of the boundary circles

Do we have any incoming boundary components at all?

Yes, so iterate over them
Now iterate over the outgoing boundary components, if we have any
\begin{verbatim}
\ifnum\tqft@outgoing>0
\pgfmathsetlength{\tqft@xa}{\tqft@xa + (\tqft@outgoing + \tqft@offset + 1) \times \tqft@separation}
\foreach \tqft@k in {1,...,\tqft@outgoing} {
  \advance\tqft@xa by -\tqft@k\tqft@s
  % \advance\tqft@xa by \tqft@s
  \pgfpathmoveto{\pgfqpoint{\tqft@xa}{-\tqft@h}}
  \pgfpatharc{0}{\pgf@tqft@upper180}{\tqft@width and \tqft@depth}
  \ifx\tikz@fig@name\pgfutil@empty
    \else
      \pgfmathtruncatemacro{\tqft@l}{\tqft@outgoing + 1 - \tqft@k}
      \advance\tqft@xa by -\tqft@width
      \pgftransformshift{\pgfqpoint{\tqft@xa}{-\tqft@h}}
      \tqft@within@nodetrue
      \pgfsyssoftpath@getcurrentpath{\tqft@bdry@path}
      \pgfsyssoftpath@setcurrentpath{\tqft@bdry@node@path}
      \pgfnode{tqft boundary circle}{centre}{}{\tikz@fig@name space outgoing \tqft@l}{}
      \pgfsyssoftpath@getcurrentpath{\tqft@bdry@node@path}
      \pgfsyssoftpath@setcurrentpath{\tqft@bdry@path}
  \fi
}\fi
\edef\tikz@temp{\noexpand\pgfusepath{%
  \iftikz@mode@fill fill,\fi%
  \iftikz@mode@draw draw\fi%}
}}%
\tikz@temp
\end{verbatim}
\end{verbatim}
\end{verbatim}
boundary circle shape
This is the shape of the boundary circles
\begin{verbatim}
\pgfdeclareshape{tqft boundary circle}{
Now we save our dimensions: height, separation, and the radii of the boundary circles
\begin{verbatim}
\.savedim{\tqft@height}{\pgfkeysvalueof{/pgf/tqft/cobordism height}}
\savedim{\tqft@separation}{\pgfkeysvalueof{/pgf/tqft/boundary separation}}
\end{verbatim}
\end{verbatim}
For the externally available anchors, we need to save the declared transformation; we save the actual transformation, not the macro that points to it. If we’re called within the main cobordism shape, the transformation is already applied so we ignore it.

\savedmacro{tqft@transformation}{% 
  iftqft@within@node 
  \let tqft@transformation=\pgfutil@empty 
  \else 
  \pgfkeysgetvalue{/pgf/tqft/flow transformation}{tqft@transformation} 
  \fi 
}

\savedanchor{tqft@centre}{% 
  \pgfpointorigin}

For completeness, we record the size of the text box (not that we expect any text, but you never know)

\savedanchor{tqft@textsize}{% 
  \pgf@y=-.5\ht\pgfnodeparttextbox\% 
  \pgf@x=-.5\wd\pgfnodeparttextbox\% 
}

These are our externally available anchors

\anchor{centre}{tqft@centre}
\anchor{center}{tqft@centre}
\anchor{text}{tqft@textsize}
\anchor{next}{tqft@process{tqft@separation}{0pt}}
\anchor{prior}{tqft@process{-tqft@separation}{0pt}}
\anchor{above}{tqft@process{0pt}{tqft@height}}
\anchor{below}{tqft@process{0pt}{-tqft@height}}

The anchor border is the ellipse, but we need to take into account the possible transformation. (This isn’t right if the origin is shifted.) At the moment, ‘0 degrees’ is interpreted in the transformed coordinate system. Should provide a system whereby that can be interpreted in the main coordinate system.

\anchorborder{
  This next \pgf@process makes the angles absolute. Comment it out to make the angles relative.
  \tqft@process{\the\pgf@x}{\the\pgf@y}
  \edef\tqft@marshal{% 
  \noexpand\pgfpointborderellipse 
  \noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}
  \noexpand\pgfqpoint{\tqft@width}{\tqft@depth}}

17
Now we define the background path. This is the upper part of the cobordism.
\begin{tikzpicture}
\begin{scope}
\pgfpathellipse\pgfboundingbox\path{\pgfpoint{0pt}{0pt}}\pgfpoint{\tqft@width}{0pt}\pgfpoint{0pt}{\tqft@depth}}
\end{scope}
\end{tikzpicture}
We draw the upper and lower arcs again with the appropriate styles
\begin{tikzpicture}
\beforebackgroundpath{
\ifthenelse{\tqft@within@node}{\else}{
\let\tikz@mode@fill=false\n\let\tikz@mode@draw=false\n\let\tikz@options=\pgfutil@empty\n\begin{scope}
\tqftset{boundary lower style contents}\tikz@mode\tikz@options\pgfpathmoveto{\pgfqpoint{\tqft@width}{0pt}}\pgfpatharc{0}{\pgf@tqft@lower180}{\tqft@width and \tqft@depth}\edef\tikz@temp{\noexpand\pgfusepath{\iftikz@mode@fill fill,\fi\iftikz@mode@draw draw\fi}}\tikz@temp\end{scope}
\begin{scope}
\tqftset{boundary upper style contents}\tikz@mode\tikz@options\pgfpathmoveto{\pgfqpoint{\tqft@width}{0pt}}\pgfpatharc{0}{\pgf@tqft@upper180}{\tqft@width and \tqft@depth}\edef\tikz@temp{\noexpand\pgfusepath{\iftikz@mode@fill fill,\fi\iftikz@mode@draw draw\fi}}\tikz@temp\end{scope}}\fi
\end{scope}\end{tikzpicture}
2.2 New Version: Picture Shapes

Issue a warning if the pic syntax is not available.
\verbatim
\ifcsname pgfk@/handlers/.pic/.@cmd\endcsname
\else\pgfwarning{This library only works with TikZ 3.0 or later; for earlier versions of TikZ use the TQFT package}\fi

For the boundaries, we need elliptical node shapes.
\verbatim\usetikzlibrary{shapes.geometric}\verbatim
We can view the cobordisms from the input or output ends, the implementation of the choice is to draw an arc from 0 to 180 or from 0 to -180 so we just need to track minus signs. These macros are for that.
\verbatim\def\pgf@tqft@minus{-}\verbatim\verbatim\let\pgf@tqft@upper\@empty\verbatim\verbatim\let\pgf@tqft@lower\pgf@tqft@minus\verbatim
Should we twist the cobordism?
\verbatim\newif\iftqft@twisted\verbatim
Split an anchoring coordinate. The y–value is simply multiplied by the cobordism height (but pointing downwards, so that 1 is level with the outgoing boundary). The x–value is multiplied by the boundary separation, but is shifted so that at the incoming boundary level, or above, then it is in line with the incoming boundaries and similarly at the outgoing boundary level, or below, it is in line with the outgoing boundaries.
\verbatim\def\tqft@split(#1,#2){%\verbatim\verbatim\pgfmathsetmacro\tqft@y{#2 * (-\tqft@val{cobordism height})}%\verbatim\verbatim\pgfmathsetmacro\tqft@x{(#1 - 1 + max(min(#2,1),0)*\tqft@val{offset}) * \tqft@val{boundary separation}}%\verbatim\verbatim\def\tqft@shift{(%\tqft@x pt, \tqft@y pt)}%\verbatim\verbatim\}%\verbatim
Now we set up all the keys that we’ll need in the course of this shape
\verbatim\tikzset{\verbatim\Fix for the fact that the alias key doesn’t use the prefix and suffix.\verbatim\verbatim\pic alias/.code={%\verbatim\verbatim\ \tikz@fig@mustbenamed\verbatim\verbatim\ \expandafter\def\expandafter\tikz@alias\expandafter{\tikz@alias\pgfnodealias{\tikz@pp@name{#1}}{\tikz@fig@name}}%\verbatim\}%\verbatim
This key is our basic installer key, setting the pic and putting us in the right key family.
\verbatim\tqft/.style={%\verbatim\verbatim\ pic type=cobordism,\verbatim\verbatim\ every tqft/.try,\verbatim\verbatim\ tqft/.cd,\verbatim\verbatim\}%\verbatim
This deals with unknown keys, passing them on to TikZ.
\verbatim\tqft/.unknown/.code={%\verbatim\verbatim\ \let\tqft@searchname=\pgfkeyscurrentname%\verbatim\verbatim\ \pgfkeysalso{%\verbatim\verbatim\ /tikz/\tqft@searchname={#1}\verbatim\verbatim\}%\verbatim\}%\verbatim
Let’s play happy families!

These set our number of boundary components and genus.

This is the “horizontal” offset of the first outgoing component from the first incoming one.

This is the “vertical” separation between boundary components.

This is the “horizontal” separation between boundary components.

These are the “horizontal” and “vertical” radii, respectively, of the boundary components.

These control the direction from which we view the cobordism.

Should we twist the cobordism?

We simulate node placement using the following key.

The next set of keys define some default shapes.
This is a little helper macro for getting the values of tqft keys.
\def\tqft@val#1{\pgfkeysvalueof{/tikz/tqft/#1}}

Now we define the code for the actual cobordism shape.
Defining the cobordism paths. This holds the full boundary path of the cobordism shape.

\gdef\tqft@fullpath{}\%

This is a list of the edge pieces without the boundary circles.

\global\let\tqft@blist\pgfutil@gobble\%

This punches the holes (if there are any) in the cobordism shape.

\gdef\tqft@gclip{}\%

This is a list of the paths for drawing the holes.

\global\let\tqft@glist\pgfutil@gobble\%

This collects any coordinates that are to be defined (it appears to be difficult to define them as we go along).

\global\let\tqft@clist\pgfutil@gobble\%

This collects any coordinates that can be used to shift the shape that aren’t to be defined using \tqft@clist.

\global\let\tqft@alist\pgfutil@gobble\%

These will be lists of the boundary components, divided into sets as to whether or not they are rendered. For the outgoing ones, we need two lists because they are rendered in the opposite order to how they are labelled.

\global\let\tqft@ibdrylist=\pgfutil@gobble
\global\let\tqft@cibdrylist=\pgfutil@gobble
\global\let\tqft@obdrylist=\pgfutil@gobble
\global\let\tqft@cobdrylist=\pgfutil@gobble
\global\let\tqft@robdrylist=\pgfutil@gobble
\global\let\tqft@rcobdrylist=\pgfutil@gobble

Is the cobordism twisted? If so, we need to reverse the order of the outgoing boundary components.

\iftqft@twisted
  \pgfmathsetmacro\tqft@outgoing@end{0}\%
  \pgfmathsetmacro\tqft@outgoing@dir{-1}\%
\else
  \pgfmathsetmacro\tqft@outgoing@end{1}\%
  \pgfmathsetmacro\tqft@outgoing@dir{1}\%
\fi

The first stage is to iterate over the incoming boundary components (if there are any), building up the various paths.

\ifnum\tqft@val{incoming boundary components}>0\relax

We have some so draw the half circle for the first component. Note that we use \pgf@tqft@upper to flip the sign of the start angle depending on the view from setting.

\xdef\tqft@fullpath{\%
  \tqft@fullpath
  (\-\tqft@val{circle x radius},0) arc[start angle=\pgf@tqft@upper180, end angle=0, x radius=\tqft@val{circle x radius}, y radius=\tqft@val{circle y radius}]\%
}\%

And add the centre to the list for available shifts.

\xdef\tqft@alist{\%
  \tqft@alist,-incoming boundary 1/{(0,0)},-incoming boundary/{(0,0)}\%
}\%
If there are more than one then for each subsequent one we add the curve between them and the corresponding arc of the boundary circle.

\begin{verbatim}
\ifnum\tqft@val{incoming boundary components}>1\relax
\foreach \k in {2,...,\tqft@val{incoming boundary components}} {
  \edef\tqft@temp{\noexpand\pgfutil@in@{,\k,}{,\tqft@val{skip incoming boundary components},}}
  \tqft@temp
  \ifpgfutil@in@
    \xdef\tqft@cibdrylist{\tqft@cibdrylist,\k}
  \else
    \xdef\tqft@ibdrylist{\tqft@ibdrylist,\k}
  \fi
}
\ifx\tqft@ibdrylist\pgfutil@gobble
\else
  \foreach \k [remember=\k as \kmo (initially 1), evaluate=\k as \xpos using (\k-1)*\tqft@val{boundary separation} - \tqft@val{circle x radius}, ] in \tqft@ibdrylist {
    \pgfmathsetmacro\xppos{(%(\kmo - 1)*\tqft@val{boundary separation} + \tqft@val{circle x radius})}
    \pgfmathsetmacro\cpos{(%(\xpos pt,0) .. controls +(0,-\tqft@val{cobordism height}/3) and +(0,-\tqft@val{cobordism height}/3) .. \xpos pt,0)}
    \edef\tqft@fullpath{%
      \tqft@fullpath
      .. controls +(0,-\tqft@val{cobordism height}/3) and +(0,-\tqft@val{cobordism height}/3) .. (\xpos pt,0)
    }
    \edef\tqft@blist{%
      \tqft@blist,incoming boundary \k/incoming/{\xppos pt,0) .. controls +(0,-\tqft@val{cobordism height}/3) and +(0,-\tqft@val{cobordism height}/3) .. (\xpos pt,0)}
    \edef\tqft@clist{%
      \tqft@clist,-between incoming \kmo\space and \k/{(\cpos pt,-\tqft@val{cobordism height}/4)}
    }
    \edef\tqft@alist{%
      \tqft@alist,-incoming boundary \k/{(%(\kmo * \tqft@val{boundary separation},0)}
    }
  }
\fi
\fi
\ifnum\tqft@val{outgoing boundary components}>0\relax
There are.
\end{verbatim}

We start by adding a curve from the end of the last incoming to the last outgoing component to the full path,

\begin{verbatim}
\pgfmathsetmacro\xpos{%
\end{verbatim}

23
\framebox{922} \((\texttt{tqft@outgoing@end} * (\texttt{tqft@val[outgoing boundary components]} -1)) + \texttt{tqft@val[offset]} * \texttt{tqft@val[boundary separation]} + \texttt{tqft@outgoing@dir} * \texttt{tqft@val[circle x radius]})\%\)
\pgfmathsetmacro{\texttt{tqft@ht}}{(\texttt{tqft@val[incoming boundary components]} -\texttt{tqft@val[outgoing boundary components]} -\texttt{offset}) * \texttt{tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} + \texttt{tqft@outgoing@dir} * \texttt{tqft@val[circle x radius]}}\%
\def{\texttt{tqft@fullpath}}{(\texttt{tqft@fullpath} \texttt{. controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) )\% \}
and the edge path.
\def{\texttt{tqft@blist}}{(\texttt{tqft@blist},between last incoming and last outgoing/incoming and outgoing/{(\texttt{tqft@val[incoming boundary components]} \texttt{* \tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \texttt{tqft@val[circle x radius]}),0}) \texttt{controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{. controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{)\%))%}}\%
In addition, we add a coordinate at the midpoint.
\pgfmathsetmacro{\texttt{tqft@ht}}{(\texttt{tqft@val[incoming boundary components]} -1) * \texttt{tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]})/2)%\%
\def{\texttt{tqft@clist}}{(\texttt{tqft@clist},between first last incoming and last outgoing/incoming and outgoing/{(\texttt{tqft@val[incoming boundary components]} \texttt{+ \tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \texttt{tqft@val[circle x radius]}),0} \texttt{controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{. controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{)\%))%}}\%
There aren’t any outgoing boundary components so we loop back to the start. We adjust the height of the control points to take into account the overall width.
\def{\texttt{tqft@fullpath}}{(\texttt{tqft@fullpath} \texttt{. controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{. controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{)\%\}
Same for the edge path.
\def{\texttt{tqft@blist}}{(\texttt{tqft@blist},between first and last incoming/incoming and outgoing/{(\texttt{tqft@val[incoming boundary components]} \texttt{* \tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \texttt{tqft@val[circle x radius]}),0} \texttt{controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{. controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{)\%\}}\%
Add a coordinate at the midpoint.
\def{\texttt{tqft@clist}}{(\texttt{tqft@clist},between first and last incoming/{(\texttt{tqft@val[incoming boundary components]} -1) * \texttt{tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \tqft@val[boundary separation]} + \texttt{tqft@val[circle x radius]} \texttt{+ \texttt{tqft@val[circle x radius]}),0} \texttt{controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{. controls } +0,-\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]} \texttt{and } +0,\texttt{tqft@ht}*\texttt{tqft@val[cobordism height]}) \texttt{)\%\}}\%
\fi
\else
There weren’t any incoming boundary components, so we test to see if there were any outgoing ones and move to the start of them.
We’re done with the incoming boundary components, now we’re set up for the outgoing ones. However we got there, if we have outgoing boundary components then we’re now located at the start of them, although we might be counting backwards.

Draw the arc for the first (well, last actually) boundary component.

And add the centre to the list for available shifts.

Do we have more than one boundary component?

Yes, so add a curve and arc for each.
Both are added to the full path.

\xdef\tqft@fullpath{\tqft@fullpath
  .. controls +(0,\tqft@val{cobordism height}/3) and +(0,\tqft@val{cobordism height})
  \tqft@val{cobordism height}) arc[end angle=\pgf@tqft@upper180, start angle=0, x radius=\tqft@outgoing@dir * \tqft@val{circle x radius}, y radius=\tqft@val{circle y radius}]
}

Just the arc for the edge paths.

\xdef\tqft@blist{\tqft@blist,between outgoing \nk\ space and \nkpo/outgoing/{
  (\xppos pt,-\tqft@val{cobordism height})
  .. controls +(0,\tqft@val{cobordism height}/3) and +(0,\tqft@val{cobordism height})
  \tqft@val{cobordism height}) ++(-2*\tqft@val{circle x radius},0)}

And a coordinate at the midpoint.

\xdef\tqft@clist{\tqft@clist,-between outgoing \nk\ space and \nkpo/{(\cpos pt,-3*\tqft@val{cobordism height}/4)}}

And add the centre to the list for available shifts.

\xdef\tqft@alist{\tqft@alist,-outgoing boundary \nk/{(\xpos pt - \tqft@val{circle x radius},-
  \tqft@val{cobordism height})}}

Now we're at the end of the outgoing boundary components (well, the start actually). What we do now depends on whether or not there are any incoming boundary components.

\ifnum\tqft@val{incoming boundary components}>0\relax

There are, so we draw the path back up.

\pgfmathsetmacro\tqft@ht{1/3 + 2/3*abs(\tqft@val{offset})/(abs(\tqft@val{offset}) + 1)}
\xdef\tqft@fullpath{\tqft@fullpath
  .. controls +(0,\tqft@ht*\tqft@val{cobordism height}) and +(0,-\tqft@ht*\tqft@val{circle x radius}),0)
}

And the edge path does the same.

\xdef\tqft@blist{\tqft@blist,between first incoming and first outgoing/incoming and outgoing/{
  ( (1 - \tqft@outgoing@end) * (\tqft@val{outgoing boundary components} - 1) + \tqft@val{cobordism height})
  \tqft@val{circle x radius},0)
}

Add a coordinate at the midpoint.

\xdef\tqft@clist{\tqft@clist,}

26
No incoming boundary components so loop back to the other end of the outgoing boundary components.

Full path.

Edge path.

Add a coordinate at the midpoint.

Now we define the clip path for the genus holes. We start with a big rectangle that ought to be big enough to contain the whole shape. We start with the top left corner.

Now the bottom right.
Together, these make a rectangle.

\begin{itemize}
\item Are there any holes?
\end{itemize}

\texttt{\pgfmathsetmacro\xpos{\tqft@val\{incoming boundary components\} > 0 ? \max(\tqft@val\{incoming boundary components\},\tqft@val\{outgoing boundary components\}) : \tqft@val\{incoming boundary components\} -1) *\tqft@val\{boundary separation\} + 2*\tqft@val\{circle x radius\}}%

\texttt{\def\tqft@gclip{\tqft@gclip (\xpos pt,-\tqft@val\{cobordism height\} -2*\tqft@val\{circle y radius\})}}%

\texttt{\pgfmathsetmacro\ypos{\tqft@val\{outgoing boundary components\} > 0 ? \tqft@val\{incoming boundary components\} > 0 ? \tqft@val\{offset\}/2 : \tqft@val\{offset\} : 0 \tqft@val\{offset\}) *\tqft@val\{boundary separation\} - \tqft@val\{circle x radius\}}%

\texttt{\def\tqft@gclip{\tqft@gclip (\xpos pt,\ypos pt)}}%

\texttt{\pgfmathsetmacro\gsize{(\tqft@val\{outgoing boundary components\} > 0 ? \tqft@val\{incoming boundary components\} > 0 ? (\tqft@val\{incoming boundary components\} + \tqft@val\{outgoing boundary components\})/2 : \tqft@val\{outgoing boundary components\}) -1) *\tqft@val\{boundary separation\} /\tqft@val\{circle x radius\} + 2}}%

Each hole should take up three half-widths, but we want a little extra on the edges so the total number of half-widths we want is $3g + 1$. Do we need to scale down the holes (we never scale up)? If so, \texttt{\gyscale} holds the overall scale factor and \texttt{\gxyscale} are the resulting horizontal and vertical measurements. The baseline is the size of the boundary circles.
Each hole should take up 2 half widths, modulo scaling, so the total width used by the holes is $2g$ leaving $w - 2g$ left for the gaps which is divided into $g + 1$ lots.

We shift in by half of one unit of excess separation.

Some useful quantities.

Now we iterate over the holes.

For the clipping path, we just want the bare hole.

Move in by half an excess separation unit and move to the left-hand extent of the hole.

Now curve up over the hole,

and return on the underside.

Lastly, move to the right-hand edge of the space taken up by this hole.

For the genus path we want to add the little “tails” which means that the two curves are different, and we need to take into account the view from direction.

Move to the starting point of the smaller curve and add that.
Add the larger of the two curves.
\begin{verbatim}
Add a coordinate at the centre of the hole.
\end{verbatim}
Now we start to lay out the cobordism Were we given a shift? If so, shift.
\begin{verbatim}
At each incoming boundary component we place an elliptical node of the right
size.
\end{verbatim}
\begin{tikzpicture}
\foreach [evaluate=k as \kpos using (k-1)*\tqft@val{boundary separation}] k in \tqft@inboundlist 
\node[
  transform shape, 
  node contents={}, 
  ellipse, 
  inner sep=0pt, 
  outer sep=0pt, 
  minimum width=2*\tqft@val{circle x radius}, 
  minimum height=2*\tqft@val{circle y radius}, 
  at={(\kpos pt,0)}, 
  name=-incoming boundary \k, 
  /tikz/tqft/every skipped boundary component/.try, 
  /tikz/tqft/every skipped incoming boundary component/.try, 
  /tikz/tqft/skipped incoming boundary component \k/.try, 
];
\end{tikzpicture}
\path node also[pic alias=-incoming boundary] (-incoming boundary 1);
\end{tikzpicture}
\begin{tikzpicture}
\ifnum\tqft@val{outgoing boundary components}>0\relax
\foreach \k [evaluate=k as \ok using int(\tqft@outgoing@end * (\tqft@val{outgoing boundary components} + 1) - \tqft@outgoing@dir * \k)] in \tqft@outboundlist 
\xdef\tqft@robdrylist{\tqft@robdrylist,\ok}
\foreach[evaluate=k as \kpos using (k-1+\tqft@val{offset})*\tqft@val{boundary separation}] k in \tqft@robdrylist 
\node[
  transform shape, 
  node contents={}, 
  ellipse, 
  inner sep=0pt, 
  outer sep=0pt, 
  minimum width=2*\tqft@val{circle x radius}, 
  minimum height=2*\tqft@val{circle y radius}, 
  at={(\kpos pt,0)}, 
  name=-outgoing boundary \k, 
  /tikz/tqft/every boundary component/.try, 
  /tikz/tqft/every outgoing boundary component/.try, 
  /tikz/tqft/outgoing boundary component \k/.try, 
];
\end{tikzpicture}
\begin{tikzpicture}
\if\tqft@cobdrylist\else
\end{tikzpicture}
\foreach \k[evaluate=\k as \ok using int(\tqft@outgoing@end * (\tqft@val{outgoing boundary components} + 1) - \tqft@outgoing@dir * \k)] in \tqft@cobdrylist {
  \def\tqft@rcobdrylist{\tqft@rcobdrylist,\ok}
}

\foreach \k in \tqft@rcobdrylist {
  \node[
    transform shape,
    node contents={},
    ellipse,
    inner sep=0pt,
    outer sep=0pt,
    minimum width=2*\tqft@val{circle x radius},
    minimum height=2*\tqft@val{circle y radius},
    at={(\xpos pt,-\tqft@val{cobordism height})},
    name=-outgoing boundary \k,
    /tikz/tqft/every skipped boundary component/.try,
    /tikz/tqft/every skipped outgoing boundary component/.try,
    /tikz/tqft/skipped outgoing boundary component \k/.try
  ];
}

Add an alias for the first.
\path node also[pic alias=-outgoing boundary] (-outgoing boundary 1);
\fi
Now we draw the lower paths of the incoming boundary components.
\ifnum\tqft@val{incoming boundary components}>0\relax
  \foreach[evaluate=\k as \xpos using ((\k-1)*\tqft@val{boundary separation})] \k in \tqft@ibdrylist {
    \path[\tikz/tqft/every lower boundary component/.try,
    \tikz/tqft/every incoming lower boundary component/.try,
    \tikz/tqft/incoming lower boundary component \k/.try
    ] (\xpos pt - \tqft@val{circle x radius},0) arc[start angle=\pgf@tqft@lower180,end angle=0,
    x radius=\tqft@val{circle x radius}, y radius =\tqft@val{circle y radius}];
  }
\fi

Same for the outgoing boundary components.
\ifnum\tqft@val{outgoing boundary components}>0\relax
  \foreach[evaluate=\k as \xpos using ((\k-1)*\tqft@val{offset})*\tqft@val{boundary separation}] \k in \tqft@robdrylist {
    \path[\tikz/tqft/every lower boundary component/.try,
    \tikz/tqft/every outgoing lower boundary component/.try,
    \tikz/tqft/outgoing lower boundary component \k/.try
    ] (\xpos pt - \tqft@val{circle x radius},-\tqft@val{cobordism height}) arc[start angle=\pgf@tqft@lower180,end angle=0,
    x radius=\tqft@val{circle x radius}, y radius =\tqft@val{circle y radius}];
  }
\fi

Full outer path, clipped against the genus holes in case it is filled.
\begin{scope}
  \path[overlay,clip] \tqft@gclip;
  \path[

Now we draw the genus path, outside the clip. We view this as part of the full cobordism path so try to apply the same style as for the full path, but if that is filled then we turn the fill off. It can be turned back on again using the styles cobordism edge or genus style. We also apply the cobordism edge style as it could be thought of as part of the non-boundary edge. Finally, it has its own style to enable overrides if the other two get confused.

Now we redraw the non-boundary paths.

There were various coordinates that we wanted to define but couldn’t. Here, we put those in place.

The last task is to draw the upper paths of the boundary components. First, incoming.
\path[
]/tikz/tqft/every upper boundary component/.try,
/tikz/tqft/every incoming upper boundary component/.try,
/tikz/tqft/incoming upper boundary component \k/.try
] (\xpos pt - \tqft@val{circle x radius},0) arc[start angle=\pgf@tqft@upper180,end angle=0, x radius=\tqft@val{circle x radius}, y radius =\tqft@val{circle y radius}];
\fi

Next, outgoing.
\ifnum\tqft@val{outgoing boundary components}>0\relax
\foreach[
  evaluate=\k as \xpos using (\k-1+\tqft@val{offset})*\tqft@val{boundary}
] \k in \tqft@robdrylist {
\path[
/tikz/tqft/every upper boundary component/.try,
/tikz/tqft/every outgoing upper boundary component/.try,
/tikz/tqft/outgoing upper boundary component \k/.try
] (\xpos pt - \tqft@val{circle x radius},-\tqft@val{cobordism height}) arc[start angle=\pgf@tqft@upper180,end angle=0, x radius=\tqft@val{circle x radius}, y radius =\tqft@val{circle y radius}];
\fi
\end{scope}

We're done! Phew.
\}
\}

34