pull-eh

Visualize pulleys with Typst and CeTZ

v0.1.1 June 25, 2025

https://github.com/SillyFreak/typst-pull-eh

Clemens Koza

Contents

I Introduction

Pull-eh lets you visualize pulleys and similar things with Typst and CeTZ. More specifically, it computes the way a cable or rope would wind around a series of wheels. This can be used to visualize physical systems. As an example, here is a gun tackle[1] in place and with pulleys separated:

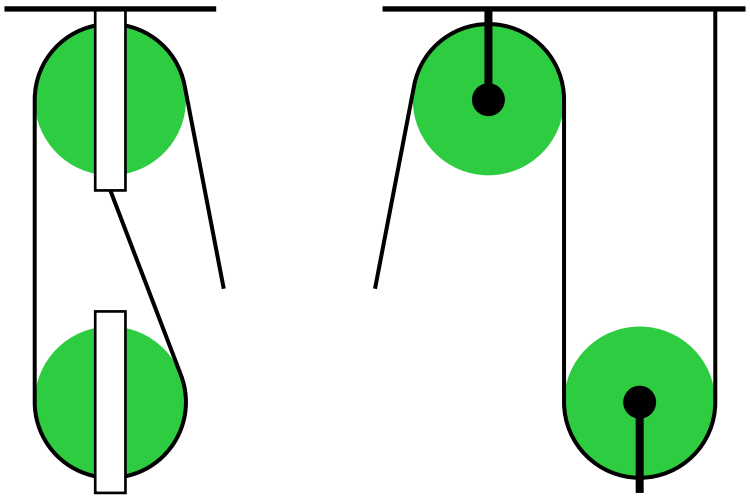

Figure 1: a gun tackle, and its separation

And the corresponding code:

```typ
#import "@preview/cetz:0.4.0"
#import "@preview/pull-eh:0.1.1"
…                                                    …
#let gun-tackle = cetz.canvas({
  import cetz.draw: *
  import pull-eh: *
…                                                    …
  // the rope; drawn over the pulleys, but hidden by the blocks
  on-layer(1, wind(
    stroke: 1.5pt,
    (rel: (1.5, -2.5), to: "pulley1"),
    (coord: "pulley1", radius: 1) + ccw,
    (coord: "pulley2", radius: 1) + ccw,
    "block1.south",
  ))
})
```

Note that pull-eh is *only* responsible for drawing the rope using `wind()`. The rest of the diagram is regular CeTZ; the pulleys themselves are circles with a `name` set. Pull-eh supports any kind of CeTZ coordinate, so the loose end of the rope can be specified relative to the center of pulley 1 (line 37), for example.

The `wind()` function works with either points (CeTZ coordinates) or pulleys (coordinates, plus a radius and winding direction); see the `wind()` function for more details. The direction is specified as a dictionary key `direction: "cw"` or `direction: "ccw"`; for convenience the `cw` and `ccw` variables can be "added" to another dictionary, as shown in lines 38 and 39.

[1] https://en.wikipedia.org/wiki/Block_and_tackle#Overview

II Module reference

II.a `pull-eh`

`wind(..args: arguments) -> shape`

Generates a CeTZ path for a rope winding around a number of points and pulleys. Each positional argument is one winding anchor; named arguments are used to style the path and passed to `cetz.draw.merge-path()`.

Winding anchors can be specified in one of three ways:

- a dictionary with the `radius`, `direction` and `coord` keys: the `coord` is resolved according to CeTZ's rules and treated as the center of a circle with the given `radius`; the rope winds around the circle in the given direction.
- a dictionary with the `radius` and `direction` keys, and any other keys other than `coord`: the other keys are collectively treated as the `coord`, i.e. the radius and winding direction are "embedded" into a CeTZ coordinate dictionary. For example, `(coord: (rel: (2, 2)), radius: 1) + cw` could be rewritten as `(rel: (2, 2), radius: 1) + cw`.
- a valid CeTZ coordinate (not necessarily a dictionary): the rope goes directly to the point, without winding radius or direction.

At least two positional arguments are required. The returned path consists of the following segments:

- For each pair of adjacent anchors, a straight line segment connecting the two. This line will either be a direct connection (if both anchors are points) or a circle-line/circle-circle tangent. Which tangent it is is defined by the winding direction.
  The line segment is omitted if its length would be zero, i.e. the anchors are touching.
- For each anchor except for the first and last, an arc that connects the (potential) line segments surrounding it.
  The arc is omitted if its angle would be zero, or if the anchor is a point anyway.

By using `cetz.draw.merge-path()`, the styles should apply to the whole path and things such as dashed strokes should work correctly without looking strange around segment transitions. Due to the need of resolving coordinates, the returned shape is actually the result of `cetz.draw.get-ctx()`.

Parameters:

`..args` (`arguments`) – positional winding anchors and named CeTZ arguments; see above for details.

`cw: dictionary`

The dictionary `(direction: "cw")` representing clockwise winding. This can be added to a dictionary instead of typing out the whole key and value:

```
1 my-dictionary + cw                                    typc
2 // is equivalent to
3 (..my-dictionary, direction: "cw")
```

`ccw: dictionary`

The dictionary `(direction: "ccw")` representing counter-clockwise winding. This can be added to a dictionary instead of typing out the whole key and value:

```
1 my-dictionary + ccw                                                typc
2 // is equivalent to
3 (..my-dictionary, direction: "ccw")
```