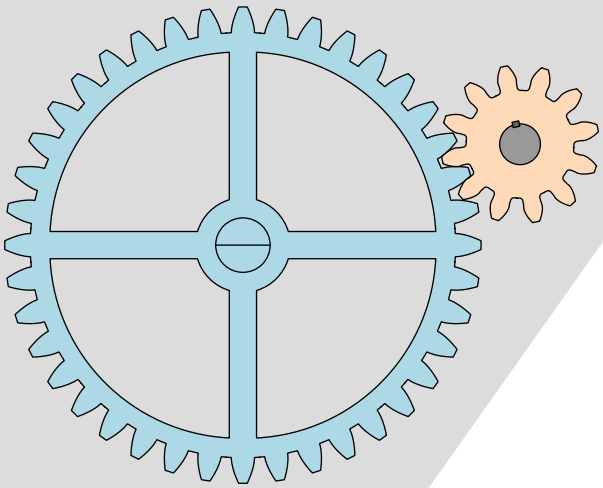
mp-gears

drawing systems of intermeshing gears
with METAPOST



Contributor
Maxime CHUPIN
maxime.chupin@tuta.com

Version 0.1, 2026-04-28
https://gitlab.gutenberg-asso.fr/mchupin/mp-gears

Abstract

This METAPOST package allows to draw system of intermeshing gears.

https://gitlab.gutenberg-asso.fr/mchupin/mp-gears

Contents

This package is in beta version—do not hesitate to report bugs, as well as requests for improvement.

1 Introduction

This package therefore allows users to model various gears and gear systems (and even spirographs). In what follows, we will assume that the reader is familiar with key concepts in gear design, such as the base radius, pitch radius, module, pressure angle, pitch, etc.

For further references, please see [3, 2, 1, 8, 9].

There is the excellent pst-gears package [7] by Manuel Luque and Hebert Voss, which served as a major source of inspiration for me. However, I believe that METAPOST [6] is a language particularly well-suited to the construction of gear systems and that it may offer a bit more flexibility.

Because with METAPOST, we do not need to declare `numeric` variable, with `mp-gears`, we define parameters (in table) associated to un `numeric` identifier and then we manipulate the *gear* object with this `numeric` identifier.

This package and its documentation are developed without any use of generative artificial intelligence: noGIA.

2 Installation

`mp-gears` is on CTAN and can also be installed via the package manager of your distribution.

`https://www.ctan.org/pkg/mp-gears`

2.1 With TEXlive under Linux or macOS

To install `mp-gears` with TEXLive, you will have to create the directory `texmf` in your `home`.

```
user $> mkdir ~/texmf
```

Then, you will have to place the `gears.mp` files in

`~/texmf/metapost/mp-gears/`

Once this is done, `mp-gears` will be loaded with the classic METAPOST input code

```
input gears
```

2.2 With MikTEX and Windows

These two systems are unknown to the author of `mp-gears`, so we refer you to the MikTEXdocumentation concerning the addition of local packages:

`http://docs.miktex.org/manual/localadditions.html`

3 Building Gears

In this section, we present the building macros. These macros define the parameters of the gears, and return *identifiers* to which the parameters are attached. Hence, these functions do not produce any drawing. However, in order to help the readers to understand, we will use *drawing macros* without presenting them. For details about them, see section 4.

3.1 External Gears

The first command is for building one gear. `mp-gears` offer three kinds of gear, *external*, *internal* and *rack*. We start with the first case, more classical. For the second kind, see sectio 3.2, and the third case is described in section 3.3.

`buildExternalGear(⟨Z1⟩,⟨m⟩,⟨a⟩,⟨c⟩, ⟨r⟩)` → `numeric`

⟨Z1⟩: `numeric` number of teeth of the gear.

⟨m⟩: `numeric` module of the gear.

⟨a⟩: `numeric` pressure angle.

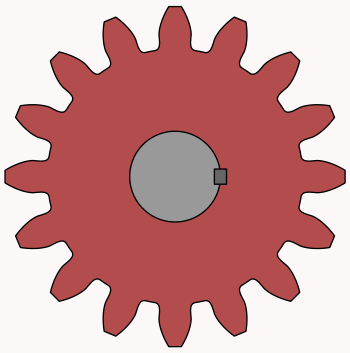
⟨c⟩: `pair` center of the gear.

⟨r⟩: `numeric` angle of rotation of the gear.

Here, a simple example.

Exemple 1

```
input gears

beginfig(1);
A = buildExternalGear(16,0.5,20,(0,0),0);
drawGear(A,(0.7,0.3,0.3),"classical");
endfig;
```



The next command is maybe more useful because it builds two enmeshed gears.

`buildExternalGearPair(⟨Z1⟩,⟨Z2⟩,⟨m⟩,⟨a⟩,⟨c⟩, ⟨r⟩, ⟨ar⟩)` → `pair`

This function builds two intermeshed gears and give a `pair` of identifiers.

⟨Z1⟩: `numeric` number of teeth of the first gear.

⟨Z2⟩: `numeric` number of teeth of the second gear.

⟨***m***⟩: `numeric` module of the system.

⟨***a***⟩: `numeric` pressure angle.

⟨***c***⟩: `pair` center of the first gear.

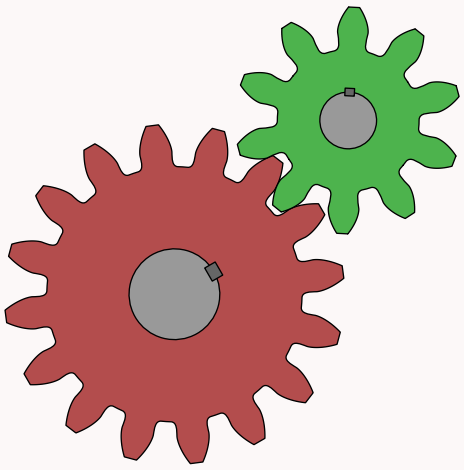
⟨***r***⟩: `numeric` angle of rotation of the first gear. The second gear will be rotated with the corresponded linked angle.

⟨***ar***⟩: `numeric` angle of rotation of the center of the second gear around the center of the first gear.

Here, a simple example.

Exemple 2

```
input gears

beginfig(1);
(A1,A2) = buildExternalGearPair(16,10,0.5,20,(0,0),30,45);
drawGear(A1,(0.7,0.3,0.3),"classical");
drawGear(A2,(0.3,0.7,0.3),"classical");
endfig;
```



You can also create a gear that needs to mesh with an existing one. To do this, use the following macro. This macro also works if the existing gear is an internal one (see section 3.2).

`buildExternalGearFor(`⟨*id*⟩`,`⟨*Z2*⟩`,`⟨*ar*⟩`)` → `numeric`

⟨***id***⟩: `numeric` identifier to the existing gear.

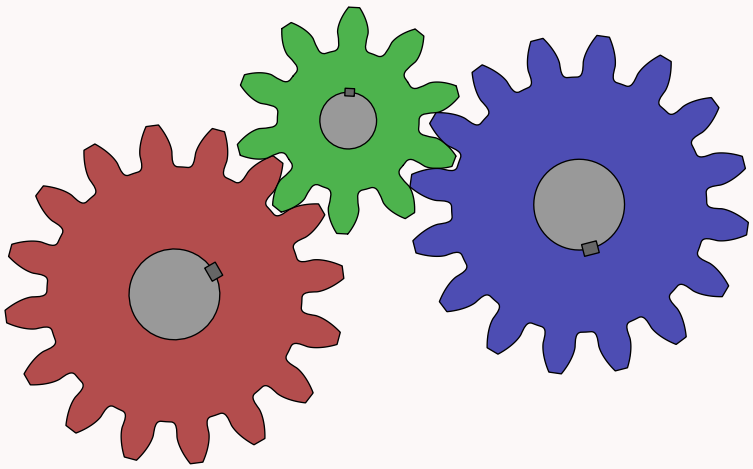
⟨***Z2***⟩: `numeric` number of teeth of the new gear.

⟨***ar***⟩**:** numeric angle of rotation of the center of the new gear around the center of the existing gear.

Here, a simple example.

Exemple 3

```
input gears

beginfig(1);
(A1,A2) = buildExternalGearPair(16,10,0.5,20,(0,0),30,45);
A3 = buildExternalGearFor(A2,16,-20);
drawGear(A1,(0.7,0.3,0.3),"classical");
drawGear(A2,(0.3,0.7,0.3),"classical");
drawGear(A3,(0.3,0.3,0.7),"classical");
endfig;
```



One can build an external gear compounded to another one in the sens that they only share the same center (axis of rotation) and the same angle of rotation.

`buildCompoundGear(`⟨*id*⟩`,`⟨*Za*⟩`,`⟨*m*⟩`,`⟨*a*⟩`)` → pair

This function builds two intermeshed gears and give a pair of identifiers.

⟨***id***⟩**:** numeric identifier to the existing gear.

⟨***Za***⟩**:** numeric number of teeth of the new gear.

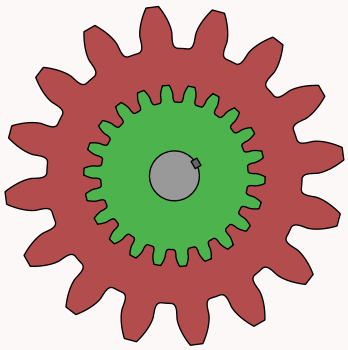
⟨***m***⟩**:** numeric module of the new gear.

⟨***a***⟩**:** numeric pressure angle of the new gear.

Here, a simple example.

Exemple 4

```
input gears

beginfig(1);
A1 = buildExternalGear(16,0.5,20,(0,0),30);
A2 = buildCompoundGear(A1,22,0.2,14);
drawGear(A1,(0.7,0.3,0.3),"classical");
drawGear(A2,(0.3,0.7,0.3),"classical");
endfig;
```



3.2 Internal Gears

The equivalent macros exist for internal gear. Fist, the simpler one to build only one internal gear.

`buildInternalGear(⟨Z1⟩,⟨m⟩,⟨a⟩,⟨c⟩, ⟨r⟩)` → `numeric`

⟨***Z1***⟩: `numeric` number of teeth of the gear.

⟨***m***⟩: `numeric` module of the gear.

⟨***a***⟩: `numeric` pressure angle.

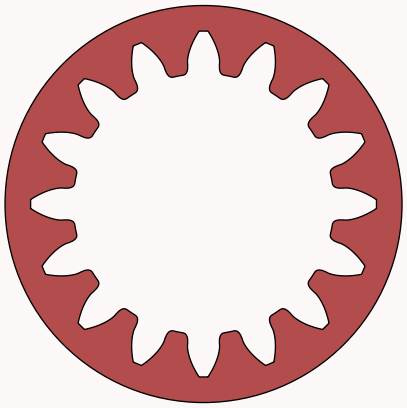
⟨***c***⟩: `pair` center of the gear.

⟨***r***⟩: `numeric` angle of rotation of the gear.

Here, a simple example.

Exemple 5

```
input gears

beginfig(1);
A = buildInternalGear(16,0.5,20,(0,0),0);
drawGear(A,(0.7,0.3,0.3),"classical");
endfig;
```



Of course, you cannot mesh two internal gears. Hence, the following macro build two gears, the first is an internal one, and the second an external one.

`buildInternalGearPair(`⟨*Z1*⟩`,`⟨*Z2*⟩`,`⟨*m*⟩`,`⟨*a*⟩`,`⟨*c*⟩`, `⟨*r*⟩`, `⟨*ar*⟩`)` → `pair`

This function builds two intermeshed gears and give a `pair` of identifiers.

⟨***Z1***⟩**:** `numeric` number of teeth of the first gear.

⟨***Z2***⟩**:** `numeric` number of teeth of the second gear.

⟨***m***⟩**:** `numeric` module of the system.

⟨***a***⟩**:** `numeric` pressure angle.

⟨***c***⟩**:** `pair` center of the first gear.

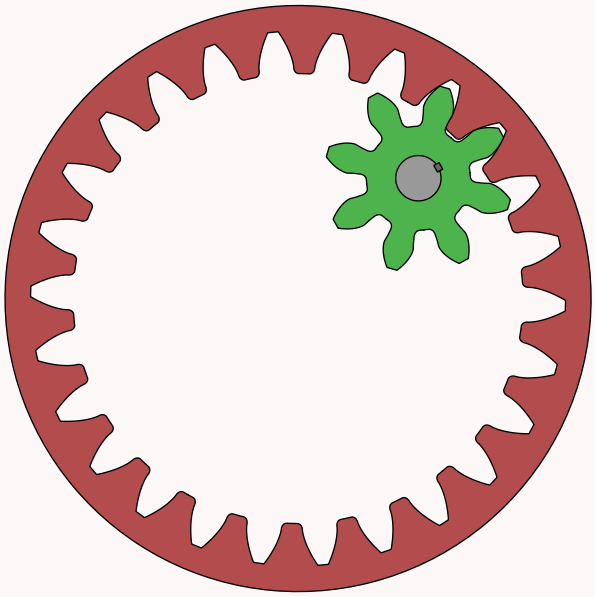
⟨***r***⟩**:** `numeric` angle of rotation of the first gear. The second gear will be rotated with the corresponded linked angle.

⟨***ar***⟩**:** `numeric` angle of rotation of the center of the second gear around the center of the first gear.

Here, a simple example.

Exemple 6

```
input gears

beginfig(1);
(A1,A2) = buildInternalGearPair(26,8,0.5,20,(0,0),40,45);
drawGear(A1,(0.7,0.3,0.3),"classical");
drawGear(A2,(0.3,0.7,0.3),"classical");
endfig;
```



We can also build a internal gear meshed to an existing external one.

`buildInternalGearFor(`⟨*id*⟩`,`⟨*Z2*⟩`,`⟨*ar*⟩`)` → `numeric`

⟨***id***⟩**:** `numeric` identifier to the existing gear.

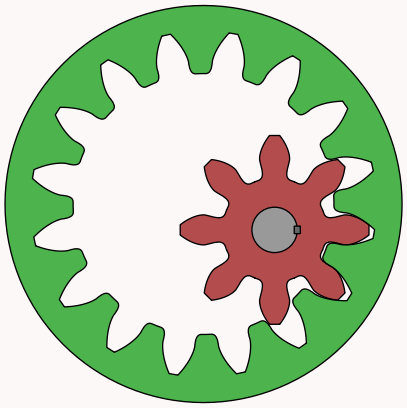
⟨***Z2***⟩**:** `numeric` number of teeth of the new gear.

⟨***ar***⟩**:** `numeric` angle of rotation of the center of the new gear around the center of the existing gear.

Here, a simple example.

Exemple 7

```
input gears

beginfig(1);
A = buildExternalGear(8,0.5,20,(0,0),0);
B = buildInternalGearFor(A,16,-20);
drawGear(A,(0.7,0.3,0.3),"classical");
drawGear(B,(0.3,0.7,0.3),"classical");
endfig;
```



3.3 Rack

There exist another kind of gear: rack. It can be viewed as an external gear with infinite radius. Because of the singularity, we use another type of object, distinct from internal and external gear. However, macros related to racks are quite similar to the ones for internal and external gears.

The rack is defined by default vertically.

`buildRack(⟨Z1⟩,⟨m⟩,⟨a⟩,⟨c⟩,⟨r⟩,⟨t⟩)` → `numeric`

⟨***Z1***⟩: `numeric` number of teeth of the rack. This should be odd and if the user chooses an even number, 1 is added.

⟨***m***⟩: `numeric` module of the rack.

⟨***a***⟩: `numeric` pressure angle.

⟨***c***⟩: `pair` center of the rack on the equivalent of the radius of an external gear (which becomes a line). It is on the middle of the middle tooth.

⟨***r***⟩: `numeric` angle of rotation of the rack around the center.

⟨***t***⟩: `numeric` translation of the rack.

Here, a simple example.

Exemple 8

```
input gears

beginfig(1);
A = buildRack(5,0.5,20,(0,0),0,0);
drawRack(A,(0.7,0.3,0.3));
endfig;
```

We can only mesh a rack with an external gear. This is done by the following macro.

`buildGearRackPair(⟨Z1⟩,⟨Z2⟩,⟨m⟩,⟨a⟩,⟨c⟩, ⟨r⟩, ⟨ar⟩)` → `pair`

⟨Z1⟩: `numeric` number of teeth of the gear.

⟨Z2⟩: `numeric` number of teeth of the rack.

⟨m⟩: `numeric` module of the system.

⟨a⟩: `numeric` pressure angle.

⟨c⟩: `pair` center of the gear.

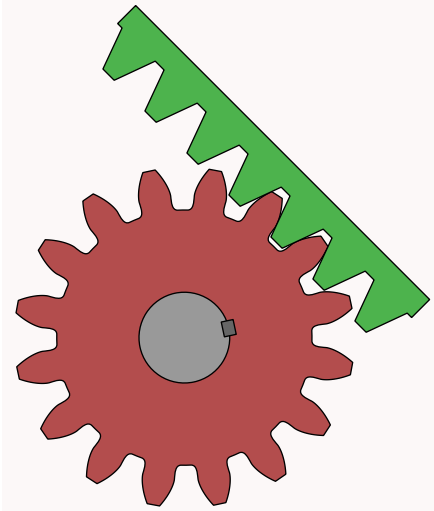
⟨r⟩: `numeric` angle of rotation of the gear. The rack will be *translated* with the corresponded length.

⟨ar⟩: `numeric` angle of rotation of the center of the rack around the center of the gear.

Here, a simple example.

Exemple 9

```
input gears

beginfig(1);
(A1,A2) = buildGearRackPair(16,7,0.5,20,(0,0),12,45);
drawGear(A1,(0.7,0.3,0.3),"classical");
drawRack(A2,(0.3,0.7,0.3));
endfig;
```



Like the other kinds of gear, we can define a rack meshed with an existing external gear.

`buildRackFor(`⟨*id*⟩`,`⟨*Z2*⟩`,`⟨*ar*⟩`)` → `numeric`

⟨***id***⟩**:** `numeric` identifier to the existing gear.

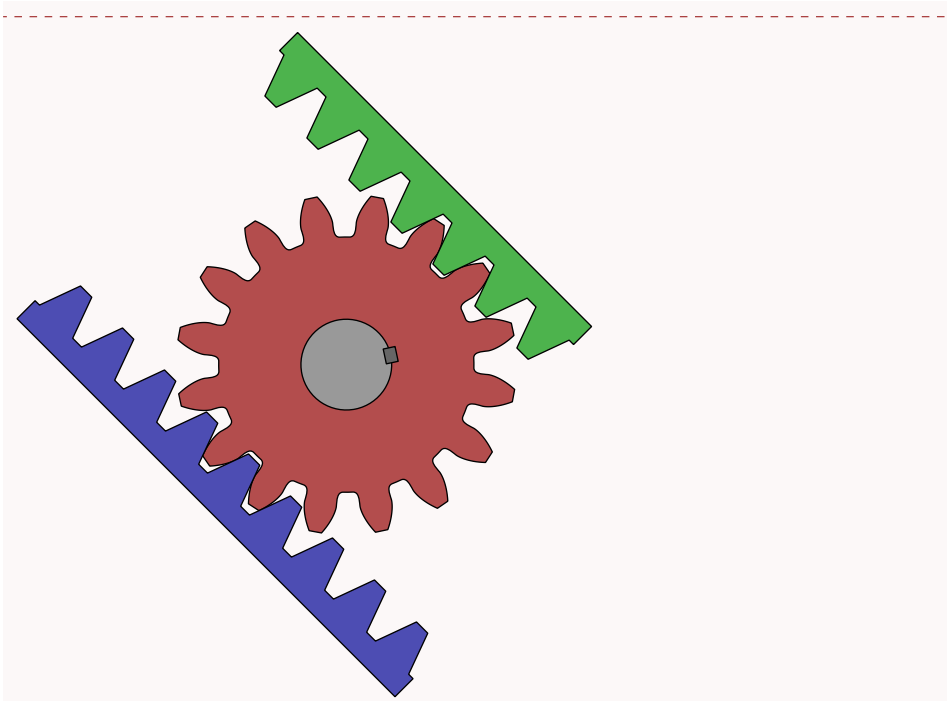
⟨***Z2***⟩**:** `numeric` number of teeth of the rack.

⟨***ar***⟩**:** `numeric` angle of rotation of the center of the rack around the center of the existing gear.

Here, a simple example.

Exemple 10

```
input gears

beginfig(1);
(A1,A2) = buildGearRackPair(16,7,0.5,20,(0,0),12,45);
A3 = buildRackFor(A1,9,180+45);
drawGear(A1,(0.7,0.3,0.3),"classical");
drawRack(A2,(0.3,0.7,0.3));
drawRack(A3,(0.3,0.3,0.7));
endfig;
```



4 Drawing

There is only one macro to draw a gear. Depending on the type of the gear, there are several styles of drawing: three for external gear and one pour internal gear.

4.1 Unit

Drawings produced with `mp-gears` use an unit for *x* and *y* axis of METAPOST. The default value is 0.5 cm. To get the unit, you can use the following macra.

```
getUnit → numeric
```

To change the value of the `mp-gears` unit, you can use the following macro.

```
setUnit(⟨u⟩)
```

⟨***u***⟩ : `numeric` the new value for `mp-gears` unit.

4.2 Gear and Rack

`drawGear(⟨id⟩,⟨c⟩,⟨s⟩)`

⟨id⟩: `numeric` identifier to the existing gear.

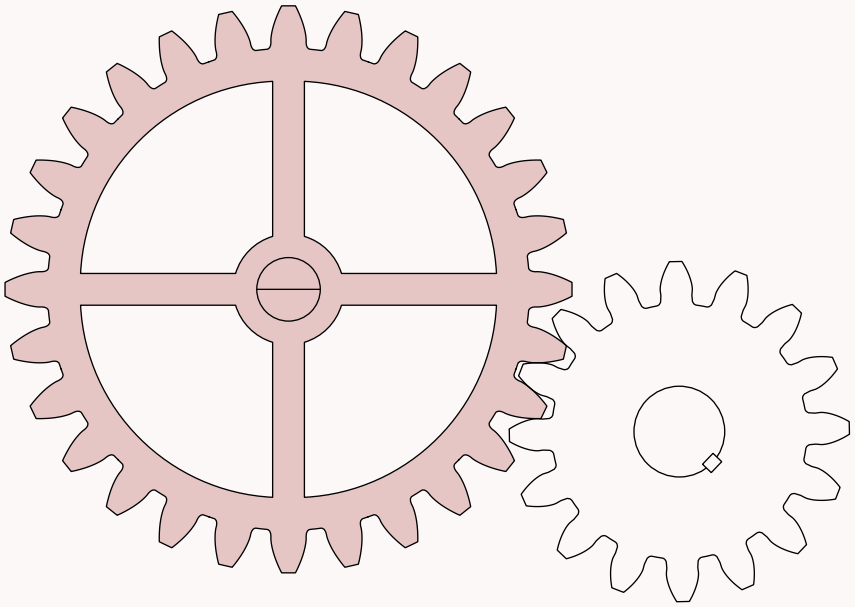
⟨c⟩: `color` the color of the gear or `string "None"`. In the last case, the gear is not colored.

⟨s⟩: `string` type of gear. `"classical"` for both internal and external gear, and `"clockwork"` and `"spirograph"` for external gear.

Because this macro produce a `picture` and a code using the `draw` METAPOST macro, it is possible to add any drawing compatible METAPOST code at this end of it. For instance, because we use luamplib [4] for this documentation, we can the PDF transparency.

Exemple 11

```
input gears

beginfig(1);
A = buildExternalGear(28,0.5,20,(0,0),0);
B = buildExternalGearFor(A,16,-20);
drawGear(A,(0.7,0.3,0.3),"clockwork") withtransparency(1,0.3);
drawGear(B,"None","classical");
endfig;
```



When the `"clockwork"` style is used, the gear will be hollow if the primitive radius is greater than three times the tooth height. Otherwise, the gear will be

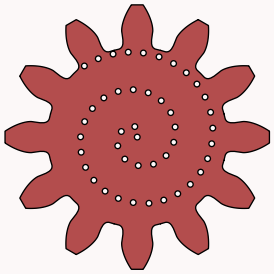
solid, much like with the `"classical"` style.

For internal gear and rack, there is only one style of drawing.

The `"spirograph"` type is illustrated in the following example.

Exemple 12

```
input gears

beginfig(1);
A = buildExternalGear(12,0.5,20,(0,0),0);
drawGear(A,(0.7,0.3,0.3),"spirograph");
endfig;
```



It comes with other macros to draw pictures produced by a spirograph. We refer to section 5.3 for details.

For racks, we have a similar macro.

`drawRack(`⟨*id*⟩`,`⟨*c*⟩`)`

⟨***id***⟩: `numeric` identifier to the existing rack.

⟨***c***⟩: `color` the color of the rack or `string "None"`. In the last case, the rack is not colored.

4.3 Different Colors

By default, lines are drawn with black color. You can change that using the following command.

`setLineColor(`⟨*c*⟩`)`

⟨***c***⟩ : `color` the new value for line color.

When the `"classical"` style is used, the axis and the key color are drawn with respectively color `(0.6,0.6,0.6)` and `(0.4,0.4,0.4)`. You can change that using the two following macros.

`setAxisColor(⟨c⟩)`

⟨***c***⟩ : `color` the new value for axis color.

`setKeyColor(⟨c⟩)`

⟨***c***⟩ : `color` the new value for key color.

5 Other Macros

5.1 Getting Parameters

In this section, we list simple macros to get parameters of the gear or the rack. Most of them do not need explanation. Once again, all the dimensions are scaled by the `mp-gears` unit.

`getPrimitiveRadius(⟨id⟩)` → `numeric`

⟨***id***⟩ : `numeric` identifier of the gear (rack does not have radius parameters).

`getTopRadius(⟨id⟩)` → `numeric`

⟨***id***⟩ : `numeric` identifier of the gear (rack does not have radius parameters).

`getBottomRadius(⟨id⟩)` → `numeric` (in degree)

⟨***id***⟩ : `numeric` identifier of the gear (rack does not have radius parameters).

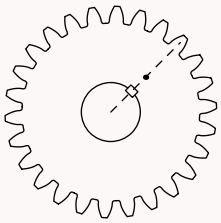
`getRadialPoint(⟨id⟩,⟨t⟩)` → `pair`

⟨***id***⟩ : `numeric` identifier of the gear (not rack).

⟨***t***⟩**:** `numeric` between 0 and 1 to get the point $c + r(\cos(\theta), \sin(\theta))$ where c is the center of the gear, r is the primitive radius, and θ the angle of rotation.

Exemple 13

```
input gears
beginfig(1);
A = buildExternalGear(26,0.2,20,(0,0),45);
drawGear(A,"None","classical");
draw getCenter(A)--getRadialPoint(A,1) dashed evenly;
drawdot getRadialPoint(A,0.5) withpen pencircle scaled 2pt;
endfig;
```



`getPressureAngle(`⟨*id*⟩`)` → `numeric` (in degree)

⟨***id***⟩ : `numeric` identifier of the gear or rack.

`getModule(`⟨*id*⟩`)` → `numeric`

⟨***id***⟩ : `numeric` identifier of the gear or the rack.

`getTeethNbr(`⟨*id*⟩`)` → `numeric`

⟨***id***⟩ : `numeric` identifier of the gear or the rack.

`getPrimitivePitch(`⟨*id*⟩`)` → `numeric`

⟨***id***⟩ : `numeric` identifier of the gear or the rack.

`getCenter(`⟨*id*⟩`)` → `pair`

⟨***id***⟩ : `numeric` identifier of the gear or the rack. For racks, the center is either in the middle tooth, or if it is meshed at the point of contact with the primitive radius of the gear.

`getToothHeight(⟨id⟩)` → `numeric`

⟨**id**⟩ : `numeric` identifier of the gear or the rack.

`getRotation(⟨id⟩)` → `numeric`

⟨**id**⟩ : `numeric` identifier of the gear or the rack.

`getTranslation(⟨id⟩)` → `numeric`

⟨**id**⟩ : `numeric` identifier of the rack (gears do not have translation parameter).

5.2 Getting construction objects

There are some macros that help to get the different construction circles and lines (respectively for gears and racks).

5.2.1 Circles

Beware, these macros only work for gears.

`primitiveCircle(⟨id⟩)` → `path`

⟨**id**⟩ : `numeric` identifier of the gear.

`baseCircle(⟨id⟩)` → `path`

⟨**id**⟩ : `numeric` identifier of the gear.

`bottomCircle(⟨id⟩)` → `path`

⟨**id**⟩ : `numeric` identifier of the gear.

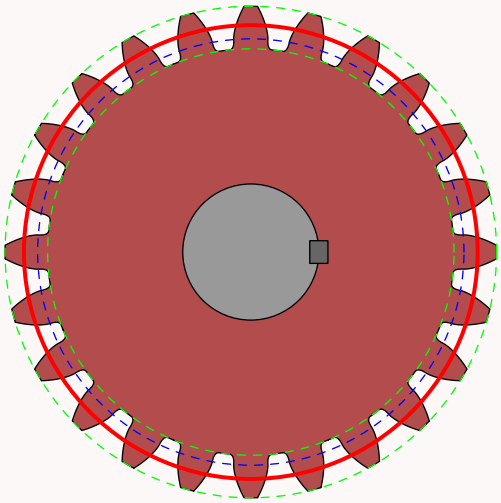
`topCircle(⟨id⟩)` → `path`

⟨**id**⟩ : `numeric` identifier of the gear.

Here, an example of use of these macros.

Exemple 14

```
input gears

beginfig(1);
A = buildExternalGear(24,0.5,20,(0,0),0);
drawGear(A,(0.7,0.3,0.3),"classical");
draw primitiveCircle(A) withpen pencircle scaled 1.5pt
     withcolor red;
draw baseCircle(A) dashed evenly  withcolor blue;
draw bottomCircle(A) dashed evenly  withcolor green;
draw topCircle(A) dashed evenly  withcolor green;
endfig;
```



5.2.2 Lines

Beware, these macros are only for rack objects.

`primitiveLine(`⟨*id*⟩`)` → `path`

⟨***id***⟩ : `numeric` identifier of the rack.

`baseLine(`⟨*id*⟩`)` → `path`

⟨***id***⟩ : `numeric` identifier of the rack.

`bottomLine(⟨id⟩)` → `path`

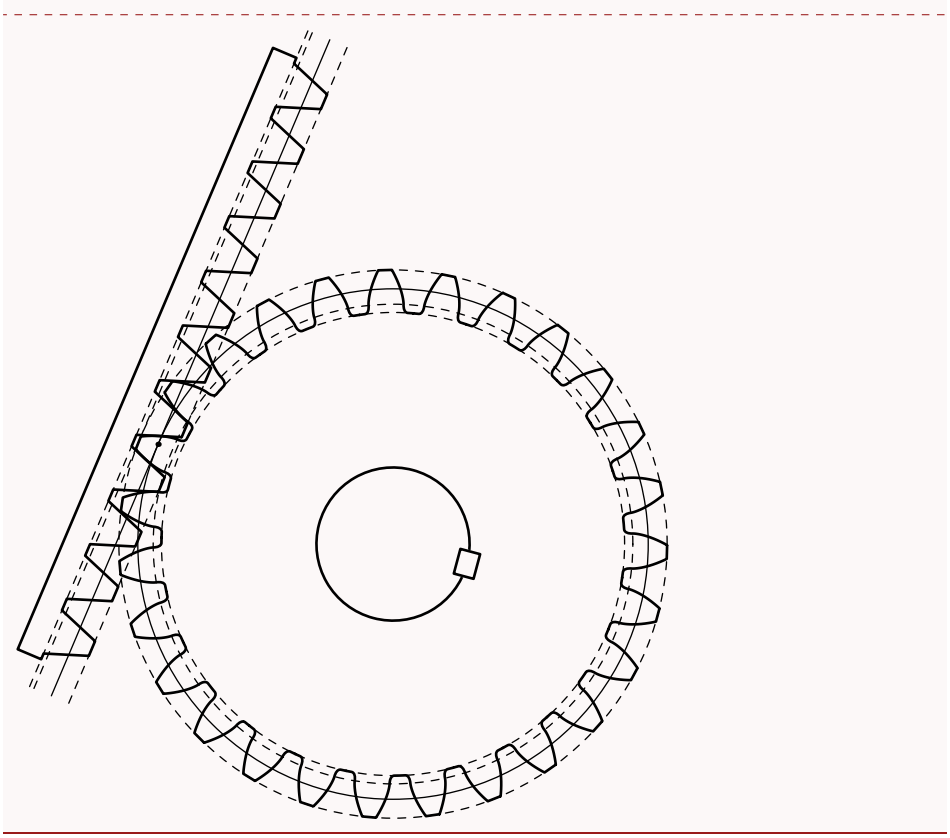
⟨**id**⟩ : `numeric` identifier of the rack.

`topLine(⟨id⟩)` → `path`

⟨**id**⟩ : `numeric` identifier of the rack.

Exemple 15

```
input gears

beginfig(1);
  (Q,P) = buildGearRackPair(27,10,0.5,20,(0,0),-15,157);
  drawGear(Q,"None","classical") withpen pencircle scaled 1pt;
  drawRack(P,"None") withpen pencircle scaled 1pt;
  drawdot getCenter(P) withpen pencircle scaled 2pt;
  draw primitiveLine(P);
  draw baseLine(P) dashed evenly;
  draw bottomLine(P) dashed evenly;
  draw topLine(P) dashed evenly;
  draw primitiveCircle(Q);
  draw baseCircle(Q) dashed evenly;
  draw bottomCircle(Q) dashed evenly;
  draw topCircle(Q) dashed evenly;
endfig;
```



5.3 Spirograph

`mp-gears` provides a number of macros for creating Spirograph illustrations. The building of a *spirograph* gear compute points along the Archimedes spiral. We can get these points using the following macros. The number of such points will depend on the radius of the external gear.

`getSpirographPointNbr(`⟨*id*⟩`)` → `numeric`

⟨***id***⟩ : `numeric` identifier of the external gear.

`getSpirographPoint(`⟨*id*⟩`,`⟨*n*⟩`)` → `numeric`

⟨***id***⟩ : `numeric` identifier of the external gear.

⟨***n***⟩**:** `numeric`, integer between 0 and `getSpirographPointNbr(`⟨*id*⟩`)`.

Thanks to all these macros, one can build the trajectories of a *spirograph* point of the gear. However, `mp-gears` provides a dedicated macro.

`spirographCurve(⟨A⟩,⟨B⟩,⟨n⟩,⟨t⟩)` → `path`

⟨A⟩: `numeric` identifier of the fixed gear.

⟨B⟩: `numeric` identifier of the rotating gear.

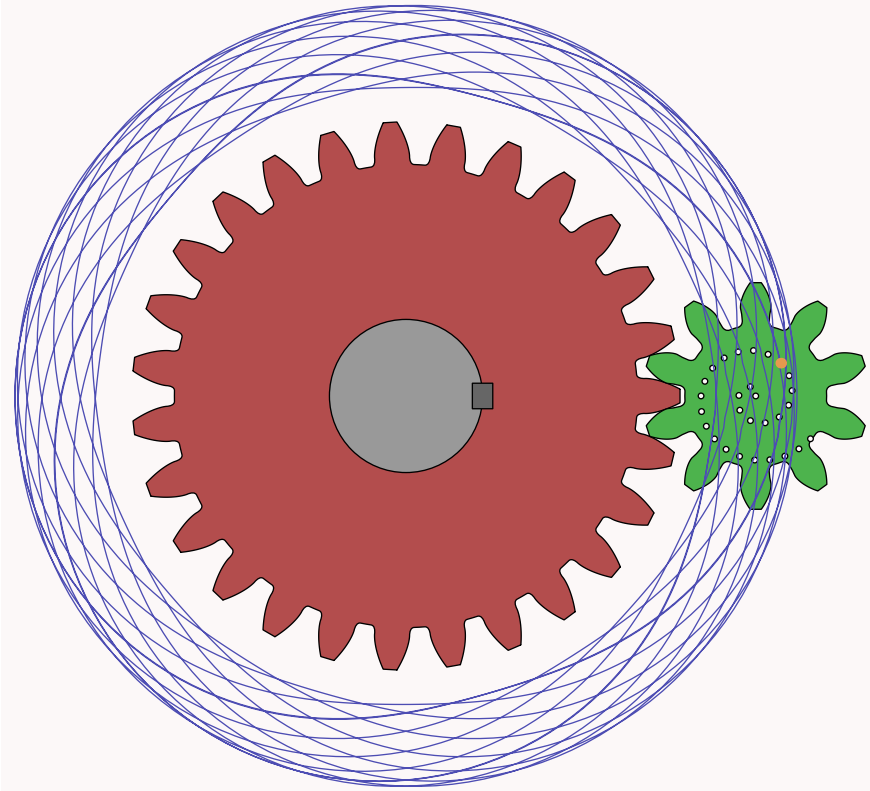
⟨n⟩: `numeric`, integer between 0 and `getSpirographPointNbr(⟨id⟩)`.

⟨t⟩: `numeric` number of revolutions around the fixed gear.

Here, are two examples. Because one revolution is 360 degrees, if we use the default number system of METAPOST, we are limited to very few revolutions.

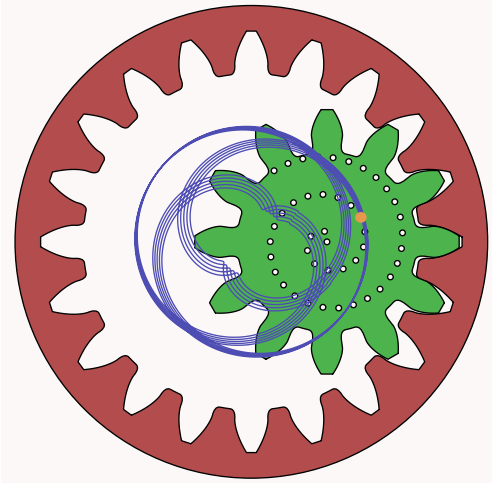
Exemple 16

```
input gears
% double number system
beginfig(1);
  N:=10;
  (Q,P) = buildExternalGearPair(27,10,0.5,20,(0,0),0,0);
  drawGear(Q,(0.7,0.3,0.3),"classical");
  drawGear(P,(0.3,0.7,0.3), "spirograph");
  draw spirographCurve(Q,P,N,15) withcolor (0.3,0.3,0.7);
  drawdot getSpirographPoint(P,N) withpen pencircle scaled 4pt
        withcolor (0.9,0.6,0.3);
endfig;
```



Exemple 17

```
input gears
% double number system
beginfig(1);
  N:=10;
  (Q,P) = buildInternalGearPair(20,12,0.5,20,(0,0),0,0);
  drawGear(Q,(0.7,0.3,0.3),"classical");
  drawGear(P,(0.3,0.7,0.3), "spirograph");
  draw spirographCurve(Q,P,N,15) withcolor (0.3,0.3,0.7);
  drawdot getSpirographPoint(P,N) withpen pencircle scaled 4pt
        withcolor (0.9,0.6,0.3);
endfig;
```



You can get or modify the size of the mark of the spirograph point on the gear with the following macros. In the `mp-gears` unit, the size is 0.15 by default.

`getSpirographPointSize` → `numeric`

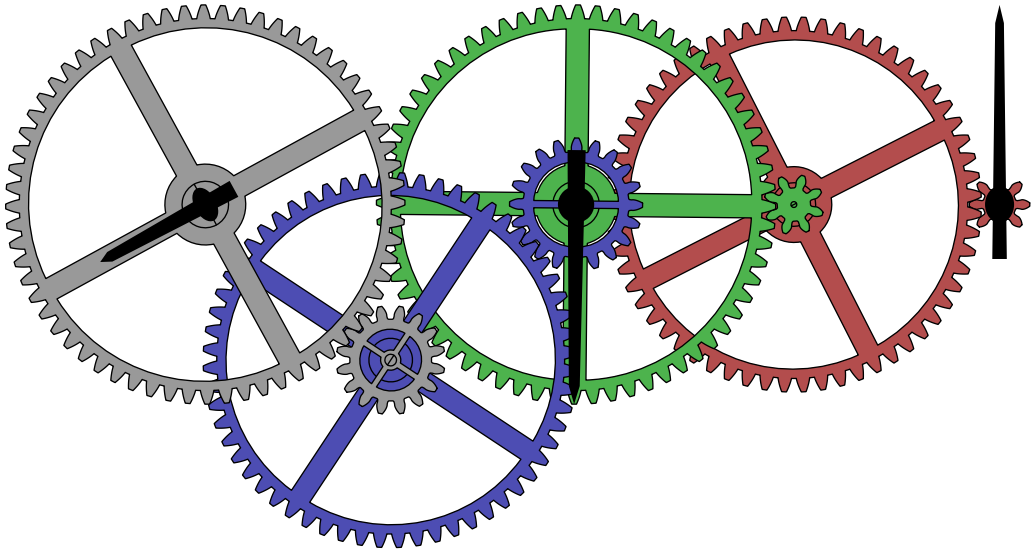
`setSpirographPointSize(`⟨*s*⟩`)`

⟨**s**⟩: `numeric` size of the spirograph points (0.15 by default).

6 Gallery

6.1 Clock Gear Train

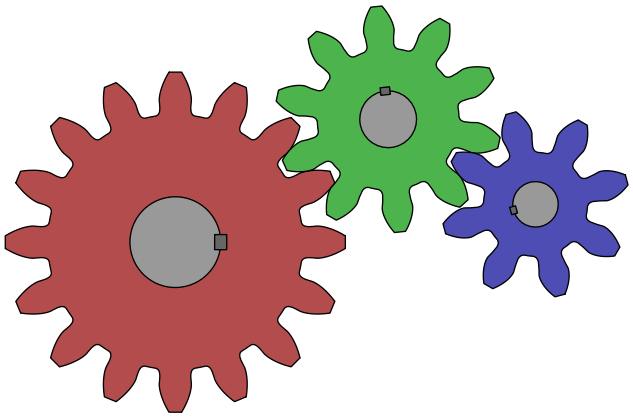
Inspired by the remarkable work of Tavmjong Bah on the website: http://tavmjong.free.fr/INKSCAPE/DRAWINGS/clock.svg and pst-gears.



```
    input gears
color couleur[];
couleur1 := (0.7,0.3,0.3);
couleur2 := (0.3,0.7,0.3);
couleur3 := (0.3,0.3,0.7);
couleur4 := (0.6,0.6,0.6);

picture aiguille;
setUnit(1cm);
m := 0.08;
alpha := 20;

aiguille := image(
    R := 60*m/2;
    fill (-0.05*R,-0.3*R)--(0.05*R,-0.3*R)
         --(0.03*R,R)--(0,1.1*R)--
    (-0.03*R,R)--cycle;
    fill fullcircle scaled (0.1*2*R);
) scaled getUnit;

beginfig(1);

    rot := 0;
    (G1,G2) = buildExternalGearPair(8,60,m,alpha
         ,(0,0),rot,180);
    G3 = buildCompoundGear(G2,8,m,alpha);
    G4 = buildExternalGearFor(G3,64,180);
    G5 = buildCompoundGear(G4,20,m,alpha);
    G6 = buildExternalGearFor(G5,60,220);
    G7 = buildCompoundGear(G6,16,m,alpha);
    G8 = buildExternalGearFor(G7,64,140);

    drawGear(G1,couleur1,"clockwork");
    drawGear(G2,couleur1,"clockwork");
    drawGear(G3,couleur2,"clockwork");
    drawGear(G4,couleur2,"clockwork");
    drawGear(G5,couleur3,"clockwork");
    drawGear(G6,couleur3,"clockwork");
    drawGear(G7,couleur4,"clockwork");
    drawGear(G8,couleur4,"clockwork");

    draw aiguille xscaled 0.8 rotated
         getRotation(G1) shifted getCenter(G1);
    draw aiguille rotated getRotation(G5)
         shifted getCenter(G5);
    draw aiguille yscaled 0.6 rotated
         getRotation(G8) shifted getCenter(G8);
endfig;
end.
```

6.2 Gears Animation

We can use the package animate [5] to animate rotation of gears.

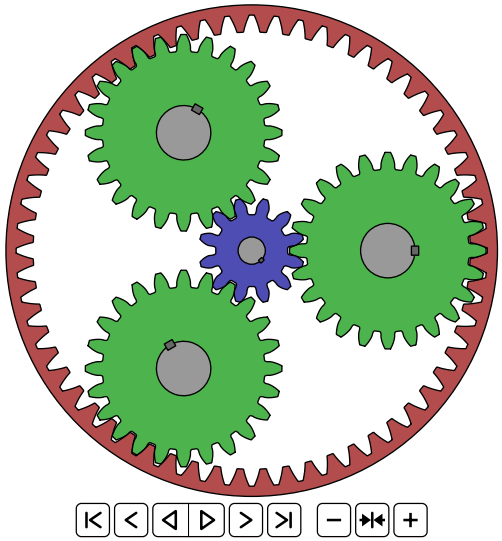


```
\begin{animateinline}[poster=first,loop,
     controls]{12}%
\multiframe{360}{iAngle=0+2}{%
\begin{mplibcode}
input gears
```

```
beginfig(1);
    rot := \iAngle;
    axerot := 30;
    axerot3 := -30;
    pair gears;
    gears := buildExternalGearPair
```

```
     (16,10,0.5,20,(0,0),rot,axerot);
G1 := xpart gears;
G2 := ypart gears;
G3 := buildExternalGearFor(G2,8,axerot3);

drawGear(G1, (0.7,0.3,0.3),"classical");
drawGear(G2, (0.3,0.7,0.3),"classical");
    drawGear(G3, (0.3,0.3,0.7),"classical");

endfig;
\end{mplibcode}
}
\end{animateinline}
```

6.3 Gears Animation (2)



```
\begin{animateinline}[poster=first, controls
     ]{12}%
\multiframe{360}{iAngle=0+2}{%
\begin{mplibcode}
input gears

beginfig(2);
    rot := \iAngle;
    axerot := 0;
    (G[7],G[8]) = buildInternalGearPair
        (60,24,0.2,20,(0,0),rot,axerot);
G9 = buildExternalGearFor(G7,24,120);
G10 = buildExternalGearFor(G7,24,240);
G11 = buildExternalGearFor(G10,12,60);
drawGear(G7,(0.7,0.3,0.3),"classical");
drawGear(G8,(0.3,0.7,0.3),"classical");
drawGear(G9,(0.3,0.7,0.3),"classical");
drawGear(G10,(0.3,0.7,0.3),"classical");
drawGear(G11,(0.3,0.3,0.7),"classical");
```

```
endfig;
\end{mplibcode}
```

```
}
\end{animateinline}
```

References

[1] André Chevalier. *Guide du dessinateur industriel*. 6th ed. Hachette Technique, 2003. ISBN: 978-2011689036.

[2] Georges Chevasson and Ange Pezet. *Formulaire du dessinateur et du technicien*. Paris: Desforges, 1977.

[3] Ginette Cuisinier and Marie-France Guissard. "Engrenages et développantes de cercle". In: *Losanges* 18 (2011). URL: https://www.sbpm.be/download/43c5d59f83254e46bd7ee36869cd2837/18Losanges/18CuisinierGuissard.pdf (visited on 04/28/2026).

[4] Philipp Gesang et al. *The luamplib package. Use LuaTeX's built-in MetaPost interpreter*. Version 2.41.0. Apr. 28, 2026. URL: https://ctan.org/pkg/luamplib (visited on 04/28/2026).

[5] Alexander Grahn. *The animate package. Create PDF and SVG animations from graphics files and inline graphics*. Oct. 14, 2024. URL: https://ctan.org/pkg/animate (visited on 04/30/2026).

[6] John Hobby and The MetaPost Team. *The metapost package. A development of Metafont for creating graphics*. URL: https://tug.org/metapost (visited on 04/28/2026).

[7] Herbert Voß. *The pst-gears package. Drawing internal and external gears*. Version 0.61. URL: https://ctan.org/pkg/pst-gears (visited on 04/28/2026).

[8] Wikipedia contributors. *Gear*. Wikipédia, l'encyclopédie libre. 2026. URL: https://en.wikipedia.org/wiki/Gear (visited on 04/28/2026).

[9] Zpag.net. *Engrenages droits à denture droite*. Consulté le 28 avril 2026. n.d. URL: https://www.zpag.net/Machines_Simples/engrenage_droit_dent_droit.htm (visited on 04/28/2026).

Command Index