CADapp

Generated by Doxygen 1.8.11

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 edge Struct Reference

structure for an edge

```
#include <drawing.h>
```

Public Member Functions

- edge ()
- edge (const edge &)=default
- edge (vertex a, vertex b, bool v=true)
- template<class Archive >
  void serialize (Archive &ar)

Public Attributes

- vertex v1
- vertex v2
- bool visi

3.1.1 Detailed Description

structure for an edge

Edge is represented by pair of vertices. It also stores the information of visibility of an edge when required

3.1.2 Constructor & Destructor Documentation

3.1.2.1 edge::edge ( )

3.1.2.2 edge::edge (const edge &) [default]

3.1.2.3 edge::edge (vertex *a*, vertex *b*, bool *v* = true)

3.1.3 Member Function Documentation

3.1.3.1 template<class Archive > void edge::serialize (Archive & *ar*) [inline]

3.1.4 Member Data Documentation

3.1.4.1 vertex edge::v1

3.1.4.2 vertex edge::v2

3.1.4.3 bool edge::visi

bool variable denoting the visibility

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

- src/drawing.h
- src/drawing.cpp

3.2 edge2D Struct Reference

2D edge

```
#include <drawing.h>
```

Public Member Functions

- edge2D ()
- edge2D (const edge2D &)=default
- edge2D (vert2D a, vert2D b, bool v=true)
- edge2D (const edge e)
- template<class Archive >
  void serialize (Archive &ar)

Public Attributes

- vert2D v1
- vert2D v2
- bool visi

3.2.1 Detailed Description

2D edge

See also

edge

3.2.2 Constructor & Destructor Documentation

3.2.2.1 edge2D::edge2D ( )

3.2.2.2 edge2D::edge2D (const edge2D &) [default]

3.2.2.3 edge2D::edge2D (vert2D *a*, vert2D *b*, bool *v* = true)

3.2.2.4 edge2D::edge2D (const edge *e*)

3.2.3 Member Function Documentation

3.2.3.1 template<class Archive > void edge2D::serialize (Archive & *ar*) [inline]

3.2.4 Member Data Documentation

3.2.4.1 vert2D edge2D::v1

3.2.4.2 vert2D edge2D::v2

3.2.4.3 bool edge2D::visi

bool variable denoting the visibility

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

- src/drawing.h
- src/drawing.cpp

3.3 face Struct Reference

structure for polygon face of a 3D object

```
#include <drawing.h>
```

Public Member Functions

- face ()
- face (const face &)=default
- void compParam ()
- template<class Archive >
  void serialize (Archive &ar)

Public Attributes

- float A
- float B
- float C
- float D
- map< string, edge > edges
- map< string, vertex > verts

3.3.1 Detailed Description

structure for polygon face of a 3D object

A polygon face is represented by a list of edges. The equation of plane of the face is represented by the A,B,C & D parameters

3.3.2 Constructor & Destructor Documentation

3.3.2.1 face::face ( )

function to compute the equation of the plane from the list of edges.

3.3.2.2 face::face (const face &) [default]

3.3.3 Member Function Documentation

3.3.3.1 void face::compParam ( )

3.3.3.2 template<class Archive > void face::serialize (Archive & *ar*) [inline]

3.3.4 Member Data Documentation

3.3.4.1 float face::A

3.3.4.2 float face::B

3.3.4.3 float face::C

3.3.4.4 float face::D

3.3.4.5 map<string, edge> face::edges

list of edges

3.3.4.6 map< string, vertex > face::verts

list of vertices

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

- src/drawing.h
- src/drawing.cpp

3.4 Object3D Class Reference

Class for representing 3D objects.

```
#include <drawing.h>
```

Public Member Functions

- Object3D ()
- Projection projectTo2D (string view)

  To compute Orthographic projections.
- void create (Projection FV, Projection TV, Projection SV, bool rightside=true, bool righthand=true)

  Initialize the 3D object using 3 Orthographic projections.
- void rotate (float alpha, float beta, float gamma)

  Rotation of the 3D object with respect to the given coordinate axes. All angles are in degrees.
- void shift (float x0, float y0, float z0)

  shifting of origin of the 3D coordinate axes
- template< class Archive >
  void serialize (Archive &ar)

Static Public Member Functions

- static void display ()

  Method to render image of the object.
- static void display_wireframe ()

Public Attributes

- map< string, face > flist
- map< string, edge > elist
- map< string, vertex > vlist

Protected Member Functions

- pair< float, float > _intersect_ratiois (edge e1, edge e2)
- pair< map< string, edge >, map< string, vertex > > _wireframe (Projection FV, Projection TV, Projection SV, bool rightside=true, bool righthand=true)
- void _overlappingEdges (map< string, edge > &els, map< string, vertex > &vls)
- void _intersectingEdges (map< string, edge > &els, map< string, vertex > &vls)
- void _dashedLines (map< string, edge > &els, map< string, face > &fls)

3.4.1 Detailed Description

Class for representing 3D objects.

A 3D object is represented by a list of faces, edges and vertices.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Object3D::Object3D ( )

3.4.3 Member Function Documentation

3.4.3.1 void Object3D::_dashedLines (map< string, edge > & *els*, map< string, face > & *fls*) [protected]

Function to mark the hidden lines as dashed in the Orthographic projection

Parameters

view	a char∗ denoting the view of the projecion. It can take values - "front", "top", "rside", "lside"

3.4.3.2 pair< float, float > Object3D::_intersect_ratiois (edge *e1*, edge *e2*) [protected]

3.4.3.3 void Object3D::_intersectingEdges (map< string, edge > & *els*, map< string, vertex > & *vls*) [protected]

Function to handle intersecting edges while generating Orthographic projection.

Parameters

view	a char∗ denoting the view of the projecion. It can take values - "front", "top", "rside", "lside"

3.4.3.4 void Object3D::_overlappingEdges (map< string, edge > & *els*, map< string, vertex > & *vls*) [protected]

Methods for Orthographic view generation /∗! Function to handle overlapping edges while generating Orthographic projection.

Parameters

view	a char∗ denoting the view of the projecion. It can take values - "front", "top", "rside", "lside"

3.4.3.5 pair< map< string, edge >, map< string, vertex > > Object3D::_wireframe (Projection *FV*, Projection *TV*, Projection *SV*, bool *rightside* = `true`, bool *righthand* = `true`) [protected]

Function that constructs the wireframe of the object i.e the edges outlning the 3D object

Parameters

FV	denotes the input front orthographic projection
TV	denotes the input top orthographic projection
SV	denoted the input side orthographic projection
rightside	boolean value indicating wether right side view is taken, default value is true
righthand	boolean value for right/left hand coordinate system to be followed, default value is true

Returns

pair containing list of possible edges and vertices

3.4.3.6 void Object3D::create (Projection *FV*, Projection *TV*, Projection *SV*, bool *rightside* = `true`, bool *righthand* = `true`)

Initialize the 3D object using 3 Orthographic projections.

This function uses 3 Orthographic projection to reconstruct the 3D object from them and initialize itself

Parameters

FV	the front Orthographic projection
TV	the top Orthographic projection
SV	the side view Orthographic projection
rightside	boolean value telling which side view is taken, default true
righthand	boolean value for right/left hand coordinate system to be followed, default true

3.4.3.7 void Object3D::display ( ) `[static]`

Method to render image of the object.

3.4.3.8 void Object3D::display_wireframe ( ) `[static]`

3.4.3.9 Projection Object3D::projectTo2D (string *view*)

To compute Orthographic projections.

Function which computes and returns an Orthographic projection of the object

Parameters

view	a char∗ denoting the view of the projecion. It can take values - "front", "top", "rside", "lside"

Returns

Object of Class Projection

3.4.3.10 void Object3D::rotate (float *alpha*, float *beta*, float *gamma*)

Rotation of the 3D object with respect to the given coordinate axes. All angles are in degrees.

Parameters

alpha	Angular displacement about the x axis
beta	Angular displacement about the y axis
gamma	Angular displacement about the z axis

3.4.3.11 template<class Archive > void Object3D::serialize (Archive & *ar*) [inline]

3.4.3.12 void Object3D::shift (float *x0*, float *y0*, float *z0*)

shifting of origin of the 3D coordinate axes

Parameters

x0	Offset in x direction
y0	Offset in y direction
z0	Offset in z direction

3.4.4 Member Data Documentation

3.4.4.1 map<string, edge> Object3D::elist

3.4.4.2 map<string, face> Object3D::flist

Function that generates the projection after processing all the edges for overlap and intersection.

Parameters

view	a char∗ denoting the view of the projecion. It can take values - "front", "top", "rside", "lside"

Returns

The corresponding orthographic view as Projection object

3.4.4.3 map<string, vertex> Object3D::vlist

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

- src/drawing.h
- src/drawing.cpp

3.5 Projection Class Reference

Class for representing projection.

```
#include <drawing.h>
```

Public Member Functions

- Projection ()
- void getProjection ()
- template<class Archive >
  void serialize (Archive &ar)

Static Public Member Functions

- static void display ()

Public Attributes

- string name
- map< string, edge2D > elist
- map< string, vert2D > vlist

3.5.1 Detailed Description

Class for representing projection.

A projection is given by a list of 2D edges and a list of 2D vertices

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Projection::Projection ( )

3.5.3 Member Function Documentation

3.5.3.1 void Projection::display ( ) [static]

function to display the projection in a window.

3.5.3.2 void Projection::getProjection ( )

Function which helps take projection specification as input

3.5.3.3 template<class Archive > void Projection::serialize (Archive & *ar*) [inline]

3.5.4 Member Data Documentation

3.5.4.1 map<string, edge2D> Projection::elist

3.5.4.2 string Projection::name

3.5.4.3 map<string, vert2D> Projection::vlist

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

- src/drawing.h
- src/drawing.cpp

3.6 vert2D Struct Reference

A 2D vertex.

```
#include <drawing.h>
```

Public Member Functions

- vert2D ()
- vert2D (float _x, float _y)
- vert2D (const vertex &v)
- vert2D (const vert2D &v)
- template<class Archive >
  void serialize (Archive &ar)

Public Attributes

- float x
- float y

3.6.1 Detailed Description

A 2D vertex.

See also

vertex

3.6.2 Constructor & Destructor Documentation

3.6.2.1 vert2D::vert2D ( )

3.6.2.2 vert2D::vert2D (float *_x*, float *_y*)

3.6.2.3 vert2D::vert2D (const vertex & *v*)

3.6.2.4 vert2D::vert2D (const vert2D & *v*)

3.6.3 Member Function Documentation

3.6.3.1 template<class Archive > void vert2D::serialize (Archive & *ar*) `[inline]`

3.6.4 Member Data Documentation

3.6.4.1 float vert2D::x

3.6.4.2 float vert2D::y

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

- src/drawing.h
- src/drawing.cpp

3.7 vertex Struct Reference

A 3D vertex.

`#include <drawing.h>`

Public Member Functions

- vertex ()
- vertex (float _x, float _y, float _z)
- vertex (const vertex &)=default
- bool operator== (const vertex &v)
- vertex operator+ (const vertex &v)
- vertex operator- (const vertex &v)
- vertex operator* (const float &f)
- template<class Archive >
  void serialize (Archive &ar)

Public Attributes

- float x
- float y
- float z

3.7.1 Detailed Description

A 3D vertex.

Structure for a point in 3 dimension.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 vertex::vertex ( )

3.7.2.2 vertex::vertex (float *_x*, float *_y*, float *_z*)

3.7.2.3 vertex::vertex (const vertex &) [default]

3.7.3 Member Function Documentation

3.7.3.1 vertex vertex::operator* (const float & *f*)

3.7.3.2 vertex vertex::operator+ (const vertex & *v*)

3.7.3.3 vertex vertex::operator- (const vertex & *v*)

3.7.3.4 bool vertex::operator== (const vertex & *v*)

3.7.3.5 template<class Archive > void vertex::serialize (Archive & *ar*) [inline]

3.7.4 Member Data Documentation

3.7.4.1 float vertex::x

3.7.4.2 float vertex::y

3.7.4.3 float vertex::z

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

- src/drawing.h
- src/drawing.cpp

Chapter 4

File Documentation

4.1 src/cinterface.cpp File Reference

Implemenation of all wrapper functions.

```
#include "drawing.h"
#include "cinterface.h"
#include <fstream>
#include <regex>
#include <math.h>
#include "../lib/cereal/types/unordered_map.hpp"
#include "../lib/cereal/types/map.hpp"
#include "../lib/cereal/types/utility.hpp"
#include "../lib/cereal/types/string.hpp"
#include "../lib/cereal/types/list.hpp"
#include "../lib/cereal/types/vector.hpp"
#include "../lib/cereal/types/memory.hpp"
#include "../lib/cereal/archives/binary.hpp"
#include "../lib/cereal/access.hpp"
```

Functions

- int _2d3dDrawWrapper (char ∗filepath, int argc, char ∗∗argv)

  2D to 3D conversion wrapper function
- int _3d2dDrawWrapper (char ∗filepath, int argc, char ∗∗argv)

  3D to 2D conversion wrapper function
- void _rotateWrapper (float alpha, float beta, float gamma, int argc, char ∗∗argv)

  Rotation wrapper function.
- int _SaveWrapper (char ∗savepath)

  Rotation wrapper function.
- void display_main (int argc, char ∗∗argv)

  Objection/Projection display function.

Variables

- int window_1
- int window_2
- bool refresh = false

4.1.1 Detailed Description

Implemenation of all wrapper functions.

This file contains the implementation of all the required wrapper funcitions required to interface between UI in C and the Backend in C++

4.1.2 Function Documentation

4.1.2.1 int _2d3dDrawWrapper (char ∗ *filepath,* int *argc,* char ∗∗ *argv*)

2D to 3D conversion wrapper function

Wrapper Function invoked when 3D object has to be reconstructed from its projections and displayed

4.1.2.2 int _3d2dDrawWrapper (char ∗ *filepath,* int *argc,* char ∗∗ *argv*)

3D to 2D conversion wrapper function

Wrapper Function invoked when 2D projections have to be made for an object and displayed

4.1.2.3 void _rotateWrapper (float *alpha,* float *beta,* float *gamma,* int *argc,* char ∗∗ *argv*)

Rotation wrapper function.

Wrapper Function invoked when the object is rotated

4.1.2.4 int _SaveWrapper (char ∗ *savepath*)

Rotation wrapper function.

Wrapper Function invoked when the object/projection have to be stored

4.1.2.5 void display_main (int *argc,* char ∗∗ *argv*)

Objection/Projection display function.

Function invoked when object and its projection shave to be displayed

4.1.3 Variable Documentation

4.1.3.1 bool refresh = false

4.1.3.2 int window_1

4.1.3.3 int window_2

4.2 src/cinterface.h File Reference

Provides wrapper functions to interface between GUI (written using GTK+ in C) with the backend (written in C++)

Functions

- int _2d3dDrawWrapper (char ∗filepath, int argc, char ∗∗argv)

  2D to 3D conversion wrapper function
- int _3d2dDrawWrapper (char ∗filepath, int argc, char ∗∗argv)

  3D to 2D conversion wrapper function
- void _rotateWrapper (float alpha, float beta, float gamma, int argc, char ∗∗argv)

  Rotation wrapper function.
- int _SaveWrapper (char ∗savepath)

  Rotation wrapper function.
- void display_main (int argc, char ∗∗argv)

  Objection/Projection display function.

4.2.1 Detailed Description

Provides wrapper functions to interface between GUI (written using GTK+ in C) with the backend (written in C++)

4.2.2 Function Documentation

4.2.2.1 int _2d3dDrawWrapper (char ∗ *filepath,* int *argc,* char ∗∗ *argv*)

2D to 3D conversion wrapper function

Wrapper Function invoked when 3D object has to be reconstructed from its projections and displayed

4.2.2.2 int _3d2dDrawWrapper (char ∗ *filepath,* int *argc,* char ∗∗ *argv*)

3D to 2D conversion wrapper function

Wrapper Function invoked when 2D projections have to be made for an object and displayed

4.2.2.3 void _rotateWrapper (float *alpha,* float *beta,* float *gamma,* int *argc,* char ∗∗ *argv*)

Rotation wrapper function.

Wrapper Function invoked when the object is rotated

4.2.2.4 int _SaveWrapper (char ∗ *savepath*)

Rotation wrapper function.

Wrapper Function invoked when the object/projection have to be stored

4.2.2.5 void display_main (int *argc,* char ∗∗ *argv*)

Objection/Projection display function.

Function invoked when object and its projection shave to be displayed

4.3 src/drawing.cpp File Reference

Implementation of the algorithms.

```
#include <map>
#include <utility>
#include <iostream>
#include <string>
#include <list>
#include <vector>
#include <GL/glew.h>
#include <GL/freeglut.h>
#include <math.h>
#include <limits>
#include "drawing.h"
#include "../lib/cereal/types/unordered_map.hpp"
#include "../lib/cereal/types/map.hpp"
#include "../lib/cereal/types/utility.hpp"
#include "../lib/cereal/types/string.hpp"
#include "../lib/cereal/types/list.hpp"
#include "../lib/cereal/types/vector.hpp"
#include "../lib/cereal/types/memory.hpp"
#include "../lib/cereal/archives/binary.hpp"
#include "../lib/cereal/access.hpp"
```

Functions

- float Abs (float f)
- void initGL ()
- void initGL3D ()
- void reshape (GLsizei width, GLsizei height)
- void reshape3D (GLsizei width, GLsizei height)
- vector< float > cross_prod (float a[3], float b[3])
- void swap (float &a, float &b)
- float _point_on_segment (vertex v1, vertex v2, vertex v)
- bool _point_behind_face (vertex v, face fc)
- void rotate_point (vertex &v, float R[3][3])
- void shift_point (vertex &v, vertex v0)

Variables

- Object3D default_ob
- Projection default_fv
- Projection default_tv
- Projection default_sv

4.3.1 Detailed Description

Implementation of the algorithms.

This file contains the implementation of all the required algorithms for the CAD application.

4.3.2 Function Documentation

4.3.2.1 bool _point_behind_face (vertex *v*, face *fc*)

4.3.2.2 float _point_on_segment (vertex *v1*, vertex *v2*, vertex *v*)

4.3.2.3 float Abs (float *f*) [inline]

4.3.2.4 vector<float> cross_prod (float *a[3]*, float *b[3]*)

4.3.2.5 void initGL ( )

4.3.2.6 void initGL3D ( )

4.3.2.7 void reshape (GLsizei *width*, GLsizei *height*)

4.3.2.8 void reshape3D (GLsizei *width*, GLsizei *height*)

4.3.2.9 void rotate_point (vertex & *v*, float *R[3][3]*)

4.3.2.10 void shift_point (vertex & *v*, vertex *v0*) [inline]

4.3.2.11 void swap (float & *a*, float & *b*) [inline]

4.3.3 Variable Documentation

4.3.3.1 Projection default_fv

4.3.3.2 Object3D default_ob

4.3.3.3 Projection default_sv

4.3.3.4 Projection default_tv

4.4 src/drawing.h File Reference

Contains Projection and Object3D class definitions.

```
#include <map>
#include <utility>
#include <string>
#include <list>
#include <vector>
#include <GL/freeglut.h>
#include "../lib/cereal/types/unordered_map.hpp"
#include "../lib/cereal/types/map.hpp"
#include "../lib/cereal/types/utility.hpp"
#include "../lib/cereal/types/string.hpp"
#include "../lib/cereal/types/list.hpp"
#include "../lib/cereal/types/vector.hpp"
#include "../lib/cereal/types/memory.hpp"
#include "../lib/cereal/archives/binary.hpp"
#include "../lib/cereal/access.hpp"
```

Classes

- struct vertex

  A 3D vertex.
- struct vert2D

  A 2D vertex.
- struct edge

  structure for an edge
- struct edge2D

  2D edge
- struct face

  structure for polygon face of a 3D object
- class Projection

  Class for representing projection.
- class Object3D

  Class for representing 3D objects.

Functions

- void initGL ()
- void reshape (GLsizei width, GLsizei height)
- void initGL3D ()
- void reshape3D (GLsizei width, GLsizei height)
- vector< float > cross_prod (float[3], float[3])

Variables

- Object3D default_ob
- Projection default_fv
- Projection default_tv
- Projection default_sv

4.4.1 Detailed Description

Contains Projection and Object3D class definitions.

This is the interface to the library developed for the CAD application which implements all the required algorithms. Libraries used:

- stl::list
- GLEW: OpenGL Extension Wrangler Library
- GLUT: OpenGL Utility Toolkit
- Cereal: Library to serialize objects and store them in binary format

OpenGL libraries are used to render orthographic views and the 3D object

4.4.2 Function Documentation

4.4.2.1 vector< float > cross_prod (float *[3]*, float *[3]*)

4.4.2.2 void initGL ( )

4.4.2.3 void initGL3D ( )

4.4.2.4 void reshape (GLsizei *width*, GLsizei *height*)

4.4.2.5 void reshape3D (GLsizei *width*, GLsizei *height*)

4.4.3 Variable Documentation

4.4.3.1 Projection default_fv

4.4.3.2 Object3D default_ob

4.4.3.3 Projection default_sv

4.4.3.4 Projection default_tv

4.5 src/GUI.c File Reference

Implements to UI fro the application.

```
#include <gtk/gtk.h>
#include "cinterface.h"
#include <stdlib.h>
```

Functions

- int main (int argc, char ∗∗argv)
- void on_btn_2d3d_clicked ()
- void on_btn_3d2d_clicked ()
- void on_btn_rotate_clicked ()
- void on_btn_filePicker_file_set ()
- void on_window_main_destroy ()
- void on_btn_save_clicked ()

Variables

- GtkWidget ∗ g_lbl_title
- GtkButton ∗ g_btn_2d3d
- GtkButton ∗ g_btn_3d2d
- GtkButton ∗ g_btn_rotate
- GtkButton ∗ g_btn_save
- GtkEntry ∗ g_txt_filePath
- GtkEntry ∗ g_txt_xAxis
- GtkEntry ∗ g_txt_yAxis
- GtkEntry ∗ g_txt_zAxis
- GtkEntry ∗ g_txt_savePath
- GtkFileChooserButton ∗ g_btn_filePicker
- GtkStatusbar ∗ g_bar_status
- long int pID = 0
- char ∗ filepath = NULL
- char ∗ savepath = NULL
- int arc = 0
- char ∗∗ arv =NULL

4.5.1 Detailed Description

Implements to UI fro the application.

Constructs the graphical UI of the applications, handles errors and calls the correct callback depending on the user interaction

4.5.2 Function Documentation

4.5.2.1 int main (int *argc,* char ∗∗ *argv*)

4.5.2.2 void on_btn_2d3d_clicked ( )

4.5.2.3 void on_btn_3d2d_clicked ( )

4.5.2.4 void on_btn_filePicker_file_set ( )

4.5.2.5 void on_btn_rotate_clicked ( )

4.5.2.6 void on_btn_save_clicked ( )

4.5.2.7 void on_window_main_destroy ( )

4.5.3 Variable Documentation

4.5.3.1 int arc = 0

4.5.3.2 char∗∗ arv =NULL

4.5.3.3 char∗ filepath = NULL

4.5.3.4 GtkStatusbar∗ g_bar_status

4.5.3.5 GtkButton∗ g_btn_2d3d

4.5.3.6 GtkButton∗ g_btn_3d2d

4.5.3.7 GtkFileChooserButton∗ g_btn_filePicker

4.5.3.8 GtkButton∗ g_btn_rotate

4.5.3.9 GtkButton∗ g_btn_save

4.5.3.10 GtkWidget∗ g_lbl_title

4.5.3.11 GtkEntry∗ g_txt_filePath

4.5.3.12 GtkEntry∗ g_txt_savePath

4.5.3.13 GtkEntry∗ g_txt_xAxis

4.5.3.14 GtkEntry∗ g_txt_yAxis

4.5.3.15 GtkEntry∗ g_txt_zAxis

4.5.3.16 long int pID = 0

4.5.3.17 char∗ savepath = NULL

Index