Politechnika
Śląska

SILESIAN UNIVERSITY OF TECHNOLOGY

Faculty of Automatic control, Electronics and Informatics

Computer programming
(110)**3D Graphic** - final report

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

Project topic:

3D graphics - write a library for drawing object defined as 3D shapes. Shapes should be stored separately as a collection of lines or arcs; have to be drawn on request.

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.

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); //Initialization
while (Canvas1.endCondition()) {
   while (true) {
      if (Canvas1.FrameInit()) { return 0; }

      // Shapes and transformations/deformations goes here

      Canvas1.keyframe();
   }
}
```

As said earlier this is a library, so most of this report will be documentations of classes and methods, but here I can write problems that I had encountered and solutions to programming and algebra problems.

For cube I had made simple solution; every Vertice's position have been set to (R, R, R) [R is outer radius of a cube circumscribed on the sphere (R is equal to edge times square root of 3 by 2) and then rotate them in a for loop where rotations in X axis is equal to 90 * i and rotation in Y axis by 0 for 4 Vertices and 90 for rest of them.

Sphere was much more challenging, well for me. I had a lot of difficulties in writing my idea of making sphere in 3D space but i came up with setting all Vertices to position (0, radius, 0) and rotating it like a cube but with constant angle by two axis (y, z) angle is equal to 180 / opt and there are 2 * (opt * (opt - 1) + 1) Vertices and 2 * opt * (2 * opt - 1) edges; opt * 2 columns and opt + 1 rows.

After Sphere I had made Cone which was straight forward, circle with opt number of angles and one Vertice above them.

Testing/ debugging :

This library has little testing problems, In tested values the most visible problem is lagging and slow working with large amounts of Vertices (fog. Sphere with opt set to 1000) and other crucial bug is when you set the opt value as a negative number which is a thing to be fixed, but everything else is working properly even if you enter distances and lengths as a negative values, most of times it will just make point go other direction.

Chapter 1

Computer programming (110)

1.0.1 3D Graphic library - specyfication 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 Allego 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:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Class Documentation

6.1 canvas Class Reference

Class for Allegro 5 canvas initiation and animation handeling.

```
#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 handeling.

Parameters

x	= width of an Allegro 5 window
y	= height of an Allego 5 window

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

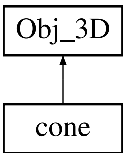
- Graphics3D.h
- Graphics3D.cpp

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)

  Constrctor 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

r	Length of an base circe radius
H	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 )
```

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

Parameters

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

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

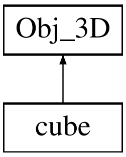
- Graphics3D.h
- Graphics3D.cpp

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

a	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 )
```

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

It also calls the constructor of an Obj_3D base class

Parameters

ia	Length of an Cube edge a{ia}
v	Vertices count
e	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 (Vertice &iv1, Vertice &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

- Vertice ∗ **v1**
- Vertice ∗ **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

Vertice

Parameters

∗v1	Pointer to the first Vertice of this Edge
∗v2	Pointer to the seccond Vertice of this Edge

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Edge() [1/2]

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

Constructor of an Edge.

Parameters

iv1	Vertice 1 for v1
iv2	Vertice 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

other	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

Vertice

Edge

Vertice::addPosition()

Parameters

ix	Value added to both Vertices in Edge xs
iy	Value added to both Vertices in Edge ys
iz	Value added to both Vertices in Edge zs

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

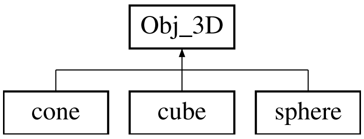
- Graphics3D.h
- Graphics3D.cpp

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=al_map_rgb(255, 0, 0), int b=2)
- void **drawPixels** (ALLEGRO_COLOR a=al_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

anchorPoint	Center of the figure
vArrSize	Size of the vArr array (array of Vertices)
eArrSize	Size of the eArr array (array of Edges)
∗*vArr*	Array of pointers for Vertices
∗*eArr*	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

v	value for vArrSize

See also

Obj_3D::vArrSize

Parameters

e	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

Vertice::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

ax	Distance for xs to move
ay	Distance for ys to move
az	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

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

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

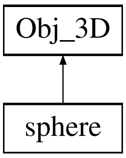
- Graphics3D.h
- Graphics3D.cpp

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

r	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

ir	Length of an sphere radius r{ir}
opt	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)

  Substracting two Vertice objects (changes main Vertice)
- Vertice & operator∗ (double k)

  Multipling 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

ix	- ix = x
iy	- ix = x
iz	- ix = x

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

ax	Value added to x
ay	Value added to y
az	Value added to z

6.7.3.2 get()

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

Same as Vertice::operator[ ].

Parameters

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

ax	Mulitplier for x
ay	Mulitplier for y
az	Mulitplier 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

type	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 )
```

Multipling Vertice object by number (changes Vertice)

Warning

This function will change Vertice before operator

Parameters

k	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

other	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

other	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	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

ax	New position of x
ay	New position of y
az	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 accorging to Anchor Point position as center by angle as parameters.

Parameters

ax	Angle in which Vertice will rotate by x axis
ay	Angle in which Vertice will rotate by y axis
az	Angle in which Vertice will rotate by z axis
anchorPoint	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 handeling.

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

Author

Robert Okenczyc (robeoke723@student.polsl.pl)

7.1.2 Function Documentation

7.1.2.1 aprox()

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

Approximate value.

Woring with more than few double types could slow down work so this fucntion is for approximate purpuses, 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