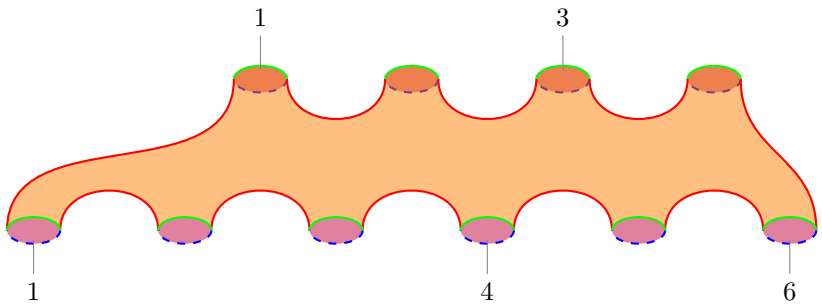
The tqft package: codebase

Andrew Stacey
loopspace@mathforge.org

? from ?



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

```
\RequirePackage{pgfkeys}
\RequirePackage{pgf}
```

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.

```
\def\pgf@tqft@minus{-}
\let\pgf@tqft@upper\@empty
\let\pgf@tqft@lower\pgf@tqft@minus
```

Some helpful extra functions.

`\tqftset` is our equivalent of `\tikzset`.

```
\def\tqftset#1{\pgfkeys{/pgf/tqft/.cd,#1}}
```

`\tqft@process` This macro applies our flow transformation to the given coordinates, with the result stored in `\pgf@x` and `\pgf@y`.

```
\def\tqft@process#1#2{%
  \edef\tqft@px{#1}
  \edef\tqft@py{#2}
  \pgf@process{
```

```
    \pgftransformreset
    \let\tikz@transform=\pgfutil@empty
    \expandafter\tikzset\expandafter{\tqft@transformation}
    \tikz@transform
    \pgfpointtransformed{\pgfqpoint{\tqft@px}{\tqft@py}}
  }
}
```

Declare some dimension registers to hold the specifications of the cobordism.

```
\newdimen\tqft@xa
\newdimen\tqft@xb
\newdimen\tqft@c
\newdimen\tqft@ch
\newdimen\tqft@h
\newdimen\tqft@s
\newdimen\tqft@w
\newif\iftqft@within@node
```

Now we set up all the keys that we'll need in the course of this shape

```
\pgfkeys{
```

Add a key to switch between the two versions.

```
  /tikz/tqft/use nodes/.is choice,
  /tikz/tqft/use nodes/true/.code={%
    \tikzset{
```

If using nodes, set the defaults

```
      tqft/.style={%
        /tikz/shape=tqft cobordism,
        /pgf/tqft,
        /tikz/every tqft/.try
      },
```

Unknowns go to `/pgf/tqft`

```
      tqft/.unknown/.code={%
        \let\tqft@searchname=\pgfkeyscurrentname%
        \pgfkeys{%
          /pgf/tqft/\tqft@searchname={##1}
        }
      },
    }%
  },
```

If not using nodes, set the defaults for the library

```
  /tikz/tqft/use nodes/false/.code={%
    \tikzset{
      tqft/.style={%
        pic type=cobordism,
        tqft/.cd,
        every tqft/.try,
      },
```

Pass unknown keys on to TikZ.

```
      tqft/.unknown/.code={%
        \let\tqft@searchname=\pgfkeyscurrentname%
        \pgfkeys{%
```

```
          /tikz/\tqft@searchname={##1}
        }
      },
    }%
  },
  /tikz/tqft/use nodes=true,
```

This deals with unknown keys, passing them on to TikZ.

```
  /pgf/tqft/.unknown/.code={%
    \let\tqft@searchname=\pgfkeyscurrentname%
    \pgfkeysalso{%
      /tikz/\tqft@searchname={#1}
    }
  },
```

Let's play happy families!

```
  /pgf/tqft/.is family,
  /pgf/tqft,
```

This sets our shape to be the boundary circle

```
  boundary circle/.style={
    /tikz/shape=tqft boundary circle
  },
```

These set our number of boundary components

```
  incoming boundary components/.initial=5,
  outgoing boundary components/.initial=4,
```

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

```
  offset/.initial=0,
```

This is the "vertical" separation between boundary components.

```
  cobordism height/.initial=2cm,
```

This is the "horizontal" separation between boundary components.

```
  boundary separation/.initial=2cm,
```

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

```
  circle width/.initial=10pt,
  circle depth/.initial=5pt,
```

These control the separation between the node and its anchors.

```
  outer xsep/.initial=0pt,
  outer ysep/.initial=0pt,
  outer sep/.style={
    outer xsep=#1,
    outer ysep=#1
  },
```

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/.code={%
    \pgfkeys{/pgf/tqft/flow transformation/.expand twice/.expand once=\pgfkeysvalueof{/pgf/t
```

```
  },
  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
      }
    }
  },
  cobordism style contents/.initial={},
  cobordism style/.code={%
    \pgfkeys{/pgf/tqft/cobordism style contents/.style={%
        /tikz/.cd,#1%
      }
    }
  },
```

The next set of keys define some default shapes.

```
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,
  },
}
```

`tqft shape` This is a generic cobordism shape

```
\pgfdeclareshape{tqft cobordism}{
```

Save our specifications: incoming and outgoing boundary components

```
  \savedmacro{\tqft@incoming}{\edef\tqft@incoming{\pgfkeysvalueof{/pgf/tqft/incoming boundar
  \savedmacro{\tqft@outgoing}{\edef\tqft@outgoing{\pgfkeysvalueof{/pgf/tqft/outgoing boundar
```

and the offset (in units of boundary components) between the leading incoming and outgoing components (regarded as a shift of the outgoing components relative to the incoming)

```
  \savedmacro{\tqft@offset}{\edef\tqft@offset{\pgfkeysvalueof{/pgf/tqft/offset}}}
```

Now we save our dimensions: height, separation, the radii of the boundary circles, and outer seps, and the heights of the control points.

```
  \saveddimen{\tqft@height}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/cobordism height}}
  \saveddimen{\tqft@separation}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/boundary separation}}
  \saveddimen{\tqft@width}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/circle width}}
  \saveddimen{\tqft@depth}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/circle depth}}
  \saveddimen{\tqft@outerxsep}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/outer xsep}}
  \saveddimen{\tqft@outerysep}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/outer ysep}}
  \saveddimen{\tqft@control}{
  \pgfkeysgetvalue{/pgf/tqft/cobordism height}{\tqft@tempa}
  \pgfkeysgetvalue{/pgf/tqft/circle depth}{\tqft@tempb}
  \pgfmathsetlength{\pgf@x}{.5 * \tqft@tempa - 4 * \tqft@tempb}
}
```

This is the internal transformation that is in place

```
   \savedmacro{\tqft@transformation}{%
  \pgfkeysgetvalue{/pgf/tqft/flow transformation}{\tqft@transformation}
}
```

For the externally available anchors, we need to save a few things as well.

Position of first incoming boundary in internal coordinates

```
   \savedanchor{\tqft@start@incoming}{%
    \pgfmathsetlength{\pgf@x}{-(max(\pgfkeysvalueof{/pgf/tqft/incoming boundary components}
    \pgfmathsetlength{\pgf@y}{.5 * \pgfkeysvalueof{/pgf/tqft/cobordism height}}
}
```

Position of first outgoing boundary in internal coordinates

```
   \savedanchor{\tqft@start@outgoing}{%
    \pgfmathsetlength{\pgf@x}{-(max(\pgfkeysvalueof{/pgf/tqft/incoming boundary components}
    \pgfmathsetlength{\pgf@y}{-.5 * \pgfkeysvalueof{/pgf/tqft/cobordism height}}
}
```

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}{\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{Opt}{\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{Opt}{\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
  \advance\pgf@xb by -\tqft@width
  \pgf@yb=\pgf@y
  \pgf@xc=.5\pgf@xa
  \advance\pgf@xc by .5\pgf@xb
  \pgf@yc=.5\pgf@ya
  \advance\pgf@yc by .5\pgf@yb
  \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=.5\pgf@xa
  \advance\pgf@xc by .5\pgf@xb
  \pgf@yc=.5\pgf@ya
  \advance\pgf@yc by .5\pgf@yb
  \advance\pgf@xc by \tqft@outerxsep\relax
  \tqft@process{\the\pgf@xc}{\the\pgf@yc}
```

```
}
\anchor{north west}{
  \tqft@start@incoming
  \pgf@xc=\pgf@x
  \pgf@yc=\pgf@y
  \advance\pgf@xc by -\tqft@width\relax
  \advance\pgf@yc by \tqft@outerysep\relax
  \advance\pgf@xc by -\tqft@outerxsep\relax
  \tqft@process{\the\pgf@xc}{\the\pgf@yc}
}
\anchor{south west}{
  \tqft@start@outgoing
  \pgf@xc=\pgf@x
  \pgf@yc=\pgf@y
  \advance\pgf@xc by -\tqft@width\relax
  \advance\pgf@yc by -\tqft@outerysep\relax
  \advance\pgf@xc by -\tqft@outerxsep\relax
  \tqft@process{\the\pgf@xc}{\the\pgf@yc}
}
\anchor{north east}{
  \tqft@start@incoming
  \pgf@xc=\pgf@x
  \pgfmathsetlength{\pgf@xc}{\pgf@xc + (\tqft@incoming - 1)*\tqft@separation}
  \pgf@yc=\pgf@y
  \advance\pgf@xc by \tqft@width\relax
  \advance\pgf@yc by \tqft@outerysep\relax
  \advance\pgf@xc by \tqft@outerxsep\relax
  \tqft@process{\the\pgf@xc}{\the\pgf@yc}
}
\anchor{south east}{
  \tqft@start@outgoing
  \pgf@xc=\pgf@x
  \pgfmathsetlength{\pgf@xc}{\pgf@xc + (\tqft@outgoing - 1)*\tqft@separation}
  \pgf@yc=\pgf@y
  \advance\pgf@xc by \tqft@width\relax
  \advance\pgf@yc by -\tqft@outerysep\relax
  \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.

```
\expandafter\pgfutil@g@addto@macro\csname pgf@sh@s@tqft cobordism\endcsname{%
  \c@pgf@counta\tqft@incoming\relax%
  \pgfmathloop%
  \ifnum\c@pgf@counta>0\relax%
  \pgfutil@ifundefined{pgf@anchor@tqft cobordism@incoming boundary\space\the\c@pgf@counta}{%
    \expandafter\xdef\csname pgf@anchor@tqft cobordism@incoming boundary\space\the\c@pgf@cou
  \noexpand\tqft@start@incoming
  \noexpand\pgfmathsetlength{\noexpand\pgf@y}{\noexpand\pgf@y + \noexpand\tqft@outerysep}
  \noexpand\pgfmathsetlength{\noexpand\pgf@x}{\noexpand\pgf@x + (\the\c@pgf@counta - 1) * \n
  \noexpand\tqft@process{\noexpand\the\noexpand\pgf@x}{\noexpand\the\noexpand\pgf@y}
  }
  }{\c@pgf@counta0\relax}%
```

```
  \advance\c@pgf@counta-1\relax%
  \repeatpgfmathloop%
}
\expandafter\pgfutil@g@addto@macro\csname pgf@sh@s@tqft cobordism\endcsname{%
  \c@pgf@counta\tqft@outgoing\relax%
  \pgfmathloop%
  \ifnum\c@pgf@counta>0\relax%
  \pgfutil@ifundefined{pgf@anchor@tqft cobordism@outgoing boundary\space\the\c@pgf@counta}{%
    \expandafter\xdef\csname pgf@anchor@tqft cobordism@outgoing boundary\space\the\c@pgf@cou
  \noexpand\tqft@start@outgoing
  \noexpand\pgfmathsetlength{\noexpand\pgf@y}{\noexpand\pgf@y - \noexpand\tqft@outerysep}
  \noexpand\pgfmathsetlength{\noexpand\pgf@x}{\noexpand\pgf@x + (\the\c@pgf@counta - 1) * \n
  \noexpand\tqft@process{\noexpand\the\noexpand\pgf@x}{\noexpand\the\noexpand\pgf@y}
  }
  }{\c@pgf@counta0\relax}%
  \advance\c@pgf@counta-1\relax%
  \repeatpgfmathloop%
}
\expandafter\pgfutil@g@addto@macro\csname pgf@sh@s@tqft cobordism\endcsname{%
  \c@pgf@counta\tqft@incoming\relax%
  \advance\c@pgf@counta-1\relax
  \pgfmathloop%
  \ifnum\c@pgf@counta>0\relax%
  \pgfutil@ifundefined{pgf@anchor@tqft cobordism@after incoming boundary\space\the\c@pgf@cou
    \expandafter\xdef\csname pgf@anchor@tqft cobordism@after incoming boundary\space\the\c@p
  \noexpand\tqft@start@incoming
  \noexpand\pgfmathsetlength{\noexpand\pgf@y}{.25 * \noexpand\pgf@y +.75 * \noexpand\tqft@co
  \noexpand\pgfmathsetlength{\noexpand\pgf@x}{\noexpand\pgf@x + (\the\c@pgf@counta - .5) * \
  \noexpand\tqft@process{\noexpand\the\noexpand\pgf@x}{\noexpand\the\noexpand\pgf@y}
  }
  }{\c@pgf@counta0\relax}%
  \advance\c@pgf@counta-1\relax%
  \repeatpgfmathloop%
}
\expandafter\pgfutil@g@addto@macro\csname pgf@sh@s@tqft cobordism\endcsname{%
  \c@pgf@counta\tqft@outgoing\relax%
  \advance\c@pgf@counta-1\relax
  \pgfmathloop%
  \ifnum\c@pgf@counta>0\relax%
  \pgfutil@ifundefined{pgf@anchor@tqft cobordism@after outgoing boundary\space\the\c@pgf@cou
    \expandafter\xdef\csname pgf@anchor@tqft cobordism@after outgoing boundary\space\the\c@p
  \noexpand\tqft@start@outgoing
  \noexpand\pgfmathsetlength{\noexpand\pgf@y}{.25 * \noexpand\pgf@y -.75 * \noexpand\tqft@co
  \noexpand\pgfmathsetlength{\noexpand\pgf@x}{\noexpand\pgf@x + (\the\c@pgf@counta - .5) * \
  \noexpand\tqft@process{\noexpand\the\noexpand\pgf@x}{\noexpand\the\noexpand\pgf@y}
  }
  }{\c@pgf@counta0\relax}%
  \advance\c@pgf@counta-1\relax%
  \repeatpgfmathloop%
}
```

Now we define the background path. This is the upper part of the cobordism.

```
  \backgroundpath{
```

Apply the internal transformation

```
    \let\tikz@transform=\pgfutil@empty
    \expandafter\tikzset\expandafter{\tqft@transformation}
    \tikz@transform
```

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
    \tqft@xb=\tqft@xa
    \advance\tqft@xb 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}
```

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}}{\pgfqpoin
     \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(\t
     \pgfpathcurveto{\pgfqpoint{\tqft@xb}{\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@ch}}{\pgfqpo
    \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@xa}{\tqft@xa + (\tqft@outgoing - 1 + \tqft@offset) * \tqft@sepa
```

If we had incoming boundaries, this is a curveto, otherwise it's a moveto

```
    \ifnum\tqft@incoming>0
     \pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\t
```

```
    \pgfpathcurveto{\pgfqpoint{\tqft@xb}{\tqft@ch}}{\pgfqpoint{\tqft@xa}{-\tqft@ch}}{\pgfqpoint{\tqft@xa}{-\tqft@h}}
   \else
    \pgfpathmoveto{\pgfqpoint{\tqft@xa}{-\tqft@h}}
   \fi
    \tqft@xb=\tqft@xa
    \advance\tqft@xb by -\tqft@w
```

Now draw the lower components

```
    \pgfpatharc{0}{\pgf@tqft@upper180}{\tqft@width and \tqft@depth}
```

Now iterate over the remaining outgoing boundary components

```
   \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
```

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

```
    \advance\tqft@xb by -\tqft@outgoing\tqft@s
    \advance\tqft@xb by \tqft@s
   \ifnum\tqft@incoming>0
    \pgfmathsetlength{\tqft@xa}{\tqft@xa - (\tqft@outgoing -1 + \tqft@offset) * \tqft@separ
    \pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\t
    \pgfpathcurveto{\pgfqpoint{\tqft@xb}{-\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@ch}}{\pgfqp
   \else
    \pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\t
    \pgfpathcurveto{\pgfqpoint{\tqft@xb}{-\tqft@ch}}{\pgfqpoint{\tqft@xa}{-\tqft@ch}}{\pgfqpoint{\tqft@xa}{-\tqft@h}}
   \fi
   \fi
```

Close the path

```
    \pgfpathclose
 }
```

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

```
 \behindbackgroundpath{
```

Apply the internal transformation

```
   \let\tikz@transform=\pgfutil@empty
   \expandafter\tikzset\expandafter{\tqft@transformation}
   \tikz@transform
```

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

```
    \pgfmathsetlength{\tqft@xa}{-(max(\tqft@incoming - 1,\tqft@outgoing - 1 + \tqft@offset)
    \pgfmathsetlength{\tqft@h}{.5 * \tqft@height}
```

This section draws the boundary circles

```
    {
```

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

```
      \tikz@mode@fillfalse%
      \tikz@mode@drawfalse%
      \let\tikz@mode=\pgfutil@empty
      \let\tikz@options=\pgfutil@empty
      \tqftset{boundary style contents}
      \tikz@mode
      \tikz@options
```

Do we have any incoming boundary components at all?

```
    \ifnum\tqft@incoming>0
```

Yes, so iterate over them

```
    \foreach \tqft@k in {1,...,\tqft@incoming} {
     \advance\tqft@xa by \tqft@k\tqft@s
     \pgfpathellipse{\pgfqpoint{\tqft@xa}{\tqft@h}}{\pgfqpoint{\tqft@width}{0pt}}{\pgfqpoint
    }
    \fi
```

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@sepa
    \foreach \tqft@k in {1,...,\tqft@outgoing} {
     \advance\tqft@xa by -\tqft@k\tqft@s
%     \advance\tqft@xa by \tqft@s
     \pgfpathellipse{\pgfqpoint{\tqft@xa}{-\tqft@h}}{\pgfqpoint{\tqft@width}{0pt}}{\pgfqpoin
    }
    \fi
```

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

This section draws the lower parts of the boundary circles

```
    {
```

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

```
      \tikz@mode@fillfalse%
      \tikz@mode@drawfalse%
      \let\tikz@mode=\pgfutil@empty
      \let\tikz@options=\pgfutil@empty
      \tqftset{boundary lower style contents}
      \tikz@mode
      \tikz@options
      \advance\tqft@xa by .5\tqft@w
```

Do we have any incoming boundary components at all?

```
    \ifnum\tqft@incoming>0
```

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}
    }
    \fi
```

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@sepa
    \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@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@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

```
    \pgfmathsetlength{\tqft@xa}{-(max(\tqft@incoming - 1,\tqft@outgoing - 1 + \tqft@offset)
    \tqft@xb=\tqft@xa
    \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@fillfalse%
      \tikz@mode@drawfalse%
      \let\tikz@mode=\pgfutil@empty
      \let\tikz@options=\pgfutil@empty
      \tqftset{cobordism style contents}
```

```
      \tikz@mode
      \tikz@options
% Do we have any incoming boundary components at all?
%    \begin{macrocode}
    \ifnum\tqft@incoming>0
```

Do we have more than one?

```
   \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
    \pgfpathmoveto{\pgfqpoint{\tqft@xb}{\tqft@h}}
     \pgfpathcurveto{\pgfqpoint{\tqft@xb}{\tqft@c}}{\pgfqpoint{\tqft@xa}{\tqft@c}}{\pgfqpoin
    }
    \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(\t
    \pgfpathmoveto{\pgfqpoint{\tqft@xb}{\tqft@h}}
     \pgfpathcurveto{\pgfqpoint{\tqft@xb}{\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@ch}}{\pgfqpo
    \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@xa}{\tqft@xa + (\tqft@outgoing - 1 + \tqft@offset) * \tqft@sepa
```

If we had incoming boundaries, this is a curveto, otherwise it's a moveto

```
    \ifnum\tqft@incoming>0
     \pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\t
     \pgfpathmoveto{\pgfqpoint{\tqft@xb}{\tqft@h}}
     \pgfpathcurveto{\pgfqpoint{\tqft@xb}{\tqft@ch}}{\pgfqpoint{\tqft@xa}{-
\tqft@ch}}{\pgfqpoint{\tqft@xa}{-\tqft@h}}
    \else
     \pgfpathmoveto{\pgfqpoint{\tqft@xa}{-\tqft@h}}
    \fi
     \tqft@xb=\tqft@xa
     \advance\tqft@xb by -\tqft@w
```

Now draw the lower components

```
     \pgfpathmoveto{\pgfqpoint{\tqft@xb}{-\tqft@h}}
```

Now iterate over the remaining outgoing boundary components

```
    \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}}
      \advance\tqft@xa by -\tqft@w
      \pgfpathmoveto{\pgfqpoint{\tqft@xa}{-\tqft@h}}
     }
     \fi
```

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

```
      \advance\tqft@xb by -\tqft@outgoing\tqft@s
      \advance\tqft@xb by \tqft@s
     \ifnum\tqft@incoming>0
      \pgfmathsetlength{\tqft@xa}{\tqft@xa - (\tqft@outgoing -1 + \tqft@offset) * \tqft@separ
      \pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\t
      \pgfpathcurveto{\pgfqpoint{\tqft@xb}{-\tqft@ch}}{\pgfqpoint{\tqft@xa}{\tqft@ch}}{\pgfqp
     \else
      \pgfmathsetlength{\tqft@ch}{min(0,max(-\tqft@h,\tqft@h - (\tqft@h - \tqft@c) * ((abs(\t
      \pgfpathcurveto{\pgfqpoint{\tqft@xb}{-\tqft@ch}}{\pgfqpoint{\tqft@xa}{-
\tqft@ch}}{\pgfqpoint{\tqft@xa}{-\tqft@h}}
     \fi
     \fi
       \edef\tikz@temp{\noexpand\pgfusepath{%
           \iftikz@mode@fill fill,\fi%
           \iftikz@mode@draw draw\fi%
       }}%
       \tikz@temp
}
```

This section draws the upper parts of the boundary circles

```
     {
```

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

```
\let\tqft@bdry@path=\pgfutil@empty
\let\tqft@bdry@node@path=\pgfutil@empty
\pgfsyssoftpath@setcurrentpath{\tqft@bdry@path}
       \tikz@mode@fillfalse%
       \tikz@mode@drawfalse%
       \let\tikz@mode=\pgfutil@empty
       \let\tikz@options=\pgfutil@empty
       \tqftset{boundary upper style contents}
       \tikz@mode
       \tikz@options
       \advance\tqft@xa by -\tqft@s
       \advance\tqft@xa by \tqft@w
```

Do we have any incoming boundary components at all?

```
     \ifnum\tqft@incoming>0
```

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@upper180}{\tqft@width and \tqft@depth}
   \ifx\tikz@fig@name\pgfutil@empty
   \else
```

```
  {
    \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 incoming \tqft@k}{}
    \pgfsyssoftpath@getcurrentpath{\tqft@bdry@node@path}
    \pgfsyssoftpath@setcurrentpath{\tqft@bdry@path}
  }
  \fi
    }
    \fi
```

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@sepa
    \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
    }
  }
}
```

`boundary circle shape` This is a the shape of the boundary circles

```
\pgfdeclareshape{tqft boundary circle}{
```

Now we save our dimensions: height, separation, and the radii of the boundary circles

```
  \saveddimen{\tqft@height}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/cobordism height}}
  \saveddimen{\tqft@separation}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/boundary separation}}
```

```
  \saveddimen{\tqft@width}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/circle width}}
  \saveddimen{\tqft@depth}{\pgf@x=\pgfkeysvalueof{/pgf/tqft/circle depth}}
```

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 intepreted 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}}
  }%
```

```
  \tqft@marshal
  \tqft@process{\the\pgf@x}{\the\pgf@y}
}
```

Now we define the background path. This is the upper part of the cobordism.

```
  \backgroundpath{
```

Apply the internal transformation if we're not within a node

```
    \let\tikz@transform=\pgfutil@empty
    \expandafter\tikzset\expandafter{\tqft@transformation}
    \tikz@transform
```

Draw the boundary circle

```
  \pgfpathellipse{\pgfqpoint{0pt}{0pt}}{\pgfqpoint{\tqft@width}{0pt}}{\pgfqpoint{0pt}{\tqft@
  }
```

We draw the upper and lower arcs again with the appropriate styles

```
  \beforebackgroundpath{
  \iftqft@within@node
  \else
    \tikz@mode@fillfalse%
    \tikz@mode@drawfalse%
    \let\tikz@mode=\pgfutil@empty
    \let\tikz@options=\pgfutil@empty
    {
    \pgfsys@beginscope
      \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
      \pgfsys@endscope
    }
    {
      \pgfsys@beginscope
      \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
      \pgfsys@endscope
    }
    \fi
  }
  }
```

2.2 New Version: Picture Shapes

Issue a warning if the pic syntax is not available.

```
\ifcsname pgfk@/handlers/.pic/.@cmd\endcsname
\else
\pgfwarning{This library only works with TikZ 3.0 or later; for earlier versions of TikZ use
\fi
```

For the boundaries, we need elliptical node shapes.

```
\usetikzlibrary{shapes.geometric}
```

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.

```
\def\pgf@tqft@minus{-}
\let\pgf@tqft@upper\@empty
\let\pgf@tqft@lower\pgf@tqft@minus
```

Should we twist the cobordism?

```
\newif\iftqft@twisted
```

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.

```
\def\tqft@split(#1,#2){%
  \pgfmathsetmacro\tqft@y{#2 * (-\tqft@val{cobordism height})}%
  \pgfmathsetmacro\tqft@x{(#1 - 1 + max(min(#2,1),0)*\tqft@val{offset}) * \tqft@val{boundary
  \def\tqft@shift{(\tqft@x pt, \tqft@y pt)}%
}%
```

Now we set up all the keys that we'll need in the course of this shape

```
\tikzset{
```

Fix for the fact that the `alias` key doesn't use the prefix and suffix.

```
    pic alias/.code={%
      \tikz@fig@mustbenamed
      \expandafter\def\expandafter\tikz@alias\expandafter{\tikz@alias\pgfnodealias{\tikz@pp@
    },
```

This key is our basic installer key, setting the pic and putting us in the right key family.

```
  tqft/.style={%
    pic type=cobordism,
    every tqft/.try,
    tqft/.cd,
  },
```

This deals with unknown keys, passing them on to TikZ.

```
  tqft/.unknown/.code={%
    \let\tqft@searchname=\pgfkeyscurrentname%
    \pgfkeysalso{%
      /tikz/\tqft@searchname={#1}
    }
  },
```

Let's play happy families!

```
  tqft/.cd,
```

These set our number of boundary components and genus.

```
  incoming boundary components/.initial=5,
  outgoing boundary components/.initial=4,
  skip incoming boundary components/.initial={},
  skip outgoing boundary components/.initial={},
  genus/.initial = 0,
```

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

```
  offset/.initial=0,
```

This is the "vertical" separation between boundary components.

```
  cobordism height/.initial=2cm,
```

This is the "horizontal" separation between boundary components.

```
  boundary separation/.initial=2cm,
```

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

```
  circle x radius/.initial=10pt,
  circle y radius/.initial=5pt,
```

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
  },
```

Should we twist the cobordism?

```
  twisted/.is if=tqft@twisted,
```

We simulate node placement using the following key.

```
  anchor/.initial = none,
```

The next set of keys define some default shapes.

```
  pair of pants/.style={
    /tikz/tqft,
    incoming boundary components=1,
    outgoing boundary components=2,
    offset=-.5
  },
  /tikz/tqft pair of pants/.style={
    /tikz/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={
    /tikz/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={
    /tikz/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={
    /tikz/tqft/cylinder to next,
  },
  cylinder/.style={
    /tikz/tqft,
    incoming boundary components=1,
    outgoing boundary components=1
  },
  /tikz/tqft cylinder/.style={
    /tikz/tqft/cylinder,
  },
  cup/.style={
    /tikz/tqft,
    incoming boundary components=1,
    outgoing boundary components=0
  },
  /tikz/tqft cup/.style={
    /tikz/tqft/cup,
  },
  cap/.style={
    /tikz/tqft,
    incoming boundary components=0,
    outgoing boundary components=1
  },
  /tikz/tqft cap/.style={
    /tikz/tqft/cap,
  },
}
```

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.

```
\tikzset{
  cobordism/.pic={
```

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 r
}%
```

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.

```
    \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 componen
      \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
      \pgfmathsetmacro\cpos{(\xpos + \xppos)/2}
```

Add the curve and the arc.

```
        \xdef\tqft@fullpath{%
          \tqft@fullpath
           .. controls +(0,-\tqft@val{cobordism height}/3) and +(0,-\tqft@val{cobordism heig
      }%
```

But for the edge path, just add the curve to the list.

```
        \xdef\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)}%
      }%
```

We add a coordinate at the midpoint of the curve.

```
      \xdef\tqft@clist{%
        \tqft@clist,-between incoming \kmo\space and \k/{(\cpos pt,-\tqft@val{cobordism heig
      }%
```

And add the centre to the list for available shifts.

```
      \xdef\tqft@alist{%
        \tqft@alist,-incoming boundary \k/{(\kmo * \tqft@val{boundary separation},0)}%
      }%
    }%
    \fi
      \fi
```

We're at the edge of the last incoming boundary component. What we do now depends on whether or not there are outgoing boundary components.

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

There are.

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

```
    \pgfmathsetmacro\xppos{%
```

```
      (\tqft@outgoing@end * (\tqft@val{outgoing boundary components} -1)
      + \tqft@val{offset}) * \tqft@val{boundary separation}
      + \tqft@outgoing@dir * \tqft@val{circle x radius}}%
      \pgfmathsetmacro\tqft@ht{abs(\tqft@val{incoming boundary components} - \tqft@val{outgo
      \pgfmathsetmacro\tqft@ht{1/3 + 2/3*\tqft@ht/(\tqft@ht + 1)}%
      \xdef\tqft@fullpath{%
        \tqft@fullpath
        .. controls +(0,-\tqft@ht*\tqft@val{cobordism height}) and +(0,\tqft@ht*\tqft@val{co
\tqft@val{cobordism height})
      }%
```

and the edge path.

```
      \xdef\tqft@blist{%
        \tqft@blist,between last incoming and last outgoing/incoming and outgoing/{%
        (\tqft@val{incoming boundary components} * \tqft@val{boundary separation} + \tqft@va
        .. controls +(0,-\tqft@ht*\tqft@val{cobordism height}) and +(0,\tqft@ht*\tqft@val{co
\tqft@val{cobordism height})}%
      }%
```

In addition, we add a coordinate at the midpoint.

```
      \pgfmathsetmacro\xppos{(\xppos + (\tqft@val{incoming boundary components} -
1) * \tqft@val{boundary separation} +\tqft@val{circle x radius})/2}%
      \xdef\tqft@clist{%
        \tqft@clist,-between last incoming and last outgoing/{(\xppos pt,-
\tqft@val{cobordism height}/2)}%
      }%
      \else
```

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.

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

Same for the edge path.

```
      \xdef\tqft@blist{%
        \tqft@blist,between first and last incoming/incoming and outgoing/{%
        (\tqft@val{incoming boundary components} * \tqft@val{boundary separation} + \tqft@va
        .. controls +(0,-\tqft@ht*\tqft@val{cobordism height}) and +(0,-
\tqft@ht*\tqft@val{cobordism height}) .. (-\tqft@val{circle x radius},0)}
      }%
```

Add a coordinate at the midpoint.

```
      \pgfmathsetmacro\xppos{(\tqft@val{incoming boundary components} -1) * \tqft@val{bounda
      \xdef\tqft@clist{%
        \tqft@clist,-between first and last incoming/{(\xppos pt,-\tqft@ht*\tqft@val{cobordi
between first incoming and last incoming/{(\xppos pt,-\tqft@ht*\tqft@val{cobordism height}*3
      }%
      \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.

```
      \ifnum\tqft@val{outgoing boundary components}>0\relax
        \pgfmathsetmacro\xppos{( \tqft@outgoing@end * (\tqft@val{outgoing boundary component
   1)+\tqft@val{offset}) * \tqft@val{boundary separation} + \tqft@outgoing@dir * \tqft@val{circ
```

Add a move to the full path,

```
      \xdef\tqft@fullpath{%
        \tqft@fullpath
        (\xppos pt, -\tqft@val{cobordism height})
      }%
      \fi
      \fi
```

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 .

```
    \ifnum\tqft@val{outgoing boundary components}>0\relax
    \pgfmathsetmacro\xppos{%
      ( \tqft@outgoing@end * (\tqft@val{outgoing boundary components} -1)
      + \tqft@val{offset}) * \tqft@val{boundary separation}
      - \tqft@outgoing@dir * \tqft@val{circle x radius}}%
```

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

```
      \xdef\tqft@fullpath{%
        \tqft@fullpath
        arc[end angle=\pgf@tqft@upper180, start angle=0, x radius=\tqft@outgoing@dir * \tqft
      }%
```

And add the centre to the list for available shifts.

```
      \xdef\tqft@alist{%
        \tqft@alist,-outgoing boundary \tqft@val{outgoing boundary components}/{(\xppos pt +
   \tqft@val{cobordism height})},-outgoing boundary/{(\tqft@val{offset}*\tqft@val{boundary sepa
   \tqft@val{cobordism height})}%
      }%
```

Do we have more than one boundary component?

```
      \ifnum\tqft@val{outgoing boundary components}>1\relax
```

Yes, so add a curve and arc for each.

```
    \foreach \k [evaluate=\k as \ok using int(\tqft@outgoing@end * (\tqft@val{outgoing bound
      \edef\tqft@temp{\noexpand\pgfutil@in@{,\ok,}{,\tqft@val{skip outgoing boundary compone
      \tqft@temp
      \ifpgfutil@in@
      \xdef\tqft@cobdrylist{\tqft@cobdrylist,\k}
      \else
      \xdef\tqft@obdrylist{\tqft@obdrylist,\k}
      \fi
    }
    \ifx\tqft@obdrylist\pgfutil@gobble
    \else
    \foreach \k [
      remember=\k as \kmo (initially 1),
        evaluate=\k as \xpos using ( \tqft@outgoing@end * (\tqft@val{outgoing boundary compo
    ] in \tqft@obdrylist {
      \pgfmathsetmacro\xppos{(\tqft@outgoing@end * (\tqft@val{outgoing boundary components}
        \pgfmathsetmacro\cpos{(\xpos + \xppos)/2}
```

```
        \pgfmathsetmacro\nk{int(\tqft@val{outgoing boundary components} - \k + 1)}
        \pgfmathsetmacro\nkpo{int(\tqft@val{outgoing boundary components} - \kmo + 1)}
```

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
      }%
```

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
      }%
```

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})}%
      }%
      }%
    \fi
    \fi
```

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{co
    \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@va
    \tqft@val{cobordism height})
        .. controls +(0,\tqft@ht*\tqft@val{cobordism height}) and +(0,-\tqft@ht*\tqft@val{co
    \tqft@val{circle x radius},0)}%
      }%
```

Add a coordinate at the midpoint.

```
      \xdef\tqft@clist{%
```

```
      \tqft@clist,-between first incoming and first outgoing/{({ ( (1 - \tqft@outgoing@end
\tqft@val{cobordism height}/2)}%
    }%
    \else
```

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

```
    \pgfmathsetmacro\xppos{(\tqft@val{outgoing boundary components} -1+\tqft@val{offset})
    \pgfmathsetmacro\tqft@ht{1/3 + 2/3*(\tqft@val{outgoing boundary components} - 1)/\tqft
```

Full path.

```
    \xdef\tqft@fullpath{%
      \tqft@fullpath
      .. controls +(0,\tqft@ht*\tqft@val{cobordism height}) and +(0,\tqft@ht*\tqft@val{cob
\tqft@val{cobordism height})
    }%
```

Edge path.

```
    \xdef\tqft@blist{%
      \tqft@blist,between first and last outgoing/incoming and outgoing/{%
   (\tqft@val{offset} * \tqft@val{boundary separation} - \tqft@val{circle x radius},-
\tqft@val{cobordism height})
      .. controls +(0,\tqft@ht*\tqft@val{cobordism height}) and +(0,\tqft@ht*\tqft@val{cob
\tqft@val{cobordism height})}%
    }%
```

Add a coordinate at the midpoint.

```
    \pgfmathsetmacro\xppos{(\tqft@val{outgoing boundary components}/2 + \tqft@val{offset}
1/2) * \tqft@val{boundary separation}}%
    \pgfmathsetmacro\tqft@ht{1 -\tqft@ht*3/4}%
    \xdef\tqft@clist{%
      \tqft@clist,-between first and last outgoing/{(\xppos pt,-\tqft@ht*\tqft@val{cobordi
between first outgoing and last outgoing/{(\xppos pt,-\tqft@ht*\tqft@val{cobordism height})}
    }%
    \fi
    \fi
```

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.

```
    \pgfmathsetmacro\xpos{%
      (
      \tqft@val{outgoing boundary components} > 0 ?
      (
      \tqft@val{incoming boundary components} > 0 ?
      min(0,\tqft@val{offset}) : \tqft@val{offset}
      ) : 0
      )
      *\tqft@val{boundary separation} - 2*\tqft@val{circle x radius}}%
    \xdef\tqft@gclip{(\xpos pt,2*\tqft@val{circle y radius}) rectangle }%
```

Now the bottom right.

```
    \pgfmathsetmacro\xpos{%
      ((
      \tqft@val{outgoing boundary components} > 0 ?
      (
```

```
      \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}}%
```

Together, these make a rectangle.

```
    \xdef\tqft@gclip{\tqft@gclip (\xpos pt,-\tqft@val{cobordism height} - 2*\tqft@val{circ
```

Are there any holes?

```
    \ifnum\tqft@val{genus}>0\relax
```

Yes, so first we need to figure out where to place them. We work out the left-hand edge of the cobordism.

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

Work out the height that the holes should be punched at.

```
    \pgfmathsetmacro\ypos{%
      (
      \tqft@val{outgoing boundary components} > 0 ?
      (
      \tqft@val{incoming boundary components} > 0 ?
      -\tqft@val{cobordism height}/2 : -1 + \tqft@val{cobordism height}/3
      ) : - \tqft@val{cobordism height}/3
      )}%
```

Start our clip path at this point

```
    \xdef\tqft@gclip{%
      \tqft@gclip
      (\xpos pt,\ypos pt)
    }%
```

Now work out the width of the cobordism, in units of circle half-widths. This may not be very accurate if there aren't any boundary components of a given type.

```
    \pgfmathsetmacro\gsize{%
      ((
      \tqft@val{outgoing boundary components} > 0 ?
      (
      \tqft@val{incoming boundary components} > 0 ?
      (\tqft@val{incoming boundary components} + \tqft@val{outgoing boundary components})/
      ) : \tqft@val{incoming 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, `\gscale` holds the overall scale factor and `\gxscale` and `gyscale` are the resulting horizontal and vertical measurements. The baseline is the size of the boundary circles.

```
      \pgfmathsetmacro\gscale{min(1,\gsize/(3*\tqft@val{genus}+1))}%
      \pgfmathsetmacro\gyscale{\tqft@val{circle y radius}*\gscale*.707}%
      \pgfmathsetmacro\gxscale{\tqft@val{circle x radius}*\gscale}%
```

Each hole should take up 2 half widths, modulo scaling, so the total width used by the holes is $2gs$ leaving $w - 2gs$ left for the gaps which is divided in to $g + 1$ lots.

```
      \pgfmathsetmacro\gsep{((\gsize - 2*\tqft@val{genus}*\gscale)/(\tqft@val{genus} + 1)*\t
```

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

```
      \xdef\tqft@gclip{%
        \tqft@gclip
        ++(\gsep/2 pt,0)
      }%
```

Some useful quantities.

```
      \pgfmathsetmacro\omrstwo{1 - 1/sqrt(2)}%
      \pgfmathsetmacro\sqrtwo{sqrt(2)}%
```

Now we iterate over the holes.

```
      \foreach[
        evaluate=\k as \kmo using int(2 * \k-1)
      ] \k in {1,...,\tqft@val{genus}} {
```

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

```
        \xdef\tqft@gclip{%
          \tqft@gclip
```

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

```
          ++(\gsep/2 pt + \omrstwo*\gxscale pt,0)
```

Now curve up over the hole,

```
          .. controls +(\gxscale*\sqrtwo/3 pt,4/3*\gyscale pt) and +(-\gxscale*\sqrtwo/3 pt,
          .. ++(\sqrtwo*\gxscale pt,0)
```

and return on the underside.

```
          .. controls +(-\gxscale*\sqrtwo/3 pt,-4/3*\gyscale pt) and +(\gxscale*\sqrtwo/3 pt
    4/3*\gyscale pt)
          .. ++(-\sqrtwo*\gxscale pt,0)
```

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

```
          ++(2*\gxscale pt -\omrstwo*\gxscale pt + \gsep/2 pt,0)
        }
```

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 acount the `view from` direction.

```
        \xdef\tqft@glist{%
          \tqft@glist,%
            hole \k/lower/{%
```

Move to the starting point of the smaller curve and add that.

```
              (\xpos pt + \k * \gsep pt +  \kmo * \gxscale pt + \gxscale pt -
    \omrstwo*\gxscale pt,\ypos pt)
          .. controls +(-\gxscale pt*\sqrtwo/3,\pgf@tqft@upper4/3*\gyscale pt) and +(\gxscal
          .. ++(-\sqrtwo*\gxscale pt,0)},%
% Move to the left-hand corner of the path, with the upper or lower chosen by the \verb+view
          hole \k/upper/{(\xpos pt + \k * \gsep pt +  \kmo * \gxscale pt - \gxscale pt,\ypos
```

Add the larger of the two curves.

```
          .. controls +(\gxscale pt*2/3,\pgf@tqft@lower8/3*\gyscale pt) and +(-
    \gxscale pt*2/3,\pgf@tqft@lower8/3*\gyscale pt)
          .. ++(2*\gxscale pt,0)}%
        }%
```

Add a coordinate at the centre of the hole.

```
        \xdef\tqft@clist{%
          \tqft@clist,-hole \k/{(\xpos pt + \k * \gsep pt +  \kmo * \gxscale pt,\ypos pt)}%
        }%
      }%
      \fi
```

Now we start to lay out the cobordism Were we given a shift? If so, shift.

```
\gdef\tqft@shift{(0,0)}%
\edef\tqft@anchor{\tqft@val{anchor}}%
\expandafter\pgfutil@in@\expandafter{\expandafter,\expandafter}\expandafter{\tqft@anchor}%
\ifpgfutil@in@
\expandafter\tqft@split\tqft@anchor\relax
\else
\edef\tqft@anchor{-\tqft@val{anchor}}%
\xdef\tqft@alist{\tqft@clist,\tqft@alist}%
\foreach \anchor/\coord in \tqft@alist
{
  \ifx\anchor\tqft@anchor\relax
  \global\let\tqft@shift\coord
  \fi
}%
\fi
\tikz@scan@one@point\pgfutil@firstofone\tqft@shift\relax
\begin{scope}[shift={(-\pgf@x,-\pgf@y)}]
```

At each incoming boundary component we place an elliptical node of the right size.

```
\ifnum\tqft@val{incoming boundary components}>0\relax
\ifx\tqft@ibdrylist\pgfutil@gobble
\xdef\tqft@ibdrylist{1}
\else
\xdef\tqft@ibdrylist{1,\tqft@ibdrylist}
\fi
      \foreach[evaluate=\k as \xpos using (\k-1)*\tqft@val{boundary separation}] \k in \tqft
  \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,0)},
          name=-incoming boundary \k,
          /tikz/tqft/every boundary component/.try,
          /tikz/tqft/every incoming boundary component/.try,
          /tikz/tqft/incoming boundary component \k/.try
        ];
```

```
      }%
\ifx\tqft@cibdrylist\pgfutil@gobble
\else
      \foreach[evaluate=\k as \xpos using (\k-1)*\tqft@val{boundary separation}] \k in \tqft
  \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,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,
        ];
}%
\fi
```

Add an alias for the first.

```
      \path node also[pic alias=-incoming boundary] (-incoming boundary 1);
      \fi
```

Same for the outgoing boundary components.

```
      \ifnum\tqft@val{outgoing boundary components}>0\relax
\ifx\tqft@obdrylist\pgfutil@gobble
\xdef\tqft@obdrylist{1}
\else
\xdef\tqft@obdrylist{1,\tqft@obdrylist}
\fi
\foreach \k [evaluate=\k as \ok using int(\tqft@outgoing@end * (\tqft@val{outgoing boundary
  \xdef\tqft@robdrylist{\tqft@robdrylist,\ok}
}
      \foreach[
        evaluate=\k as \xpos 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={(\xpos pt,-\tqft@val{cobordism height})},
          name=-outgoing boundary \k,
          /tikz/tqft/every boundary component/.try,
          /tikz/tqft/every outgoing boundary component/.try,
          /tikz/tqft/outgoing boundary component \k/.try
        ];
}%
\ifx\tqft@cobdrylist\pgfutil@gobble
\else
```

```
\foreach \k[evaluate=\k as \ok using int(\tqft@outgoing@end * (\tqft@val{outgoing boundary c
 in \tqft@cobdrylist {
  \xdef\tqft@rcobdrylist{\tqft@rcobdrylist,\ok}
}
      \foreach[
        evaluate=\k as \xpos using (\k-1+\tqft@val{offset})*\tqft@val{boundary separation}
      ] \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
        ];
      }%
\fi
```

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
        \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 a
      }%
      \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
      ] \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 ang
        }%
      \fi
```

Full outer path, clipped against the genus holes in case it is filled.

```
      \begin{scope}
      \path[overlay,clip] \tqft@gclip;
      \path[
```

```
        /tikz/tqft/cobordism/.try,
        pic actions,
        /tikz/tqft/cobordism outer path/.try,
      ] \tqft@fullpath;
      \end{scope}
```

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.

```
\ifx\tqft@glist\pgfutil@gobble
\else
\foreach \tqft@gstyle/\tqft@gside/\tqft@gpath in \tqft@glist {
      \path[
        /tikz/tqft/cobordism/.try,
        pic actions,
        fill=none,
        shade=none,
        /tikz/tqft/cobordism edge/.try,
        /tikz/tqft/genus style/.try,
        /tikz/tqft/genus \tqft@gside/.try,
        /tikz/tqft/\tqft@gstyle/.try,
        /tikz/tqft/\tqft@gstyle\space\tqft@gside/.try,
      ] \tqft@gpath;
}
\fi
```

Now we redraw the non-boundary paths.

```
\ifx\tqft@blist\pgfutil@gobble
\else
\foreach \tqft@bstyle/\tqft@btype/\tqft@bpath in \tqft@blist {
      \path[
        /tikz/tqft/cobordism edge/.try,
        /tikz/tqft/cobordism outer edge/.try,
        /tikz/tqft/between \tqft@btype/.try,
        /tikz/tqft/\tqft@bstyle/.try,
      ] \tqft@bpath;
}
\fi
```

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

```
      \ifx\tqft@clist\pgfutil@gobble
      \else
      \foreach \name/\coord in \tqft@clist {
        \path \coord node[coordinate,node contents={},name=\name];
      }
      \fi
```

The last task is to draw the upper paths of the boundary components. First, incoming.

```
      \ifnum\tqft@val{incoming boundary components}>0\relax
      \foreach[evaluate=\k as \xpos using (\k-1)*\tqft@val{boundary separation}] \k in \tqft
```

```
        \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 a
      }
      \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 ang
      }
      \fi
\end{scope}
```

We're done! Phew.

```
  }
}
```