COP290

Generated by Doxygen 1.8.15

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

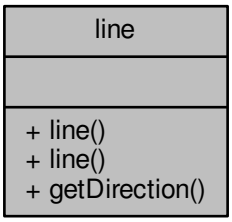
Chapter 3

Class Documentation

3.1 line Class Reference

`#include <point.h>`

Collaboration diagram for line:



Public Member Functions

- line (point p1, point p2)
- line (float a, float b, float c)
- point getDirection ()

3.1.1 Constructor & Destructor Documentation

3.1.1.1 line() [1/2]

```
line::line (
            point p1,
            point p2 )
```

3.1.1.2 line() [2/2]

```
line::line (
             float a,
             float b,
             float c )
```

3.1.2 Member Function Documentation

3.1.2.1 getDirection()

```
point line::getDirection ( )
```

Returns

direction pointed to by the line

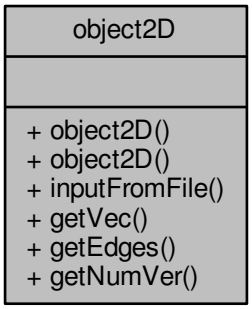
The documentation for this class was generated from the following file:

- point.h

3.2 object2D Class Reference

```
#include <object2D.h>
```

Collaboration diagram for object2D:



Public Member Functions

- object2D ()
- object2D (vector< glm::vec4 > vertices_vec_2d, vector< vector< int > > line_2d)
- void inputFromFile (string filename)
- vector< glm::vec4 > getVec ()
- vector< vector< int > > getEdges ()
- int getNumVer ()

3.2.1 Constructor & Destructor Documentation

3.2.1.1 object2D() [1/2]

```
object2D::object2D ( )
```

constructor for the 2D figure class

3.2.1.2 object2D() [2/2]

```
object2D::object2D (
            vector< glm::vec4 > vertices_vec_2d,
            vector< vector< int > > line_2d )
```

3.2.2 Member Function Documentation

3.2.2.1 getEdges()

```
vector<vector<int> > object2D::getEdges ( )
```

function to return the edges between all the vertices of the object in form of adjacency list.

3.2.2.2 getNumVer()

```
int object2D::getNumVer ( )
```

function to return the number of vertices in the object.

3.2.2.3 getVec()

```
vector<glm::vec4> object2D::getVec ( )
```

function to return all the coordinates of the 2d object

3.2.2.4 inputFromFile()

```
void object2D::inputFromFile (
            string filename )
```

function to input the 2D figure from file

Parameters

filename	File to be taken as input Then it check the consistency If inconsistant, raises exception else constructs the object

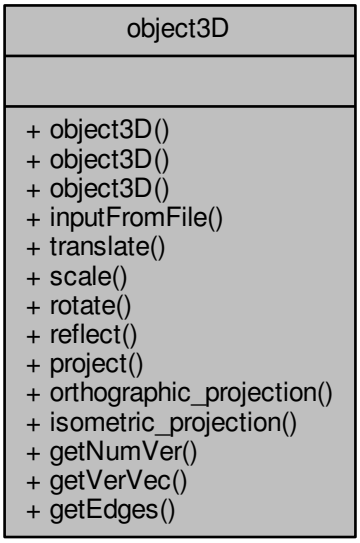
The documentation for this class was generated from the following file:

- object2D.h

3.3 object3D Class Reference

`#include <object3D.h>`

Collaboration diagram for object3D:



Public Member Functions

- object3D ()
- object3D (vector< glm::vec4 > vertices_vec_3d, vector< vector< int > > line_3d)
- object3D (orthographic obj)
- void inputFromFile (string filename)
- void translate (float x=0.0, float y=0.0, float z=0.0)
- void scale (float x=1.0, float y=1.0, float z=1.0)
- void rotate (float theta, line axis)
- void reflect (plane p)
- object2D project (plane p)
- orthographic orthographic_projection ()
- object2D isometric_projection ()
- int getNumVer ()
- vector< glm::vec4 > getVerVec ()
- vector< vector< int > > getEdges ()

3.3.1 Constructor & Destructor Documentation

3.3.1.1 object3D() [1/3]

```
object3D::object3D ( )
```

Constructs a blank 3D figure

3.3.1.2 object3D() [2/3]

```
object3D::object3D (
            vector< glm::vec4 > vertices_vec_3d,
            vector< vector< int > > line_3d )
```

contructor of object3D class given the vertices coordinates and the connection of points stored in adjacency list.

Parameters

vertices_vec_3d	vertices co-ordinates
line_3d	adjacency list of connection of points.

3.3.1.3 object3D() [3/3]

```
object3D::object3D (
            orthographic obj )
```

Function to interactively input the points and their connections in form of lines and then it check the consistency If inconsistent, raises exception else constructs the object.

3.3.2 Member Function Documentation

3.3.2.1 getEdges()

```
vector<vector<int> > object3D::getEdges ( )
```

function to return the edges between all the vertices of the object in form of adjacency list.

3.3.2.2 getNumVer()

```
int object3D::getNumVer ( )
```

function to return the number of vertices in the object.

3.3.2.3 getVerVec()

```
vector<glm::vec4> object3D::getVerVec ( )
```

function to return all the coordinates of the 2d object

3.3.2.4 inputFromFile()

```
void object3D::inputFromFile (
            string filename )
```

Function to input the 2D figure from file and then it check the consistency If inconsistent, raises exception else constructs the object.

Parameters

filename	File to be taken as input

3.3.2.5 isometric_projection()

```
object2D object3D::isometric_projection ( )
```

Calculates isometric projection of the 3D Figure

3.3.2.6 orthographic_projection()

```
orthographic object3D::orthographic_projection ( )
```

Calculates orthographic projections of the 3D Figure

3.3.2.7 project()

```
object2D object3D::project (
            plane p )
```

Projects the 3D Figure on a given plane

Parameters

p	plane on which to project

3.3.2.8 reflect()

```
void object3D::reflect (
            plane p )
```

Reflects an object about a given plane

Parameters

p	plane about which to reflect

3.3.2.9 rotate()

```
void object3D::rotate (
            float theta,
            line axis )
```

Rotates the figure around a given axis by a given angle

Parameters

theta	angle to be rotated by
axis	axis about which object is to be rotated

3.3.2.10 scale()

```
void object3D::scale (
            float x = 1.0,
            float y = 1.0,
            float z = 1.0 )
```

Scales the 3D Figure proportionately

Parameters

x	amount of scaling in x-direction
y	amount of scaling in y-direction
z	amount of scaling in z-direction

3.3.2.11 translate()

```
void object3D::translate (
            float x = 0.0,
```

```
float y = 0.0,
float z = 0.0 )
```

Translates the 3D Figure to a different 3D location.

Parameters

x	amount of translation in x-direction
y	amount of translation in y-direction
z	amount of translation in z-direction

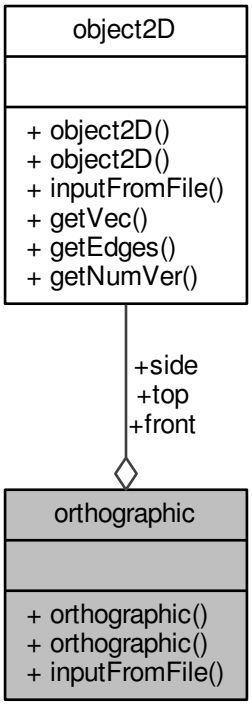
The documentation for this class was generated from the following file:

- object3D.h

3.4 orthographic Class Reference

```
#include <orthographic.h>
```

Collaboration diagram for orthographic:



Public Member Functions

- orthographic (object2D f, object2D s, object2D t)
- orthographic ()
- void inputFromFile (string filename)

Public Attributes

- object2D front
- object2D side
- object2D top

3.4.1 Constructor & Destructor Documentation

3.4.1.1 orthographic() [1/2]

```
orthographic::orthographic (
            object2D f,
            object2D s,
            object2D t )
```

constructor of the orthographic views

Parameters

f	Takes the reference of the front view 2D object
ba	Takes the reference of the back view 2D object
l	Takes the reference of the left view 2D object
r	Takes the reference of the right view 2D object
t	Takes the reference of the top view 2D object
bo	Takes the reference of the bottom view 2D object

3.4.1.2 orthographic() [2/2]

```
orthographic::orthographic ( )
```

3.4.2 Member Function Documentation

3.4.2.1 inputFromFile()

```
void orthographic::inputFromFile (
            string filename )
```

Parameters

filename	File from which input is taken.

3.4.3 Member Data Documentation

3.4.3.1 front

```
object2D orthographic::front
```

3.4.3.2 side

```
object2D orthographic::side
```

3.4.3.3 top

```
object2D orthographic::top
```

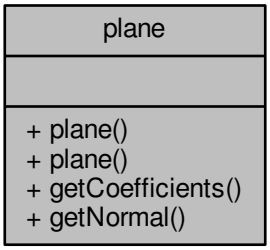
The documentation for this class was generated from the following file:

- orthographic.h

3.5 plane Class Reference

```
#include <point.h>
```

Collaboration diagram for plane:



Public Member Functions

- plane (float A, float B, float C, float D)
- plane (point p, line nor)
- float * getCoefficients ()
- point getNormal ()

3.5.1 Constructor & Destructor Documentation

3.5.1.1 plane() [1/2]

```
plane::plane (
            float A,
            float B,
            float C,
            float D )
```

3.5.1.2 plane() [2/2]

```
plane::plane (
            point p,
            line nor )
```

3.5.2 Member Function Documentation

3.5.2.1 getCoefficients()

```
float* plane::getCoefficients ( )
```

Returns an array containing the co-efficients of the plane a,b,c,d respectively

3.5.2.2 getNormal()

```
point plane::getNormal ( )
```

Returns the normal of the plane

Returns

A vector representing the components along x, y, and z directions

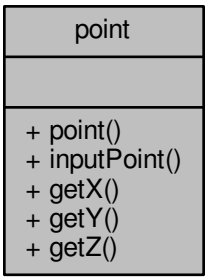
The documentation for this class was generated from the following file:

- point.h

3.6 point Class Reference

`#include <point.h>`

Collaboration diagram for point:



Public Member Functions

- point (float x_coord, float y_coord, float z_coord)
- void inputPoint ()
- float getX ()
- float getY ()
- float getZ ()

3.6.1 Constructor & Destructor Documentation

3.6.1.1 point()

```
point::point (
            float x_coord,
            float y_coord,
            float z_coord )
```

class to store the point for the graphics object

Parameters

x_coord	X-coordinate with default initilisation zero
y_coord	Y-coordinate with default initilisation zero
z_coord	Z-coordinate with default initilisation zero

3.6.2 Member Function Documentation

3.6.2.1 getX()

```
float point::getX ( )
```

returns the x-coordinate of the point

3.6.2.2 getY()

```
float point::getY ( )
```

returns the y-coordinate of the point

3.6.2.3 getZ()

```
float point::getZ ( )
```

returns the z-coordinate of the point

3.6.2.4 inputPoint()

```
void point::inputPoint ( )
```

input the x,y,z coordinate in that order from stdin

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

- point.h

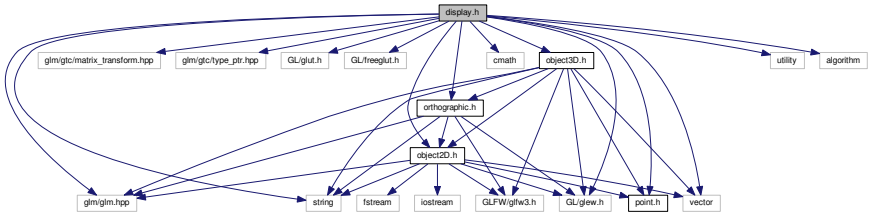
Chapter 4

File Documentation

4.1 display.h File Reference

```
#include <GL/glew.h>
#include <glm/glm.hpp>
#include "glm/gtc/matrix_transform.hpp"
#include <glm/gtc/type_ptr.hpp>
#include <GL/glut.h>
#include <GL/freeglut.h>
#include <string>
#include <vector>
#include <cmath>
#include "point.h"
#include "object2D.h"
#include "orthographic.h"
#include "object3D.h"
#include <utility>
#include <algorithm>
```

Include dependency graph for display.h:



Functions

- void initialiseGlut ()
- void drawAxis ()
- void drawAxisOrtho ()
- void keyboard (int key, int x, int y)
- void makeLine (glm::vec4 v1, glm::vec4 v2)
- void clear ()

- void clearSolid ()
- void renderWire3D ()
- void renderWire2D ()
- void makeTriangle (glm::vec4 a, glm::vec4 b, glm::vec4 c, glm::vec3 d)
- glm::vec3 cross (glm::vec4 a, glm::vec4 b)
- void renderSolid3D ()
- void makeWireFrame (object3D obj)
- void makeWireFrame (object2D obj)
- void makeSolid (object3D obj)
- void makeOrthographic (orthographic obj)
- void renderOrtho ()
- void translatedRender (glm::vec4 tr, glm::vec4 sca, int view_number)
- glm::vec4 changeCoord (glm::vec4 ver, int view_number)
- void makeIsometric (object2D obj)
- void renderWireIso ()

Variables

- float rX
- float rY
- int global_argc
- char ** global_argv
- object3D rendering_obj3D
- object3D global_obj3D
- object2D rendering_obj2D
- object2D global_obj2D
- orthographic rendering_objOrtho
- orthographic global_orthoObj

4.1.1 Function Documentation

4.1.1.1 changeCoord()

```
glm::vec4 changeCoord (
            glm::vec4 ver,
            int view_number )
```

function to rotate the coordinate system for orthographic views.

4.1.1.2 clear()

```
void clear ( )
```

function to clear the glut screen in wireframe model

4.1.1.3 clearSolid()

```
void clearSolid ( )
```

function to clear the glut screen in the 3D models

4.1.1.4 cross()

```
glm::vec3 cross (
            glm::vec4 a,
            glm::vec4 b )
```

function to take the cross product of two vectors

4.1.1.5 drawAxis()

```
void drawAxis ( )
```

function to display the axis in the glut display window

4.1.1.6 drawAxisOrtho()

```
void drawAxisOrtho ( )
```

function to display axis in orthographic view

4.1.1.7 initialiseGlut()

```
void initialiseGlut ( )
```

function to initialize the glut window for display

4.1.1.8 keyboard()

```
void keyboard (
            int key,
            int x,
            int y )
```

function to control the 3D figure rotation from the keyboard arrow keys

4.1.1.9 makeIsometric()

```
void makeIsometric (
            object2D obj )
```

4.1.1.10 makeLine()

```
void makeLine (
            glm::vec4 v1,
            glm::vec4 v2 )
```

function to draw a line in the glut window between two points

Parameters

v1	takes first input vertex as glm vector
v2	takes second input vertex as glm vector

4.1.1.11 makeOrthographic()

```
void makeOrthographic (
            orthographic obj )
```

function to actually setup call back functions for rendering orthographic views

4.1.1.12 makeSolid()

```
void makeSolid (
            object3D obj )
```

function to actually setup call back functions for rendering 3D solid model

4.1.1.13 makeTriangle()

```
void makeTriangle (
            glm::vec4 a,
            glm::vec4 b,
            glm::vec4 c,
            glm::vec3 d )
```

function to draw triangle between 3 points and make surface from the normal

Parameters

a	first vertex
b	second vertex
c	third vertex
d	normal vector to the surface

4.1.1.14 makeWireFrame() [1/2]

```
void makeWireFrame (
            object3D obj )
```

function to actually setup call back functions for rendering 3D wireframe object

4.1.1.15 makeWireFrame() [2/2]

```
void makeWireFrame (
            object2D obj )
```

function to actually setup call back functions for rendering 2D wireframe object

4.1.1.16 renderOrtho()

```
void renderOrtho ( )
```

function to display the orthographic views on the glut screen

4.1.1.17 renderSolid3D()

```
void renderSolid3D ( )
```

function to render 3D solid model with surface shading

4.1.1.18 renderWire2D()

```
void renderWire2D ( )
```

function to render 2D wireframe model

4.1.1.19 renderWire3D()

```
void renderWire3D ( )
```

function to render 3D wireframe model

4.1.1.20 renderWireIso()

```
void renderWireIso ( )
```

4.1.1.21 translatedRender()

```
void translatedRender (
            glm::vec4 tr,
            glm::vec4 sca,
            int view_number )
```

function to correctly translate the orthographic views for display

4.1.2 Variable Documentation

4.1.2.1 global_argc

```
int global_argc
```

4.1.2.2 global_argv

```
char** global_argv
```

4.1.2.3 global_obj2D

```
object2D global_obj2D
```

4.1.2.4 global_obj3D

```
object3D global_obj3D
```

4.1.2.5 global_orthoObj

```
orthographic global_orthoObj
```

4.1.2.6 rendering_obj2D

```
object2D rendering_obj2D
```

4.1.2.7 rendering_obj3D

```
object3D rendering_obj3D
```

4.1.2.8 rendering_objOrtho

```
orthographic rendering_objOrtho
```

4.1.2.9 rX

```
float rX
```

4.1.2.10 rY

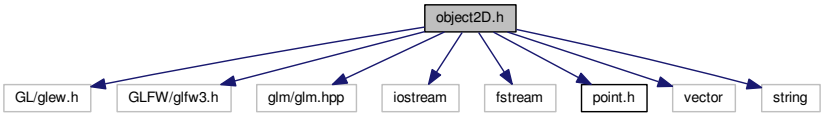
```
float rY
```

4.2 object2D.h File Reference

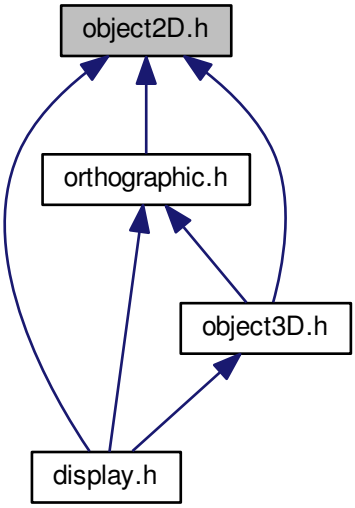
This file contains the class object2D, which is used to model a 2D figure.

```
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include <iostream>
#include <fstream>
#include "point.h"
#include <vector>
#include <string>
```

Include dependency graph for object2D.h:



This graph shows which files directly or indirectly include this file:



Classes

- class object2D

4.2.1 Detailed Description

This file contains the class object2D, which is used to model a 2D figure.

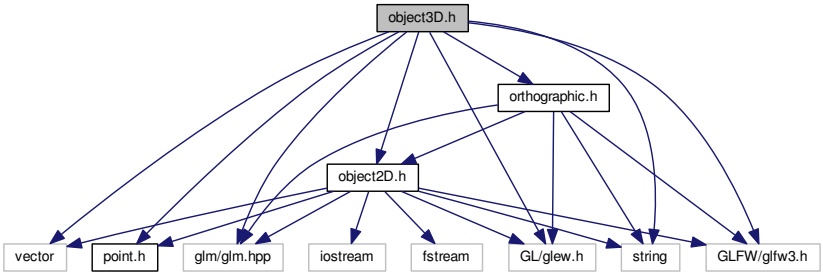
4.3 object3D.h File Reference

This file contains the class object3D which is used to model a 3D object.

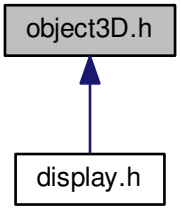
```
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include <string>
#include <vector>
#include "point.h"
#include "object2D.h"
```

```
#include "orthographic.h"
```

Include dependency graph for object3D.h:



This graph shows which files directly or indirectly include this file:



Classes

- class object3D

4.3.1 Detailed Description

This file contains the class object3D which is used to model a 3D object.

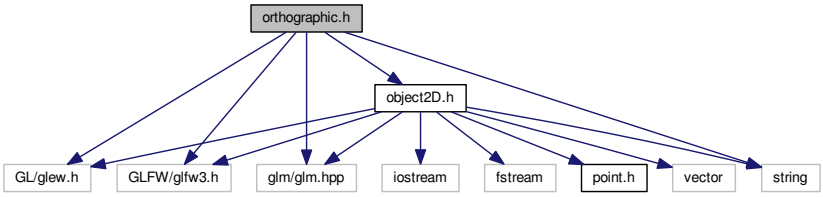
4.4 orthographic.h File Reference

This file contains the class orthographic which is used to model orthographic projections of a 3D object.

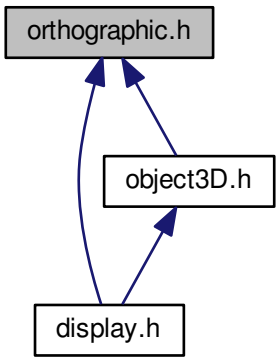
```
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include "object2D.h"
```

```
#include <string>
```

Include dependency graph for orthographic.h:



This graph shows which files directly or indirectly include this file:



Classes

- class orthographic

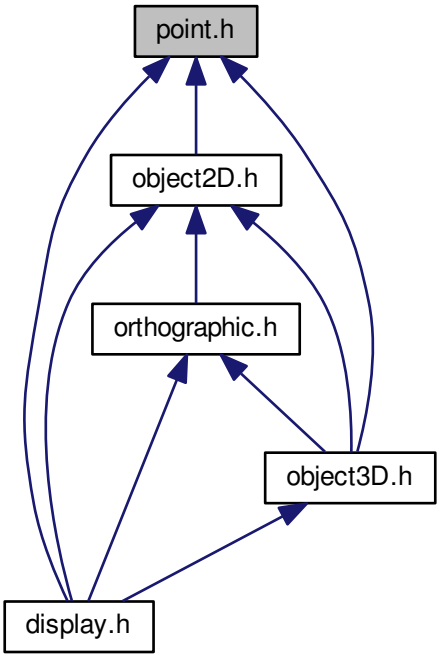
4.4.1 Detailed Description

This file contains the class orthographic which is used to model orthographic projections of a 3D object.

4.5 point.h File Reference

This file contains the classes used to model various geometric objects such as point, line and a plane.

This graph shows which files directly or indirectly include this file:



Classes

- class point
- class line
- class plane

4.5.1 Detailed Description

This file contains the classes used to model various geometric objects such as point, line and a plane.

Index