

CADapp

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	edge Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Constructor & Destructor Documentation	6
3.1.2.1	edge()	6
3.1.2.2	edge(const edge &)=default	6
3.1.2.3	edge(vertex a, vertex b, bool v=true)	6
3.1.3	Member Function Documentation	6
3.1.3.1	serialize(Archive &ar)	6
3.1.4	Member Data Documentation	6
3.1.4.1	v1	6
3.1.4.2	v2	6
3.1.4.3	visi	6
3.2	edge2D Struct Reference	6
3.2.1	Detailed Description	7
3.2.2	Constructor & Destructor Documentation	7
3.2.2.1	edge2D()	7
3.2.2.2	edge2D(const edge2D &)=default	7

3.2.2.3	edge2D(vert2D a, vert2D b, bool v=true)	7
3.2.2.4	edge2D(const edge e)	7
3.2.3	Member Function Documentation	7
3.2.3.1	serialize(Archive &ar)	7
3.2.4	Member Data Documentation	7
3.2.4.1	v1	7
3.2.4.2	v2	7
3.2.4.3	visi	7
3.3	face Struct Reference	7
3.3.1	Detailed Description	8
3.3.2	Constructor & Destructor Documentation	8
3.3.2.1	face()	8
3.3.2.2	face(const face &)=default	8
3.3.3	Member Function Documentation	8
3.3.3.1	compParam()	8
3.3.3.2	serialize(Archive &ar)	8
3.3.4	Member Data Documentation	8
3.3.4.1	A	8
3.3.4.2	B	8
3.3.4.3	C	8
3.3.4.4	D	8
3.3.4.5	edges	8
3.3.4.6	verts	9
3.4	Object3D Class Reference	9
3.4.1	Detailed Description	10
3.4.2	Constructor & Destructor Documentation	10
3.4.2.1	Object3D()	10
3.4.3	Member Function Documentation	10
3.4.3.1	_dashedLines(map< string, edge > &els, map< string, face > &fls)	10
3.4.3.2	_intersect_ratiois(edge e1, edge e2)	10

3.4.3.3	_intersectingEdges(map< string, edge > &els, map< string, vertex > &vls)	10
3.4.3.4	_overlappingEdges(map< string, edge > &els, map< string, vertex > &vls)	10
3.4.3.5	_wireframe(Projection FV, Projection TV, Projection SV, bool rightside=true, bool righthand=true)	10
3.4.3.6	create(Projection FV, Projection TV, Projection SV, bool rightside=true, bool righthand=true)	11
3.4.3.7	display()	11
3.4.3.8	display_wireframe()	11
3.4.3.9	projectTo2D(string view)	11
3.4.3.10	rotate(float alpha, float beta, float gamma)	12
3.4.3.11	serialize(Archive &ar)	12
3.4.3.12	shift(float x0, float y0, float z0)	12
3.4.4	Member Data Documentation	12
3.4.4.1	elist	12
3.4.4.2	flist	12
3.4.4.3	vlist	12
3.5	Projection Class Reference	13
3.5.1	Detailed Description	13
3.5.2	Constructor & Destructor Documentation	13
3.5.2.1	Projection()	13
3.5.3	Member Function Documentation	13
3.5.3.1	display()	13
3.5.3.2	getProjection()	13
3.5.3.3	serialize(Archive &ar)	14
3.5.4	Member Data Documentation	14
3.5.4.1	elist	14
3.5.4.2	name	14
3.5.4.3	vlist	14
3.6	vert2D Struct Reference	14
3.6.1	Detailed Description	14
3.6.2	Constructor & Destructor Documentation	15

3.6.2.1	vert2D()	15
3.6.2.2	vert2D(float _x, float _y)	15
3.6.2.3	vert2D(const vertex &v)	15
3.6.2.4	vert2D(const vert2D &v)	15
3.6.3	Member Function Documentation	15
3.6.3.1	serialize(Archive &ar)	15
3.6.4	Member Data Documentation	15
3.6.4.1	x	15
3.6.4.2	y	15
3.7	vertex Struct Reference	15
3.7.1	Detailed Description	16
3.7.2	Constructor & Destructor Documentation	16
3.7.2.1	vertex()	16
3.7.2.2	vertex(float _x, float _y, float _z)	16
3.7.2.3	vertex(const vertex &)=default	16
3.7.3	Member Function Documentation	16
3.7.3.1	operator*(const float &f)	16
3.7.3.2	operator+(const vertex &v)	16
3.7.3.3	operator-(const vertex &v)	16
3.7.3.4	operator==(const vertex &v)	16
3.7.3.5	serialize(Archive &ar)	16
3.7.4	Member Data Documentation	16
3.7.4.1	x	16
3.7.4.2	y	16
3.7.4.3	z	16

4 File Documentation	17
4.1 src/cinterface.cpp File Reference	17
4.1.1 Detailed Description	18
4.1.2 Function Documentation	18
4.1.2.1 _2d3dDrawWrapper(char *filepath, int argc, char **argv)	18
4.1.2.2 _3d2dDrawWrapper(char *filepath, int argc, char **argv)	18
4.1.2.3 _rotateWrapper(float alpha, float beta, float gamma, int argc, char **argv)	18
4.1.2.4 _SaveWrapper(char *savepath)	18
4.1.2.5 display_main(int argc, char **argv)	18
4.1.3 Variable Documentation	18
4.1.3.1 refresh	18
4.1.3.2 window_1	18
4.1.3.3 window_2	18
4.2 src/cinterface.h File Reference	18
4.2.1 Detailed Description	19
4.2.2 Function Documentation	19
4.2.2.1 _2d3dDrawWrapper(char *filepath, int argc, char **argv)	19
4.2.2.2 _3d2dDrawWrapper(char *filepath, int argc, char **argv)	19
4.2.2.3 _rotateWrapper(float alpha, float beta, float gamma, int argc, char **argv)	19
4.2.2.4 _SaveWrapper(char *savepath)	19
4.2.2.5 display_main(int argc, char **argv)	20
4.3 src/drawing.cpp File Reference	20
4.3.1 Detailed Description	21
4.3.2 Function Documentation	21
4.3.2.1 _point_behind_face(vertex v, face fc)	21
4.3.2.2 _point_on_segment(vertex v1, vertex v2, vertex v)	21
4.3.2.3 Abs(float f)	21
4.3.2.4 cross_prod(float a[3], float b[3])	21
4.3.2.5 initGL()	21
4.3.2.6 initGL3D()	21

4.3.2.7	reshape(GLsizei width, GLsizei height)	21
4.3.2.8	reshape3D(GLsizei width, GLsizei height)	21
4.3.2.9	rotate_point(vertex &v, float R[3][3])	21
4.3.2.10	shift_point(vertex &v, vertex v0)	21
4.3.2.11	swap(float &a, float &b)	21
4.3.3	Variable Documentation	21
4.3.3.1	default_fv	21
4.3.3.2	default_ob	21
4.3.3.3	default_sv	21
4.3.3.4	default_tv	21
4.4	src/drawing.h File Reference	21
4.4.1	Detailed Description	23
4.4.2	Function Documentation	23
4.4.2.1	cross_prod(float[3], float[3])	23
4.4.2.2	initGL()	23
4.4.2.3	initGL3D()	23
4.4.2.4	reshape(GLsizei width, GLsizei height)	23
4.4.2.5	reshape3D(GLsizei width, GLsizei height)	23
4.4.3	Variable Documentation	23
4.4.3.1	default_fv	23
4.4.3.2	default_ob	23
4.4.3.3	default_sv	23
4.4.3.4	default_tv	23
4.5	src/GUI.c File Reference	23
4.5.1	Detailed Description	24
4.5.2	Function Documentation	24
4.5.2.1	main(int argc, char **argv)	24
4.5.2.2	on_btn_2d3d_clicked()	24
4.5.2.3	on_btn_3d2d_clicked()	24
4.5.2.4	on_btn_filePicker_file_set()	24

4.5.2.5	on_btn_rotate_clicked()	24
4.5.2.6	on_btn_save_clicked()	25
4.5.2.7	on_window_main_destroy()	25
4.5.3	Variable Documentation	25
4.5.3.1	arc	25
4.5.3.2	arv	25
4.5.3.3	filepath	25
4.5.3.4	g_bar_status	25
4.5.3.5	g_btn_2d3d	25
4.5.3.6	g_btn_3d2d	25
4.5.3.7	g_btn_filePicker	25
4.5.3.8	g_btn_rotate	25
4.5.3.9	g_btn_save	25
4.5.3.10	g_lbl_title	25
4.5.3.11	g_txt_filePath	25
4.5.3.12	g_txt_savePath	25
4.5.3.13	g_txt_xAxis	25
4.5.3.14	g_txt_yAxis	25
4.5.3.15	g_txt_zAxis	25
4.5.3.16	pID	25
4.5.3.17	savepath	25
Index		27

Chapter 1

Class Index

1.1 Class List

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

edge	Structure for an edge	5
edge2D	2D edge	6
face	Structure for polygon face of a 3D object	7
Object3D	Class for representing 3D objects	9
Projection	Class for representing projection	13
vert2D	A 2D vertex	14
vertex	A 3D vertex	15

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/ cinterface.cpp	Implemenation of all wrapper functions	17
src/ cinterface.h	Provides wrapper functions to interface between GUI (written using GTK+ in C) with the backend (written in C++)	18
src/ drawing.cpp	Implementation of the algorithms	20
src/ drawing.h	Contains Projection and Object3D class definitions	21
src/ GUI.c	Implements to UI fro the application	23

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

<i>view</i>	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_ratios (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

<i>view</i>	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

<i>view</i>	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

<i>FV</i>	denotes the input front orthographic projection
<i>TV</i>	denotes the input top orthographic projection
<i>SV</i>	denoted the input side orthographic projection
<i>rightside</i>	boolean value indicating wether right side view is taken, default value is true
<i>righthand</i>	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

<i>FV</i>	the front Orthographic projection
<i>TV</i>	the top Orthographic projection
<i>SV</i>	the side view Orthographic projection
<i>rightside</i>	boolean value telling which side view is taken, default true
<i>righthand</i>	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

<i>view</i>	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

<i>alpha</i>	Angular displacement about the x axis
<i>beta</i>	Angular displacement about the y axis
<i>gamma</i>	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

<i>x0</i>	Offset in x direction
<i>y0</i>	Offset in y direction
<i>z0</i>	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

<i>view</i>	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

Implementation 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

Implementation of all wrapper functions.

This file contains the implementation of all the required wrapper functions 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 have 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](#)
- GtkWidget * [g_btn_2d3d](#)
- GtkWidget * [g_btn_3d2d](#)
- GtkWidget * [g_btn_rotate](#)
- GtkWidget * [g_btn_save](#)
- GtkWidget * [g_txt_filePath](#)
- GtkWidget * [g_txt_xAxis](#)
- GtkWidget * [g_txt_yAxis](#)
- GtkWidget * [g_txt_zAxis](#)
- GtkWidget * [g_txt_savePath](#)
- GtkWidget * [g_btn_filePicker](#)
- GtkWidget * [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 GtkWidget* g_bar_status

4.5.3.5 GtkWidget* g_btn_2d3d

4.5.3.6 GtkWidget* g_btn_3d2d

4.5.3.7 GtkWidget* g_btn_filePicker

4.5.3.8 GtkWidget* g_btn_rotate

4.5.3.9 GtkWidget* g_btn_save

4.5.3.10 GtkWidget* g_lbl_title

4.5.3.11 GtkWidget* g_txt_filePath

4.5.3.12 GtkWidget* g_txt_savePath

4.5.3.13 GtkWidget* g_txt_xAxis

4.5.3.14 GtkWidget* g_txt_yAxis

4.5.3.15 GtkWidget* g_txt_zAxis

4.5.3.16 long int pID = 0

4.5.3.17 char* savepath = NULL

Index

- _2d3dDrawWrapper
 - cinterface.cpp, [18](#)
 - cinterface.h, [19](#)
 - _3d2dDrawWrapper
 - cinterface.cpp, [18](#)
 - cinterface.h, [19](#)
 - _SaveWrapper
 - cinterface.cpp, [18](#)
 - cinterface.h, [19](#)
 - _dashedLines
 - Object3D, [10](#)
 - _intersect_ratios
 - Object3D, [10](#)
 - _intersectingEdges
 - Object3D, [10](#)
 - _overlappingEdges
 - Object3D, [10](#)
 - _point_behind_face
 - drawing.cpp, [21](#)
 - _point_on_segment
 - drawing.cpp, [21](#)
 - _rotateWrapper
 - cinterface.cpp, [18](#)
 - cinterface.h, [19](#)
 - _wireframe
 - Object3D, [10](#)
- A
 - face, [8](#)
- Abs
 - drawing.cpp, [21](#)
- arc
 - GUI.c, [25](#)
- arv
 - GUI.c, [25](#)
- B
 - face, [8](#)
- C
 - face, [8](#)
- cinterface.cpp
 - _2d3dDrawWrapper, [18](#)
 - _3d2dDrawWrapper, [18](#)
 - _SaveWrapper, [18](#)
 - _rotateWrapper, [18](#)
 - display_main, [18](#)
 - refresh, [18](#)
 - window_1, [18](#)
 - window_2, [18](#)
- cinterface.h
 - _2d3dDrawWrapper, [19](#)
 - _3d2dDrawWrapper, [19](#)
 - _SaveWrapper, [19](#)
 - _rotateWrapper, [19](#)
 - display_main, [19](#)
- compParam
 - face, [8](#)
- create
 - Object3D, [11](#)
- cross_prod
 - drawing.cpp, [21](#)
 - drawing.h, [23](#)
- D
 - face, [8](#)
- default_fv
 - drawing.cpp, [21](#)
 - drawing.h, [23](#)
- default_ob
 - drawing.cpp, [21](#)
 - drawing.h, [23](#)
- default_sv
 - drawing.cpp, [21](#)
 - drawing.h, [23](#)
- default_tv
 - drawing.cpp, [21](#)
 - drawing.h, [23](#)
- display
 - Object3D, [11](#)
 - Projection, [13](#)
- display_main
 - cinterface.cpp, [18](#)
 - cinterface.h, [19](#)
- display_wireframe
 - Object3D, [11](#)
- drawing.cpp
 - _point_behind_face, [21](#)
 - _point_on_segment, [21](#)
 - Abs, [21](#)
 - cross_prod, [21](#)
 - default_fv, [21](#)
 - default_ob, [21](#)
 - default_sv, [21](#)
 - default_tv, [21](#)
 - initGL3D, [21](#)
 - initGL, [21](#)
 - reshape, [21](#)
 - reshape3D, [21](#)
 - rotate_point, [21](#)

- shift_point, 21
 - swap, 21
- drawing.h
 - cross_prod, 23
 - default_fv, 23
 - default_ob, 23
 - default_sv, 23
 - default_tv, 23
 - initGL3D, 23
 - initGL, 23
 - reshape, 23
 - reshape3D, 23
- edge, 5
 - edge, 6
 - serialize, 6
 - v1, 6
 - v2, 6
 - visi, 6
- edge2D, 6
 - edge2D, 7
 - serialize, 7
 - v1, 7
 - v2, 7
 - visi, 7
- edges
 - face, 8
- elist
 - Object3D, 12
 - Projection, 14
- face, 7
 - A, 8
 - B, 8
 - C, 8
 - compParam, 8
 - D, 8
 - edges, 8
 - face, 8
 - serialize, 8
 - verts, 8
- filepath
 - GUI.c, 25
- flist
 - Object3D, 12
- g_bar_status
 - GUI.c, 25
- g_btn_2d3d
 - GUI.c, 25
- g_btn_3d2d
 - GUI.c, 25
- g_btn_filePicker
 - GUI.c, 25
- g_btn_rotate
 - GUI.c, 25
- g_btn_save
 - GUI.c, 25
- g_lbl_title
 - GUI.c, 25
- g_txt_filePath
 - GUI.c, 25
- g_txt_savePath
 - GUI.c, 25
- g_txt_xAxis
 - GUI.c, 25
- g_txt_yAxis
 - GUI.c, 25
- g_txt_zAxis
 - GUI.c, 25
- GUI.c
 - arc, 25
 - arv, 25
 - filepath, 25
 - g_bar_status, 25
 - g_btn_2d3d, 25
 - g_btn_3d2d, 25
 - g_btn_filePicker, 25
 - g_btn_rotate, 25
 - g_btn_save, 25
 - g_lbl_title, 25
 - g_txt_filePath, 25
 - g_txt_savePath, 25
 - g_txt_xAxis, 25
 - g_txt_yAxis, 25
 - g_txt_zAxis, 25
 - main, 24
 - on_btn_2d3d_clicked, 24
 - on_btn_3d2d_clicked, 24
 - on_btn_filePicker_file_set, 24
 - on_btn_rotate_clicked, 24
 - on_btn_save_clicked, 24
 - on_window_main_destroy, 25
 - pID, 25
 - savepath, 25
- getProjection
 - Projection, 13
- initGL3D
 - drawing.cpp, 21
 - drawing.h, 23
- initGL
 - drawing.cpp, 21
 - drawing.h, 23
- main
 - GUI.c, 24
- name
 - Projection, 14
- Object3D, 9
 - _dashedLines, 10
 - _intersect_ratiois, 10
 - _intersectingEdges, 10
 - _overlappingEdges, 10
 - _wireframe, 10
 - create, 11

- display, [11](#)
- display_wireframe, [11](#)
- elist, [12](#)
- flist, [12](#)
- Object3D, [10](#)
- projectTo2D, [11](#)
- rotate, [12](#)
- serialize, [12](#)
- shift, [12](#)
- vlist, [12](#)
- on_btn_2d3d_clicked
 - GUI.c, [24](#)
- on_btn_3d2d_clicked
 - GUI.c, [24](#)
- on_btn_filePicker_file_set
 - GUI.c, [24](#)
- on_btn_rotate_clicked
 - GUI.c, [24](#)
- on_btn_save_clicked
 - GUI.c, [24](#)
- on_window_main_destroy
 - GUI.c, [25](#)
- operator*
 - vertex, [16](#)
- operator+
 - vertex, [16](#)
- operator-
 - vertex, [16](#)
- operator==
 - vertex, [16](#)
- pID
 - GUI.c, [25](#)
- projectTo2D
 - Object3D, [11](#)
- Projection, [13](#)
 - display, [13](#)
 - elist, [14](#)
 - getProjection, [13](#)
 - name, [14](#)
 - Projection, [13](#)
 - serialize, [13](#)
 - vlist, [14](#)
- refresh
 - cinterface.cpp, [18](#)
- reshape
 - drawing.cpp, [21](#)
 - drawing.h, [23](#)
- reshape3D
 - drawing.cpp, [21](#)
 - drawing.h, [23](#)
- rotate
 - Object3D, [12](#)
- rotate_point
 - drawing.cpp, [21](#)
- savepath
 - GUI.c, [25](#)
- serialize
 - edge, [6](#)
 - edge2D, [7](#)
 - face, [8](#)
 - Object3D, [12](#)
 - Projection, [13](#)
 - vert2D, [15](#)
 - vertex, [16](#)
- shift
 - Object3D, [12](#)
- shift_point
 - drawing.cpp, [21](#)
- src/GUI.c, [23](#)
- src/cinterface.cpp, [17](#)
- src/cinterface.h, [18](#)
- src/drawing.cpp, [20](#)
- src/drawing.h, [21](#)
- swap
 - drawing.cpp, [21](#)
- v1
 - edge, [6](#)
 - edge2D, [7](#)
- v2
 - edge, [6](#)
 - edge2D, [7](#)
- vert2D, [14](#)
 - serialize, [15](#)
 - vert2D, [15](#)
 - x, [15](#)
 - y, [15](#)
- vertex, [15](#)
 - operator*, [16](#)
 - operator+, [16](#)
 - operator-, [16](#)
 - operator==, [16](#)
 - serialize, [16](#)
 - vertex, [16](#)
 - x, [16](#)
 - y, [16](#)
 - z, [16](#)
- verts
 - face, [8](#)
- visi
 - edge, [6](#)
 - edge2D, [7](#)
- vlist
 - Object3D, [12](#)
 - Projection, [14](#)
- window_1
 - cinterface.cpp, [18](#)
- window_2
 - cinterface.cpp, [18](#)
- x
 - vert2D, [15](#)
 - vertex, [16](#)

y

vert2D, [15](#)
vertex, [16](#)

z

vertex, [16](#)