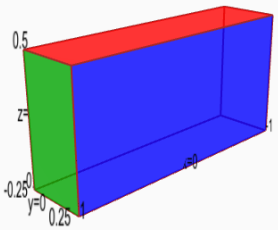
2015-04-22-172526

Jonathan Seeley

4/22/2015

Contents

```
show(cube(color=['red', 'blue', 'green'], frame_thickness=2,
         frame_color='brown', opacity=0.8).transform(scale=[2,1/2,1]), \
            frame=True)
```




```
print cube
#print(help(cube(frame_thickness=2,frame_color='brown',opacity=0.8)))
```

```
<function cube at 0x7eff06a26050>
Help on Graphics3dGroup in module sage.plot.plot3d.base object:

class Graphics3dGroup(Graphics3d)
 |  File: sage/plot/plot3d/base.pyx (starting at line 1422)
 |
 |  This class represents a collection of 3d objects. Usually they are formed
 |  implicitly by summing.
 |
 |  Method resolution order:
 |      Graphics3dGroup
 |      Graphics3d
 |      sage.structure.sage_object.SageObject
 |      __builtin__.object
```

```
 |
 |  Methods defined here:
 |
 |  __add__(...)
 |      File: sage/plot/plot3d/base.pyx (starting at line 1441)
 |
 |      We override this here to make large sums more efficient.
 |
 |      EXAMPLES::
 |
 |          sage: G = sum(tetrahedron(opacity=1-t/11).translate(t, 0, 0) for t in
range(10))
 |          sage: G
 |          Graphics3d Object
 |          sage: len(G.all)
 |          10
 |
 |      We check that :trac:‘17258‘ is solved::
 |
 |          sage: g = point3d([0,-2,-2]); g += point3d([2,-2,-2])
 |          sage: len(g.all)
 |          2
 |          sage: h = g + arrow([0,-2,-2], [2,-2,-2])
 |          sage: len(g.all)
 |          2
 |          sage: g == h
 |          False
 |
 |  __init__(...)
 |      File: sage/plot/plot3d/base.pyx (starting at line 1427)
 |
 |      EXAMPLES::
 |
 |          sage: sage.plot.plot3d.base.Graphics3dGroup([icosahedron(),
dodecahedron(opacity=.5)])
 |          Graphics3d Object
 |          sage: type(icosahedron() + dodecahedron(opacity=.5))
 |          <class 'sage.plot.plot3d.base.Graphics3dGroup'>
 |
 |  bounding_box(...)
 |      File: sage/plot/plot3d/base.pyx (starting at line 1471)
 |
 |      Box that contains the bounding boxes of
 |      all the objects that make up ‘‘self‘‘.
 |
 |      EXAMPLES::
 |
 |          sage: A = sphere((0,0,0), 5)
 |          sage: B = sphere((1, 5, 10), 1)
 |          sage: A.bounding_box()
 |          ((-5.0, -5.0, -5.0), (5.0, 5.0, 5.0))
```

```
|          sage: B.bounding_box()
|          ((0.0, 4.0, 9.0), (2.0, 6.0, 11.0))
|          sage: (A+B).bounding_box()
|          ((-5.0, -5.0, -5.0), (5.0, 6.0, 11.0))
|          sage: (A+B).show(aspect_ratio=1, frame=True)
|
|          sage: sage.plot.plot3d.base.Graphics3dGroup([]).bounding_box()
|          ((0.0, 0.0, 0.0), (0.0, 0.0, 0.0))
|
|  flatten(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1637)
|
|      Try to reduce the depth of the scene tree by consolidating groups
|      and transformations.
|
|      EXAMPLES::
|
|          sage: G = sum([circle((0, 0), t) for t in [1..10]], sphere()); G
|          Graphics3d Object
|          sage: G.flatten()
|          Graphics3d Object
|          sage: len(G.all)
|          2
|          sage: len(G.flatten().all)
|          11
|
|  jmol_repr(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1602)
|
|      The jmol representation of a group is simply the concatenation of
|      the representation of its objects.
|
|      EXAMPLES::
|
|          sage: G = sphere() + sphere((1,2,3))
|          sage: G.jmol_repr(G.default_render_params())
|          [[['isosurface sphere_1  center {0.0 0.0 0.0} sphere 1.0\ncolor isosurface
[102,102,255]']],
|           [['isosurface sphere_2  center {1.0 2.0 3.0} sphere 1.0\ncolor isosurface
[102,102,255]']]]
|
|  json_repr(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1529)
|
|      The JSON representation of a group is simply the concatenation of the
|      representations of its objects.
|
|      EXAMPLES::
|
|          sage: G = sphere() + sphere((1, 2, 3))
|          sage: G.json_repr(G.default_render_params())
```

```
|          [[["{vertices:..."]], [["{vertices:..."]]]
|
|  obj_repr(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1574)
|
|      The obj representation of a group is simply the concatenation of
|      the representation of its objects.
|
|      EXAMPLES::
|
|          sage: G = tetrahedron() + tetrahedron().translate(10, 10, 10)
|          sage: G.obj_repr(G.default_render_params())
|          [['g obj_1',
|            'usemtl ...',
|            ['v 0 0 1',
|             'v 0.942809 0 -0.333333',
|             'v -0.471405 0.816497 -0.333333',
|             'v -0.471405 -0.816497 -0.333333'],
|            ['f 1 2 3', 'f 2 4 3', 'f 1 3 4', 'f 1 4 2'],
|            []],
|           [['g obj_2',
|             'usemtl ...',
|             ['v 10 10 11',
|              'v 10.9428 10 9.66667',
|              'v 9.5286 10.8165 9.66667',
|              'v 9.5286 9.1835 9.66667'],
|             ['f 5 6 7', 'f 6 8 7', 'f 5 7 8', 'f 5 8 6'],
|             []]]]
|
|  set_texture(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1515)
|
|      EXAMPLES::
|
|          sage: G = dodecahedron(color='red', opacity=.5) + icosahedron((3, 0, 0),
color='blue')
|          sage: G
|          Graphics3d Object
|          sage: G.set_texture(color='yellow')
|          sage: G
|          Graphics3d Object
|
|  tachyon_repr(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1542)
|
|      The tachyon representation of a group is simply the concatenation of
|      the representations of its objects.
|
|      EXAMPLES::
|
|          sage: G = sphere() + sphere((1,2,3))
```

```
|          sage: G.tachyon_repr(G.default_render_params())
|          [['Sphere center 0.0 0.0 0.0 Rad 1.0 texture...'],
|           ['Sphere center 1.0 2.0 3.0 Rad 1.0 texture...']]
|
|  texture_set(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1616)
|
|      The texture set of a group is simply the union of the textures of
|      all its objects.
|
|      EXAMPLES::
|
|          sage: G = sphere(color='red') + sphere(color='yellow')
|          sage: [t for t in G.texture_set() if t.color == colors.red] # one red texture
|          [Texture(texture..., red, ff0000)]
|          sage: [t for t in G.texture_set() if t.color == colors.yellow] # one yellow
texture
|          [Texture(texture..., yellow, ffff00)]
|
|          sage: T = sage.plot.plot3d.texture.Texture('blue'); T
|          Texture(texture..., blue, 0000ff)
|          sage: G = sphere(texture=T) + sphere((1, 1, 1), texture=T)
|          sage: len(G.texture_set())
|          1
|
|  transform(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1496)
|
|      Transforming this entire group simply makes a transform group with
|      the same contents.
|
|      EXAMPLES::
|
|          sage: G = dodecahedron(color='red', opacity=.5) + icosahedron(color='blue')
|          sage: G
|          Graphics3d Object
|          sage: G.transform(scale=(2,1/2,1))
|          Graphics3d Object
|          sage: G.transform(trans=(1,1,3))
|          Graphics3d Object
|
|  x3d_str(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1556)
|
|      The x3d representation of a group is simply the concatenation of
|      the representation of its objects.
|
|      EXAMPLES::
|
|          sage: G = sphere() + sphere((1,2,3))
|          sage: print G.x3d_str()
```

```
 |          <Transform translation='0 0 0'>
 |          <Shape><Sphere radius='1.0'/><Appearance><Material diffuseColor='0.4 0.4 1.0'
shininess='1' specularColor='0.0 0.0 0.0'/></Appearance></Shape>
 |          </Transform>
 |          <Transform translation='1 2 3'>
 |          <Shape><Sphere radius='1.0'/><Appearance><Material diffuseColor='0.4 0.4 1.0'
shininess='1' specularColor='0.0 0.0 0.0'/></Appearance></Shape>
 |          </Transform>
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __qualname__ = 'Graphics3dGroup'
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from Graphics3d:
 |
 |  __radd__(...)
 |      x.__radd__(y) <==> y+x
 |
 |  __str__(...)
 |      File: sage/plot/plot3d/base.pyx (starting at line 150)
 |
 |      EXAMPLES::
 |
 |          sage: S = sphere((0, 0, 0), 1)
 |          sage: str(S)
 |          'Graphics3d Object'
 |
 |  aspect_ratio(...)
 |      File: sage/plot/plot3d/base.pyx (starting at line 213)
 |
 |      Set or get the preferred aspect ratio of ``self``.
 |
 |      INPUT:
 |
 |      - ``v`` -- (default: ``None``) must be a list or tuple of length three,
 |        or the integer ``1``. If no arguments are provided then the
 |        default aspect ratio is returned.
 |
 |      EXAMPLES::
 |
```

```
|          sage: D = dodecahedron()
|          sage: D.aspect_ratio()
|          [1.0, 1.0, 1.0]
|          sage: D.aspect_ratio([1,2,3])
|          sage: D.aspect_ratio()
|          [1.0, 2.0, 3.0]
|          sage: D.aspect_ratio(1)
|          sage: D.aspect_ratio()
|          [1.0, 1.0, 1.0]
|
|  default_render_params(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 467)
|
|      Return an instance of RenderParams suitable for plotting this object.
|
|      EXAMPLES::
|
|          sage: type(dodecahedron().default_render_params())
|          <class 'sage.plot.plot3d.base.RenderParams'>
|
|  export_jmol(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 651)
|
|      A jmol scene consists of a script which refers to external files.
|      Fortunately, we are able to put all of them in a single zip archive,
|      which is the output of this call.
|
|      EXAMPLES::
|
|          sage: out_file = tmp_filename(ext=".jmol")
|          sage: G = sphere((1, 2, 3), 5) + cube() + sage.plot.plot3d.shapes.Text("hi")
|          sage: G.export_jmol(out_file)
|          sage: import zipfile
|          sage: z = zipfile.ZipFile(out_file)
|          sage: z.namelist()
|          ['obj_...pmesh', 'SCRIPT']
|
|          sage: print z.read('SCRIPT')
|          data "model list"
|          2
|          empty
|          Xx 0 0 0
|          Xx 5.5 5.5 5.5
|          end "model list"; show data
|          select *
|          wireframe off; spacefill off
|          set labelOffset 0 0
|          background [255,255,255]
|          spin OFF
|          moveto 0 -764 -346 -545 76.39
|          centerAt absolute {0 0 0}
```

```
|          zoom 100
|          frank OFF
|          set perspectivedepth ON
|          isosurface sphere_1  center {1.0 2.0 3.0} sphere 5.0
|          color isosurface  [102,102,255]
|          pmesh obj_... "obj_...pmesh"
|          color pmesh  [102,102,255]
|          select atomno = 1
|          color atom  [102,102,255]
|          label "hi"
|          isosurface fullylit; pmesh o* fullylit; set antialiasdisplay on;
|
|          sage: print z.read(z.namelist()[0])
|          24
|          0.5 0.5 0.5
|          -0.5 0.5 0.5
|          ...
|          -0.5 -0.5 -0.5
|          6
|          5
|          0
|          1
|          ...
|
|  frame_aspect_ratio(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 247)
|
|      Set or get the preferred frame aspect ratio of ``self``.
|
|      INPUT:
|
|      - ``v`` -- (default: ``None``) must be a list or tuple of
|        length three, or the integer ``1``. If no arguments are
|        provided then the default frame aspect ratio is returned.
|
|      EXAMPLES::
|
|          sage: D = dodecahedron()
|          sage: D.frame_aspect_ratio()
|          [1.0, 1.0, 1.0]
|          sage: D.frame_aspect_ratio([2,2,1])
|          sage: D.frame_aspect_ratio()
|          [2.0, 2.0, 1.0]
|          sage: D.frame_aspect_ratio(1)
|          sage: D.frame_aspect_ratio()
|          [1.0, 1.0, 1.0]
|
|  mtl_str(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 877)
|
|      Return the contents of a .mtl file, to be used to provide coloring
```

```
|      information for an .obj file.
|
|      EXAMPLES::
|
|          sage: G = tetrahedron(color='red') + tetrahedron(color='yellow', opacity=0.5)
|          sage: print G.mtl_str()
|          newmtl ...
|          Ka 0.5 5e-06 5e-06
|          Kd 1.0 1e-05 1e-05
|          Ks 0.0 0.0 0.0
|          illum 1
|          Ns 1
|          d 1
|          newmtl ...
|          Ka 0.5 0.5 5e-06
|          Kd 1.0 1.0 1e-05
|          Ks 0.0 0.0 0.0
|          illum 1
|          Ns 1
|          d 0.500000000000000
|
|  obj(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 622)
|
|      An .obj scene file (as a string) containing the this object.
|
|      A .mtl file of the same name must also be produced for
|      coloring.
|
|      EXAMPLES::
|
|          sage: from sage.plot.plot3d.shapes import ColorCube
|          sage: print ColorCube(1, ['red', 'yellow', 'blue']).obj()
|          g obj_1
|          usemtl ...
|          v 1 1 1
|          v -1 1 1
|          v -1 -1 1
|          v 1 -1 1
|          f 1 2 3 4
|          ...
|          g obj_6
|          usemtl ...
|          v -1 -1 1
|          v -1 1 1
|          v -1 1 -1
|          v -1 -1 -1
|          f 21 22 23 24
|
|  rotate(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 397)
```

```
|
|      Return ``self`` rotated about the vector `v` by `\theta` radians.
|
|      EXAMPLES::
|
|          sage: from sage.plot.plot3d.shapes import Cone
|          sage: v = (1,2,3)
|          sage: G = arrow3d((0, 0, 0), v)
|          sage: G += Cone(1/5, 1).translate((0, 0, 2))
|          sage: C = Cone(1/5, 1, opacity=.25).translate((0, 0, 2))
|          sage: G += sum(C.rotate(v, pi*t/4) for t in [1..7])
|          sage: G.show(aspect_ratio=1)
|
|          sage: from sage.plot.plot3d.shapes import Box
|          sage: Box(1/3, 1/5, 1/7).rotate((1, 1, 1), pi/3).show(aspect_ratio=1)
|
|  rotateX(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 417)
|
|      Return ``self`` rotated about the `x`-axis by the given angle.
|
|      EXAMPLES::
|
|          sage: from sage.plot.plot3d.shapes import Cone
|          sage: G = Cone(1/5, 1) + Cone(1/5, 1, opacity=.25).rotateX(pi/2)
|          sage: G.show(aspect_ratio=1)
|
|  rotateY(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 429)
|
|      Return ``self`` rotated about the `y`-axis by the given angle.
|
|      EXAMPLES::
|
|          sage: from sage.plot.plot3d.shapes import Cone
|          sage: G = Cone(1/5, 1) + Cone(1/5, 1, opacity=.25).rotateY(pi/3)
|          sage: G.show(aspect_ratio=1)
|
|  rotateZ(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 441)
|
|      Return ``self`` rotated about the `z`-axis by the given angle.
|
|      EXAMPLES::
|
|          sage: from sage.plot.plot3d.shapes import Box
|          sage: G = Box(1/2, 1/3, 1/5) + Box(1/2, 1/3, 1/5, opacity=.25).rotateZ(pi/5)
|          sage: G.show(aspect_ratio=1)
|
|  save(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1334)
```

```
|
|      Save to file.
|
|      Save the graphic to an image file (of type: PNG, BMP, GIF, PPM, or TIFF)
|      rendered using Tachyon, or pickle it (stored as an SOBJ so you can load it
|      later) depending on the file extension you give the filename.
|
|      INPUT:
|
|      - ``filename`` -- Specify where to save the image or object.
|
|      - ``**kwds`` -- When specifying an image file to be rendered by Tachyon,
|        any of the viewing options accepted by show() are valid as keyword
|        arguments to this function and they will behave in the same way.
|        Accepted keywords include: ``viewer``, ``verbosity``, ``figsize``,
|        ``aspect_ratio``, ``frame_aspect_ratio``, ``zoom``, ``frame``, and
|        ``axes``. Default values are provided.
|
|      EXAMPLES::
|
|          sage: f = tmp_filename() + '.png'
|          sage: G = sphere()
|          sage: G.save(f)
|
|      We demonstrate using keyword arguments to control the appearance of the
|      output image::
|
|          sage: G.save(f, zoom=2, figsize=[5, 10])
|
|      But some extra parameters don't make since (like ``viewer``, since
|      rendering is done using Tachyon only). They will be ignored::
|
|          sage: G.save(f, viewer='jmol') # Looks the same
|
|      Since Tachyon only outputs PNG images, PIL will be used to convert to
|      alternate formats::
|
|          sage: cube().save(tmp_filename(ext='.gif'))
|
|  save_image(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1312)
|
|      Save an image representation of self.  The image type is
|      determined by the extension of the filename.  For example,
|      this could be ``.png``, ``.jpg``, ``.gif``, ``.pdf``,
|      ``.svg``.  Currently this is implemented by calling the
|      :meth:`save` method of self, passing along all arguments and
|      keywords.
|
|      .. Note::
|
```

```
|          Not all image types are necessarily implemented for all
|          graphics types.  See :meth:‘save‘ for more details.
|
|      EXAMPLES::
|
|          sage: f = tmp_filename() + '.png'
|          sage: G = sphere()
|          sage: G.save_image(f)
|
|  scale(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 373)
|
|      Return ‘‘self‘‘ scaled in the x, y, and z directions.
|
|      EXAMPLES::
|
|          sage: G = dodecahedron() + dodecahedron(opacity=.5).scale(2)
|          sage: G.show(aspect_ratio=1)
|          sage: G = icosahedron() + icosahedron(opacity=.5).scale([1, 1/2, 2])
|          sage: G.show(aspect_ratio=1)
|
|      TESTS::
|
|          sage: G = sphere((0, 0, 0), 1)
|          sage: G.scale(2)
|          Graphics3d Object
|          sage: G.scale(1, 2, 1/2).show(aspect_ratio=1)
|          sage: G.scale(2).bounding_box()
|          ((-2.0, -2.0, -2.0), (2.0, 2.0, 2.0))
|
|  show(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 1090)
|
|      INPUT:
|
|      - ‘‘viewer‘‘ -- string (default: 'jmol'), how to view
|        the plot
|
|        * 'jmol': Interactive 3D viewer using Java
|
|        * 'tachyon': Ray tracer generates a static PNG image
|
|        * 'java3d': Interactive OpenGL based 3D
|
|        * 'canvas3d': Web-based 3D viewer powered by JavaScript and
|          <canvas> (notebook only)
|
|      - ‘‘filename‘‘ -- string (default: a temp file); filename
|        without extension to save the image file(s) to
|
|      - ‘‘verbosity‘‘ -- display information about rendering
```

```
|          the figure
|
|      -  ‘‘figsize‘‘ -- (default: 5); x or pair [x,y] for
|          numbers, e.g., [5,5]; controls the size of the output figure. E.g.,
|          with Tachyon the number of pixels in each direction is 100 times
|          figsize[0]. This is ignored for the jmol embedded renderer.
|
|      -  ‘‘aspect_ratio‘‘ -- (default: "automatic") -- aspect
|          ratio of the coordinate system itself. Give [1,1,1] to make spheres
|          look round.
|
|      -  ‘‘frame_aspect_ratio‘‘ -- (default: "automatic")
|          aspect ratio of frame that contains the 3d scene.
|
|      -  ‘‘zoom‘‘ -- (default: 1) how zoomed in
|
|      -  ‘‘frame‘‘ -- (default: True) if True, draw a
|          bounding frame with labels
|
|      -  ‘‘axes‘‘ -- (default: False) if True, draw coordinate
|          axes
|
|      -  ‘‘**kwds‘‘ -- other options, which make sense for particular
|          rendering engines
|
|      CHANGING DEFAULTS: Defaults can be uniformly changed by importing a
|      dictionary and changing it. For example, here we change the default
|      so images display without a frame instead of with one::
|
|          sage: from sage.plot.plot3d.base import SHOW_DEFAULTS
|          sage: SHOW_DEFAULTS['frame'] = False
|
|      This sphere will not have a frame around it::
|
|          sage: sphere((0,0,0))
|          Graphics3d Object
|
|      We change the default back::
|
|          sage: SHOW_DEFAULTS['frame'] = True
|
|      Now this sphere is enclosed in a frame::
|
|          sage: sphere((0,0,0))
|          Graphics3d Object
|
|      EXAMPLES: We illustrate use of the ‘‘aspect_ratio‘‘ option::
|
|          sage: x, y = var('x,y')
|          sage: p = plot3d(2*sin(x*y), (x, -pi, pi), (y, -pi, pi))
|          sage: p.show(aspect_ratio=[1,1,1])
```

```
|
|      This looks flattened, but filled with the plot::
|
|          sage: p.show(frame_aspect_ratio=[1,1,1/16])
|
|      This looks flattened, but the plot is square and smaller::
|
|          sage: p.show(aspect_ratio=[1,1,1], frame_aspect_ratio=[1,1,1/8])
|
|      This example shows indirectly that the defaults
|      from :func:‘~sage.plot.plot.plot‘ are dealt with properly::
|
|          sage: plot(vector([1,2,3]))
|          Graphics3d Object
|
|      We use the 'canvas3d' backend from inside the notebook to get a view of
|      the plot rendered inline using HTML canvas::
|
|          sage: p.show(viewer='canvas3d')
|
|      From the command line, you probably want to specify an explicit
|      filename::
|
|          sage: basedir = tmp_dir()
|          sage: basename = os.path.join(basedir, "xyz")
|          sage: p.show(filename=basename)
|
|      In this doctest, we see many files because we test various
|      viewers::
|
|          sage: sorted(os.listdir(basedir))
|          ['xyz-size500.spt', 'xyz-size500.spt.zip', 'xyz.mtl', 'xyz.obj', 'xyz.png']
|
|  tachyon(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 545)
|
|      An tachyon input file (as a string) containing the this object.
|
|      EXAMPLES::
|
|          sage: print sphere((1, 2, 3), 5, color='yellow').tachyon()
|          begin_scene
|          resolution 400 400
|                   camera
|                   ...
|                 plane
|                   center -2000 -1000 -500
|                   normal 2.3 2.4 2.0
|                   TEXTURE
|                       AMBIENT 1.0 DIFFUSE 1.0 SPECULAR 1.0 OPACITY 1.0
|                       COLOR 1.0 1.0 1.0
```

```
|                     TEXFUNC 0
|              Texdef texture...
|            Ambient 0.333333333333 Diffuse 0.666666666667 Specular 0.0 Opacity 1
|             Color 1.0 1.0 0.0
|             TexFunc 0
|              Sphere center 1.0 -2.0 3.0 Rad 5.0 texture...
|          end_scene
|
|          sage: G = icosahedron(color='red') + sphere((1,2,3), 0.5, color='yellow')
|          sage: G.show(viewer='tachyon', frame=false)
|          sage: print G.tachyon()
|          begin_scene
|          ...
|          Texdef texture...
|            Ambient 0.333333333333 Diffuse 0.666666666667 Specular 0.0 Opacity 1
|             Color 1.0 1.0 0.0
|             TexFunc 0
|          TRI V0 ...
|          Sphere center 1.0 -2.0 3.0 Rad 0.5 texture...
|          end_scene
|
|  testing_render_params(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 478)
|
|      Return an instance of RenderParams suitable for testing this object.
|
|      In particular, it opens up '/dev/null' as an auxiliary zip
|      file for jmol.
|
|      EXAMPLES::
|
|          sage: type(dodecahedron().testing_render_params())
|          <class 'sage.plot.plot3d.base.RenderParams'>
|
|  translate(...)
|      File: sage/plot/plot3d/base.pyx (starting at line 347)
|
|      Return ``self`` translated by the given vector (which can be
|      given either as a 3-iterable or via positional arguments).
|
|      EXAMPLES::
|
|          sage: icosahedron() + sum(icosahedron(opacity=0.25).translate(2*n, 0, 0) for n
in [1..4])
|          Graphics3d Object
|          sage: icosahedron() + sum(icosahedron(opacity=0.25).translate([-2*n, n, n^2])
for n in [1..4])
|          Graphics3d Object
|
|      TESTS::
|
```

```
 |          sage: G = sphere((0, 0, 0), 1)
 |          sage: G.bounding_box()
 |          ((-1.0, -1.0, -1.0), (1.0, 1.0, 1.0))
 |          sage: G.translate(0, 0, 1).bounding_box()
 |          ((-1.0, -1.0, 0.0), (1.0, 1.0, 2.0))
 |          sage: G.translate(-1, 5, 0).bounding_box()
 |          ((-2.0, 4.0, -1.0), (0.0, 6.0, 1.0))
 |
 |  viewpoint(...)
 |      File: sage/plot/plot3d/base.pyx (starting at line 453)
 |
 |      Return the viewpoint of this plot.
 |
 |      Currently only a stub for x3d.
 |
 |      EXAMPLES::
 |
 |          sage: type(dodecahedron().viewpoint())
 |          <class 'sage.plot.plot3d.base.Viewpoint'>
 |
 |  x3d(...)
 |      File: sage/plot/plot3d/base.pyx (starting at line 494)
 |
 |      An x3d scene file (as a string) containing the this object.
 |
 |      EXAMPLES::
 |
 |          sage: print sphere((1, 2, 3), 5).x3d()
 |          <X3D version='3.0' profile='Immersive' xmlns:xsd='http://www.w3.org/2001
/XMLSchema-instance' xsd:noNamespaceSchemaLocation='
http://www.web3d.org/specifications/x3d-3.0.xsd '>
 |          <head>
 |          <meta name='title' content='sage3d'/>
 |          </head>
 |          <Scene>
 |          <Viewpoint position='0 0 6'/>
 |          <Transform translation='1 2 3'>
 |          <Shape><Sphere radius='5.0'/><Appearance><Material diffuseColor='0.4 0.4 1.0'
shininess='1' specularColor='0.0 0.0 0.0'/></Appearance></Shape>
 |          </Transform>
 |          </Scene>
 |          </X3D>
 |
 |          sage: G = icosahedron() + sphere((0,0,0), 0.5, color='red')
 |          sage: print G.x3d()
 |          <X3D version='3.0' profile='Immersive' xmlns:xsd='http://www.w3.org/2001
/XMLSchema-instance' xsd:noNamespaceSchemaLocation='
http://www.web3d.org/specifications/x3d-3.0.xsd '>
 |          <head>
 |          <meta name='title' content='sage3d'/>
 |          </head>
```

```
 |          <Scene>
 |          <Viewpoint position='0 0 6'/>
 |          <Shape>
 |          <IndexedFaceSet coordIndex='...'>
 |            <Coordinate point='...'/>
 |          </IndexedFaceSet>
 |          <Appearance><Material diffuseColor='0.4 0.4 1.0' shininess='1'
specularColor='0.0 0.0 0.0'/></Appearance></Shape>
 |          <Transform translation='0 0 0'>
 |          <Shape><Sphere radius='0.5'/><Appearance><Material diffuseColor='1.0 0.0 0.0'
shininess='1' specularColor='0.0 0.0 0.0'/></Appearance></Shape>
 |          </Transform>
 |          </Scene>
 |          </X3D>
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from Graphics3d:
 |
 |  texture
 |      File: sage/plot/plot3d/base.pxd (starting at line 5)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes inherited from Graphics3d:
 |
 |  __new__ = <built-in method __new__ of type object>
 |      T.__new__(S, ...) -> a new object with type S, a subtype of T
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from sage.structure.sage_object.SageObject:
 |
 |  __hash__(...)
 |      File: sage/structure/sage_object.pyx (starting at line 230)
 |
 |  __repr__(...)
 |      File: sage/structure/sage_object.pyx (starting at line 124)
 |
 |      Default method for string representation.
 |
 |      .. NOTE::
 |
 |          Do not overwrite this method. Instead, implement
 |          a ``_repr_`` (single underscore) method.
 |
 |      EXAMPLES:
 |
 |      By default, the string representation coincides with
 |      the output of the single underscore ``_repr_``::
 |
 |          sage: P.<x> = QQ[]
 |          sage: repr(P) == P._repr_()  #indirect doctest
 |          True
```

```
|
|      Using :meth:‘rename‘, the string representation can
|      be customized::
|
|          sage: P.rename('A polynomial ring')
|          sage: repr(P) == P._repr_()
|          False
|
|      The original behaviour is restored with :meth:‘reset_name‘.::
|
|          sage: P.reset_name()
|          sage: repr(P) == P._repr_()
|          True
|
|  category(...)
|      File: sage/structure/sage_object.pyx (starting at line 400)
|
|  db(...)
|      File: sage/structure/sage_object.pyx (starting at line 371)
|
|      Dumps self into the Sage database.  Use db(name) by itself to
|      reload.
|
|      The database directory is ‘‘$HOME/.sage/db‘‘
|
|  dump(...)
|      File: sage/structure/sage_object.pyx (starting at line 343)
|
|      Same as self.save(filename, compress)
|
|  dumps(...)
|      File: sage/structure/sage_object.pyx (starting at line 349)
|
|      Dump ‘‘self‘‘ to a string ‘‘s‘‘, which can later be reconstituted
|      as ‘‘self‘‘ using ‘‘loads(s)‘‘.
|
|      There is an optional boolean argument ‘‘compress‘‘ which defaults to ‘‘True‘‘.
|
|      EXAMPLES::
|
|          sage: O=SageObject(); O.dumps()
|          'x\x9ck‘J.NLO\xd5+.)*M.)-\x02\xb2\x80\xdc\xf8\xfc\xa4\xac\xd4\xe4\x12\xae‘
\xdb\x1f\xc2,d\xd41,d\xd2\x03\x00\xb7X\x10\xf1'
|          sage: O.dumps(compress=False)
|          '\x80\x02csage.structure.sage_object\nSageObject\nq\x01)\x81q\x02.'
|
|  parent(...)
|      File: sage/structure/sage_object.pyx (starting at line 435)
|
|      Return the type of ‘‘self‘‘ to support the coercion framework.
|
```

```
|      EXAMPLES::
|
|          sage: t = log(sqrt(2) - 1) + log(sqrt(2) + 1); t
|          log(sqrt(2) + 1) + log(sqrt(2) - 1)
|          sage: u = t.maxima_methods()
|          sage: u.parent()
|          <class 'sage.symbolic.maxima_wrapper.MaximaWrapper'>
|
|  rename(...)
|      File: sage/structure/sage_object.pyx (starting at line 47)
|
|      Change self so it prints as x, where x is a string.
|
|      .. NOTE::
|
|         This is *only* supported for Python classes that derive
|         from SageObject.
|
|      EXAMPLES::
|
|          sage: x = PolynomialRing(QQ, 'x', sparse=True).gen()
|          sage: g = x^3 + x - 5
|          sage: g
|          x^3 + x - 5
|          sage: g.rename('a polynomial')
|          sage: g
|          a polynomial
|          sage: g + x
|          x^3 + 2*x - 5
|          sage: h = g^100
|          sage: str(h)[:20]
|          'x^300 + 100*x^298 - '
|          sage: h.rename('x^300 + ...')
|          sage: h
|          x^300 + ...
|
|      Real numbers are not Python classes, so rename is not supported::
|
|          sage: a = 3.14
|          sage: type(a)
|          <type 'sage.rings.real_mpfr.RealLiteral'>
|          sage: a.rename('pi')
|          Traceback (most recent call last):
|          ...
|          NotImplementedError: object does not support renaming: 3.14000000000000
|
|      .. NOTE::
|
|         The reason C-extension types are not supported by default
|         is if they were then every single one would have to carry
|         around an extra attribute, which would be slower and waste
```

```
 |         a lot of memory.
 |
 |         To support them for a specific class, add a
 |         ``cdef public __custom_name`` attribute.
 |
 |  reset_name(...)
 |      File: sage/structure/sage_object.pyx (starting at line 104)
 |
 |      Remove the custrom name of an object.
 |
 |      EXAMPLES::
 |
 |          sage: P.<x> = QQ[]
 |          sage: P
 |          Univariate Polynomial Ring in x over Rational Field
 |          sage: P.rename('A polynomial ring')
 |          sage: P
 |          A polynomial ring
 |          sage: P.reset_name()
 |          sage: P
 |          Univariate Polynomial Ring in x over Rational Field
 |
 |  version(...)
 |      File: sage/structure/sage_object.pyx (starting at line 303)
 |
 |      The version of Sage.
 |
 |      Call this to save the version of Sage in this object.
 |      If you then save and load this object it will know in what
 |      version of Sage it was created.
 |
 |      This only works on Python classes that derive from SageObject.
None
```