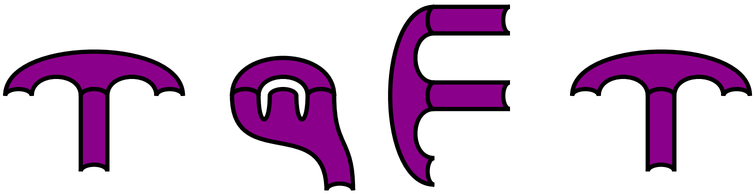
The **tqft** Ti*k*Z Library: Documentation

Andrew Stacey
loopspace@mathforge.org

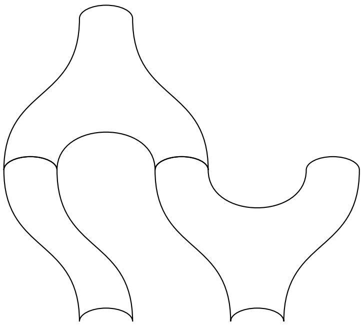
v2.3 from 2024/05/31



1 Introduction

This package defines some Ti*k*Z/PGF picture shapes that can be used to construct the diagrams common in Topological Quantum Field Theory (TQFT). An example follows:

```
\begin{tikzpicture}[tqft/cobordism/.style={draw}]
\pic[tqft/pair of pants,name=a];
\pic[tqft/cylinder to next,anchor=incoming boundary
    1,name=c,at=(a-outgoing boundary 1)];
\pic[tqft/reverse pair of pants, anchor=incoming
    boundary 1,at=(a-outgoing boundary 2)];
\end{tikzpicture}
```



2 Contents

3 History

This is the second version of the TQFT package. The first version used nodes to draw the shapes. However, with the advent of Ti*k*Z3.0 came the ability to define subdrawings called `pics` which were a bit like nodes but were geared more towards drawings than containers for text. This seems a much more suitable mechanism for drawing these diagrams and so the package has been rewritten to make use of this new facility.

The second version is designed to be similar to the first, but with some improvements. The original version was distributed as a `.sty` file and so is loaded using `\usepackage{tqft}`. The newer version is a Ti*k*Z library and so is loaded using `\usetikzlibrary{tqft}`. This makes it possible to use both in the same document. This is not recommended, but an attempt has been made to make it possible to switch between the two methods (mainly to stop this documentation file complaining every time I compile it). This hasn't been extensively tested so use with caution. To make the switch use the key `/tikz/tqft/use nodes=<true|false>`. By default, the one loaded last should be in effect at the start of the document.

4 Keys

Before giving any details, a word is in order about the keys involved in this package. There are many options and keys that can be set via the `\pgfkeys` system (which is used for setting options in Ti*k*Z). Such keys live in a "directory" but often that can be omitted. For example, in the Ti*k*Z command `\draw[red] (0,0) -- (1,0);` the key `red` is actually in the "directory" `/tikz` but it is not necessary to specify that as it is assumed. Defining a "directory" helps separate keys and ensure that there is no conflict. The keys in this library are (mostly) defined in the directories `/tikz/tqft` (the newer version) and `/pgf/tqft` (the old version).

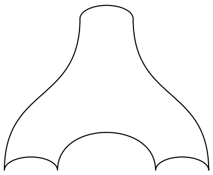
Invoking `/tikz/tqft` itself sets the "current directory" to whichever directory is right for the current version in force and so all subsequent keys do not need prefixing. Moreover, any unknown keys are passed on to the `/tikz` directory so there is (or should be!) no harm in mixing `tqft` specific keys with ordinary Ti*k*Z keys. Some examples take advantage of this switch so when copying and modifying examples from this document, it is important to remember that the first `tqft` specific key needs an explicit `tqft/` prefix.

5 Version 2: Pics via a TikZ Library

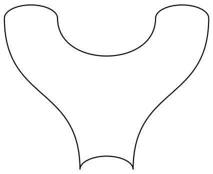
5.1 The Shapes

There is only one picture shape: `cobordism`. This is a cobordism between a number of incoming circles and a number of outgoing circles, where the numbers of boundary components can be specified as options to the shape. There are certain common shapes that are predefined as aliases to the main shape with specified boundaries. The list of predefined shapes follows. The names are all in the `tqft` family, but an alias is made so that `tqft <shape>` will work without any further qualification.

1. `pair of pants`

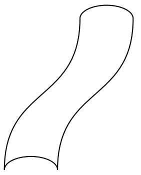
   

2. `reverse pair of pants`

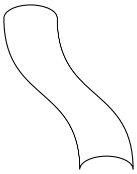
3. `cylinder to prior`

   This is a cylinder that has been skewed to one side, thus following the same path as the `pair of pants` cobordism but with only one outgoing boundary component. The name `to prior` is because it goes towards the lower-numbered component on the `pair of pants`.

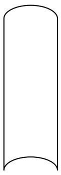
4. `cylinder to next`

This is a cylinder that has been skewed to one side, thus following the same path as the `pair of pants` cobordism but with only one outgoing boundary component. The name `to next` is because it goes towards the higher-numbered component on the `pair of pants`.



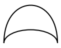
5. `cylinder`

   This is a straight cylinder.



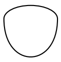
6. `cap`

   This is a cap.



7. `cup`

   This is a cup (an upside-down cap).



The general shape is controlled by the following keys:

`view from`	• To get a simulated 3D effect, the cobordism is drawn as if viewed from a slight angle. The value of this key determines whether the cobordism is viewed from the direction of the incoming boundary components or the outgoing ones. This key can take the values `incoming` and `outgoing`. The default is `outgoing`.
`cobordism height`	• This is the height of the cobordism ("height" interpreted in its own internal coordinate system). With no offset (q.v.), this would be the distance between the centres of the first incoming and first outgoing boundary components.
`boundary separation`	• This is the distance between the centres of the boundary components of the same type.
`circle x radius`	• This is the half-width of the boundary circles.
`circle y radius`	• This is the half-depth of the boundary circles ("depth" since, in the internal coordinate system, this corresponds to the z-axis out of the page).
`incoming boundary components`	• The number of incoming boundary components (can be zero).

- `skip incoming boundary components` A list of incoming boundary components to be skipped.
- `outgoing boundary components` The number of outgoing boundary components (can be zero).
- `skip outgoing boundary components` A list of outgoing boundary components to be skipped.
- `offset` This offsets the first outgoing boundary component horizontally relative to the first incoming boundary component. It is a dimensionless number (not necessarily an integer) and is interpreted so that a value of 1 aligns the first outgoing boundary component with the second incoming boundary component.
- `genus` This defines the number of holes in the shape. These are spread out in a horizontal line in the middle of the shape.
- `twisted` This is a boolean that, if set, makes the cobordism *twisted* in that the edges cross as they pass from the incoming to outgoing boundaries. This probably won't look good with a non-zero genus, but there's nothing stopping you doing it.

5.2 Styling

There are various options for styling the diagrams. To understand how they work, it is important to know the order in which a cobordism is drawn and how many pieces it decomposes into. This is the following list, with the corresponding keys:

1. The boundary circles are drawn. These are actually elliptical nodes (and thus can be individually styled). Applicable styles:
    - `every boundary component`,
    - `every incoming boundary component`,
      or `every outgoing boundary component`,
    - `incoming boundary component <n>`,
      or `outgoing boundary component <n>`
2. The lower edges of the boundary circles are redrawn. These are individual arcs.
    - `every lower boundary component`,
    - `every incoming lower boundary component`,
      or `every outgoing lower boundary component`,
    - `incoming lower boundary component <n>`,
      or `outgoing lower boundary component <n>`
3. The full edge of the cobordism is drawn. This is a closed path so can be sensibly filled. It is clipped against a path defined by the genus of the cobordism which results in holes if it is filled (or shaded or anything else that goes in to the interior).
    - `cobordism`,

- also, any actions specified on the `pic` are applied here (specifically, the `pic actions` key is invoked; see the Ti*k*Z manual for full details on this),
- `cobordism outer path`.

4. Any holes specified by the genus are now drawn. These are styled to give the 3D impression, and this follows the direction specified by the `view from` key. The paths are split so that each can be individually styled.

   - `cobordism`, this style is applied because the curves defined by the genus can be thought of as part of the edge of the cobordism shape.
   - `pic actions`, for the same reason as above.

     However, following this key then `fill=none,shade=none` is issued. This is because even if the main shape is filled or shaded, the paths drawing the holes should almost certainly not be.
   - `cobordism edge`, the same logic applies to the edge path (q.v.).
   - `genus style`,
   - `genus upper` or `genus lower`,
   - `hole <n>`,
   - `hole <n> upper` or `hole <n> lower`.

5. The non-boundary edge of the cobordism is redrawn. This is split in to pieces to allow for individual styling.

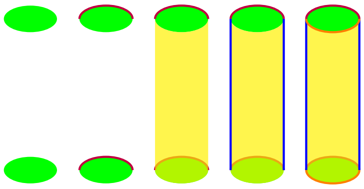
   - `cobordism edge`,
   - `cobordism outer edge`,
   - `between incoming`, or `between outgoing`, or `between incoming and outgoing`. The latter is for the two sides, but note that if the cobordism has no incoming or no outgoing components then it also applies to the "over the top" edge.
   - `<anchor>`, where the `<anchor>` is the name of the anchor that lies on the midpoint of the curve, so it will be one of:
     - `between incoming <n> and <n+1>`,
     - `between outgoing <n> and <n+1>`,
     - `between first incoming and first outgoing`,
     - `between last incoming and last outgoing`,
     - `between first and last incoming`,
     - `between first and last outgoing`.

6. The upper edges of the boundary circles are redrawn. These are arcs.

   - `every upper boundary component`,
   - `every incoming upper boundary component`,
     or `every outgoing upper boundary component`,
   - `incoming upper boundary component <n>`,
     or `outgoing upper boundary component <n>`

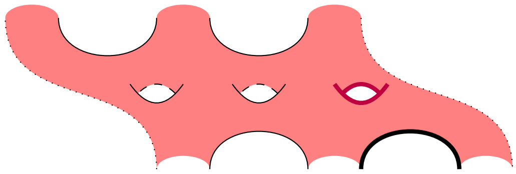
The fact that there are so many is to allow different style to be applied to different pieces and to give as much control as possible, whilst still making it fairly straightforward to draw a simple cobordism. The duplication of paths is to allow certain composite pieces to be *filled*. Here is a progressively built up cobordism.

```
\begin{tikzpicture}[tqft/view from=incoming]
\begin{scope}[tqft/every boundary
    component/.style={fill=green, fill opacity=1}]
\pic[tqft/cylinder,at={(1,0)}];
\begin{scope}[tqft/every lower boundary
    component/.style={draw=purple,thick}]
\pic[tqft/cylinder,at={(2,0)}];
\begin{scope}[tqft/cobordism/.style={fill=yellow,fill
    opacity=.7}]
\pic[tqft/cylinder,at={(3,0)}];
\begin{scope}[tqft/cobordism
    edge/.style={draw,thick,blue}]
\pic[tqft/cylinder,fill=yellow,fill
    opacity=.7,at={(4,0)}];
\begin{scope}[tqft/every upper boundary
    component/.style={draw,thick,orange}]
\pic[tqft/cylinder,fill=yellow,fill
    opacity=.7,at={(5,0)}];
\end{scope}
\end{scope}
\end{scope}
\end{scope}
\end{scope}
\end{tikzpicture}
```



Here's an example with lots of styling.

```
\begin{tikzpicture}
\pic[
  tqft,
  incoming boundary components=3,
  outgoing boundary components=3,
  offset=1,
  genus=3,
  hole 3/.style={ultra thick, purple,solid},
  genus lower/.style={dashed},
  fill=red!50,
  cobordism edge/.style={draw},
  between incoming and outgoing/.style={dotted},
  between outgoing 2 and 3/.style={ultra thick},
];
\end{tikzpicture}
```



5.3 Anchors

The cobordism is a `pic` so does not have any native anchors. Nevertheless, a multitude of coordinates are defined that simulate the anchors associated with nodes. There is also support for specifying the shape to be located relative to a particular anchor.

The `\pic` should be named via the `name=<prefix>` key, whereupon the anchors are prefixed by this value. The pseudo-anchors defined have the naming convention `<prefix>-<anchor name>` (at the moment, it doesn't check to see if the `name` key has been specified so if it isn't then the pseudo-anchors are still defined but with an empty prefix). They are:

- `incoming boundary <n>`, these are in fact elliptical nodes and so also define actual anchors.
- `incoming boundary` is an alias for `incoming boundary 1`.
- `outgoing boundary <n>`, same.
- `outgoing boundary` is an alias for `outgoing boundary 1`.
- `between incoming <n> and <n+1>`, this lies on the midpoint of the curve between successive boundary components.
- `between outgoing <n> and <n+1>`, this lies on the midpoint of the curve between successive boundary components.
- `between first incoming and first outgoing` is on the edge between the first incoming and first outgoing boundary components; note that this is only defined if there are both incoming and outgoing boundary components.

- `between last incoming and last outgoing` is on the edge between the last incoming and last outgoing boundary components; note that this is only defined if there are both incoming and outgoing boundary components.
- `between first and last incoming`; this is only defined if there are no outgoing components.

  This is also available via the alias `between first incoming and last incoming`.
- `between first and last outgoing`; this is only defined if there are no incoming components.

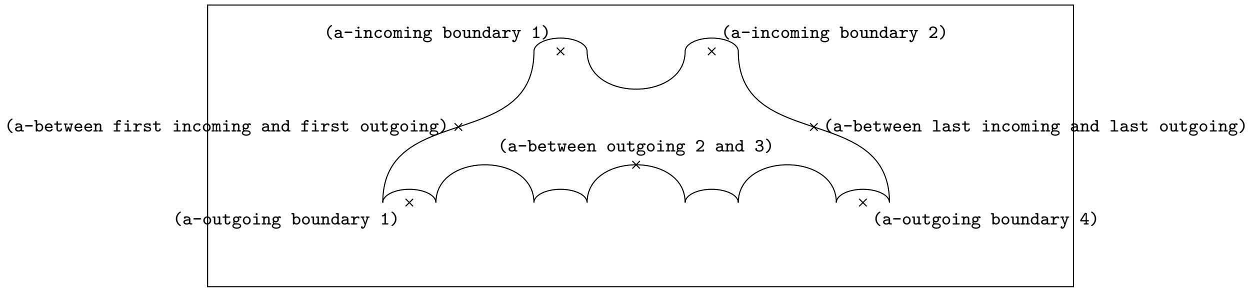
  This is also available via the alias `between first outgoing and last outgoing`.
- `hole <n>`; if the genus is non-zero, this points to the centre of the nth hole.

To place the shape relative to an anchor, use the `tqft/anchor` key. The argument should be just the name of the anchor without the leading `<prefix>-`. The `anchor` key can also take another type of argument. If its argument is of the form `(x,y)` then this is taken as a pseudo-coordinate[1]. It is interpreted as being x boundary components across and y times the cobordism height down. However, if an `offset` is specified then the resulting x value is shifted so that if $y < 0$ then $(1, y)$ is in line with the first incoming boundary component and if $y > 1$ then $(1, y)$ is in line with the first outgoing boundary component. If $0 < y < 1$ then $(1, y)$ linearly interpolates between the first incoming and first outgoing boundary components. Thus $(1, 0)$ is the first incoming boundary component, $(1, 1)$ the first outgoing boundary component, $(0, 0)$ is one unit to the left of the first incoming, and $(1, 2)$ one unit below the first outgoing. Note that the picture is shifted to put this point at the current coordinate.

```
\begin{tikzpicture}
\pic[tqft, incoming boundary components=2,outgoing
    boundary components=4,offset=-1,draw,name=a];
\foreach \anchor/\placement in
{
between first incoming and first outgoing/left,
between last incoming and last outgoing/right,
between outgoing 2 and 3/above,
incoming boundary 1/above left,
incoming boundary 2/above right,
outgoing boundary 1/below left,
outgoing boundary 4/below right}
\draw[overlay,shift=(a-\anchor)] plot[mark=x]
    coordinates{(0,0)} node[\placement]
    {\scriptsize\texttt{(a-\anchor)}};
\path (a-incoming boundary) +(0,.5) (a-outgoing
    boundary) +(0,-1);
\end{tikzpicture}
```

---

[1] Note that due to the presence of the comma, this type of argument must be protected by braces.


(a-incoming boundary 1)
(a-incoming boundary 2)
(a-between first incoming and first outgoing)
(a-between last incoming and last outgoing)
(a-between outgoing 2 and 3)
(a-outgoing boundary 1)
(a-outgoing boundary 4)

5.4 Notes

1. Like `node`s, `pic`s need the `transform shape` key to be set to take note of external transformations (other than shifts). The tqft pic uses nodes internally and those nodes have `transform shape` automatically set so this should *just work*.

2. There is an additional `every tqft` key which is run when the `tqft` key is invoked (which might be via some other key). This is better placed than the `every pic` key since that applies to a surrounding scope rather than to the `pic` itself.

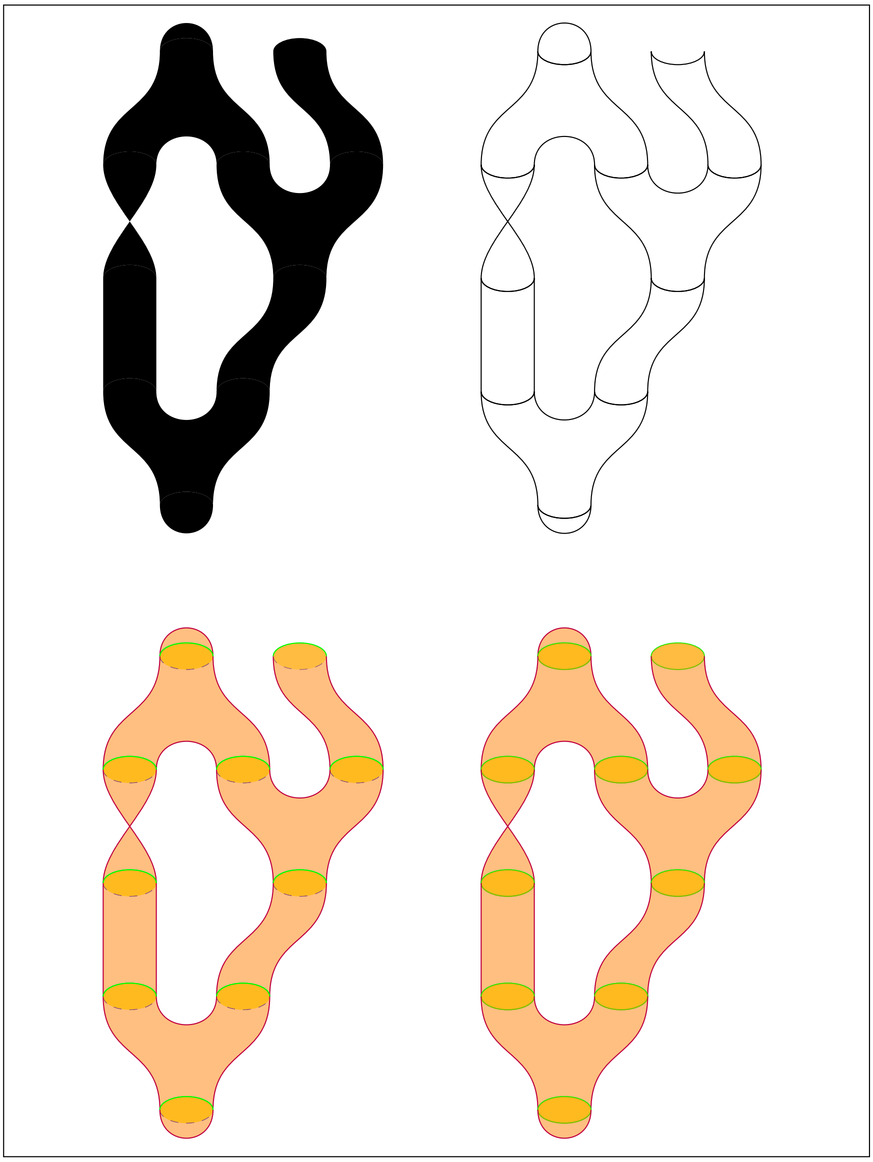
3. If the `tqft` key is invoked, either implicitly or explicitly, then the `pic type` is set to `cobordism`. This has the side effect that the invoking syntax has to be completely set by keys; so the `pic (<name>) at (<coord>) {<type>}` cannot be used. Rather, the `name` and `at` have to be specified by keys and the `type` omitted.

4. If upgrading from the previous version of TQFT, as well as shifting from a `node` to a `pic`, the following changes have been made in the implementation:

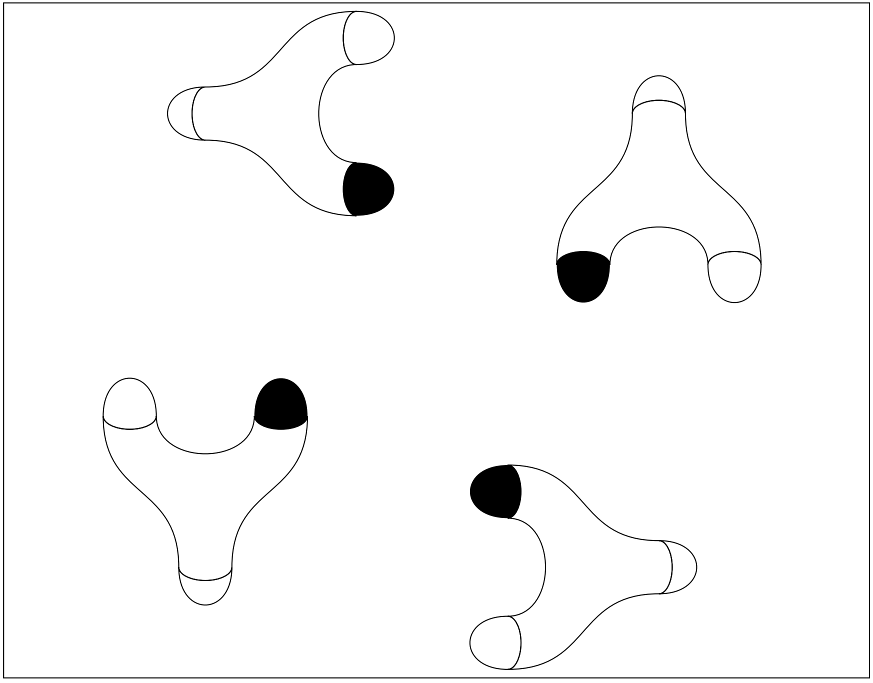
    - The `flow` key has not made it across to the new version. Use `transform shape` and apply your own transformation.
    - The `circle width` and `circle depth` are now `circle x radius` and `circle y radius` (the old names weren't correct anyway).
    - The bounding box is a little better, particularly for cobordisms with only one type of boundary component.

5.5 More Examples

```
\begin{tikzpicture}[tqft/cobordism
    height=1.5cm,tqft/boundary separation=1.5cm]
\foreach \coord/\style in {
{(0,0)}/{tqft/view from=outgoing, fill},
{(5,0)}/{tqft/view from=incoming,draw},
{(0,-8)}/{fill=orange, fill opacity=.5,tqft/every lower
    boundary component/.style={draw,blue, ultra
    thin,dashed},tqft/every upper boundary
    component/.style={draw,green},tqft/cobordism
    edge/.style={draw,purple},tqft/every boundary
    component/.style={fill=yellow}},
{(5,-8)}/{fill=orange, fill opacity=.5,tqft/cobordism
    edge/.style={draw,purple},tqft/every boundary
    component/.style={fill=yellow,draw=green}}
} {
  \begin{scope}[every tqft/.style/.expand once=\style]
\pic[tqft/cap,name=h,at=\coord];
\pic[tqft/pair of pants,anchor=incoming boundary
    1,name=a,at=(h-outgoing boundary 1)];
\pic[tqft/cylinder to
    next,anchor={(0,1)},name=d,at=(a-outgoing boundary
    2)];
\pic[tqft/reverse pair of pants,anchor=incoming boundary
    1,name=b,at=(a-outgoing boundary 2)];
\pic[tqft/cylinder to prior,anchor=incoming boundary
    1,name=c,at=(b-outgoing boundary 1)];
\pic[tqft/cylinder,twisted,anchor=incoming boundary
    1,name=e,at=(a-outgoing boundary 1)];
\pic[tqft/cylinder,anchor=incoming boundary
    1,name=f,at=(e-outgoing boundary 1)];
\pic[tqft/reverse pair of pants,anchor=incoming boundary
    1,name=g,at=(f-outgoing boundary 1)];
\pic[tqft/cup,anchor=incoming boundary
    1,name=i,at=(g-outgoing boundary 1)];
\end{scope}
}
\end{tikzpicture}
```

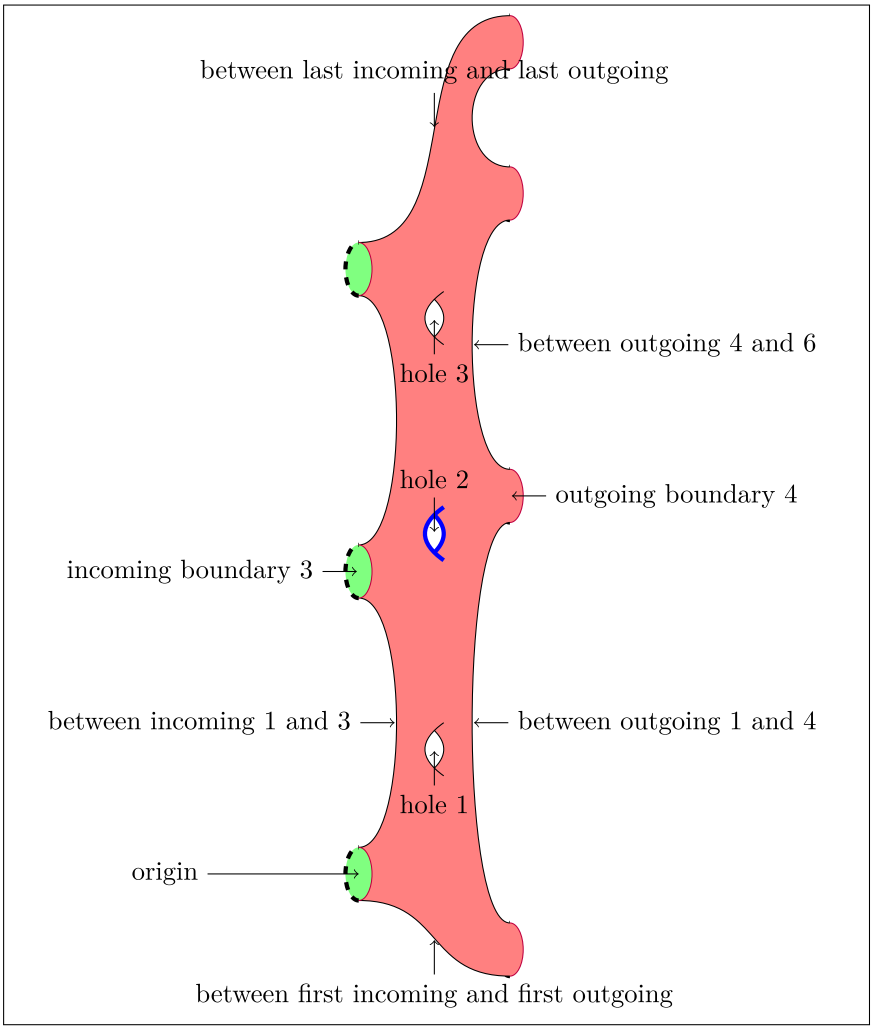


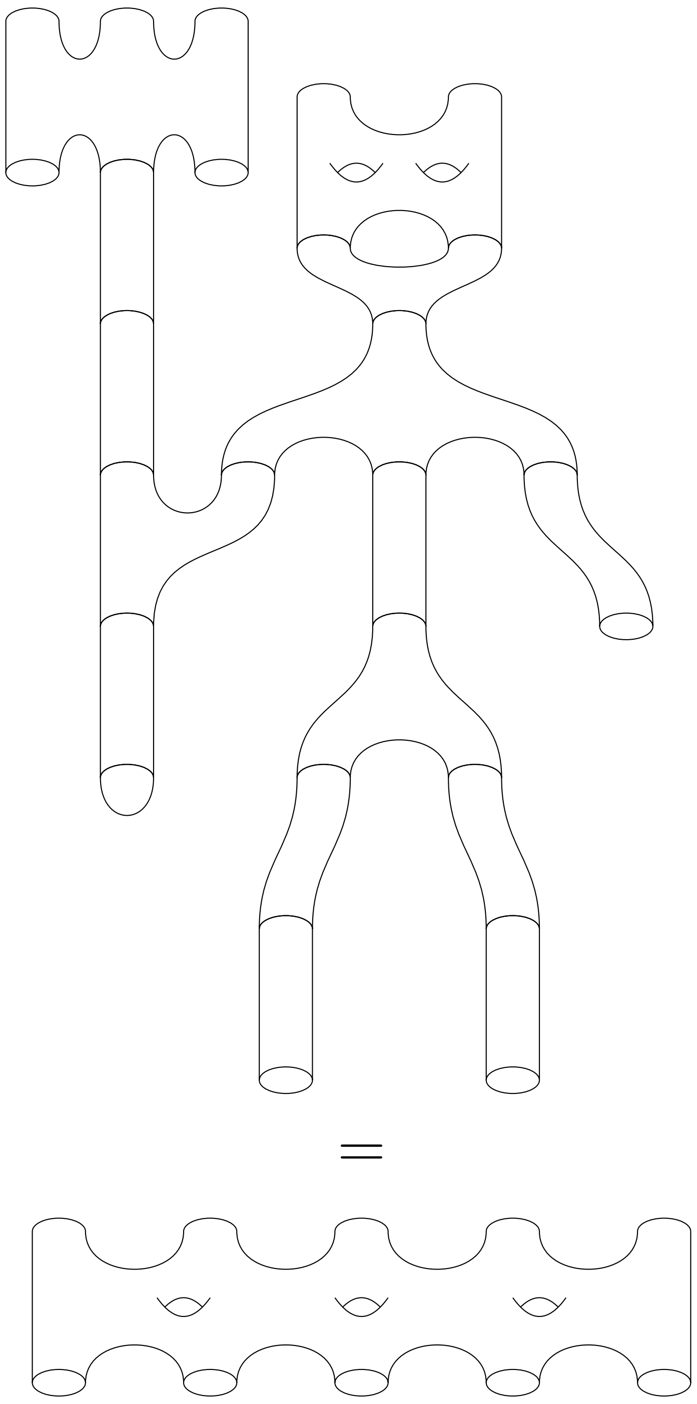
```
\begin{tikzpicture}[every tqft/.append style={transform
    shape}]
\foreach \ang in {0,90,180,270} {
\begin{scope}[rotate=\ang]
\pic[draw,tqft/pair of pants,name=a,at={(3,3)}];
\pic[draw,tqft/cap,anchor=outgoing boundary
    1,at=(a-incoming boundary 1)];
\pic[fill,tqft/cup,anchor=incoming boundary
    1,at=(a-outgoing boundary 1)];
\pic[draw,tqft/cup,anchor=incoming boundary
    1,at=(a-outgoing boundary 2)];
\end{scope}
}
\end{tikzpicture}
```



```
\begin{tikzpicture}[
  tqft,
  every outgoing boundary
      component/.style={fill=blue!50},
  outgoing boundary component
      3/.style={fill=none,draw=red},
  every incoming boundary
      component/.style={fill=green!50},
  every lower boundary component/.style={draw,ultra
      thick, dashed},
  every upper boundary component/.style={draw,purple},
  cobordism/.style={fill=red!50},
  cobordism edge/.style={draw},
  genus=3,
  hole 2/.style={ultra thick, blue},
  view from=incoming,
  anchor=between incoming 1 and 2
]
\pic[rotate=90,
  %every node/.style={transform shape},
  name=a,tqft,incoming boundary components=5,skip
      incoming boundary components={2,4},outgoing
      boundary components=7,skip outgoing boundary
      components={2,3,5},offset=-.5];

\begin{scope}[every pin edge/.style={<-}]
\foreach \anchor/\ang in {
  hole 1/-90,
  hole 2/90,
  hole 3/-90,
  incoming boundary 3/180,
  outgoing boundary 4/0,
  between last incoming and last outgoing/90,
  between first incoming and first outgoing/-90,
  between incoming 1 and 3/180,
  between outgoing 1 and 4/0,
  between outgoing 4 and 6/0
} {
  \node[pin=\ang:\anchor,at=(a-\anchor),inner sep=0pt]
      {};
}
\draw[<-] (0,0) -- ++(-2,0) node[left] {origin};
\end{scope}
\end{tikzpicture}
```


between last incoming and last outgoing
between outgoing 4 and 6
hole 3
hole 2
outgoing boundary 4
incoming boundary 3
between incoming 1 and 3
between outgoing 1 and 4
hole 1
origin
between first incoming and first outgoing



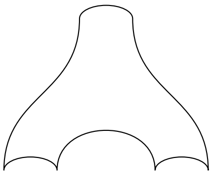
6 Version 1: Nodes via a Style File

As mentioned in the introduction, this is the original method of drawing cobordism diagrams using nodes and is no longer updated (though I will fix bugs if I can). If drawing a new diagram, use the pic method from Section 5.

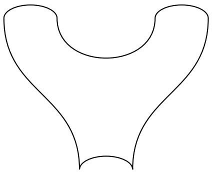
6.1 The Node Shapes

There are only two shapes, `tqft cobordism` and `tqft boundary circle`. The first, which is the main shape, is a cobordism between a number of incoming circles and a number of outgoing circles, where the numbers of boundary components can be specified as options to the shape. The second is just the boundary circle. It is used as a sub-node of the first to add extra anchors, but can be used by itself. There are certain common shapes that are predefined as aliases to the main shape with specified boundaries. The list of predefined shapes follows. The names are all in the `tqft` family, but an alias is made so that `tqft nodeshape` will work without any further qualification.

1. `pair of pants`

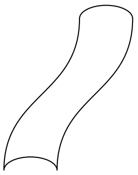
   

2. `reverse pair of pants`

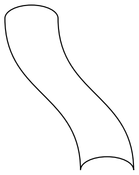
3. `cylinder to prior`

   This is a cylinder that has been skewed to one side, thus following the same path as the `pair of pants` cobordism but with only one outgoing boundary component. The name `to prior` is because it goes towards the lower-numbered component on the `pair of pants`.

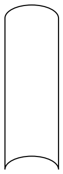
4. `cylinder to next`

   This is a cylinder that has been skewed to one side, thus following the same path as the `pair of pants` cobordism but with only one outgoing boundary component. The name `to next` is because it goes towards the higher-numbered component on the `pair of pants`.



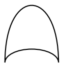
5. `cylinder`

   This is a straight cylinder.

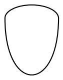
6. `cap`

   This is a cap.

7. `cup`

   This is a cup (an upside-down cap).

The general shape is controlled by the following keys:

`flow`
- A cobordism "flows" from its incoming to its outgoing boundaries. This key controls the direction of that flow. The shape is transformed so that the incoming-outgoing axis aligns with the argument. However, the transformation may be more than just a rotation as the shape is set up so that the numbering of the boundary components is always left-to-right or top-to-bottom (as appropriate). Currently, this key can take the values `north`, `south` (default), `east`, and `west`.

`view from`
- To get a simulated 3D effect, the cobordism is drawn as if viewed from a slight angle. The value of this key determines whether the cobordism is viewed from the direction of the incoming boundary components or the outgoing ones. This key can take the values `incoming` and `outgoing`. The default is `outgoing`.

`cobordism height`
- This is the height of the cobordism ("height" interpreted in its own internal coordinate system). With no offset (q.v.), this would be the distance between the centres of the first incoming and first outgoing boundary components.

`boundary separation`
- This is the distance between the centres of the boundary components of the same type.

`circle width`	This is the half-width of the boundary circles.
`circle depth`	This is the half-depth of the boundary circles ("depth" since, in the internal coordinate system, this corresponds to the z-axis out of the page).
`incoming boundary components`	The number of incoming boundary components (can be zero).
`outgoing boundary components`	The number of outgoing boundary components (can be zero).
`offset`	This offsets the first outgoing boundary component horizontally relative to the first incoming boundary component. It is a dimensionless number (not necessarily an integer) and is interpreted so that a value of 1 aligns the first outgoing boundary component with the second incoming boundary component.

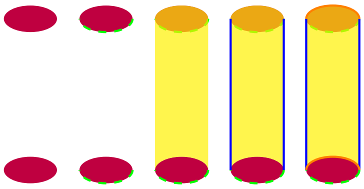
6.2 Styling

There are various options for styling the diagrams. To understand how they work, it is important to know the order in which a cobordism is drawn and how many pieces it decomposes into. This is the following list, with the corresponding key:

1. The boundary circles are drawn. `boundary style`
2. The lower edges of the boundary circles are redrawn. `boundary lower style`
3. The cobordism itself is drawn.
4. The non-boundary edge of the cobordism is redrawn. `cobordism style`
5. The upper edges of the boundary circles are redrawn. `boundary upper style`

The fact that there are so many is to allow different style to be applied to different pieces. The duplication is to allow certain composite pieces to be *filled*. All of these items can be styled separately. The style given to the node itself is passed on to the third item in that list, the cobordism itself. The styles of the others are controlled by a series of keys, each of should be a list of styles to be applied to that component. Not all options make sense, in particular only the first and third can be filled. (That is, the `fill` style is ignored on the other components.) Here is a progressively built up cobordism.

```
\begin{tikzpicture}
\begin{scope}[tqft/boundary style={fill=purple,fill
    opacity=1}]
\node[tqft/cylinder] at (1,0) {};
\begin{scope}[tqft/boundary lower
    style={draw,dashed,green,thick}]
\node[tqft/cylinder] at (2,0) {};
\begin{scope}
\node[tqft/cylinder,fill=yellow,fill opacity=.7] at
    (3,0) {};
\begin{scope}[tqft/cobordism style={draw,thick,blue}]
\node[tqft/cylinder,fill=yellow,fill opacity=.7] at
    (4,0) {};
\begin{scope}[tqft/boundary upper
    style={draw,thick,orange}]
\node[tqft/cylinder,fill=yellow,fill opacity=.7] at
    (5,0) {};
\end{scope}
\end{scope}
\end{scope}
\end{scope}
\end{scope}
\end{tikzpicture}
```



6.3 Anchors

As with all PGF node shapes, there are certain anchors defined by the `tqft` shape. These are the `center` (and `centre`) anchors and the `incoming boundary n`, `outgoing boundary n` anchors. The positioning of the `center` anchor is slightly unusual in that if there are no, say, incoming boundary components then the centre anchor is still at the same height above the outgoing boundary components as if there were incoming boundary components. The reason for this is two-fold: computing the *actual* centre of the shape in such circumstances would be tricky, and when aligning these shapes it is more useful to have the anchors consistent across shapes of varying boundary components.

There are also the directional anchors `north`, `south`, `east`, `west`, `north east`, `north west`, `south east`, `south west`. The `east` and `west` anchors are placed at the midpoints of the sides. The `north` and `south` anchors are placed in a vertical line with the `center` anchor and vertically aligned with the centres of the corresponding boundary circles. The other four directional anchors are placed at the corners of the cobordism (the placement of these anchors in the case that there are no boundary circles in the corresponding direction may change in future versions).

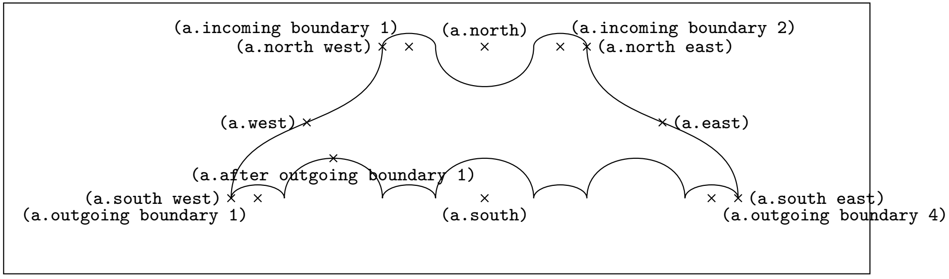
The `incoming boundary n` and `outgoing boundary n` are placed at the centres of the corresponding boundary components, with the numbering starting at the left or the top as appropriate to the flow of the cobordism. A hack borrowed from the `regular polygon` shape ensures that there are always enough anchors for the boundary components.

There are also anchors placed at the midpoint of the cobordism edge between the boundary circles. The names of these are `after incoming boundary n` and `after outgoing boundary n`.

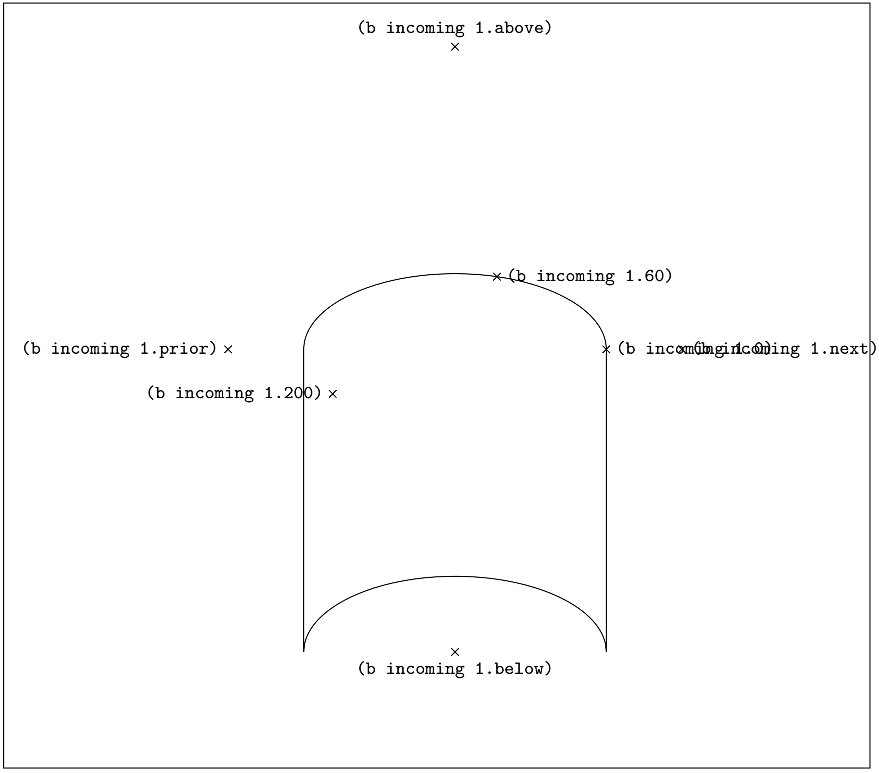
The above anchors can all be “floated” off the cobordism using the keys `outer sep`, `outer xsep`, and `outer ysep`. The last two are the ones actually used, the first is a shortcut for setting both simultaneously.

There are also “sub-nodes”. Provding the main node is named, each boundary circle is covered by a `tqft boundary circle` node. This means that the anchors of the `tqft boundary circle` can be used. These cannot be used for placing the main shape, but can be used afterwards. These are not affected by the `outer (x/y)sep` keys. The names of these sub-nodes are of the form `name incoming n` and `name outgoing n` where `name` is the name of the main node. The `tqft boundary circle` shape is based on an ellipse and defines a boundary so the syntax `(name.angle)` works as expected. It also defines anchors `next`, `prior`, `above`, and `below`. These correspond to where the boundary circle in the prescribed direction should be placed.

```
\begin{tikzpicture}
\node[tqft, incoming boundary components=2,outgoing
    boundary components=4,offset=-1,draw] (a) {};
\foreach \anchor/\placement in
{
north/above,
south/below,
east/right,
west/left,
north west/left,
south west/left,
north east/right,
south east/right,
incoming boundary 1/above left,
incoming boundary 2/above right,
outgoing boundary 1/below left,
outgoing boundary 4/below right,
after outgoing boundary 1/below}
\draw[shift=(a.\anchor)] plot[mark=x] coordinates{(0,0)}
    node[\placement] {\scriptsize\texttt{(a.\anchor)}};
\end{tikzpicture}
```



```
\begin{tikzpicture}
\node[tqft,cylinder, circle width=2cm, circle depth=1cm,
    cobordism height=4cm,boundary separation=3cm,draw]
    (b) {};
\foreach \anchor/\placement in
{
prior/left,
next/right,
above/above,
below/below,
0/right,
60/right,
200/left}
\draw[shift=(b incoming 1.\anchor)] plot[mark=x]
    coordinates{(0,0)} node[\placement]
    {\scriptsize\texttt{(b incoming 1.\anchor)}};
\end{tikzpicture}
```



6.4 Improvements

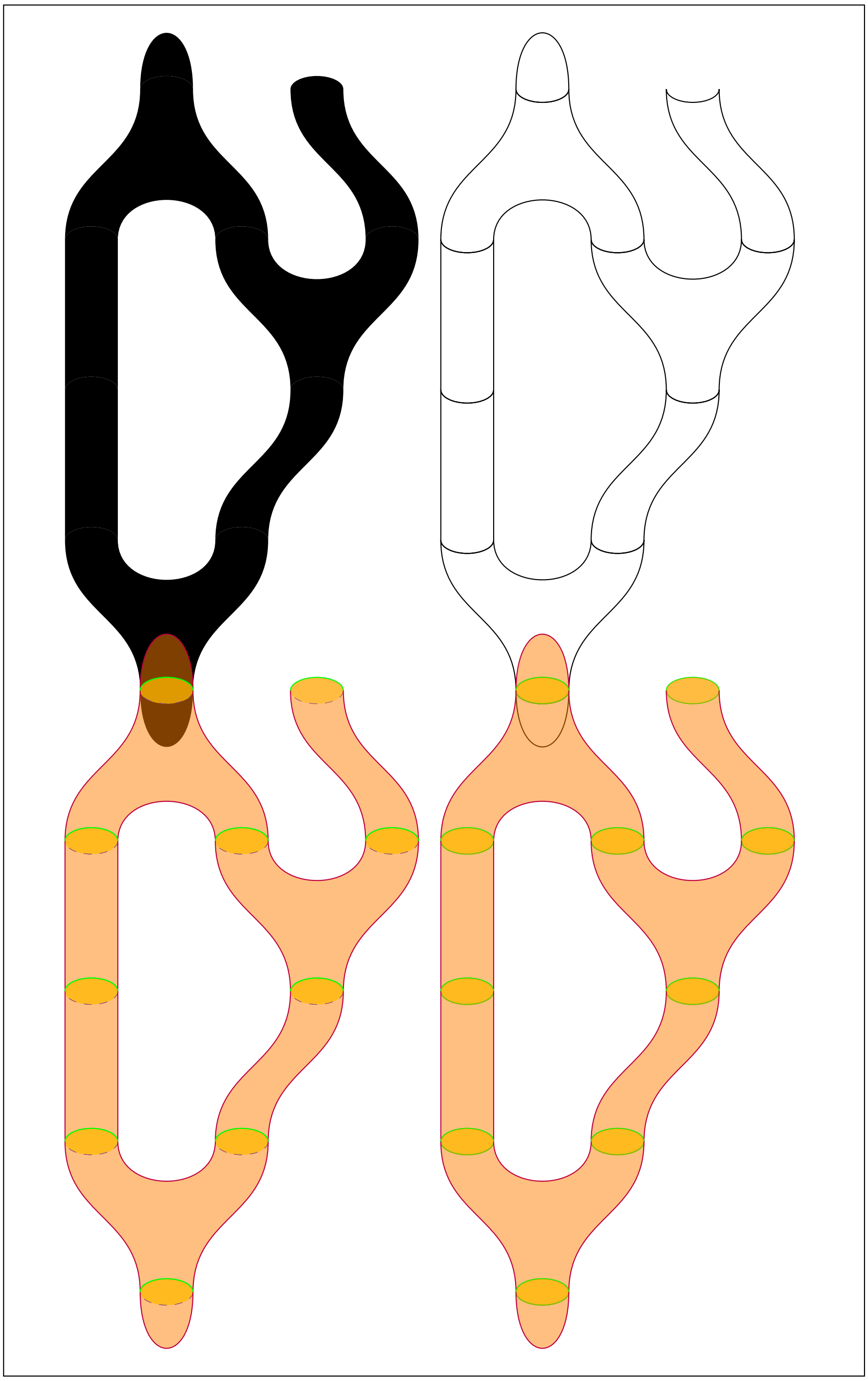
Here are some ideas for extending this, and some minor "bugs".

1. Make `incoming boundary` an alias of `incoming boundary 1` so that if there is only one incoming boundary component then we don't need to specify the number (ditto outgoing).
2. No thought has been given as to where the text gets placed if it is specified.

3. Add the ability to hide certain boundary components. This is useful if the shapes are not specified in their natural order so certain boundary components should be hidden behind earlier drawn shapes.
4. Some style options on the main node get passed to the other pieces (`fill opacity` being one). This shouldn't happen, or should happen by design not by accident.
5. The bounding box isn't as good as it could be.
6. Add a way to specify more directions for the flow.
7. Add the ability to apply different styles to the incoming and outgoing components.

6.5 More Examples

```
\begin{tikzpicture}[tqft/cobordism
    height=1.5cm, tqft/boundary separation=1.5cm]
\foreach \coord/\style in {
{(0,0)}/{tqft/view from=outgoing, fill},
{(5,0)}/{tqft/view from=incoming,draw},
{(0,-8)}/{fill=orange, fill opacity=.5,tqft/boundary
    lower style={draw,blue,ultra
    thin,dashed},tqft/boundary upper
    style={draw,green},tqft/cobordism
    style={draw,purple},tqft/boundary
    style={fill=yellow}},
{(5,-8)}/{fill=orange, fill opacity=.5,tqft/cobordism
    style={draw,purple},tqft/boundary
    style={fill=yellow,draw=green}}
} {
\begin{scope}[every node/.style/.expand once=\style]
\node[tqft/cap] (h) at \coord {};
\node[tqft/pair of pants,anchor=incoming boundary 1] (a)
    at (h.outgoing boundary 1) {};
\node[tqft/cylinder to next,anchor=incoming boundary 1]
    (d) at (a.incoming boundary 2) {};
\node[tqft/reverse pair of pants,anchor=incoming
    boundary 1] (b) at (a.outgoing boundary 2) {};
\node[tqft/cylinder to prior,anchor=incoming boundary 1]
    (c) at (b.outgoing boundary 1) {};
\node[tqft/cylinder,anchor=incoming boundary 1] (e) at
    (a.outgoing boundary 1) {};
\node[tqft/cylinder,anchor=incoming boundary 1] (f) at
    (e.outgoing boundary 1) {};
\node[tqft/reverse pair of pants,anchor=incoming
    boundary 1] (g) at (f.outgoing boundary 1) {};
\node[tqft/cup,anchor=incoming boundary 1] (i) at
    (g.outgoing boundary 1) {};
\end{scope}
}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\node[draw,tqft/pair of pants] (a) {};
\node[draw,tqft/cap,anchor=outgoing boundary 1] at
    (a.incoming boundary 1) {};
\node[fill,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 1) {};
\node[draw,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 2) {};
\begin{scope}[tqft/flow=east]
\node[draw,tqft/pair of pants] (a) at (4,0) {};
\node[draw,tqft/cap,anchor=outgoing boundary 1] at
    (a.incoming boundary 1) {};
\node[fill,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 1) {};
\node[draw,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 2) {};
\end{scope}
\begin{scope}[tqft/flow=north]
\node[draw,tqft/pair of pants] (a) at (0,−4) {};
\node[draw,tqft/cap,anchor=outgoing boundary 1] at
    (a.incoming boundary 1) {};
\node[fill,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 1) {};
\node[draw,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 2) {};
\end{scope}
\begin{scope}[tqft/flow=west]
\node[draw,tqft/pair of pants] (a) at (4,−4) {};
\node[draw,tqft/cap,anchor=outgoing boundary 1] at
    (a.incoming boundary 1) {};
\node[fill,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 1) {};
\node[draw,tqft/cup,anchor=incoming boundary 1] at
    (a.outgoing boundary 2) {};
\end{scope}
\end{tikzpicture}
```