autofletcher

version 0.1.0

This module provides functions to (sort of) abstract away manual placement of coordinates by leveraging typst's partial function application.

Contents

1 Introduction

The main entry-point is `place-nodes()`, which returns a list of indices and a list of partially applied `node()` functions, with the pre-calculated positions.

All coordinates here are elastic, as defined in the fletcher manual. Fractional coordinates don't work that well, from what I've seen.

1.1 About placers

A placer is a function that takes the index of current child, and the total number of children, and returns the coordinates for that child relative to the parent.

Some built-in placers are provided:

- `placer()` which allows easily creating placers from a list of positions. This should be good enough for most uses. See this example
- `arc-placer()` and its special instance `circle-placer` are built-in placers for circular structures. See these examples
- `tree-placer`, which places nodes as children in a tree. See this example

It's relatively easy to create custom placers if needed. See here

1.2 About spread

It appears that fletcher "squeezes" large distances along the left-right axis, as long as the coordinates in-between are empty. This is why it's useful to spread out the first generation of children, even by a large factor. Their children would then occupy the spaces in-between instead of overlapping.

This, however, does not appear to be true for the up-down axis.

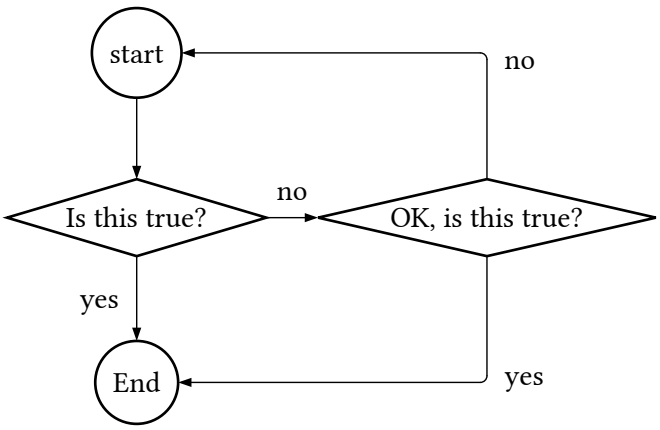
2 Examples

Import the module with:

```
#import "@preview/autofletcher:0.1.0": *
```

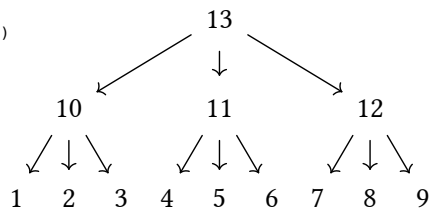
2.1 Flowchart

```
#diagram(
  spacing: (0.2cm, 1.5cm),
  node-stroke: 1pt,
  {
    let r = (0, 0)
    let flowchart-placer = placer((0, 1), (1, 0))

    node(r, [start], shape: shapes.circle)
    // question is a node function with the position pre-applied
    let ((iquestion, ), (question, )) = place-nodes(r, 1, flowchart-placer, spread: 20)

    question([Is this true?], shape: shapes.diamond)
    edge(r, iquestion, "-|>")

    let ((iend, ino), (end, no)) = place-nodes(iquestion, 2, flowchart-placer, spread: 10)

    end([End], shape: shapes.circle)
    no([OK, is this true?], shape: shapes.diamond)

    edge(iquestion, iend, "-|>", label: [yes])
    edge(iquestion, ino, "-|>", label: [no])

    edge(ino, iend, "-|>", label: [yes], corner: right)

    edge(ino, r, "-|>", label: [no], corner: left)

  })
```



2.2 Tree diagram

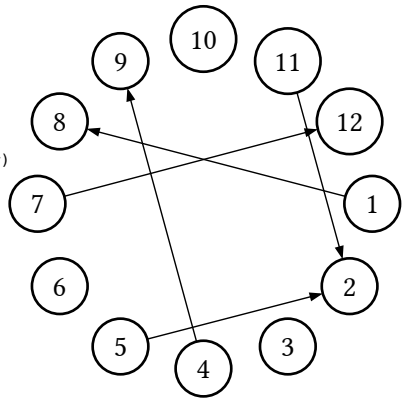
```
#diagram(
spacing: (0.0cm, 0.5cm),
{
  let r = (0, 0)
  node(r, [13])

  let (idxs0, (c1, c2, c3)) = place-nodes(r, 3, tree-placer, spread: 10)

  c1([10])
  c2([11])
  c3([12])

  edges(r, idxs0, "->")

  for (i, parent) in idxs0.enumerate() {
    let (idxs, (c1, c2, c3)) = place-nodes(parent, 3, tree-placer, spread: 2)

    c1([#(i * 3 + 1)])
    c2([#(i * 3 + 2)])
    c3([#(i * 3 + 3)])

    edges(parent, idxs, "->")
  }
})
```



2.3 Arc placer

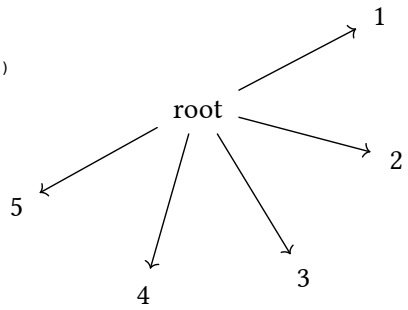
with `circle-placer`:

```
#diagram(
spacing: (1.5cm, 1.5cm),
node-stroke: 1pt,
{
  let r = (0, 0)

  let (idxs, nodes) = place-nodes(r, 12, circle-placer)

  for (i, ch) in nodes.enumerate() {
    ch([#{i + 1}], shape: shapes.circle)
  }

  edge(idxs.at(0), idxs.at(7), "-|>")
  edge(idxs.at(3), idxs.at(8), "-|>")
  edge(idxs.at(4), idxs.at(1), "-|>")
  edge(idxs.at(10), idxs.at(1), "-|>")
  edge(idxs.at(6), idxs.at(11), "-|>")
})
```



With `arc-placer`:

```
#diagram(
spacing: (1.5cm, 1.5cm),
{
  let placer = arc-placer(-30deg, length: calc.pi, radius: 1.2)
  let r = (0, 0)
  node(r, [root])

  let (idxs, nodes) = place-nodes(r, 5, placer, spread: 1)

  for (i, ch) in nodes.enumerate() {
    ch([#{i + 1}])
  }

  edges(r, idxs, "->")
})
```



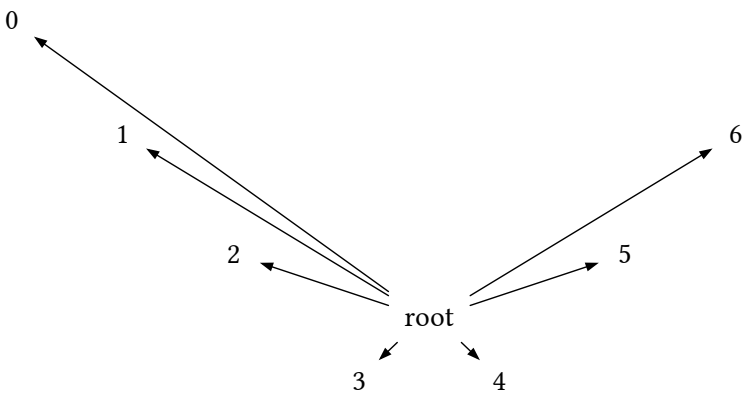
2.4 Custom placers

If the built-in placers don't fit your needs, you can create a custom placer; that is, a function that calculates the relative positions for each child. It should accept, in order:

1. (`int`) the index of the child
2. (`int`) the total number of children

and it should return a pair of coordinates, `(x, y)`.

```
#let custom-placer(i, num-total) = {
  // custom logic here
  let x = i - num-total/2
  let y = calc.min(- x, + x) + 1
  return (x, y)
}

#diagram({
  let r = (0, 0)
  node(r, [root])

  let (idxs, nodes) = place-nodes(r, 7, custom-placer, spread: 1)
  for (i, ch) in nodes.enumerate() {
    ch([#i])
  }
  edges(r, idxs, "-|>")
})
```


0
1
2
root
3
4
5
6

3 API reference

- arc-placer()
- edges()
- place-nodes()
- placer()
- tree-placer()

Variables:

- circle-placer()

arc-placer

Returns a placer that places children in a circular arc

It appears this breaks spread, probably because it uses fractional coordinates. Also, don't mix it with other non-fractional placers. It messes up the graph

Parameters

```
arc-placer(
  start: angle float,
  length: angle float,
  radius: float
)
```

start `angle` or `float`

The starting angle of the arc

length `angle` or `float`

The length of the arc

Default: `2*calc.pi`

radius `float`

The radius of the circle

Default: `1`

edges

Convenience function that draws edges between a parent node and its children, given the coordinates of the parent and children.

Parameters

```
edges(
  parent: coordinates,
  children: array of coordinates,
  ..options: any
)
```

parent `coordinates`

The coordinates of the parent node

children `array of coordinates`

The coordinates of the children nodes

..options `any`

Additional options to pass to `edge`

place-nodes

Calculates the positions of `num-children` children of `parent` node.

Returns a pair of arrays. The first array contains the coordinates of the children, and the second array contains the nodes partially applied with the calculated positions.

Parameters

```
place-nodes(
  parent: coordinates,
  num-children: int,
  placer: function,
  spread: int
) -> (array of coordinates + array of nodes)
```

parent `coordinates`

The coordinates of the parent node

num-children `int`

The number of children to place

placer `function`

The function to calculate the relative positions of the children

spread `int`

A multiplier for the x coordinate, “spreads” children out. Increase this for high parent nodes.

Default: 1

placer

Returns a generic placer, where children are placed according to the given relative positions. If more children are present than there are positions, positions are repeated.

This is probably sufficient for most use cases.

Parameters

placer(..placements: `coordinates`) -> `function`

..placements `coordinates`

Relative positions to assign to children

tree-placer

Calculates the relative position of a child node, like in a tree

Don't call this directly; instead, pass this as a parameter to `place-nodes`.

Parameters

tree-placer(
  i: `int`,
  num-total: `int`
)

i `int`

The index of the child node

num-total `int`

The total number of children

circle-placer

A pre-defined arc placer that places children in a full circle.