

# Contents

|          |                              |          |
|----------|------------------------------|----------|
| <b>1</b> | <b>Introduction</b>          | <b>2</b> |
| 1.1      | Objects types list . . . . . | 2        |
| <b>2</b> | <b>Temp: all functions</b>   | <b>2</b> |
| <b>3</b> | <b>The type line3</b>        | <b>4</b> |
| <b>4</b> | <b>The type sphere3</b>      | <b>4</b> |

# Introduction

Before using this package, make sure, that you have this settings:

```
settings.outformat = "pdf";  
settings.render = 0;  
settings.prc = false;
```

and specified size of picture by `size3`. Also, you have to wrap your code into function (say `main`) and put `with_geometry3d(main)`; after `main` function ends.

## Objects types list

The package *geometry3d.asy* is the extension of the module *geometry.asy*. Basically, this package provides you a tools to create a really nice 3D pictures in solid geometry.

Here is all types, defined in this module

`basis3` – a 3D ray  
`curve3` – a 3D ray  
`ray3` – a 3D ray  
`vector3` – a 3D vector  
`line3` – a 3D line  
`plane3` – a plane  
`sphere3` – a sphere

## Temp: all functions

```
void drawAllObjects();
```

this function draws all objects on the scene with front-back feature and is called by default in function `with_geometry3d`.

```
void withGeometry3d(void main());
```

this function is meant to be ending of your programm, executing essential function for drawing figures properly.

```
void add2dFrame();
```

add 2D frame in order to be able to draw a 2D figures

```
void drawCurve(picture pic=currentpicture, curve3 curve, pen  
frontpen=currentpen, pen backpen=currentpen+dashed);
```

draw curve with pens `frontpen` and `backpen` respectively.

```
circle3 circle3(triple A, triple B, triple C);  
    returns circumcircle of triangle  $ABC$ .
```

```
circle3 incircle3(triple A, triple B, triple C);  
    returns incircle of triangle  $ABC$ .
```

```
transform3 orthogonalproject(plane3 p);  
    returns transform3, which projects in direction of normal to the plane  $p$ .
```

```
triple foot3(triple A, line3 l);  
    return the foot of the perpendicular dropped from point  $A$  onto the line  $l$ .
```

```
triple foot3(triple A, plane3 p);  
    return the foot of the perpendicular dropped from point  $A$  to the plane  $p$ .
```

```
void markrightangle3(triple A, triple B, triple C, real n=5, pen  
p=currentpen);  
    marks right angle  $\angle ABC$  with pen  $p$ , size of real  $n$ .
```

```
real distance3(triple A, triple B);  
    returns distance between two points  $A$  and  $B$ .
```

```
triple midpoint3(triple A, triple B);  
    returns the midpoint of segment  $AB$ .
```

```
basis3 get_basis(projection P = currentprojection);  
    returns the basis of the projection  $P$  formed from vectors  $\vec{x} = P.camera$ ,  $\vec{y} = \vec{x} \times \vec{u}$ ,  $\vec{z} = \vec{x} \times \vec{y}$ , where  $\vec{u} = P.up$ .
```

```
triple calcCoordsInBasis(basis3 basis, triple A);  
    returns coordinates of point  $A$  (which coordinates are given in standart basis  $\{\vec{x}, \vec{y}, \vec{z}\}$ ) in basis  $basis$ .
```

```
triple changeBasis(basis3 basis1, basis3 basis2, triple A);  
    returns coordinates of point  $A$  (which coordinates are given in basis  $basis1$ ) in basis  $basis2$ .
```

```
pair project3(triple A);
```

returns 2D-coordinates  $(x', y')$  of `triple A` as if it was drawn as a plain point  $A'$  with coordinates  $(x', y')$ .

WARNING! It won't work unless you specified size of image with `size3`.

```
path project3(path3 p);
```

returns 2D-path formed from `project3(node)` for each node of nodes of `path3 p`.

```
void markangle3(picture pic = currentpicture, Label L = "", int n = 1, real
radius = 0, real space = 0, explicit triple A, explicit triple B, explicit
triple C, pair align = dir(1), arrowbar3 arrow3 = None, pen p = currentpen,
filltype filltype_ = NoFill, margin margin = NoMargin, marker marker =
nomarker);
```

marks angle  $\angle ABC$  with pen `p`, filled with `filltype_`, drawing arrow with `arrow3`.

## The type `line3`

## The type `sphere3`

Represent sphere `sphere(C,r);` as a circle `Circle(project3(C),r);` from package `graph`.