My Project

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

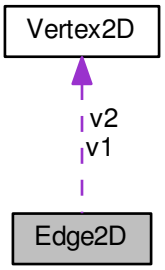
Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Edge2D Class Reference

Collaboration diagram for Edge2D:



Public Attributes

- Vertex2D **v1**
- Vertex2D **v2**

3.1.1 Detailed Description

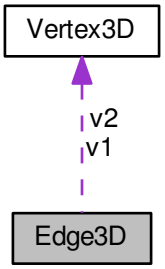
This class defines an edge of a 2D object which contains two 2D vertices of the edge.

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

- cad_v1_0.cpp
- generate2D_v2.0.cpp

3.2 Edge3D Class Reference

Collaboration diagram for Edge3D:



Public Attributes

- Vertex3D **v1**
- Vertex3D **v2**

3.2.1 Detailed Description

This class defines an edge of a 3D object which contains two 3D vertices of the edge.

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

- cad_v1_0.cpp
- generate2D_v2.0.cpp

3.3 Model2D Class Reference

Public Member Functions

- int **edge1Length** ()
- int **vertex1Length** ()
- int **edge2Length** ()
- int **vertex2Length** ()
- int **edge3Length** ()
- int **vertex3Length** ()
- int **edge1Length** ()
- int **vertex1Length** ()
- int **edge2Length** ()
- int **vertex2Length** ()
- int **edge3Length** ()
- int **vertex3Length** ()

Public Attributes

- vector< Vertex2D > vertexVector1
  vector of vertices in the model
- vector< Vertex2D > **vertexVector2**
- vector< Vertex2D > **vertexVector3**
- vector< Edge2D > edgesVector1
  vector of edges in the model
- vector< Edge2D > **edgesVector2**
- vector< Edge2D > **edgesVector3**

3.3.1 Detailed Description

This class defines a model of a 2D object.

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

- cad_v1_0.cpp
- generate2D_v2.0.cpp

3.4 Model3D Class Reference

Public Member Functions

- int addEdge (Vertex3D v1, Vertex3D v2, int length)
  Method to add edge to the model.
- int **edgeLength** ()
- int **vertexLength** ()
- int addEdge (Vertex3D v1, Vertex3D v2, int length)
  Method to add edge to the model.
- int **edgeLength** ()
- int **vertexLength** ()

Public Attributes

- vector< Vertex3D > vertexVector
  vector of vertices in the model
- vector< Edge3D > edgesVector
  vector of edges in the model

3.4.1 Detailed Description

This class defines a model of a 3D object.

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

- cad_v1_0.cpp
- generate2D_v2.0.cpp

3.5 Vertex2D Class Reference

Public Member Functions

- void vertex2DAddNeighbour (Vertex2D v)

  Method to add neighbouring vertices to the vector.
- void vertex2DAddNeighbour (Vertex2D v)

  Method to add neighbouring vertices to the vector.

Public Attributes

- float **x**
- float **y**
- string name

  Name of vertex.
- vector< Vertex2D > vertex2DEdgeVector

  vector of names of vertices which have edge to particular vertex
- double **x**
- double **y**

3.5.1 Detailed Description

This class defines a vertex of a 2D object which contains two coordinates and the vector of edges from that vertex.

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

- cad_v1_0.cpp
- generate2D_v2.0.cpp

3.6 Vertex3D Class Reference

Public Member Functions

- void vertex3DAddNeighbour (Vertex3D v)

  Method to add neighbouring vertices to vertex.
- void vertex3DAddNeighbour (Vertex3D v)

  Method to add neighbouring vertices to vertex.

Public Attributes

- float x

  3 coordinates of vertex
- float **y**
- float **z**
- string name

  Name of vertex.
- vector< Vertex3D > vertex3DEdgeVector

  vector of edges
- double x

  3 coordinates of vertex
- double **y**
- double **z**

3.6.1 Detailed Description

This class defines a vertex of a 3D object which contains three coordinates and the vector of edges from that vertex.

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

- cad_v1_0.cpp
- generate2D_v2.0.cpp

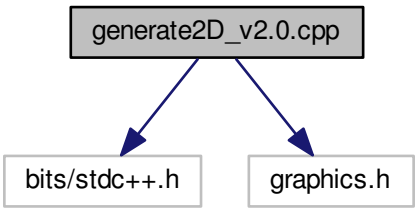
Chapter 4

File Documentation

4.1 generate2D_v2.0.cpp File Reference

```
#include <bits/stdc++.h>
#include <graphics.h>
```

Include dependency graph for generate2D_v2.0.cpp:



Classes

- class Vertex2D
- class Edge2D
- class Model2D
- class Vertex3D
- class Edge3D
- class Model3D

Macros

- #define **least** 0.0001

Functions

- void getRotaionMatrixz (float x, float y, float z, double a[ ][3])
- void **getRotaionMatrixx** (float x, float y, float z, double a[ ][3])
- void **getRotaionMatrixy** (float x, float y, float z, double a[ ][3])
- bool **sameValue** (int x, int y)
- Vertex2D getProjection (Vertex3D v, double rotationMatrix[ ][3], int count)
- Model2D rotatedModel3D (Model3D model, Model2D outputModel, double rotationMatrix[ ][3], int count)
- void graphicalOutput (Model2D outputModel)
- int **findIndex** (Model2D outputModel, string n)
- bool **sameVertex** (Vertex2D v1, Vertex2D v2)
- Model2D positive (Model2D outputModel)
- Model2D projectedModel (Model3D model, Model2D outputModel)
- void **graphicsOutput** (Model2D outputModel)
- void generate2D (Model3D model, float x, float y, float z)
- vector< string > tokenizeString (string str)
- Model3D createModel3D (ifstream &v1, ifstream &v2, ifstream &e1, ifstream &e2)
- void getVertex (Model3D model, ofstream &vertex)
- void getEdges (Model3D model, ofstream &edge)
- void generate3D (ifstream &v1, ifstream &v2, ifstream &e1, ifstream &e2, float x, float y, float z)
- Model3D model3DFromInputFiles (ifstream &v, ifstream &e)
- void interfaceFor3Dto2D ()
- void interfaceFor2Dto3D ()
- void interfaceforCAD (int num)
- int main ()

4.1.1 Function Documentation

4.1.1.1 Model3D createModel3D (ifstream & *v1,* ifstream & *v2,* ifstream & *e1,* ifstream & *e2*)

Create a 3D Model from input .txt files of vertices and edges for 2D file.

4.1.1.2 void generate2D (Model3D *model,* float *x,* float *y,* float *z*)

Function to generate the 2D representation of the vector of edges and vertices parsed as a file along with viewing direction coordinates. Obtains the rotation matrix from the viewing direction coordinates

Generate the rotated 3D model

Obtains the rotation matrix from the viewing direction coordinates

Generate the rotated 3D model

Obtains the rotation matrix from the viewing direction coordinates

Generate the rotated 3D model

Creates a 2D model from rotated 3D Model

Make positive

Final graphical output in form of files

4.1.1.3 void generate3D (ifstream & *v1,* ifstream & *v2,* ifstream & *e1,* ifstream & *e2,* float *x,* float *y,* float *z*)

Function to generate 3D model from given vertex and edges vector of both faces along with viewing direction. Function to create the 3D model

Reduce the problem to projecting from 3D to 2D views

4.1.1.4 void getEdges (**Model3D** *model,* ofstream & *edge*)

Function to get edges from a 3D model in a file.

4.1.1.5 **Vertex2D** getProjection (**Vertex3D** *v,* double *rotationMatrix[ ][3],* int *count*)

Function to get the projections of each vertex of the model when rotated using the specified rotation matrix.

4.1.1.6 void getRotaionMatrixz (float *x,* float *y,* float *z,* double *a[ ][3]*)

Function to obtain the rotation matrix using the viewing direction obtained from the user.

4.1.1.7 void getVertex (**Model3D** *model,* ofstream & *vertex*)

Function to get vertices from a 3D model in a file.

4.1.1.8 void graphicalOutput (**Model2D** *outputModel*)

Function to obtain a graphical interface to view the model after applying all transformations.

4.1.1.9 void interfaceFor2Dto3D ( )

Function for interface in case of 2D to 3D contains buttons to upload vertex and edge files. Generate button begins processing to display the model.

4.1.1.10 void interfaceFor3Dto2D ( )

Function for interface in case of 3D to 2D contains buttons to upload vertex and edge files. Generate button begins processing to display the model.

4.1.1.11 void interfaceforCAD (int *num*)

Function for initial interface showing 2 buttons for 3D to 2D and 2D to 3D conversion.

4.1.1.12 int main ( )

Main function initializing buttons for 2D to 3D and 3D to 2D.

4.1.1.13 Model3D model3DFromInputFiles (ifstream & *v,* ifstream & *e*)

Create a 3D Model from input set of files

4.1.1.14 Model2D positive (Model2D *outputModel*)

Make all the vertices positive by shifting

4.1.1.15 Model2D projectedModel (Model3D *model,* Model2D *outputModel*)

Function to get 2D model from the 3D Model after removing one of the corrdinates in each of the view.

4.1.1.16 Model2D rotatedModel3D (Model3D *model,* Model2D *outputModel,* double *rotationMatrix[ ][3],* int *count*)

Function to output the complete transformed model

4.1.1.17 vector<string> tokenizeString (string *str*)

Function to create a 3D model using the vertices and edges vector of both the 2D faces and store it in the model object.

Index