

pt

TikZ-MEC

v.0.5a – 2012/10/21

Francesco Clemente*

1 Introduction

This library widens the application field of the *TikZ* package defining some new *shapes* used in structural mechanics' two-dimensional diagrams, useful in the description of mechanical problems related to (internally and externaly) loaded structures. The library defines the shapes of various *constraints* used in the above mentioned diagrams, providing a *TikZ*-user-friendly interface for their usage.

If similar features are provided by other libraries or packages, please let me know that through an e-mail to the address specified at the bottom of the page. Advices, corrections and demans are welcomed too. I'll try to answer as soon as possible.

1.1 License

Copyright © 2012 by Francesco Clemente. This work is “maintained” (as per LPPL maintenance status) by Francesco Clemente. The *documentation* and the *code* of this library may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in <http://www.latex-project.org/lppl.txt> and version 1.3 or later is part of all distributions of L^AT_EX version 2005/12/01 or later. This software comes with *absolutely no warranty*.

1.2 Installation

Since the library is currently in development and that it can go towards huge changes, it is better to install it simply copying the file `tikzlibrarymec.code.tex` in the working directory, using it for single documents. The above mentioned file is generated compiling the file `tikz-mec.ins` with `latex`. To generate this documentation file in .pdf format, you need to compile the file `tikz-mec.dtx` with `pdflatex`.

1.3 Loading the library

Once installed, to be able to use the *TikZ*-MEC features just add in the preamble of your document the following line:









```
\usetikzlibrary{mec}
```

TikZ-MEC is based on *TikZ* and uses the `decorations.markings` library, then it is necessary to load both the package and the library too.

*e-mail: francesco dot clemente00 at gmail dot com

2 Shapes

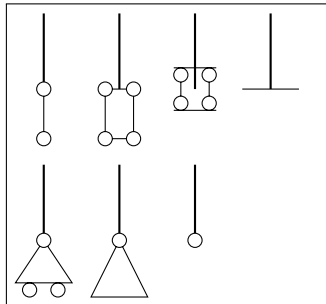
The library defines the following constraints:

- Support (**support**): 
- Hinge A (**hinge**): 
- Hinge B (**hinge b**): 
- Pendulum (**pendulum**): 
- Double pendulum A (**double pendulum**): 
- Double pendulum B (**double pendulum b**): 
- Fixed support (**fixed support**): 
- Ground (**ground**): 

3 Usage examples

In this section we will see some code examples which show the *TikZ-MEC* features. As we will see, all the defined constraints are *node shapes*. So drawing a constraint is the same as drawing a *TikZ* node, with all advantages and disadvantages (discussed in the *TikZ* documentation) that this implies.

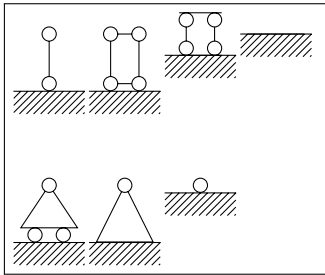
All of the defined constraints connected to a rod which shows their default anchor points:



```
\begin{tikzpicture}
  \node[draw, support](A) at (0,0){};
  \draw[thick] (A) -- +(0,1);
  \node[draw, hinge](A) at (1,0){};
  \draw[thick] (A) -- +(0,1);
  \node[draw, hinge b](A) at (2,0){};
  \draw[thick] (A) -- +(0,1);
  \node[draw, pendulum](A) at (0,2){};
  \draw[thick] (A) -- +(0,1);
  \node[draw, double pendulum](A) at (1,2){};
  \draw[thick] (A) -- +(0,1);
  \node[draw, double pendulum b](A) at (2,2){};
  \draw[thick] (A) -- +(0,1);
  \node[draw, fixed support](A) at (3,2){};
  \draw[thick] (A) -- +(0,1);
\end{tikzpicture}
```

The previous example shows that the default anchor point of the double pendulum version B is at the center of the shape. This leads to a bad graphic result. A future example will show the correct usage of the constraint which takes advantage of its **north** and **south** anchors.

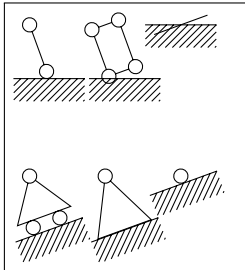
Grounded constraints:



```
\begin{tikzpicture}
  \node[draw,support,grounded] (A) at (0,0){};
  \node[draw,hinge,grounded] (B) at (1,0){};
  \node[draw,hinge b,grounded] (C) at (2,0){};
  \node[draw,pendulum,grounded] (D) at (0,2){};
  \node[draw,double pendulum,grounded] (E) at (1,2){};
  \node[draw,double pendulum b,grounded] (F) at (2,2){};
  \node[draw,fixed support,grounded] (G) at (3,2){};
\end{tikzpicture}
```

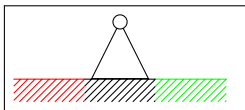
Even if it is possible to draw a ground at the bottom of the double pendulum B like any other shape, it is better to avoid that because it is an *internal* (to the structure) constraint. The equivalent external constraint (from the mechanics point of view) is the double pendulum.

When you need to *rotate* a grounded constraint you don't have to use the standard `rotate` key, but you should specify the rotation angle to the `grounded` key. Doing so you will be able to rotate both the constraint and the ground:



```
\begin{tikzpicture}
  % right
  \node[draw,support,grounded=20] (A) at (0,0){};
  \node[draw,hinge,grounded=20] (B) at (1,0){};
  \node[draw,hinge b,grounded=20] (C) at (2,0){};
  % wrong
  \node[draw,pendulum,grounded,rotate=20] (D) at (0,2){};
  \node[draw,double pendulum,grounded,rotate=20] (E) at (1,2){};
  \node[draw,fixed support,grounded,rotate=20] (F) at (2,2){};
\end{tikzpicture}
```

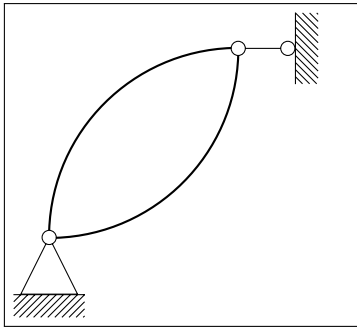
When you define a grounded constraint, the ground is automatically labeled in the form `<node label>ground` where `<node label>` is the label given to the constraint on which the `ground` is connected. Thanks to that it is possible to refer external objects to the ground through the anchors shown in section ??¹. Of course, if you don't define a name for the node, you'll not be able to use the ground anchors:



```
\begin{tikzpicture}
  \coordinate (a) at (0,0);
  \node[draw,hinge,grounded] (A) at (a){};
  \node[draw,ground,red,anchor=east] (l) at (Aground.west){};
  \node[draw,ground,green,anchor=west] (r) at (Aground.east){};
\end{tikzpicture}
```

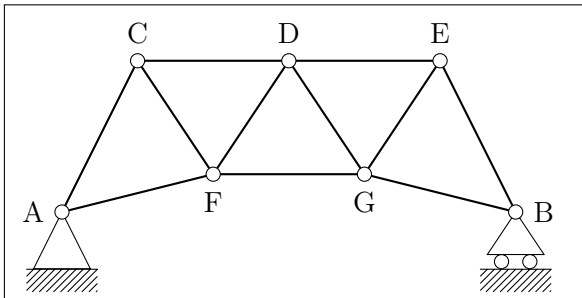
When you need to draw a diagram in which are present curved rods, you *shouldn't* refer to the nodes ending with a circle (i.e. all of them excluding the fixed support and both the versions of the double pendulum) using the usual way. You should instead consider the real starting point and set the arc starting angle accordingly (with default dimensions I suggest to correct the angle by 2°). It is possible to use the same syntax used in the standard *circle* shape:

¹The author wishes to thank Claudio Fiandrino for the suggestion of the feature and for helping in the code implementation



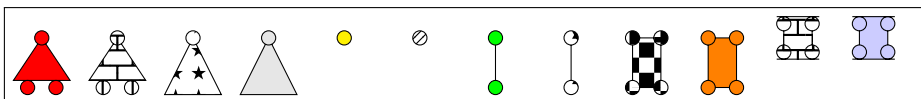
```
\begin{tikzpicture}
  \coordinate (a) at (0,0);
  \coordinate (b) at (2.5,2.5);
  \node[draw,hinge,grounded] (A) at (a){};
  \node[draw,pendulum,grounded=90] (B) at (b){};
  \draw[thick] (A.90) arc[radius=2.5,%
    start angle=178, end angle=92];
  \draw[thick] (B.180) arc[radius=2.5,%
    start angle=358, end angle=272];
\end{tikzpicture}
```

A complete structure which shows where could be useful the B version of the hinge:



```
\usetikzlibrary{positioning}
\begin{tikzpicture}[node distance=1mm]
  \coordinate (a) at (0,0) node[left=of a] {A};
  \coordinate (b) at (6,0) node[right=of b] {B};
  \coordinate (c) at (1,2) node[above=of c] {C};
  \coordinate (d) at (3,2) node[above=of d] {D};
  \coordinate (e) at (5,2) node[above=of e] {E};
  \coordinate (f) at (2,0.5) node[below=of f] {F};
  \coordinate (g) at (4,0.5) node[below=of g] {G};
  \node[hinge,draw,grounded] (A) at (a){};
  \node[support,draw,grounded] (B) at (b){};
  \node[hinge b,draw] (C) at (c){};
  \node[hinge b,draw] (D) at (d){};
  \node[hinge b,draw] (E) at (e){};
  \node[hinge b,draw] (F) at (f){};
  \node[hinge b,draw] (G) at (g){};
  \draw[thick] (A) -- (C) -- (F) -- (D) -- (G) -- (E) -- (B) -- (G)
    -- (F) -- (A) (C) -- (D) -- (E);
\end{tikzpicture}
```

An example showing the possibility to use TikZ fill and pattern operations on the various constraints:

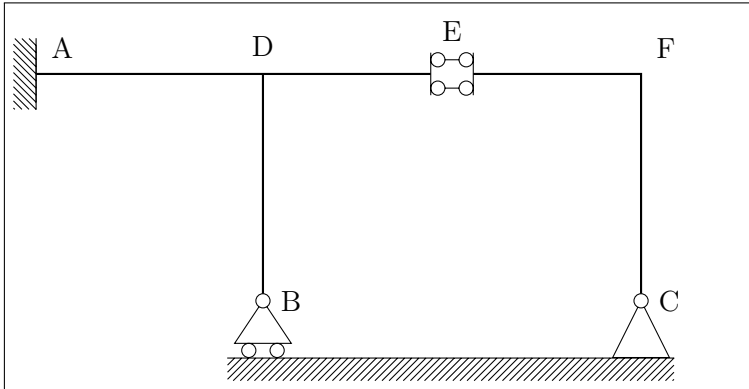


```

\usetikzlibrary{patterns}
\begin{tikzpicture}
  \node[draw,support,fill=red] (A) at (0,0){};
  \node[draw,support,pattern=bricks] (B) at (1,0){};
  \node[draw,hinge,pattern=fivepointed stars] (C) at (2,0){};
  \node[draw,hinge,fill=gray!20] (D) at (3,0){};
  \node[draw,hinge b,fill=yellow] (E) at (4,0){};
  \node[draw,hinge b,pattern=north east lines] (F) at (5,0){};
  \node[draw,pendulum,fill=green] (G) at (6,0){};
  \node[draw,pendulum,pattern=sixpointed stars] (G) at (7,0){};
  \node[draw,double pendulum,pattern=checkerboard] (H) at (8,0){};
  \node[draw,double pendulum,fill=orange] (I) at (9,0){};
  \node[draw,double pendulum b,pattern=bricks] (J) at (10,0){};
  \node[draw,double pendulum b,fill=blue!20] (K) at (11,0){};
\end{tikzpicture}

```

A diagram in which the `double pend b` is correctly used. The key `stretch factor` is also used: it allows to lengthen the ground (the width is not modified by the key). As you can see from the example below, the use of the `stretch factor` key requires to define the ground separately from the constraint. The `left` and `right` anchors come in help when you need to refer a stretched ground to a particular constraint (see section ??).



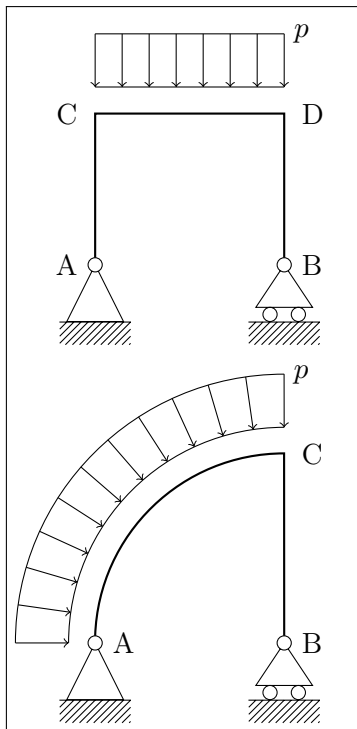
```

\usetikzlibrary{positioning}
\begin{tikzpicture}[node distance=1mm]
  \coordinate (a) at (0,3) node[above right=of a] {A};
  \coordinate (b) at (3,0) node[right=of b] {B};
  \coordinate (c) at (8,0) node[right=of c] {C};
  \coordinate (d) at (3,3) node[above=of d] {D};
  \coordinate (e) at (5.5,3) node[above=3mm of e] {E};
  \coordinate (f) at (8,3) node[above right=of f] {F};
  \node[fixed support,draw,grounded=-90] (A) at (a){};
  \node[support,draw] (B) at (b){};
  \node[hinge,draw] (C) at (c){};
  \node[draw,ground,stretch factor=6.3,anchor=left] (GR) at (B.bottom){};
  \node[double pendulum b,draw,rotate=90] (E) at (e){};
  \draw[thick] (A) -- (d) -- (E.north) (E.south) -- (f) -- (C) (d) -- (B);
\end{tikzpicture}

```

3.1 Adding loads

TikZ-MEC allows to add distributed loads on parts (i.e. links or part of them) of a structure. Lets see some examples:



```

\usetikzlibrary{positioning}
\begin{tikzpicture}[node distance=1mm]
% coordinates
\coordinate (a) at (0,0) node[left=of a] {A};
\coordinate (b) at (2.5,0) node[right=of b] {B};
\coordinate (c) at (0,2) node[left=of c] {C};
\coordinate (d) at (2.5,2) node[right=of d] {D};
% constraints and structure
\node[hinge,draw,grounded] (A) at (a){};
\node[support,grounded,draw] (B) at (b){};
\draw[thick] (A) -- (c) -- (d) -- (B);
% load
\path[draw straight load] (c) -- (d);
\begin{scope}[shift={(0,-5)}]
% coordinates
\coordinate (a) at (0,0) node[right=of a] {A};
\coordinate (b) at (2.5,0) node[right=of b] {B};
\coordinate (c) at (2.5,2.5) node[right=of c] {C};
% constraints and structure
\node[hinge,draw,grounded] (A) at (a){};
\node[support,grounded,draw] (B) at (b){};
\draw[thick] (A) arc[start angle=178,end angle=90,
radius=2.5cm] -- (B);
% load
\path[start angle=180,end angle=90,radius=2.5cm,
draw curved load] (a) arc;
\end{scope}
\end{tikzpicture}

```

As you can see, the actual version of the library allows to add loads on straight and curved links using two different keys: `draw straight load` and `draw curved load`. Every load is generated using an independent path and, by default, it is labeled with a p positioned on the top right of it.

For straight links, you don't need to specify anything else. On the contrary, for curved links, you need to let the library know in time which are the **start angle**, **end angle** and (eventually **x** and **y**) **radius** of the loaded link. This is why those keys need to be anticipated as options to the path. If not, an error will occur because the library will not find their values.

You can personalize a load appearance, using some keys:

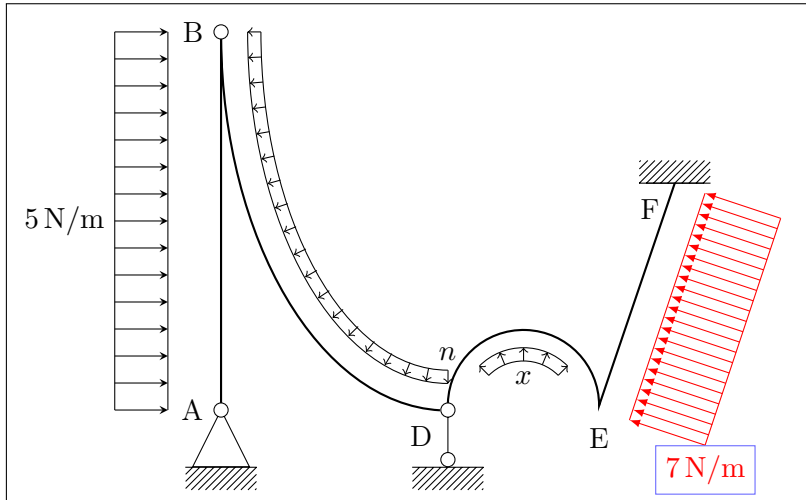
`draw` $\langle load\ type \rangle$ `load={` $[[\langle options \rangle]]\{\langle label \rangle\}\{\langle position \rangle\}$ `}` as previously explained, $\langle load\ type \rangle$ is either **straight** or **curved**. $\langle label \rangle$ is the label attached to the load and the optional $\langle options \rangle$ can be used to personalize it (just as in a normal node). The value stored as $\langle position \rangle$ can vary between 0 and 1 and it changes the position of the label along the farther (from the structure) line of the load.

load distance it sets the distance between the load and the structure. It is set to 10 by default;

forces distance this changes the distance between the single arrows (the forces) that compose the load and is useful to make visible the differences of modulus between different loads (default is 10);

forces length its purpose is the same as for **forces distance**, but it allows to change the length of the forces. Its default value is 20.

Using those keys you can easily get results like this:



```
\usetikzlibrary{positioning}
\begin{tikzpicture}[node distance=1mm]
\coordinate (a) at (0,0) node[left= of a]{A};
\coordinate (b) at (0,5) node[left= of b]{B};
\coordinate (d) at (3,0) node[below left= of d]{D};
\coordinate (e) at (5,0) node[below= of e]{E};
\coordinate (f) at (6,3) node[below left= of f]{F};
% constraints
\node[draw,hinge,grounded] (A) at (a){};
\node[draw,pendulum, grounded] (D) at (d){};
\node[draw,fixed support,grounded=180] (F) at (f){};
\node[draw,hinge b] (B) at (b){};
% structure
\draw[thick] (A) -- (B) (B.270)
  arc[start angle=181,end angle=268,x radius=3,y radius=5] (D.90)
  arc[start angle=178,end angle=0,radius=1cm] -- (F);
% loads
\path[>stealth,load distance=20pt,
  draw straight load={[left]{\SI{5}{\newton/\meter}}}{0.5}] (a) -- (b);
\path[>latex,red,forces length=30pt,forces distance=4pt, load distance=12pt,
  draw straight load={[[draw=blue!60,below]]{\SI{7}{\newton/\meter}}}{1}]
  (f) -- (e);
\path (4,0)+(45:1cm) coordinate (S);
\path[forces length=5pt,load distance=5pt,start angle=45,end angle=135,%
  radius=1cm,draw curved load={[below]{$x$}}{0.5}] (S) arc;
\path[forces length=5pt,start angle=180,end angle=270,%
  x radius=3cm,y radius=5cm,draw curved load={[above]{$n$}}{0.999}]
  (b) arc;
\end{tikzpicture}
```


4 TODO List

The following is a (unordered and incomplete) list of goals needed to improve the library. If you think that something is missing or something has to be deleted from the list, please let me know using the e-mail address in the first page. Well, if you really want to delete some items in the list, you should send me a solution too.

- Verify if the defined shapes don't violate a ISO/UNI standard (if it exists)
- Check the anchors defined for the various shapes and, if necessary, define new and delete others of them
- Modify the shading in the pendulum which now leads to a too dark lower circle
- Find other shared parameters between constraints and the ground
- Create a good interface to draw structures loads (in progress)
- Create a good interface to draw internal stresses distributions
- Solve precision problems affecting (mostly) the appearance of distributed loads on elliptical arcs

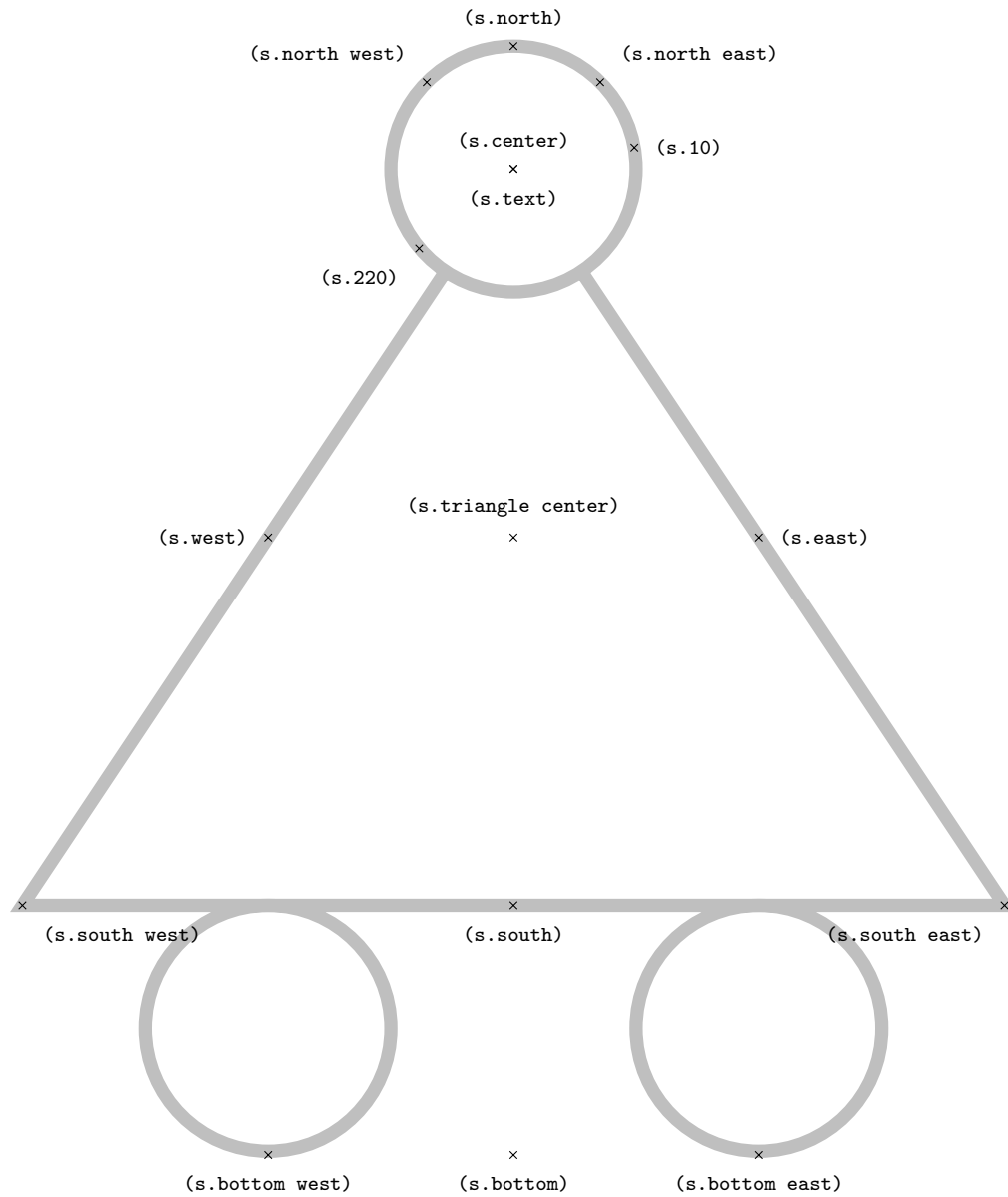
Change History

v.0.1		v.0.4	
General: First public release	1	General: Translated the documentation in english and widely commented the library code	1
v.0.1a			
General: Added the “double pendulum b” shape	1	v.0.5	
v.0.2		General: Added support for distributed loads on straight links	1
General: Documented “double pendulum b” and added examples	1	v.0.5a	
v.0.3		General: Added support for distributed loads on curved paths (using decora- tions)	1
General: Added the stretch factor key, which allows to stretch (lengthen) the ground	1		

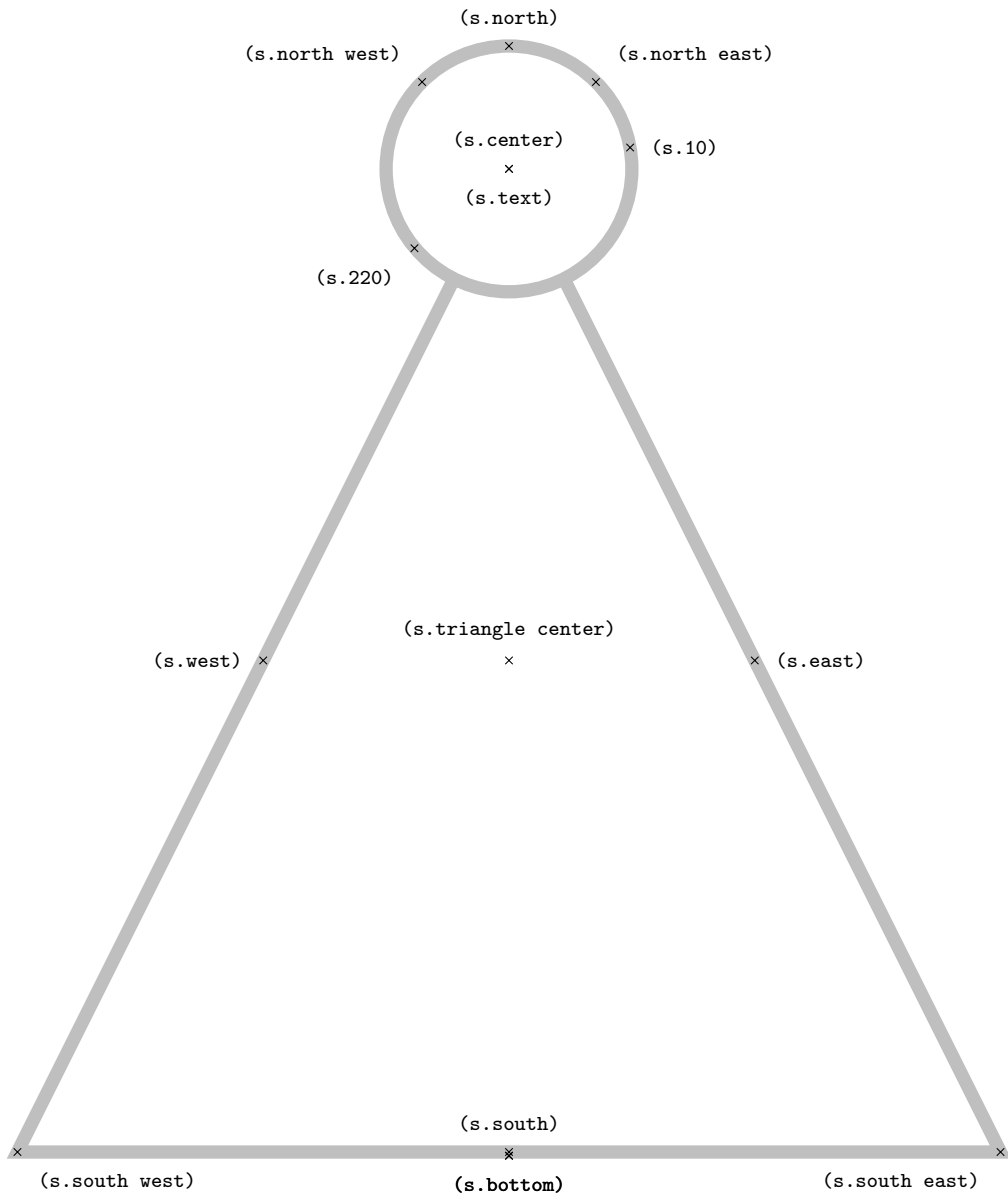
5 Anchors

In this section you'll find all the defined anchors for all the defined shapes. They can be used to refer external objects (like rods and forces) to the constraints and to the ground.

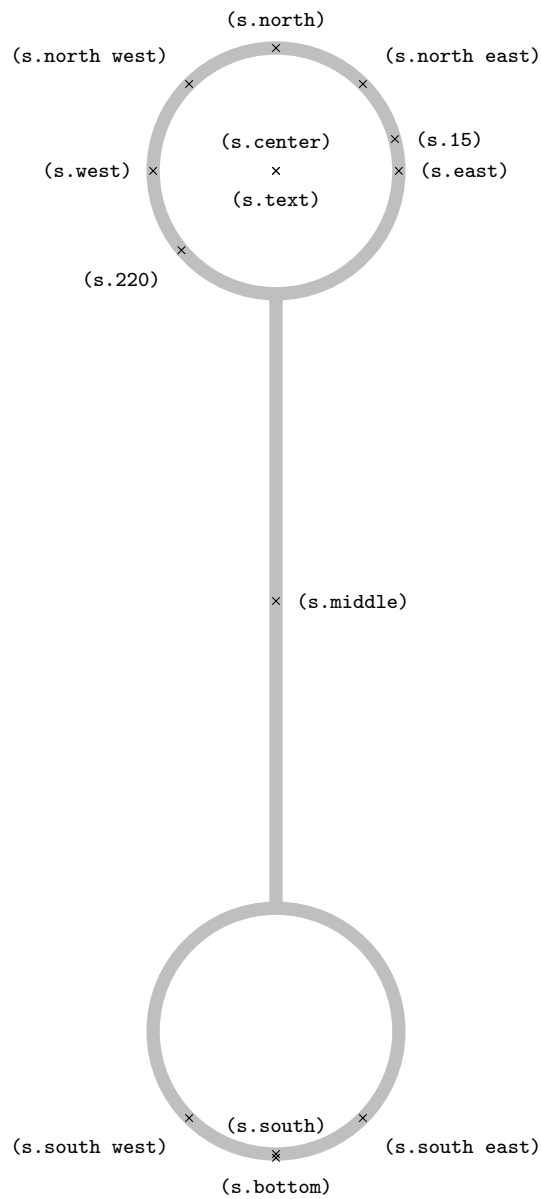
SUPPORT



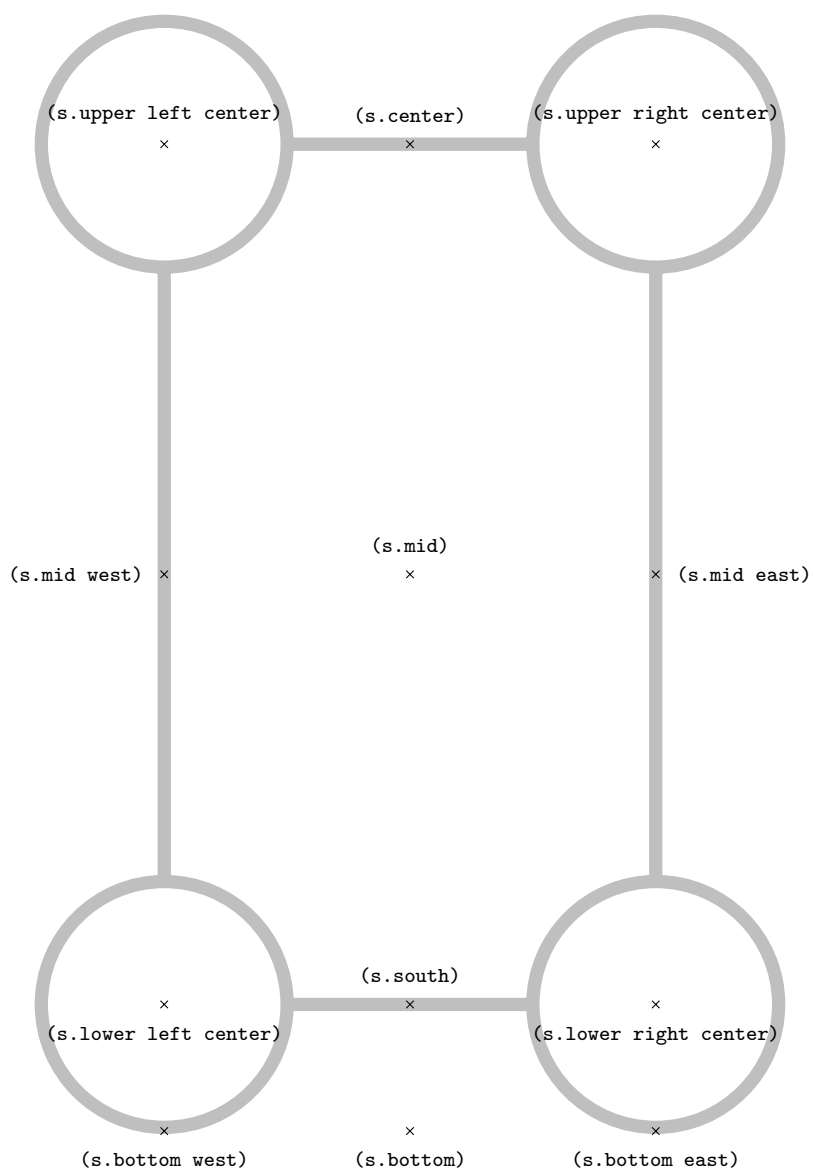
HINGE



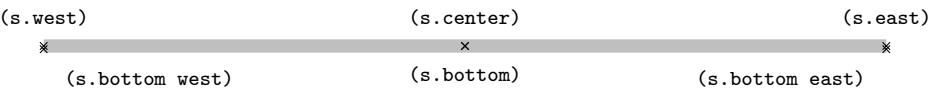
PENDULUM



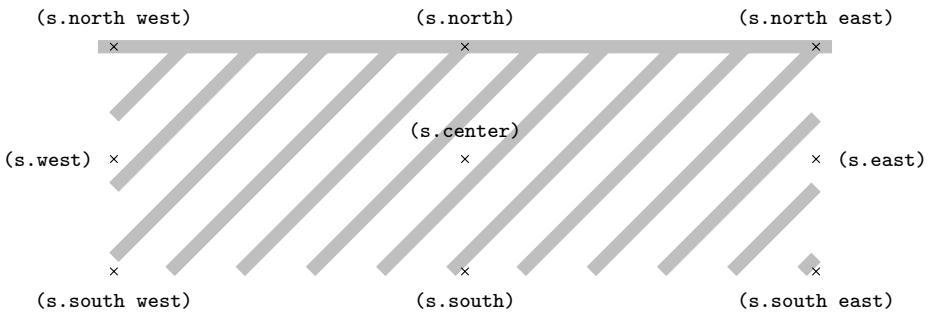
DOUBLE PENDULUM



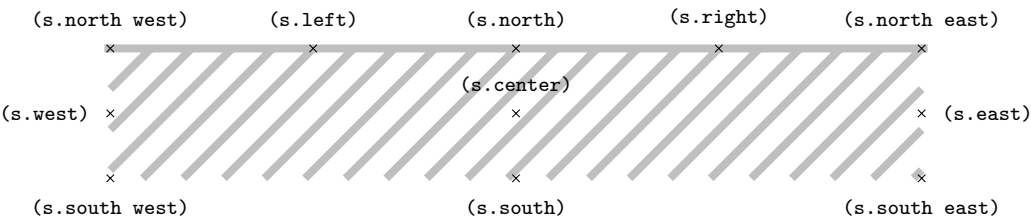
FIXED SUPPORT



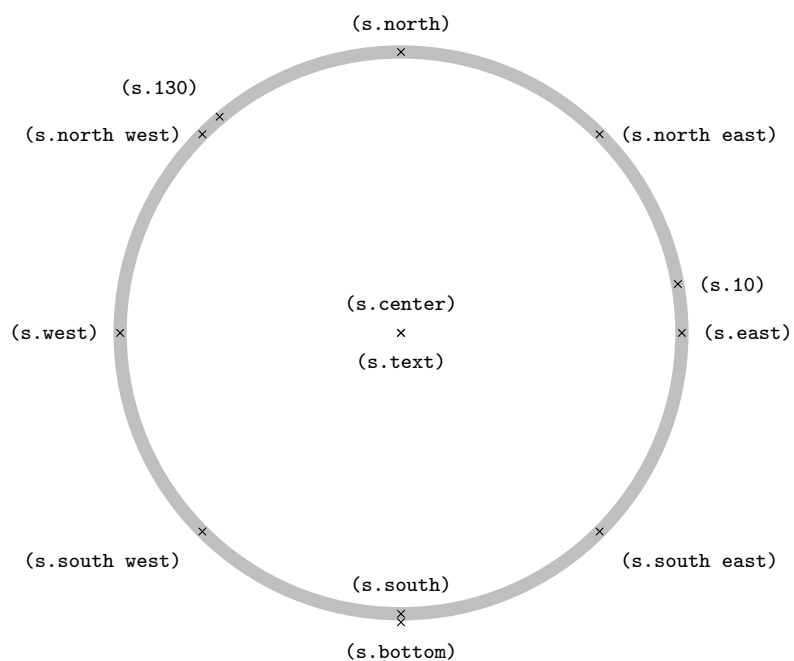
GROUND



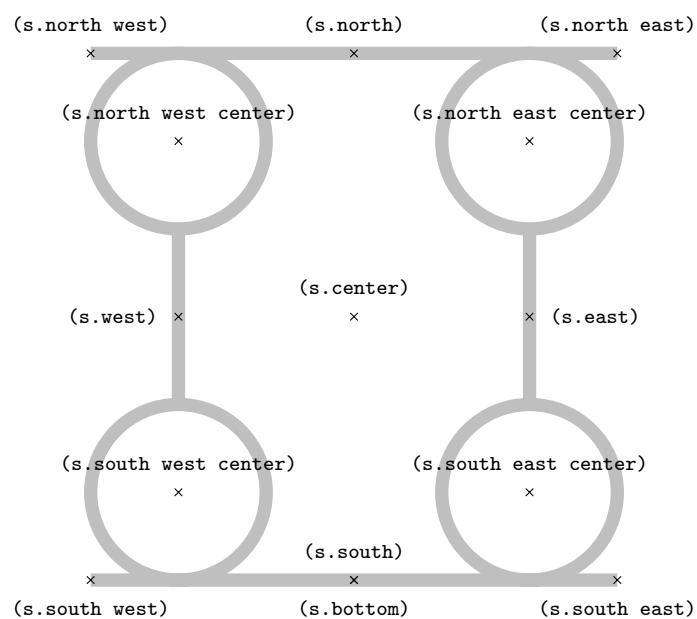
STRETCHED GROUND



HINGE B



DOUBLE PENDULUM B



6 Library Code

First of all, some styles which simplify the user interface are defined.

grounded `grounded` adds a node (shaped *ground*) at the baseline of a constraint allowing its rotation accordingly to it. The rotation angle has to be specified as a parameter of the key (i.e. `grounded=20`)

```
1 \tikzset{%
2   Ground/.style={%
3     append after command={;%
4       \node[draw,ground,anchor=north,rotate=#1] (\tikzlastnode ground) at
5         (\tikzlastnode.bottom){}}},%
6   Ground/.default=0,%
7   grounded/.style={Ground=#1,rotate=#1},%
8   grounded/.default=0}%
```

loads Here we define useful keys and decorations to easily draw distributed loads on a structure or truss. It is possible to add such loads on straight and curved (i.e. circular and elliptical arcs) paths.

Define decoration that draws the arrows that composes a load on a straight path

```
9 \pgfdeclaredecoration{mec@draw@straight@load@forces}{initial}{%
```

The width of each decoration is equal to the path length (stored in `\pgfdecoratedpathlength`) divided by the number of arrows. The number of arrows is calculated dividing the length of the path by the distance between the arrows (contained in the `force distance` key) and rounding the result to assure an integer number

```
10 \state{initial}[width=\pgfdecoratedpathlength/%
11   round(\pgfdecoratedpathlength/\pgfkeysvalueof{/tikz/force distance})]{%
```

Loading parameters used to personalize the load appearance

```
12   \pgf@xb=\pgfkeysvalueof{/tikz/load distance}
13   \pgf@xa=\pgfkeysvalueof{/tikz/force length}
```

`\pgf@xa` gets the distance between the path and the beginning of the arrows

```
14   \advance\pgf@xa by\pgf@xb
```

Draw the arrow perpendicular to the path. No need to add an arrow tip (using, for example, `\pgfsetarrows{>}`) at this level. It will be added at higher level in the style definition (this allows customization at user level defining `>`)

```
15   \pgfpathmoveto{\pgfpoint{0pt}{\pgf@xa}}
16   \pgfpathlineto{\pgfpoint{0pt}{\pgf@xb}}
17   \pgfusepath{stroke}
18 }
19 \state{final}{%
```

The last arrow is not drawn in the `initial` state, so it is needed to add it. Same operations as above, but starting from the last point of the path

```
20   \pgf@xb=\pgfkeysvalueof{/tikz/load distance}
21   \pgf@xa=\pgfkeysvalueof{/tikz/force length}
22   \advance\pgf@xa by\pgf@xb
23   \pgfpathmoveto{%
24     \pgfpointadd{\pgfpoint{0pt}{\pgf@xa}}{\pgfpointdecoratedpathlast}}
25   \pgfpathlineto{%
26     \pgfpointadd{\pgfpoint{0pt}{\pgf@xb}}{\pgfpointdecoratedpathlast}}
27   \pgfusepath{stroke}
28 }
29 }
```


Define a decoration that draws the box containing the arrows of a load on a straight path

```
30 \pgfdeclaredcoration{mec@draw@straight@load@box}{initial}{%
```

Here we use a fake width and draw only once (thanks to the `next state` option) the two paths (above and below the arrows)

```
31 \state{initial}[width=1pt,next state=final]{
32   \pgf@xa=\pgfkeysvalueof{/tikz/load distance}
33   \pgf@xb=\pgfkeysvalueof{/tikz/forces length}
34   \advance\pgf@xb by\pgf@xa
```

Draw the path far from the structure

```
35   \pgfpathmoveto{\pgfpoint{0pt}{\pgf@xa}}
36   \pgfpathlineto{%
37     \pgfpointadd{\pgfpoint{0pt}{\pgf@xa}}{\pgfpointdecoratedpathlast}}%
```

Draw the path near the structure

```
38   \pgfpathmoveto{\pgfpoint{0pt}{\pgf@xb}}
39   \pgfpathlineto{%
40     \pgfpointadd{\pgfpoint{0pt}{\pgf@xb}}{\pgfpointdecoratedpathlast}}%
41   \pgfusepath{stroke}
42 }
43 \state{final}{%
44   \pgfpathmoveto{\pgfpointdecoratedpathlast}
45 }
46 }
```

Define a decoration that draws the arrows on a curved (i.e. circular or elliptical) path. Everything is quite similar to the straight case

```
47 \pgfdeclaredcoration{mec@draw@curved@load@forces}{initial}{%
```

Same width as for the straight one. The only difference is the presence of the multiplier 0.999 to avoid artifacts affecting the last arrow (bad positioning)

```
48 \state{initial}[width=0.999*\pgfdecoratedpathlength/%
49   round(\pgfdecoratedpathlength/\pgfkeysvalueof{/tikz/forces distance})]{%
50   \pgf@xb=\pgfkeysvalueof{/tikz/load distance}
51   \pgf@xa=\pgfkeysvalueof{/tikz/forces length}
52   \advance\pgf@xa by\pgf@xb
53   \pgfpathmoveto{\pgfpoint{0pt}{\pgf@xa}}
54   \pgfpathlineto{\pgfpoint{0pt}{\pgf@xb}}
55   \pgfusepath{stroke}
56 }
57 \state{final}{%
58   \pgf@xb=\pgfkeysvalueof{/tikz/load distance}
59   \pgf@xa=\pgfkeysvalueof{/tikz/forces length}
60   \advance\pgf@xa by\pgf@xb
61   \pgfpathmoveto{%
62     \pgfpointadd{\pgfpoint{0pt}{\pgf@xa}}{\pgfpointdecoratedpathlast}}
63   \pgfpathlineto{%
64     \pgfpointadd{\pgfpoint{0pt}{\pgf@xb}}{\pgfpointdecoratedpathlast}}
65   \pgfusepath{stroke}
66 }
67 }
```

Define a decoration that draws the box containing the arrows of a load on a curved path

```
68 \pgfdeclaredcoration{mec@draw@curved@load@box}{initial}{%
69 \state{initial}[width=1pt,next state=final]{%
```

It is needed to store the values of the x radius and y radius and to add or subtract (depending on the position of the load) to them the distance between the load and the structure

```

70 \pgf@xa=\pgfkeysvalueof{/tikz/load distance}
71 \pgf@ya=\pgf@xa
72 \pgf@xb=\pgfkeysvalueof{/tikz/x radius}
73 \pgf@yb=\pgfkeysvalueof{/tikz/y radius}
74 \pgf@xc=\pgfkeysvalueof{/tikz/forces length}

```

If the load is drawn internally (i.e. `start angle` is smaller than `end angle`) the final radius of the box lines are both smaller than the one of the truss. So `\mec@adjust@factor` is set to -1 (using the `\pgfmathtruncatemacro` I can automatically discard the decimal part) and is used as a multiplier of the lengths

```

75 \pgfmathtruncatemacro{\mec@adjust@factor}{%
76 \pgfkeysvalueof{/tikz/start angle} <%
77 \pgfkeysvalueof{/tikz/end angle} ? -1 : 1}
78 \multiply\pgf@xa by\mec@adjust@factor
79 \multiply\pgf@xc by\mec@adjust@factor
80 \advance\pgf@xb by\pgf@xa
81 \advance\pgf@yb by\pgf@xa
82 \pgfpathmoveto{\pgfpoint{0pt}{\pgf@ya}}

```

It is needed to adjust the `start angle` and `end angle` to correctly draw the arcs adding to it a different factor based on the position (internal or external) of the load

```

83 \pgfmathtruncatemacro{\mec@adjust@angle}{%
84 (90*\mec@adjust@factor)-\pgfkeysvalueof{/tikz/start angle}}
85 \pgfpatharc{\pgfkeysvalueof{/tikz/start angle}+\mec@adjust@angle}{%
86 {\pgfkeysvalueof{/tikz/end angle}+\mec@adjust@angle}%
87 {\pgf@yb and \pgf@xb}% switched
88 \advance\pgf@ya by\mec@adjust@factor\pgf@xc
89 \advance\pgf@xb by\pgf@xc
90 \advance\pgf@yb by\pgf@xc
91 \pgfpathmoveto{\pgfpoint{0pt}{\pgf@ya}}
92 \pgfpatharc{\pgfkeysvalueof{/tikz/start angle}+\mec@adjust@angle}{%
93 {\pgfkeysvalueof{/tikz/end angle}+\mec@adjust@angle}%
94 {\pgf@yb and \pgf@xb}% switched
95 \pgfusepath{stroke}
96 }
97 \state{final}{%
98 \pgfpathmoveto{\pgfpointdecoratedpathlast}
99 }
100 }

```

Now some styles are defined to allow an easy use of the above defined decorations.

```

101 \tikzset{load distance/.initial=10pt,
102 forces distance/.initial=10pt,
103 forces length/.initial=20pt,
104 total distance/.initial=30pt,
105 draw curved load/.style 2 args={%
106 total distance={\pgfkeysvalueof{/tikz/forces length}+%
107 \pgfkeysvalueof{/tikz/load distance}},%
108 postaction={->,decorate,decoration={mec@draw@curved@load@forces}},%
109 postaction={decorate,decoration={mec@draw@curved@load@box}},%
110 postaction={decorate,decoration={markings,%
111 raise=\pgfkeysvalueof{/tikz/total distance}},%
112 mark=at position #2 with \node #1;}}},

```

```

113     draw curved load/.default={\right}{p$}{0.999},
114     draw straight load/.style 2 args={%
115         total distance={\pgfkeysvalueof{/tikz/forces length}+%
116             \pgfkeysvalueof{/tikz/load distance}},%
117         postaction={->,decorate,decoration={mec@draw@straight@load@forces}},%
118         postaction={decorate,decoration={mec@draw@straight@load@box}},%
119         postaction={decorate,decoration={markings,%
120             raise=\pgfkeysvalueof{/tikz/total distance},%
121             mark=at position #2 with \node #1;}}},
122     draw straight load/.default={\right}{p$}{0.999}
123 }

```

stretch factor The value passed to **stretch factor** will be used to compute the final length of the interested ground. The default value is 1 (no stretch present)

```
124 \tikzset{stretch factor/.initial=1}
```

Here starts the library's shapes definitions. All the defined shapes are based on the standard shape **circle**, using as reference dimension its radius (saved in **\radius**)

support Starting defining the support

```

125 \pgfdeclareshape{support}{%
    Load the "saved anchors" from the circle (\centerpoint and \radius)
126   \inheritssavedanchors[from=circle]
    Create a new dimension which is one half of the circle radius (the "last" saved anchor)
127   \saveddimen{\halfradius}{%
128       \pgf@x=.5\pgf@x}
    Load some anchors from the circle and redefine others to adapt them to the new dimension
    \halfradius
129   \anchorborder{
130       \pgf@xa=\pgf@x%
131       \pgf@ya=\pgf@y%
132       \edef\pgf@marshal{%
133           \noexpand\pgfpointborderellipse
134           {\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}
135           {\noexpand\pgfqpoint{\halfradius}{\halfradius}}}% <- edited here
136       }%
137       \pgf@marshal%
138       \pgf@xa=\pgf@x%
139       \pgf@ya=\pgf@y%
140       \centerpoint%
141       \advance\pgf@x by\pgf@xa%
142       \advance\pgf@y by\pgf@ya%
143   }
144   \inheritanchor[from=circle]{center}
145   \inheritanchor[from=circle]{mid}
146   \inheritanchor[from=circle]{base}
147   \anchor{north}{\centerpoint\advance\pgf@y by\halfradius}
148   \anchor{south}{%
149       \centerpoint
150       \pgf@xa=\radius
151       \advance\pgf@y by-3\pgf@xa}
152   \anchor{west}{%
153       \centerpoint
154       \pgf@xa=\radius

```

```

155     % y = -1.5*r*tan(alpha)
156     \advance\pgf@x by-1\pgf@xa
157     \advance\pgf@y by-1.5\pgf@xa
158 }
159 \anchor{east}{%
160     \centerpoint
161     \pgf@xa=\radius
162     % y = 1.5*r*tan(alpha)
163     \advance\pgf@x by1\pgf@xa
164     \advance\pgf@y by-1.5\pgf@xa
165 }
166 \anchor{mid west}{%
167     \centerpoint
168     \advance\pgf@x by-\halfradius\pgfmathsetlength\pgf@y{.5ex}
169 }
170 \anchor{mid east}{%
171     \centerpoint
172     \advance\pgf@x by\halfradius\pgfmathsetlength\pgf@y{.5ex}
173 }
174 \anchor{base west}{\centerpoint\advance\pgf@x by-\halfradius\pgf@y=0pt}
175 \anchor{base east}{\centerpoint\advance\pgf@x by\halfradius\pgf@y=0pt}
176 \anchor{north west}{%
177     \centerpoint
178     \pgf@xa=\halfradius
179     \advance\pgf@x by-0.707107\pgf@xa
180     \advance\pgf@y by0.707107\pgf@xa
181 }
182 \anchor{south west}{%
183     \centerpoint
184     \pgf@xa=\radius
185     \advance\pgf@x by-2\pgf@xa
186     \advance\pgf@y by-3\pgf@xa
187 }
188 \anchor{north east}{%
189     \centerpoint
190     \pgf@xa=\halfradius
191     \advance\pgf@x by0.707107\pgf@xa
192     \advance\pgf@y by0.707107\pgf@xa
193 }
194 \anchor{south east}{%
195     \centerpoint
196     \pgf@xa=\radius
197     \advance\pgf@x by2\pgf@xa
198     \advance\pgf@y by-3\pgf@xa
199 }
200 \anchor{triangle center}{%
201     \centerpoint
202     \pgf@xa=\radius
203     \advance\pgf@y by-1.5\pgf@xa
204 }

```

Add anchors at the base of the shape, useful to correctly place the ground (considering one half of the line width)

```

205 \anchor{bottom}{%
206     \centerpoint
207     \pgf@xa=\radius

```

```

208     \pgf@xb=\halfradius
209     \advance\pgf@y by-3\pgf@xa
210     \advance\pgf@y by-2\pgf@xb
211     \advance\pgf@y by-.5\pgflinewidth
212 }
213 \anchor{bottom west}{%
214     \centerpoint
215     \pgf@xa=\radius
216     \pgf@xb=\halfradius
217     \advance\pgf@y by-3\pgf@xa
218     \advance\pgf@y by-2\pgf@xb
219     \advance\pgf@y by-.5\pgflinewidth
220     \advance\pgf@x by-\pgf@xa
221 }
222 \anchor{bottom east}{%
223     \centerpoint
224     \pgf@xa=\radius
225     \pgf@xb=\halfradius
226     \advance\pgf@y by-3\pgf@xa
227     \advance\pgf@y by-2\pgf@xb
228     \advance\pgf@y by-.5\pgflinewidth
229     \advance\pgf@x by\pgf@xa
230 }

```

Draw the shape

```

231 \backgroundpath{%
232     \pgf@x=\radius

```

Radius is also containing the “minimum width” and “minimum height”. This ensures that even with no text the shape will be drawn, unless of course that minimums are set to 0pt. Now save \radius and \halfradius (that will be the radius of the drawn circles)

```

233     \pgfutil@tempdima=\pgf@x%
234     \pgfutil@tempdimb=\halfradius%

```

Define the west triangle corner “b” and the east triangle corner “c”

```

235     \pgf@xb=-2\pgf@x%
236     \pgf@yb=-3\pgf@x%
237     \pgf@xc= 2\pgf@x%
238     \pgf@yc=-3\pgf@x%
239     % (half inner angle is 33.69)

```

If text is present, shift shape to the center

```

240     \centerpoint
241     \advance\pgf@xb by\pgf@x
242     \advance\pgf@yb by\pgf@y
243     \advance\pgf@xc by\pgf@x
244     \advance\pgf@yc by\pgf@y

```

Save centerpoint in “a” (triangle top corner)

```

245     \pgf@xa=\pgf@x
246     \pgf@ya=\pgf@y

```

Below are good for debugging purposes

```

247     %\message{^^JTop : \the\pgf@xa,\the\pgf@ya}
248     %\message{^^JWest: \the\pgf@xb,\the\pgf@yb}
249     %\message{^^JEast: \the\pgf@xc,\the\pgf@yc}
250     %\message{^^JCent: \the\pgf@x,\the\pgf@y}
251     %\message{^^JR: \the\pgfutil@tempdima}

```

Draw the triangle (considering overlapped upper circle). The starting point is $0.5\backslash\text{radius}\cos(\alpha)$ below and $0.5\backslash\text{radius}\sin(\alpha)$ on the left of the center point (because $\backslash\text{halfradius} = 0.5\backslash\text{radius}$). Then the shift is $(0.5547, 0.832)\backslash\text{halfradius}$

```
252 \advance\pgf@xa by-0.5547\pgfutil@tempdimb%
253 \advance\pgf@ya by-0.832\pgfutil@tempdimb%
254 \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}%
255 \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
256 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
```

Shifting “a” to upper right edge and back to upper left edge to close the path

```
257 \advance\pgf@xa by1.1094\pgfutil@tempdimb%
258 \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}}%
259 \advance\pgf@xa by-1.1094\pgfutil@tempdimb%
260 \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}%
261 \pgfpathclose%
```

Restore “a” in $\backslash\text{centerpoint}$ (triangle top corner)

```
262 \advance\pgf@xa by0.5547\pgfutil@tempdimb%
263 \advance\pgf@ya by0.832\pgfutil@tempdimb%
```

Move “b” to west circle center and “c” to east circle center

```
264 \advance\pgf@xb by 1\pgfutil@tempdima
265 \advance\pgf@yb by -\pgfutil@tempdimb
266 \advance\pgf@xc by-1\pgfutil@tempdima
267 \advance\pgf@yc by -\pgfutil@tempdimb
```

Drawing the three circles

```
268 \edef\pgf@marshal{%
269 \noexpand\pgfpathcircle{%
270 \noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}
271 {\the\pgfutil@tempdimb}%
272 \noexpand\pgfpathcircle{%
273 \noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}
274 {\the\pgfutil@tempdimb}%
275 \noexpand\pgfpathcircle{%
276 \noexpand\pgfqpoint{\the\pgf@xc}{\the\pgf@yc}}
277 {\the\pgfutil@tempdimb}%
278 }\pgf@marshal
279 }%
280 }
```

hinge Start of hinge definition. Similar operations on anchors to what has been made for the support

```
281 \pgfdeclareshape{hinge}{%
282 \inheritsavedanchors[from=circle]
283 \saveddimen{\halfradius}{%
284 \pgf@x=.5\pgf@x}
285 \anchorborder{
286 \pgf@xa=\pgf@x%
287 \pgf@ya=\pgf@y%
288 \edef\pgf@marshal{%
289 \noexpand\pgfpointborderellipse
290 {\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}
291 {\noexpand\pgfqpoint{\halfradius}{\halfradius}}}% <- edited here
292 }%
293 \pgf@marshal%
294 \pgf@xa=\pgf@x%
```

```

295     \pgf@ya=\pgf@y%
296     \centerpoint%
297     \advance\pgf@x by\pgf@xa%
298     \advance\pgf@y by\pgf@ya%
299 }
300 \inheritanchor[from=circle]{center}
301 \inheritanchor[from=circle]{mid}
302 \inheritanchor[from=circle]{base}
303 \anchor{north}{\centerpoint\advance\pgf@y by\halfradius}
304 \anchor{south}{%
305     \centerpoint
306     \pgf@xa=\radius
307     \advance\pgf@y by-4\pgf@xa
308 }
309 \anchor{west}{%
310     \centerpoint
311     \pgf@xa=\radius
312     %  $y = -2*r*\tan(\alpha)$ 
313     \advance\pgf@x by-1\pgf@xa
314     \advance\pgf@y by-2\pgf@xa
315 }
316 \anchor{east}{%
317     \centerpoint
318     \pgf@xa=\radius
319     %  $y = 2*r*\tan(\alpha)$ 
320     \advance\pgf@x by1\pgf@xa
321     \advance\pgf@y by-2\pgf@xa
322 }
323 \anchor{mid west}{%
324     \centerpoint
325     \advance\pgf@x by-\halfradius\pgfmathsetlength\pgf@y{.5ex}}
326 \anchor{mid east}{%
327     \centerpoint
328     \advance\pgf@x by\halfradius\pgfmathsetlength\pgf@y{.5ex}}
329 \anchor{base west}{\centerpoint\advance\pgf@x by-\halfradius\pgf@y=0pt}
330 \anchor{base east}{\centerpoint\advance\pgf@x by\halfradius\pgf@y=0pt}
331 \anchor{north west}{%
332     \centerpoint
333     \pgf@xa=\halfradius
334     \advance\pgf@x by-0.707107\pgf@xa
335     \advance\pgf@y by0.707107\pgf@xa
336 }
337 \anchor{south west}{%
338     \centerpoint
339     \pgf@xa=\radius
340     \advance\pgf@x by-2\pgf@xa
341     \advance\pgf@y by-4\pgf@xa
342 }
343 \anchor{north east}{%
344     \centerpoint
345     \pgf@xa=\halfradius
346     \advance\pgf@x by0.707107\pgf@xa
347     \advance\pgf@y by0.707107\pgf@xa
348 }
349 \anchor{south east}{%
350     \centerpoint

```

```

351     \pgf@xa=\radius
352     \advance\pgf@x by2\pgf@xa
353     \advance\pgf@y by-4\pgf@xa
354 }
355 \anchor{bottom}{%
356     \centerpoint
357     \pgf@xa=\radius
358     \advance\pgf@y by-4\pgf@xa
359     \advance\pgf@y by-.5\pgflinewidth
360 }
361 \anchor{triangle center}{%
362     \centerpoint
363     \pgf@xa=\radius
364     \advance\pgf@y by-2\pgf@xa
365 }

```

Defining the shape

```

366 \backgroundpath{%
367     \pgf@x=\radius
368     \pgfutil@tempdima=\pgf@x%
369     \pgfutil@tempdimb=\halfradius%
370     % west triangle corner "b"
371     \pgf@xb=-2\pgf@x%
372     \pgf@yb=-4\pgf@x%
373     % east triangle corner "c"
374     \pgf@xc= 2\pgf@x%
375     \pgf@yc=-4\pgf@x%

```

Half angle is 26.565°

```

376     % If text is present shift shape to center
377     \centerpoint
378     \advance\pgf@xb by\pgf@x
379     \advance\pgf@yb by\pgf@y
380     \advance\pgf@xc by\pgf@x
381     \advance\pgf@yc by\pgf@y
382     % Save centerpoint in "a" (triangle top corner)
383     \pgf@xa=\pgf@x
384     \pgf@ya=\pgf@y

```

Draw the triangle (considering upper circle). See the `shape support` for details. The shift is $(0.4472, 0.8944)\text{radius}$. Starting from upper left edge

```

385     \advance\pgf@xa by-0.4472\pgfutil@tempdimb%
386     \advance\pgf@ya by-0.8944\pgfutil@tempdimb%
387     \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}%
388     \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
389     \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
390     % changing "a" to upper right edge...
391     \advance\pgf@xa by0.8944\pgfutil@tempdimb%
392     \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}}%
393     % ...and back to upper left to close the path
394     \advance\pgf@xa by-0.8944\pgfutil@tempdimb%
395     \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}%
396     \pgfpathclose%

```

Restore the center point in "a" (triangle top corner) and draw the circle

```

397     \advance\pgf@xa by0.4472\pgfutil@tempdimb%
398     \advance\pgf@ya by0.8944\pgfutil@tempdimb%

```



```

399     \edef\pgf@marshal{%
400         \noexpand\pgfpathcircle{%
401             \noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}
402             {\the\pgfutil@tempdimb}%
403     }\pgf@marshal
404 }%
405 }

```

hinge b Starting the definition of the alternative shape for the hinge, useful as an internal constraint.
Again same operations on anchors

```

406 \pgfdeclareshape{hinge b}{%
407     \inheritsavedanchors[from=circle]
408     \saveddimen{\halfradius}{%
409         \pgf@x=.5\pgf@x}
410     \anchorborder{
411         \pgf@xa=\pgf@x%
412         \pgf@ya=\pgf@y%
413         \edef\pgf@marshal{%
414             \noexpand\pgfpoinborderellipse
415             {\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}
416             {\noexpand\pgfqpoint{\halfradius}{\halfradius}}}% <- edited here
417         }%
418         \pgf@marshal%
419         \pgf@xa=\pgf@x%
420         \pgf@ya=\pgf@y%
421         \centerpoint%
422         \advance\pgf@x by\pgf@xa%
423         \advance\pgf@y by\pgf@ya%
424     }
425     \inheritanchor[from=circle]{center}
426     \inheritanchor[from=circle]{mid}
427     \inheritanchor[from=circle]{base}
428     \anchor{north}{\centerpoint\advance\pgf@y by\halfradius}
429     \anchor{south}{\centerpoint\advance\pgf@y by-\halfradius}
430     \anchor{west}{\centerpoint\advance\pgf@x by-\halfradius}
431     \anchor{east}{\centerpoint\advance\pgf@x by\halfradius}
432     \anchor{mid west}{%
433         \centerpoint
434         \advance\pgf@x by-\halfradius\pgfmathsetlength\pgf@y{.5ex}
435     }
436     \anchor{mid east}{%
437         \centerpoint
438         \advance\pgf@x by\halfradius\pgfmathsetlength\pgf@y{.5ex}
439     }
440     \anchor{base west}{%
441         \centerpoint
442         \advance\pgf@x by-\halfradius\pgf@y=0pt
443     }
444     \anchor{base east}{%
445         \centerpoint
446         \advance\pgf@x by\halfradius\pgf@y=0pt
447     }
448     \anchor{north west}{%
449         \centerpoint
450         \pgf@xa=\halfradius
451         \advance\pgf@x by-0.707107\pgf@xa

```

```

452   \advance\pgf@y by0.707107\pgf@xa
453 }
454 \anchor{south west}{%
455   \centerpoint
456   \pgf@xa=\halfradius
457   \advance\pgf@x by-0.707107\pgf@xa
458   \advance\pgf@y by-0.707107\pgf@xa
459 }
460 \anchor{north east}{%
461   \centerpoint
462   \pgf@xa=\halfradius
463   \advance\pgf@x by0.707107\pgf@xa
464   \advance\pgf@y by0.707107\pgf@xa
465 }
466 \anchor{south east}{%
467   \centerpoint
468   \pgf@xa=\halfradius
469   \advance\pgf@x by0.707107\pgf@xa
470   \advance\pgf@y by-0.707107\pgf@xa
471 }
472 \anchor{bottom}{%
473   \centerpoint
474   \advance\pgf@y by-\halfradius
475   \advance\pgf@y by-0.5\pgflinewidth
476 }

```

The shape is just a circle with the radius equal to \halfradius

```

477 \backgroundpath{%
478   \pgfutil@tempdima=\halfradius%
479   \pgfpathcircle{\centerpoint}{\pgfutil@tempdima}%
480 }%
481 }

```

pendulum Starting the definition of the pendulum. Same operations on anchors

```

482 \pgfdeclareshape{pendulum}{%
483   \inheritsavedanchors[from=circle]
484   \saveddimen{\halfradius}{%
485     \pgf@x=.5\pgf@x}
486   \anchorborder{
487     \pgf@xa=\pgf@x%
488     \pgf@ya=\pgf@y%
489     \edef\pgf@marshal{%
490       \noexpand\pgfpointborderellipse
491       {\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}
492       {\noexpand\pgfqpoint{\halfradius}{\halfradius}}}% <- edited here
493   }%
494   \pgf@marshal%
495   \pgf@xa=\pgf@x%
496   \pgf@ya=\pgf@y%
497   \centerpoint%
498   \advance\pgf@x by\pgf@xa%
499   \advance\pgf@y by\pgf@ya%
500 }
501 \inheritanchor[from=circle]{center}
502 \inheritanchor[from=circle]{mid}
503 \inheritanchor[from=circle]{base}

```

```

504 \anchor{north}{\centerpoint\advance\pgf@y by\halfradius}
505 \anchor{south}{%
506   \centerpoint
507   \pgf@xa=\radius
508   \advance\pgf@y by-4\pgf@xa
509 }
510 \anchor{west}{\centerpoint\advance\pgf@x by-\halfradius}
511 \anchor{east}{\centerpoint\advance\pgf@x by\halfradius}
512 \anchor{mid west}{%
513   \centerpoint
514   \advance\pgf@x by-\halfradius\pgfmathsetlength\pgf@y{.5ex}
515 }
516 \anchor{mid east}{%
517   \centerpoint
518   \advance\pgf@x by\halfradius\pgfmathsetlength\pgf@y{.5ex}
519 }
520 \anchor{base west}{\centerpoint\advance\pgf@x by-\halfradius\pgf@y=0pt}
521 \anchor{base east}{\centerpoint\advance\pgf@x by\halfradius\pgf@y=0pt}
522 \anchor{north west}{%
523   \centerpoint
524   \pgf@xa=\halfradius
525   \advance\pgf@x by-0.707107\pgf@xa
526   \advance\pgf@y by0.707107\pgf@xa
527 }
528 \anchor{south west}{%
529   \centerpoint
530   \pgf@xa=\radius
531   \pgf@xb=\halfradius
532   \advance\pgf@x by-0.707107\pgf@xb
533   \advance\pgf@y by-3\pgf@xa
534   \advance\pgf@y by-1.707107\pgf@xb
535 }
536 \anchor{north east}{%
537   \centerpoint
538   \pgf@xa=\halfradius
539   \advance\pgf@x by0.707107\pgf@xa
540   \advance\pgf@y by0.707107\pgf@xa
541 }
542 \anchor{south east}{%
543   \centerpoint
544   \pgf@xa=\radius
545   \pgf@xb=\halfradius
546   \advance\pgf@x by0.707107\pgf@xb
547   \advance\pgf@y by-3\pgf@xa
548   \advance\pgf@y by-1.707107\pgf@xb
549 }
550 \anchor{bottom}{%
551   \centerpoint
552   \pgf@xa=\radius
553   \advance\pgf@y by-4\pgf@xa
554   \advance\pgf@y by-.5\pgflinewidth
555 }
556 \anchor{middle}{%
557   \centerpoint
558   \pgf@xa=\radius
559   \pgf@xb=\halfradius

```

```

560 \advance\pgf@y by-1\pgf@xa
561 \advance\pgf@y by-1.5\pgf@xb
562 }

```

Define the shape

```

563 \backgroundpath{%
564 \pgf@x=\radius
565 \pgfutil@tempdima=\pgf@x%
566 \pgfutil@tempdimb=\halfradius%

```

Define the lower circle center “b”

```

567 \pgf@xb=0\pgfutil@tempdima%
568 \pgf@yb=-3\pgfutil@tempdima%
569 \advance\pgf@yb by-\pgfutil@tempdimb
570 % If text is present shift shape to center
571 \centerpoint
572 \advance\pgf@xb by\pgf@x
573 \advance\pgf@yb by\pgf@y
574 % Save centerpoint in "a" (top circle center)
575 \pgf@xa=\pgf@x
576 \pgf@ya=\pgf@y

```

Draw the vertical axis of the shape (considering upper and lower circle) starting from the upper edge

```

577 \advance\pgf@ya by-\pgfutil@tempdimb%
578 \advance\pgf@yb by\pgfutil@tempdimb%
579 \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}%
580 \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
581 \pgfpathclose

```

Restore “a” in center point (top circle center) and “b” in the center of the lower circle. After that, draw the two circles

```

582 \advance\pgf@ya by\pgfutil@tempdimb%
583 \advance\pgf@yb by-\pgfutil@tempdimb%
584 \edef\pgf@marshal{%
585 \noexpand\pgfpathcircle{%
586 \noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}
587 {\the\pgfutil@tempdimb}%
588 \noexpand\pgfpathcircle{%
589 \noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}
590 {\the\pgfutil@tempdimb}%
591 }\pgf@marshal
592 }%
593 }

```

double pendulum Beginning of definition of the double pendulum. Same operations on anchors

```

594 \pgfdeclareshape{double pendulum}{%
595 \inheritsavedanchors[from=circle]
596 \saveddimen{\halfradius}{%
597 \pgf@x=.5\pgf@x}
598 \inheritanchor[from=circle]{center}
599 \anchor{mid}{% y = centerpoint - 1\radius - 1.5\halfradius
600 \centerpoint
601 \pgf@xa=\radius
602 \pgf@xb=\halfradius
603 \advance\pgf@y by-1\pgf@xa
604 \advance\pgf@y by-1.5\pgf@xb

```

```

605 }
606 \anchor{mid west}{%
607   % y = centerpoint - 1\radius - 1.5\halfradius
608   % x = centerpoint - radius
609   \centerpoint
610   \pgf@xa=\radius
611   \pgf@xb=\halfradius
612   \advance\pgf@x by-1\pgf@xa
613   \advance\pgf@y by-1\pgf@xa
614   \advance\pgf@y by-1.5\pgf@xb
615 }
616 \anchor{mid east}{%
617   % y = centerpoint - 1\radius - 1.5\halfradius
618   % x = centerpoint + radius
619   \centerpoint
620   \pgf@xa=\radius
621   \pgf@xb=\halfradius
622   \advance\pgf@x by\pgf@xa
623   \advance\pgf@y by-1\pgf@xa
624   \advance\pgf@y by-1.5\pgf@xb
625 }
626 \anchor{south}{%
627   \centerpoint
628   \pgf@xa=\radius
629   \pgf@xb=\halfradius
630   \advance\pgf@y by-3\pgf@xa
631   \advance\pgf@y by-\pgf@xb
632 }
633 \anchor{upper left center}{%
634   \centerpoint
635   \advance\pgf@x by-\radius
636 }
637 \anchor{upper right center}{%
638   \centerpoint
639   \advance\pgf@x by\radius
640 }
641 \anchor{lower left center}{%
642   \centerpoint
643   \pgf@xa=\radius
644   \pgf@xb=\halfradius
645   \advance\pgf@x by-\pgf@xa
646   \advance\pgf@y by-3\pgf@xa
647   \advance\pgf@y by-\pgf@xb
648 }
649 \anchor{lower right center}{%
650   \centerpoint
651   \pgf@xa=\radius
652   \pgf@xb=\halfradius
653   \advance\pgf@x by\pgf@xa
654   \advance\pgf@y by-3\pgf@xa
655   \advance\pgf@y by-\pgf@xb
656 }
657 \anchor{bottom}{%
658   \centerpoint
659   \pgf@xa=\radius
660   \pgf@xb=\halfradius

```

```

661   \advance\pgf@y by-3\pgf@xa
662   \advance\pgf@y by-2\pgf@xb
663   \advance\pgf@y by-.5\pgflinewidth
664 }
665 \anchor{bottom west}{%
666   \centerpoint
667   \pgf@xa=\radius
668   \pgf@xb=\halfradius
669   \advance\pgf@y by-3\pgf@xa
670   \advance\pgf@y by-2\pgf@xb
671   \advance\pgf@y by-.5\pgflinewidth
672   \advance\pgf@x by-\pgf@xa
673 }
674 \anchor{bottom east}{%
675   \centerpoint
676   \pgf@xa=\radius
677   \pgf@xb=\halfradius
678   \advance\pgf@y by-3\pgf@xa
679   \advance\pgf@y by-2\pgf@xb
680   \advance\pgf@y by-.5\pgflinewidth
681   \advance\pgf@x by\pgf@xa
682 }

```

Define the shape

```

683 \backgroundpath{%
684   \pgf@x=\radius
685   \pgfutil@tempdima=\pgf@x%
686   \pgfutil@tempdimb=\halfradius%

```

Define the rectangle upper right corner “b” and lower left corner “c”

```

687   \pgf@xb=\pgfutil@tempdima%
688   \pgf@yb=0\pgfutil@tempdima%
689   \pgf@xc=-\pgfutil@tempdima%
690   \pgf@yc=-3\pgfutil@tempdima%
691   \advance\pgf@yc by-\pgfutil@tempdimb%
692   % If text is present shift shape to center
693   % You need to shift more, but to get the idea
694   \centerpoint
695   \advance\pgf@xb by\pgf@x
696   \advance\pgf@yb by\pgf@y
697   \advance\pgf@xc by\pgf@x
698   \advance\pgf@yc by\pgf@y
699   % Save centerpoint in "a" (top rectangle centerpoint)
700   \pgf@xa=\pgf@x
701   \pgf@ya=\pgf@y

```

The drawn rectangle is made of multiple paths so it is necessary to create another unique path that will be filled correctly when needed (patterns are working too). This is because the fill operation is defined only for single paths

```

702   \tikz@mode
703   \iftikz@mode@fill
704     \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}
705     \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yc}}
706     \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}
707     \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yb}}
708     \pgfpathclose
709     \pgfusepath{fill}\fi

```

Draw sides of rectangle (considering circles; this leads to multiple paths) starting from upper left edge and going counterclockwise

```

710 \advance\pgf@xc by\pgfutil@tempdimb%
711 \pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yb}}%
712 % moving to upper right edge
713 \advance\pgf@xb by-\pgfutil@tempdimb%
714 \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
715 % moving to lower right edge
716 \advance\pgf@xb by\pgfutil@tempdimb%
717 \advance\pgf@yb by-\pgfutil@tempdimb%
718 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
719 \advance\pgf@yc by\pgfutil@tempdimb%
720 \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yc}}%
721 % moving to lower left edge
722 \advance\pgf@yc by-\pgfutil@tempdimb%
723 \advance\pgf@xb by-\pgfutil@tempdimb%
724 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yc}}%
725 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
726 % moving to upper right edge
727 \advance\pgf@xc by-\pgfutil@tempdimb%
728 \advance\pgf@yc by\pgfutil@tempdimb%
729 \pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
730 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yb}}%
731 % closing the path
732 \advance\pgf@xc by\pgfutil@tempdimb%
733 \advance\pgf@yb by\pgfutil@tempdimb%
734 \pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yb}}%
735 \pgfpathclose%

```

Restore “b” and “c” and draw the circles

```

736 \advance\pgf@xb by\pgfutil@tempdimb%
737 \advance\pgf@xc by-\pgfutil@tempdimb%
738 \advance\pgf@yc by-\pgfutil@tempdimb%
739 \edef\pgf@marshal{%
740     \noexpand\pgfpathcircle{%
741         \noexpand\pgfqpoint{\the\pgf@xc}{\the\pgf@yb}}
742         {\the\pgfutil@tempdimb}%
743     \noexpand\pgfpathcircle{%
744         \noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}
745         {\the\pgfutil@tempdimb}%
746     \noexpand\pgfpathcircle{%
747         \noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yc}}
748         {\the\pgfutil@tempdimb}%
749     \noexpand\pgfpathcircle{%
750         \noexpand\pgfqpoint{\the\pgf@xc}{\the\pgf@yc}}
751         {\the\pgfutil@tempdimb}%
752 } \pgf@marshal
753 }%
754 }

```

fixed support Starting defining the fixed support. Same operations on anchors

```

755 \pgfdeclareshape{fixed support}{%
756     \inheritsavedanchors[from=circle]
757     \saveddimen{\halfradius}{%
758         \pgf@x=.5\pgf@x}
759     \inheritanchor[from=circle]{center}

```

```

760 \anchor{west}{%
761   \centerpoint
762   \pgf@xa=\radius
763   \advance\pgf@x by-2\pgf@xa
764 }
765 \anchor{east}{%
766   \centerpoint
767   \pgf@xa=\radius
768   \advance\pgf@x by2\pgf@xa
769 }
770 \anchor{bottom}{%
771   \centerpoint
772 }
773 \anchor{bottom west}{%
774   \centerpoint
775   \pgf@xa=\radius
776   \advance\pgf@x by-2\pgf@xa
777   \advance\pgf@y by-.5\pgflinewidth
778 }
779 \anchor{bottom east}{%
780   \centerpoint
781   \pgf@xa=\radius
782   \advance\pgf@x by2\pgf@xa
783   \advance\pgf@y by-.5\pgflinewidth
784 }

```

Define the shape

```

785 \backgroundpath{%
786   \pgf@x=\radius
787   \pgfutil@tempdima=\pgf@x%
788   \pgfutil@tempdimb=\half\pgf@x%

```

Define the line edges starting from the left edge “b” and ending to the right edge “c”

```

789   \pgf@xb=-2\pgfutil@tempdima
790   \pgf@yb=0\pgfutil@tempdima%
791   \pgf@xc=2\pgfutil@tempdima
792   \pgf@yc=0\pgfutil@tempdima%
793   % If text is present shift shape to center
794   \centerpoint
795   \advance\pgf@xb by\pgf@x
796   \advance\pgf@yb by\pgf@y
797   \advance\pgf@xc by\pgf@x
798   \advance\pgf@yc by\pgf@y
799   % Save centerpoint in "a" (middle line point)
800   \pgf@xa=\pgf@x
801   \pgf@ya=\pgf@y

```

Draw the line from left edge to right edge

```

802   \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
803   \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
804   \pgfpathclose
805 }%
806 }

```

ground Beginning of the ground definition in such a way that allows to stretch its length via the **stretch** factor key. Same operations for anchors

```

807 \pgfdeclareshape{ground}{%

```



```

808 \inheritsavedanchors[from=circle]
809 \saveddimen{\halfradius}{%
810   \pgf@x=.5\pgf@x}

```

Get the multiplied value of `\radius`, which is `(stretch factor)\radius`. Using a *saved* dimension it is possible to correctly update anchors when the ground is longer than default

```

811 \saveddimen{\multipliedradius}{%
812   \pgf@x=2\pgf@x% now equal to \radius
813   \pgfmathsetlength{\pgf@x}{\pgfkeysvalueof{/tikz/stretch factor}*\pgf@x}
814 }
815 \inheritanchor[from=circle]{center}
816 \anchor{north}{%
817   \centerpoint
818   \pgf@xa=\radius%
819   \advance\pgf@y by0.8\pgf@xa
820 }
821 \anchor{south}{%
822   \centerpoint
823   \pgf@xa=\radius%
824   \advance\pgf@y by-0.8\pgf@xa
825 }
826 \anchor{west}{%
827   \centerpoint
828   \pgf@xa=\multipliedradius%
829   \advance\pgf@x by-2.5\pgf@xa
830 }
831 \anchor{east}{%
832   \centerpoint
833   \pgf@xa=\multipliedradius%
834   \advance\pgf@x by2.5\pgf@xa
835 }
836 \anchor{north west}{%
837   \centerpoint
838   \pgf@xa=\radius%
839   \pgf@xb=\multipliedradius%
840   \advance\pgf@y by0.8\pgf@xa
841   \advance\pgf@x by-2.5\pgf@xb
842 }
843 \anchor{north east}{%
844   \centerpoint
845   \pgf@xa=\radius%
846   \pgf@xb=\multipliedradius%
847   \advance\pgf@y by0.8\pgf@xa
848   \advance\pgf@x by2.5\pgf@xb
849 }
850 \anchor{south west}{%
851   \centerpoint
852   \pgf@xa=\radius%
853   \pgf@xb=\multipliedradius%
854   \advance\pgf@y by-0.8\pgf@xa
855   \advance\pgf@x by-2.5\pgf@xb
856 }
857 \anchor{south east}{%
858   \centerpoint
859   \pgf@xa=\radius%
860   \pgf@xb=\multipliedradius%

```

```

861 \advance\pgf@y by-0.8\pgf@xa
862 \advance\pgf@x by2.5\pgf@xb
863 }
864 \anchor{left}{%
865 \centerpoint
866 \pgf@xa=\radius%
867 \pgf@xb=\multipliedradius%
868 \advance\pgf@y by0.8\pgf@xa
869 \advance\pgf@x by-2.5\pgf@xb
870 \advance\pgf@x by2.5\pgf@xa
871 }
872 \anchor{right}{%
873 \centerpoint
874 \pgf@xa=\radius%
875 \pgf@xb=\multipliedradius%
876 \advance\pgf@y by0.8\pgf@xa
877 \advance\pgf@x by2.5\pgf@xb
878 \advance\pgf@x by-2.5\pgf@xa
879 }

```

Define the shape

```

880 \backgroundpath{%
881 \pgf@x=\radius
882 \pgfutil@tempdima=\pgf@x%

```

The horizontal distance between lines has been chosen equal to $5\text{\radius}/10$ (ten lines every five \radiuses), that is equal to $\text{\radius}/2$

```

883 \divide\pgf@x by2\relax
884 \pgfutil@tempdimb=\pgf@x%

```

Calculating beginning and end of lines starting from the first one on the left. The x coordinate of the start point of first line is at left of the center point by a distance equal to $2.5(\text{stretch factor})\text{\radius} - \text{\radius}/2$. Its y coordinate is 0.8\radius above of the center point. The x coordinate of the end point of the first line is $2.5(\text{stretch factor})\text{\radius}$ on the left of the center point and the y coordinate is $0.8\text{\radius} - \text{\radius}/2$ above of it (considering lines slanted by 45° ; this leads to $\tan 45 = 1$)

```

885 % start of first line
886 \pgf@xb=-2.5\pgfutil@tempdima%
887 \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/tikz/stretch factor}*\pgf@xb}
888 \advance\pgf@xb by\pgfutil@tempdimb
889 \pgf@yb=0.8\pgfutil@tempdima%
890 % end of first line
891 \pgf@xc=-2.5\pgfutil@tempdima%
892 \pgfmathsetlength{\pgf@xc}{\pgfkeysvalueof{/tikz/stretch factor}*\pgf@xc}
893 \pgf@yc=\pgf@yb%
894 \advance\pgf@yc by-1\pgfutil@tempdimb% \tan(45)=1
895 % If text is present shift shape to center
896 \centerpoint
897 \advance\pgf@xb by\pgf@x
898 \advance\pgf@yb by\pgf@y
899 \advance\pgf@xc by\pgf@x
900 \advance\pgf@yc by\pgf@y
901 % Save centerpoint in "a" (rectangle center)
902 \pgf@xa=\pgf@x
903 \pgf@ya=\pgf@y

```

Draw the sloped lines starting from upper left edge. This can be divided in 3 phases. Phase 1: I know start and end point of the lines.

1:

```
904 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
905 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
```

2: shift the x coordinate on the right by $\text{\radius}/2$ and the y coordinate below by the same amount (again $\tan 45 = 1$) and then draw

```
906 \advance\pgf@xb by\pgfutil@tempdimb
907 \advance\pgf@yc by-\pgfutil@tempdimb
908 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
909 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
```

3: same as 2

```
910 \advance\pgf@xb by\pgfutil@tempdimb
911 \advance\pgf@yc by-\pgfutil@tempdimb
912 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
913 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
```

Phase 2: I know the start point and length of the lines, so I can calculate the end point. The number of lines depends on the value of `stretch factor`. If this is equal to 1, the number of lines in all phases is equal to 13. If `stretch factor` is not equal to 1 the number of lines in phase 2 is equal to $10(\text{stretch factor}) - 3$.

4:

```
914 \advance\pgf@xb by\pgfutil@tempdimb
915 \pgf@xc=\pgf@xb%
916 \advance\pgf@xc by-1.6\pgfutil@tempdima
917 \pgf@yc=\pgf@yb%
918 \advance\pgf@yc by-1.6\pgfutil@tempdima
919 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
920 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
```

5 to n : create a new macro called `\mec@centrallines` to store the number of lines to be drawn in phase 2 (minus one, the number 4, that is drawn separately). Then use a `\foreach` statement to draw in a compact way all of the lines. Since the cycle creates a group, transformations on coordinates need to be preceded by `\global`, otherwise the effect would be lost at the end of each cycle

```
921 \pgfmathsetmacro{\mec@centrallines}{%
922 10*\pgfkeysvalueof{/tikz/stretch factor}-4}
923 \foreach \x in {1,...,\mec@centrallines}{%
924 \global\advance\pgf@xb by\pgfutil@tempdimb
925 \global\advance\pgf@xc by\pgfutil@tempdimb
926 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
927 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
928 }
```

Phase 3: I know the end point of the lines because it lies on the right margin of the shape.

$n + 1$:

```
929 \advance\pgf@xc by\pgfutil@tempdimb
930 \pgf@xb=2.5\pgfutil@tempdima%
931 \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/tikz/stretch factor}*\pgf@xb}
932 \pgf@yb=0.8\pgfutil@tempdima%
933 \advance\pgf@yb by\pgf@xa
934 \advance\pgf@yb by-1.6\pgfutil@tempdima
935 \advance\pgf@yb by\pgf@xb % should be equal to 1
936 \advance\pgf@yb by-\pgf@xc
```

Reconsider the presence of text on redefined coordinates and draw the line

```

937 \advance\pgf@xb by\pgf@xa
938 \advance\pgf@yb by\pgf@ya
939 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
940 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%

```

$n + 2$:

```

941 \advance\pgf@xc by\pgfutil@tempdimb
942 \advance\pgf@yb by-\pgfutil@tempdimb
943 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
944 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%

```

$n + 3$

```

945 \advance\pgf@xc by\pgfutil@tempdimb
946 \advance\pgf@yb by-\pgfutil@tempdimb
947 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
948 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%

```

Additional phase (4): draw the horizontal line (restoring the default values of points “b” and “c”). The line is lengthened in both directions by 0.3pgflinewidth to avoid edges of the sloped lines to be visible

```

949 \pgf@xb=0pt\relax%
950 \advance\pgf@xb by-2.5\pgfutil@tempdima
951 \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/tikz/stretch factor}*\pgf@xb}
952 \advance\pgf@xb by-0.3\pgflinewidth
953 \pgf@yb=0pt\relax%
954 \advance\pgf@yb by0.8\pgfutil@tempdima
955 \pgf@xc=0pt\relax%
956 \advance\pgf@xc by2.5\pgfutil@tempdima
957 \pgfmathsetlength{\pgf@xc}{\pgfkeysvalueof{/tikz/stretch factor}*\pgf@xc}
958 \advance\pgf@xc by0.3\pgflinewidth
959 \pgf@yc=0pt\relax%
960 \advance\pgf@yc by0.8\pgfutil@tempdima
961 % Reconsidering the presence of text on redefined coordinates
962 \advance\pgf@xb by\pgf@xa
963 \advance\pgf@yb by\pgf@ya
964 \advance\pgf@xc by\pgf@xa
965 \advance\pgf@yc by\pgf@ya
966 % draw the horizontal line
967 \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
968 \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%
969 \pgfpathclose
970 }%
971 }

```

able pendulum b Starting defining the double pendulum internal variant. Same operations for anchors

```

972 \pgfdeclareshape{double pendulum b}{%
973 \inheritsavedanchors[from=circle]
974 \saveddimen{\halfradius}{%
975 \pgf@x=.5\pgf@x}
976 \inheritanchor[from=circle]{center}
977 \anchor{north}{%
978 \centerpoint
979 \pgf@xa=\radius%
980 \pgf@xb=\halfradius%
981 \advance\pgf@y by\pgf@xa

```

```

982     \advance\pgf@y by\pgf@xb
983   }
984   \anchor{north east}{%
985     \centerpoint
986     \pgf@xa=\radius%
987     \pgf@xb=\halfradius%
988     \advance\pgf@x by\pgf@xa
989     \advance\pgf@x by\pgf@xb
990     \advance\pgf@y by\pgf@xa
991     \advance\pgf@y by\pgf@xb
992   }
993   \anchor{north west}{%
994     \centerpoint
995     \pgf@xa=\radius%
996     \pgf@xb=\halfradius%
997     \advance\pgf@x by-\pgf@xa
998     \advance\pgf@x by-\pgf@xb
999     \advance\pgf@y by\pgf@xa
1000    \advance\pgf@y by\pgf@xb
1001  }
1002  \anchor{south}{%
1003    \centerpoint
1004    \pgf@xa=\radius%
1005    \pgf@xb=\halfradius%
1006    \advance\pgf@y by-\pgf@xa
1007    \advance\pgf@y by-\pgf@xb
1008  }
1009  \anchor{south east}{%
1010    \centerpoint
1011    \pgf@xa=\radius%
1012    \pgf@xb=\halfradius%
1013    \advance\pgf@x by\pgf@xa
1014    \advance\pgf@x by\pgf@xb
1015    \advance\pgf@y by-\pgf@xa
1016    \advance\pgf@y by-\pgf@xb
1017  }
1018  \anchor{south west}{%
1019    \centerpoint
1020    \pgf@xa=\radius%
1021    \pgf@xb=\halfradius%
1022    \advance\pgf@x by-\pgf@xa
1023    \advance\pgf@x by-\pgf@xb
1024    \advance\pgf@y by-\pgf@xa
1025    \advance\pgf@y by-\pgf@xb
1026  }
1027  \anchor{west}{%
1028    \centerpoint
1029    \pgf@xa=\radius%
1030    \advance\pgf@x by-\pgf@xa
1031  }
1032  \anchor{east}{%
1033    \centerpoint
1034    \pgf@xa=\radius%
1035    \advance\pgf@x by\pgf@xa
1036  }
1037  \anchor{north east center}{%

```

```

1038     \centerpoint
1039     \pgf@xa=\radius%
1040     \advance\pgf@x by\pgf@xa
1041     \advance\pgf@y by\pgf@xa
1042 }
1043 \anchor{north west center}{%
1044     \centerpoint
1045     \pgf@xa=\radius%
1046     \advance\pgf@x by-\pgf@xa
1047     \advance\pgf@y by\pgf@xa
1048 }
1049 \anchor{south east center}{%
1050     \centerpoint
1051     \pgf@xa=\radius%
1052     \advance\pgf@x by\pgf@xa
1053     \advance\pgf@y by-\pgf@xa
1054 }
1055 \anchor{south west center}{%
1056     \centerpoint
1057     \pgf@xa=\radius%
1058     \advance\pgf@x by-\pgf@xa
1059     \advance\pgf@y by-\pgf@xa
1060 }

```

Even if it shouldn't be necessary, define the `bottom` anchor, which is used to correctly place the ground

```

1061 \anchor{bottom}{%
1062     \centerpoint
1063     \pgf@xa=\radius%
1064     \pgf@xb=\halfradius%
1065     \advance\pgf@y by-\pgf@xa
1066     \advance\pgf@y by-\pgf@xb
1067 }

```

Define the shape

```

1068 \backgroundpath{%
1069     \pgf@x=\radius
1070     \pgfutil@tempdima=\pgf@x%
1071     \pgfutil@tempdimb=\halfradius%

```

Define the rectangle box upper right corner "b" and lower left corner "c". The length of one side is 3\radius

```

1072     \pgf@xb=\pgfutil@tempdima%
1073     \advance\pgf@xb by\pgfutil@tempdimb
1074     \pgf@yb=\pgfutil@tempdima%
1075     \advance\pgf@yb by\pgfutil@tempdimb
1076     \pgf@xc=-\pgfutil@tempdima%
1077     \advance\pgf@xc by-\pgfutil@tempdimb
1078     \pgf@yc=-\pgfutil@tempdima%
1079     \advance\pgf@yc by-\pgfutil@tempdimb
1080     % If text is present shift shape to center
1081     \centerpoint
1082     \advance\pgf@xb by\pgf@x
1083     \advance\pgf@yb by\pgf@y
1084     \advance\pgf@xc by\pgf@x
1085     \advance\pgf@yc by\pgf@y
1086     % Save centerpoint in "a" (top rectangle centerpoint)

```

```

1087      \pgf@xa=\pgf@x
1088      \pgf@ya=\pgf@y

```

Just like the double pendulum, the shape is made of multiple paths so it is necessary to create another unique path that will be filled correctly when needed (patterns are working too)

```

1089      \tikz@mode
1090      \iftikz@mode@fill
1091      \advance\pgf@xb by-\pgfutil@tempdimb
1092      \advance\pgf@xc by\pgfutil@tempdimb
1093      \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}
1094      \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yc}}
1095      \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}
1096      \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yb}}
1097      \pgfpathclose
1098      \pgfusepath{fill}
1099      \advance\pgf@xb by\pgfutil@tempdimb
1100      \advance\pgf@xc by-\pgfutil@tempdimb
1101      \fi

```

Draw upper and lower sides of rectangle box starting from upper side

```

1102      \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}%
1103      \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yb}}%
1104      \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yc}}%
1105      \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%

```

Draw left and right side of the inner box shifting “b” to upper right inner corner and “c” to lower left inner corner

```

1106      \advance\pgf@xb by-\pgfutil@tempdimb
1107      \advance\pgf@yb by-\pgfutil@tempdima
1108      %\advance\pgf@yb by-.5\pgflinewidth
1109      \advance\pgf@xc by\pgfutil@tempdimb
1110      \advance\pgf@yc by\pgfutil@tempdima
1111      %\advance\pgf@yb by.5\pgflinewidth
1112      \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}
1113      \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yc}}
1114      \pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yc}}
1115      \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yb}}

```

Transform coordinates to fit the circle centers and draw them

```

1116      \advance\pgf@yb by\pgfutil@tempdimb
1117      \advance\pgf@yc by-\pgfutil@tempdimb
1118      \edef\pgf@marshal{%
1119          \noexpand\pgfpathcircle{%
1120              \noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}
1121              {\the\pgfutil@tempdimb}%
1122          \noexpand\pgfpathcircle{%
1123              \noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yc}}
1124              {\the\pgfutil@tempdimb}%
1125          \noexpand\pgfpathcircle{%
1126              \noexpand\pgfqpoint{\the\pgf@xc}{\the\pgf@yc}}
1127              {\the\pgfutil@tempdimb}%
1128          \noexpand\pgfpathcircle{%
1129              \noexpand\pgfqpoint{\the\pgf@xc}{\the\pgf@yb}}
1130              {\the\pgfutil@tempdimb}%
1131      }\pgf@marshal
1132      }%

```

