

3D graphics

1.0

Generated by Doxygen 1.9.1

1 Computer programming (110)	1
1.0.1 3D Graphic library - specification and documentation	1
1.1 Brief description:	1
1.2 Usage:	1
1.3 Else:	1
2 Example program:	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Class Documentation	11
6.1 canvas Class Reference	11
6.1.1 Detailed Description	11
6.2 cone Class Reference	11
6.2.1 Detailed Description	12
6.2.2 Constructor & Destructor Documentation	12
6.2.2.1 cone()	12
6.3 cube Class Reference	13
6.3.1 Detailed Description	13
6.3.2 Constructor & Destructor Documentation	13
6.3.2.1 cube()	13
6.4 Edge Class Reference	15
6.4.1 Detailed Description	15
6.4.2 Constructor & Destructor Documentation	16
6.4.2.1 Edge() [1/2]	16
6.4.2.2 Edge() [2/2]	16
6.4.3 Member Function Documentation	16
6.4.3.1 operator=()	16
6.4.3.2 position()	17
6.5 Obj_3D Class Reference	17
6.5.1 Detailed Description	18
6.5.2 Constructor & Destructor Documentation	18
6.5.2.1 Obj_3D()	18
6.5.3 Member Function Documentation	20
6.5.3.1 printAll()	20
6.5.3.2 reposition()	20
6.5.3.3 rotateFigure()	21

6.6 sphere Class Reference	21
6.6.1 Detailed Description	22
6.6.2 Constructor & Destructor Documentation	22
6.6.2.1 sphere()	22
6.7 Vertice Class Reference	22
6.7.1 Detailed Description	23
6.7.2 Constructor & Destructor Documentation	23
6.7.2.1 Vertice()	24
6.7.3 Member Function Documentation	24
6.7.3.1 addPosition()	24
6.7.3.2 get()	24
6.7.3.3 multiplePosition()	26
6.7.3.4 operator>() [1/2]	26
6.7.3.5 operator>() [2/2]	27
6.7.3.6 operator*()	27
6.7.3.7 operator+()	27
6.7.3.8 operator-()	28
6.7.3.9 operator[]()	28
6.7.3.10 rePosition()	29
6.7.3.11 rotate()	29
7 File Documentation	31
7.1 Graphics3D.h File Reference	31
7.1.1 Detailed Description	32
7.1.2 Function Documentation	32
7.1.2.1 aprox()	32
Index	33

Chapter 1

Computer programming (110)

1.0.1 3D Graphic library - specification and documentation

Student: Robert Okenczyc 298990 **Instructor:** Piotr Fabian **Year:** 2021 (II sem)

1.1 Brief description:

This library contains classes with definitions of simple 3D shapes and methods to modify them, print them as plain data or show them in Allegro 5 window as points or/and lines. Every method was written to accept or return 3D point data. There are 3 basic shapes:

1. Cube - with edge **a** as a parameter,
2. Sphere - with radius **r** and **opt*** as a paremeters**,
3. **Cone - with base radius **r**, figure height **H** and ****opt***** as a parameters.

opt - Because some shapes have some form of a circle in them there has to be a parameter for optimisation purposes, as in higher number, fog. **opt = 1000** in a sphere, it could run very poorly because there will be a huge amount of vertices to calculate (for **opt = 1000** there are **999001** of vertices)

1.2 Usage:

As it is a library usage would be more like documentation thats why there will be only some basic informations, Inside of this library there is class called **canvas** which contains methods and informations for initializing Allegro 5 window with proper data, example of using it in your project:

```
canvas Canvas1 (width_of_a_window, height_of_a_window, title_for_window); //Initialision
while (Canvas1.endCondition()) {
    while (true) {
        if (Canvas1.FrameInit()) { return 0; }

        // Shapes and transformations/deformations goes here

        Canvas1.keyframe();
    }
}
```

1.3 Else:

All other pieces of information will be shown in documentation and examples. (Working program example at: See [example page](#) for more info.)

Chapter 2

Example program:

Simple program with some of the methods form this library with comments:

```
#include <iostream>
#include <math.h>
#include "Graphics3D.h"
using namespace std;
int main() {
    cube c1 = cube(400 / sqrt(3)); //initiation of a cube by edge with 400 / sqrt(3) length
    c1.reposition(500, 500); //repositioning cube from (0,0) point to (500, 500) with anchor point

    sphere s1 = sphere(400, 15); //initiation of a sphere with radius 400 and 15 latitude lines
    s1.printAll(); //printing all plain data
    cone con1 = cone(200, 200, 20); //initiation of a cone with base radius of 200, 200 height and with a
        base circle with 20 vertices
    con1.printAll();
    canvas Canvas1 (1000, 1000, "3D graphics project"); //initiation of new canvas
    while (Canvas1.endCondition()) {
        while (true) {
            if (Canvas1.FrameInit()) { return 0; }
            c1.draw(al_map_rgb(255, 0, 255), 2); // drawing all edges of cube with pink color and line
            thickness of 2px
            s1.draw(al_map_rgb(255, 255, 255), 2);
            c1.rotateFigure(0.1, 0.1, 0.1); // rotation figure by (0.1, 0.1, 0.1) in 3D
            s1.rotateFigure(-0.1, -0.1, -0.1);
            con1.draw(al_map_rgb(255, 255, 0), 2);
            con1.drawPixels(al_map_rgb(255, 55, 255), 2); //drawing all vertices as purple points with
            radius of 2px;
            con1.rotateFigure(-0.1, -0.1, -0.1);
            Canvas1.keyframe();
        }
    }
    return 0;
}
```


Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

canvas	11
Edge	15
Obj_3D	17
cone	11
cube	13
sphere	21
Vertice	22

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

canvas	Class for Allegro 5 canvas initiation and animation handling	11
cone	Class with inherited methods from Obj_3D that contains constructor for Cone figure	11
cube	Class with inherited methods from Obj_3D that contains constructor for Cube figure	13
Edge	Edge is a line between two Vertices in 3D space	15
Obj_3D	Basic Figure class, base class for some of the next Figure classes. It contains every method to manipulate Figures	17
sphere	Class with inherited methods from Obj_3D that contains constructor for Sphere figure	21
Vertice	Vertice is point in 3D space, it is the most basic class of 3D classes	22

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Graphics3D.h	31
--	----

Chapter 6

Class Documentation

6.1 canvas Class Reference

Class for Allegro 5 canvas initiation and animation handling.

```
#include <Graphics3D.h>
```

Public Member Functions

- **canvas** (int ix=1000, int iy=1000, const char *ititle="3D program")
- void **keyframe** ()
- bool **endCondition** ()
- bool **FrameInit** (ALLEGRO_COLOR bg=al_map_rgb(0, 0, 0))

6.1.1 Detailed Description

Class for Allegro 5 canvas initiation and animation handling.

Parameters

<i>x</i>	= width of an Allegro 5 window
<i>y</i>	= height of an Allegro 5 window

The documentation for this class was generated from the following files:

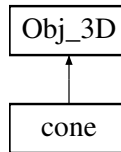
- [Graphics3D.h](#)
- Graphics3D.cpp

6.2 cone Class Reference

Class with inherited methods from [Obj_3D](#) that contains constructor for Cone figure.

```
#include <Graphics3D.h>
```

Inheritance diagram for cone:



Public Member Functions

- [cone](#) (double *ir*, double *iH*, int *opt*=10)
Constructor for a cone, made in (0, 0, 0) position.

Additional Inherited Members

6.2.1 Detailed Description

Class with inherited methods from [Obj_3D](#) that contains constructor for Cone figure.

It also calls the constructor of an [Obj_3D](#) base class

Parameters

<i>r</i>	Length of an base circe radius
<i>H</i>	Length of a height of cone

6.2.2 Constructor & Destructor Documentation

6.2.2.1 cone()

```

cone::cone (
    double ir,
    double iH,
    int opt = 10 )

```

Constrector for a cone, made in (0, 0, 0) position.

Parameters

<i>ir</i>	Length of an base circle radius <i>r</i> { <i>ir</i> }
<i>iH</i>	Length of an height of the cone <i>H</i> { <i>iH</i> }
<i>opt</i>	For information aobut <i>opt</i> check documentation main site

The documentation for this class was generated from the following files:

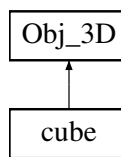
- [Graphics3D.h](#)
- Graphics3D.cpp

6.3 cube Class Reference

Class with inherited methods from [Obj_3D](#) that contains constructor for Cube figure.

```
#include <Graphics3D.h>
```

Inheritance diagram for cube:



Public Member Functions

- [cube](#) (double ia, int v=8, int e=12)
Constrctor for a Cube, made in (0, 0, 0) position.

Additional Inherited Members

6.3.1 Detailed Description

Class with inherited methods from [Obj_3D](#) that contains constructor for Cube figure.

Parameters

<i>a</i>	Length of an Cube edge
----------	------------------------

6.3.2 Constructor & Destructor Documentation

6.3.2.1 cube()

```

cube::cube (
    double ia,
    int v = 8,
    int e = 12 )
  
```

Constructor for a Cube, made in (0, 0, 0) position.

It also calls the constructor of an [Obj_3D](#) base class

Parameters

<i>ia</i>	Length of an Cube edge a{ia}
<i>v</i>	Vertices count
<i>e</i>	Edges count

The documentation for this class was generated from the following files:

- [Graphics3D.h](#)
- [Graphics3D.cpp](#)

6.4 Edge Class Reference

[Edge](#) is a line between two Vertices in 3D space.

```
#include <Graphics3D.h>
```

Public Member Functions

- [Edge](#) & **operator=** (const [Edge](#) &other)
Pasting data from [Edge](#) other to original one (before '=' operator)
- [Edge](#) ([Vertex](#) &iv1, [Vertex](#) &iv2)
Constructor of an [Edge](#).
- [Edge](#) ()
Constructor without parameters (default)
- void **position** (double ix, double iy=0, double iz=0)
Adding values to Vertices in [Edge](#) values.

Public Attributes

- [Vertex](#) * **v1**
- [Vertex](#) * **v2**

Friends

- ostream & **operator**<< (ostream &c, const [Edge](#) &orig)

6.4.1 Detailed Description

[Edge](#) is a line between two Vertices in 3D space.

See also

[Vertex](#)

Parameters

*v1	Pointer to the first Vertex of this Edge
*v2	Pointer to the second Vertex of this Edge

6.4.2 Constructor & Destructor Documentation

6.4.2.1 [Edge\(\)](#) [1/2]

```
Edge::Edge (
    Vertex & iv1,
    Vertex & iv2 )
```

Constructor of an [Edge](#).

Parameters

iv1	Vertex 1 for v1
iv2	Vertex 2 for v2

See also

[Edge](#)

6.4.2.2 [Edge\(\)](#) [2/2]

```
Edge::Edge ( )
```

Constructor without parameters (default)

Values of the data will be set to nullptr

6.4.3 Member Function Documentation

6.4.3.1 [operator=\(\)](#)

```
Edge & Edge::operator= (
    const Edge & other )
```

Pasting data from [Edge](#) *other* to original one (before '=' operator)

Warning

This function will change [Edge](#) before operator

Parameters

<i>other</i>	The Edge which from data comes
--------------	--

Returns

Main [Edge](#) that has been changed

6.4.3.2 position()

```
void Edge::position (
    double ix,
    double iy = 0,
    double iz = 0 )
```

Adding values to Vertices in [Edge](#) values.

See also

[Vertex](#)

[Edge](#)

[Vertex::addPosition\(\)](#)

Parameters

<i>ix</i>	Value added to both Vertices in Edge xs
<i>iy</i>	Value added to both Vertices in Edge ys
<i>iz</i>	Value added to both Vertices in Edge zs

The documentation for this class was generated from the following files:

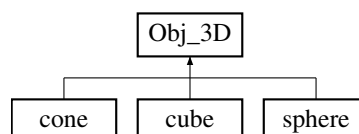
- [Graphics3D.h](#)
- [Graphics3D.cpp](#)

6.5 Obj_3D Class Reference

Basic Figure class, base class for some of the next Figure classes. It contains every method to manipulate Figures.

```
#include <Graphics3D.h>
```

Inheritance diagram for [Obj_3D](#):



Public Member Functions

- [Obj_3D](#) (int v, int e)
Constructor of an [Obj_3D](#).
- void [rotateFigure](#) (double ax, double ay=0, double az=0, [Vertice](#) anchorPoint=[Vertice](#)(0, 0, 0))
Rotates every [Vertice](#) in figure by 3D axis with anchorPoint in center.
- void [reposition](#) (double ax, double ay=0, double az=0)
Sets new position of figure.
- void [printAll](#) ()
- void [draw](#) (ALLEGRO_COLOR a=a_l_map_rgb(255, 0, 0), int b=2)
- void [drawPixels](#) (ALLEGRO_COLOR a=a_l_map_rgb(255, 255, 255), int r=1)

Public Attributes

- [Vertice](#) anchorPoint
- int vArrSize
- int eArrSize
- [Vertice](#) * vArr
- [Edge](#) * eArr

6.5.1 Detailed Description

Basic Figure class, base class for some of the next Figure classes. It contains every method to manipulate Figures.

See also

Sphere
Cube
Cone

Parameters

<i>anchorPoint</i>	Center of the figure
<i>vArrSize</i>	Size of the vArr array (array of Vertices)
<i>eArrSize</i>	Size of the eArr array (array of Edges)
<i>*vArr</i>	Array of pointers for Vertices
<i>*eArr</i>	Arrat of ponters for Edges

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Obj_3D()

```
Obj_3D::Obj_3D (
    int v,
    int e )
```

Constructor of an [Obj_3D](#).

Parameters

<i>v</i>	value for vArrSize
----------	--------------------

See also

Obj_3D::vArrSize

Parameters

<i>e</i>	value for eArrSize
----------	--------------------

See also

Obj_3D::eArrSize

6.5.3 Member Function Documentation

6.5.3.1 printAll()

```
void Obj_3D::printAll ( )
```

@biref Prints all data (Vertices, Edges) from a figure as a plain text

See also

[Vertex::printAll\(\)](#)

Edge::operator<<

6.5.3.2 reposition()

```
void Obj_3D::reposition (
    double ax,
    double ay = 0,
    double az = 0 )
```

Sets new position of figure.

Parameters

<i>ax</i>	Distance for xs to move
<i>ay</i>	Distance for ys to move
<i>az</i>	Distance for zs to move

6.5.3.3 rotateFigure()

```
void Obj_3D::rotateFigure (
    double ax,
    double ay = 0,
    double az = 0,
    Vertice anchorPoint = Vertice(0, 0, 0) )
```

Rotates every [Vertice](#) in figure by 3D axis with anchorPoint in center.

See also

[Vertice::rotate\(\)](#)

Parameters

<i>ax</i>	Angle in which Figure will rotate by x axis
<i>ay</i>	Angle in which Figure will rotate by y axis
<i>az</i>	Angle in which Figure will rotate by z axis
<i>anchorPoint</i>	Center pont for axis to be rotated of

The documentation for this class was generated from the following files:

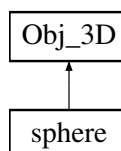
- [Graphics3D.h](#)
- [Graphics3D.cpp](#)

6.6 sphere Class Reference

Class with inherited methods from [Obj_3D](#) that contains constructor for Sphere figure.

```
#include <Graphics3D.h>
```

Inheritance diagram for sphere:



Public Member Functions

- [sphere](#) (double ir, int opt=10)
Constrctor for a sphere, made in (0, 0, 0) position.

Additional Inherited Members

6.6.1 Detailed Description

Class with inherited methods from [Obj_3D](#) that contains constructor for Sphere figure.

It also calls the constructor of an [Obj_3D](#) base class

Parameters

<i>r</i>	Length of an Sphere radius
----------	----------------------------

6.6.2 Constructor & Destructor Documentation

6.6.2.1 sphere()

```
sphere::sphere (
    double ir,
    int opt = 10 )
```

Constrctor for a sphere, made in (0, 0, 0) position.

Parameters

<i>ir</i>	Length of an sphere radius r{ir}
<i>opt</i>	For information aobut opt check documentation main site

The documentation for this class was generated from the following files:

- [Graphics3D.h](#)
- [Graphics3D.cpp](#)

6.7 Vertice Class Reference

[Vertice](#) is point in 3D space, it is the most basic class of 3D classes.

```
#include <Graphics3D.h>
```

Public Member Functions

- [Vertice](#) ()
Vertice Constructor with no values (default: (0, 0, 0))
- [Vertice](#) (double ix, double iy, double iz)

- Vertice* Constructor with values (ix, iy, iz)
- *Vertice* & **operator+** (*Vertice* &other)
Adding two *Vertice* objects (changes main *Vertice*)
- *Vertice* & **operator-** (*Vertice* &other)
Subtracting two *Vertice* objects (changes main *Vertice*)
- *Vertice* & **operator*** (double k)
Multiplying *Vertice* object by number (changes *Vertice*)
- *Vertice* **operator[]** (int i)
Getting data *Vertice* as one number (x, y or z)
- double **operator()** (char type)
Getting data *Vertice* as one number (x, y or z)
- double **operator()** (int type)
Getting data *Vertice* as one number (x, y or z)
- double **get2D** (char type)
- double **get2D** (int type)
- double **get** (int i)
Same as *Vertice::operator[]*.
- void **rePosition** (double ax, double ay, double az)
Sets new position for x, y, z.
- void **addPosition** (double ax, double ay, double az)
Adds values according to *Vertice* data.
- void **multiplePosition** (double kx, double ky=1, double kz=1)
Multiplies values according to *Vertice* data.
- void **rotate** (double ax, double ay=0, double az=0, *Vertice* anchorPoint=*Vertice*(0, 0, 0))
Rotates the *Vertice* accorging to Anchor Point position as center by angle as parameters.
- void **printAll** ()
Prints all the data as plain text.
- ostream & **printAll** (ostream &c)

Friends

- ostream & **operator<<** (ostream &c, const *Vertice* &orig)

6.7.1 Detailed Description

Vertice is point in 3D space, it is the most basic class of 3D classes.

Parameters

x	- width
y	- height
z	- depth

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Vertice()

```
Vertice::Vertice (
    double ix,
    double iy,
    double iz )
```

[Vertice](#) Constructor with values (ix, iy, iz)

Parameters

<i>ix</i>	- ix = x
<i>iy</i>	- iy = y
<i>iz</i>	- iz = z

See also

[Vertice](#)

6.7.3 Member Function Documentation

6.7.3.1 addPosition()

```
void Vertice::addPosition (
    double ax,
    double ay,
    double az )
```

Adds values according to [Vertice](#) data.

See also

[Vertice::operator+\(\)](#)

Parameters

<i>ax</i>	Value added to x
<i>ay</i>	Value added to y
<i>az</i>	Value added to z

6.7.3.2 get()

```
double Vertice::get (
    int i )
```

Same as [Vertice::operator\[\]](#).

Parameters

<i>i</i>	0-2
----------	-----

See also

[Vertice::operator\[\]\(\)](#)

6.7.3.3 multiplePosition()

```
void Vertice::multiplePosition (
    double kx,
    double ky = 1,
    double kz = 1 )
```

Multiplies values according to [Vertice](#) data.

See also

[Vertice::operator*\(\)](#)

Parameters

<i>ax</i>	Multitplier for x
<i>ay</i>	Multitplier for y
<i>az</i>	Multitplier for z

6.7.3.4 operator()() [1/2]

```
double Vertice::operator() (
    char type ) [inline]
```

Getting data [Vertice](#) as one number (x, y or z)

See also

[Vertice](#)

Parameters

<i>type</i>	x, y, z according to Vertice values
-------------	---

Returns

'x' for x, 'y' for y, 'z' for z; as one value

6.7.3.5 operator>() [2/2]

```
double Vertice::operator() (
    int type ) [inline]
```

Getting data [Vertice](#) as one number (x, y or z)

See also

[Vertice::operator\[\]\(\)](#)

6.7.3.6 operator*()

```
Vertice & Vertice::operator* (
    double k )
```

Multiplying [Vertice](#) object by number (changes [Vertice](#))

Warning

This function will change [Vertice](#) before operator

Parameters

<i>k</i>	multiplier for every value in Vertice (x, y ,z)
----------	---

See also

[Vertice](#)

Returns

Main [Vertice](#) that has been changed

6.7.3.7 operator+()

```
Vertice & Vertice::operator+ (
    Vertice & other )
```

Adding two [Vertice](#) objects (changes main [Vertice](#))

Warning

This function will change [Vertice](#) before operator

Parameters

<i>other</i>	Vertice after '+' operator to add
--------------	---

Returns

Main [Vertice](#) that has been changed

6.7.3.8 operator-()

```
Vertice & Vertice::operator- (
    Vertice & other )
```

Substracting two [Vertice](#) objects (changes main [Vertice](#))

Warning

This function will change [Vertice](#) before operator

Parameters

<i>other</i>	Vertice after '-' will be substractor to first object
--------------	---

Returns

Main [Vertice](#) that has been changed

6.7.3.9 operator[]()

```
Vertice Vertice::operator[] (
    int i )
```

Getting data [Vertice](#) as one number (x, y or z)

See also

[Vertice](#)

Parameters

<i>i</i>	0-2 for values inside of Vertice
----------	--

Returns

0 for x, 1 for y, 2 for z; as one value

6.7.3.10 rePosition()

```
void Vertice::rePosition (
    double ax,
    double ay,
    double az )
```

Sets new position for x, y, z.

Parameters

<i>ax</i>	New position of x
<i>ay</i>	New position of y
<i>az</i>	New position of z

6.7.3.11 rotate()

```
void Vertice::rotate (
    double ax,
    double ay = 0,
    double az = 0,
    Vertice anchorPoint = Vertice(0, 0, 0) )
```

Rotates the [Vertice](#) according to Anchor Point position as center by angle as parameters.

Parameters

<i>ax</i>	Angle in which Vertice will rotate by x axis
<i>ay</i>	Angle in which Vertice will rotate by y axis
<i>az</i>	Angle in which Vertice will rotate by z axis
<i>anchorPoint</i>	Center pont for axis to be rotated of

The documentation for this class was generated from the following files:

- [Graphics3D.h](#)
- [Graphics3D.cpp](#)

Chapter 7

File Documentation

7.1 Graphics3D.h File Reference

```
#include <iostream>
#include <math.h>
#include <allegro5/allegro.h>
#include <allegro5/allegro_primitives.h>
```

Classes

- class [Vertice](#)
Vertice is point in 3D space, it is the most basic class of 3D classes.
- class [Edge](#)
Edge is a line between two Vertices in 3D space.
- class [Obj_3D](#)
Basic Figure class, base class for some of the next Figure classes. It contains every method to manipulate Figures.
- class [cube](#)
Class with inherited methods from [Obj_3D](#) that contains constructor for Cube figure.
- class [sphere](#)
Class with inherited methods from [Obj_3D](#) that contains constructor for Sphere figure.
- class [cone](#)
Class with inherited methods from [Obj_3D](#) that contains constructor for Cone figure.
- class [canvas](#)
Class for Allegro 5 canvas initiation and animation handling.

Assignment 110 3 dimensional graphics

Write a library drawing objects defined as simple 3-dimensional shapes. These shapes should be stored separately e.g. as a collection of lines, arcs etc. and drawn on request. Use the Allegro library.

Version

1.0

Date

2021-06-24

Copyright

Copyright (c) 2021

- `#define pi` 3.14159265359
- `#define eps` 6
- `#define depth` 200
- `double aprox` (double x, int y)
Approximate value.
- `ostream & operator<<` (ostream &c, const [Vertice](#) &orig)
- `ostream & operator<<` (ostream &c, const [Edge](#) &orig)

7.1.1 Detailed Description

AuthorRobert Okenczyc (robeoke723@student.polsl.pl)

7.1.2 Function Documentation

7.1.2.1 aprox()

```
double aprox (  
    double x,  
    int y )
```

Approximate value.

Working with more than few double types could slow down work so this function is for approximate purposes, but using it could result in wrong positions after few changes

Parameters

x	value to approx
y	how many decimal numbers will there be (10^y)

Returns

Approximated value with 10^y of decimal numbers

Index

addPosition
 Vertex, [24](#)
aprox
 Graphics3D.h, [32](#)

canvas, [11](#)
cone, [11](#)
 cone, [12](#)
cube, [13](#)
 cube, [13](#)

Edge, [15](#)
 Edge, [16](#)
 operator=, [16](#)
 position, [17](#)

get
 Vertex, [24](#)
Graphics3D.h, [31](#)
 aprox, [32](#)

multiplePosition
 Vertex, [26](#)

Obj_3D, [17](#)
 Obj_3D, [18](#)
 printAll, [20](#)
 reposition, [20](#)
 rotateFigure, [21](#)

operator*
 Vertex, [27](#)
operator()
 Vertex, [26](#), [27](#)
operator+
 Vertex, [27](#)
operator-
 Vertex, [28](#)
operator=
 Edge, [16](#)
operator[]
 Vertex, [28](#)

position
 Edge, [17](#)
printAll
 Obj_3D, [20](#)

rePosition
 Vertex, [29](#)
reposition
 Obj_3D, [20](#)

rotate
 Vertex, [29](#)
rotateFigure
 Obj_3D, [21](#)

sphere, [21](#)
 sphere, [22](#)

Vertex, [22](#)
 addPosition, [24](#)
 get, [24](#)
 multiplePosition, [26](#)
 operator*, [27](#)
 operator(), [26](#), [27](#)
 operator+, [27](#)
 operator-, [28](#)
 operator[], [28](#)
 rePosition, [29](#)
 rotate, [29](#)
 Vertex, [23](#)