Some Infos

The methods I planned to use in the application have the following form:

- algorithm<TypeOfGlyph><TypeOfAlgorithm>()
- utilitys<TypeOfGlyph>()
- (maybe) formatChangeNestedDisks()

all other functions are really low level stuff and the draw functions where just temporary!

NestedDisks

solution, objective =algorithmNestedDisksStackingMinMin(circles, mode)

- *circles*: assuming we get 3 nestings the input *circles* is a list of lists which all have five entrys $[x, y, r_1, r_2, r_3]$ where $r_1 > r_2 > r_3$.
- *mode*: *mode* can be "relative" or "absolute"
- *solution*: has the same structure as *circles*
- *objective*: accumulated value of the calculated utilitys

result, numberOfNestings=formatChangeNestedDisks(circles)

- *circles*: assuming we get 3 nestings the input *circles* is a list of lists which all have five entrys $[x, y, r_1, r_2, r_3]$ where $r_1 > r_2 > r_3$.
- *numberOfNestings*: number of nestings of a single glyph. In our case three.
- *result*: a new representation of our Data, $[x, y, r_1, r_2, r_3]$ and $[x', y', r'_1, r'_2, r'_3]$ get transformed to $[x, y, r_1], [x, y, r_2], [x, y, r_3], [x', y', r'_1], [x', y', r'_2], [x', y', r'_3]$.

utilitysNestedDisks(circles)

- *circles*: assuming we get 3 nestings the input *circles* is a list of lists which all have five entrys $[x, y, r_1, r_2, r_3]$ where $r_1 > r_2 > r_3$.
- outputs some utility Values (variablenames should be discriptive)

Hawaiian

result=algorithmHawaiianLeftToRight(circles)

- *circles*: assuming we get 3 nestings the input *circles* is a list of lists which all have five entrys $[x, y, r_1, r_2, r_3]$ where $r_1 > r_2 > r_3$.
- *result*: not the same structure as circles!!!!!!. a new representation of our Data, $[x, y, r_1, r_2, r_3]$ and $[x', y', r'_1, r'_2, r'_3]$ get transformed to
  $[x_1, y_1, r_1], [x_2, y_2, r_2], [x_3, y_3, r_3], [x'_1, y'_1, r'_1], [x'_2, y'_2, r'_2], [x'_3, y'_3, r'_3]$

utilitysHawaiian(circles,numberOfNestings)

- *numberOfNestings*: number of nestings of a single glyph. In our case three. Is always given by $len(circles[0] - 2)$
- for the rest see NestedDisks

PieCharts

resultCircles, resultPieces, resultAngles=algorithmPieChartsStacking(circles, piePieces)

- *circles*: a list of bounding circles, eg. $[x, y, r], [x', y', r']$.....
- *piePieces*: a list of dividing lines on the boundary given in radient.
  It is always assumed that there is a "base"-dividing line at $\alpha_0 = 0$.
  Therefore we get for a glyph, which contains 3 pieces, the following list: $[\alpha_1, \alpha_2]$ and α_0 is implicit!!!!!!!!!!
- *resultCircles*: same structure as circles
- *resultPieces*: same structure as piePieces
- *resultAngles*: contains for every glyph an angle, in the visualization the dividing lines must be rotated counter clockwise with that angle

utilitysPieCharts(circles,piePieces,angles)

- structure of circles, piepieces, angles is given by the output above (result...).

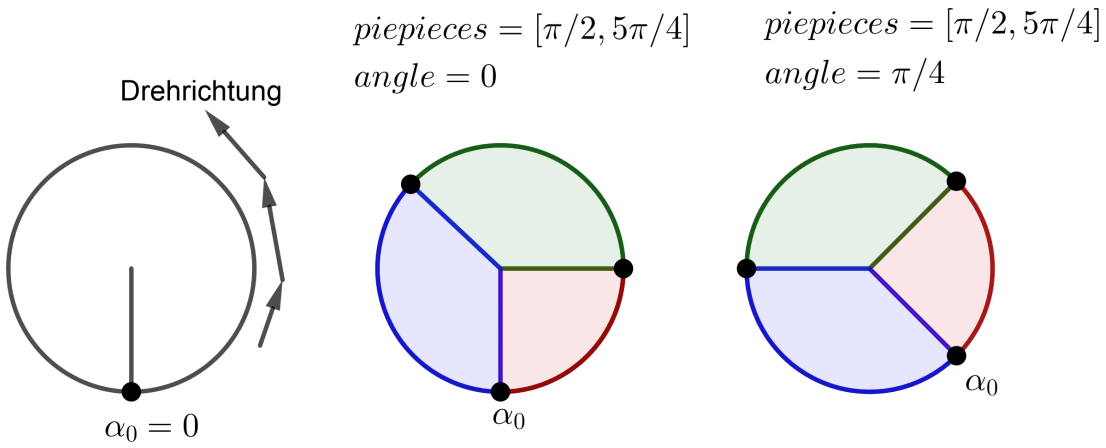

Abbildung 1: position of the lines in a drawing