3dplot Reference Manual

0.1

Generated by Doxygen 1.3.5

Mon May 3 15:20:15 2004

Contents

Chapter 1

3dplot Data Structure Index

1.1 3dplot Data Structures

Here are the data structures with brief descriptions:

Chapter 2

3dplot File Index

2.1 3dplot File List

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

Chapter 3

3dplot Data Structure Documentation

3.1 config Struct Reference

`#include <config.h>`

Data Fields

- int **WinHeight**
- int **WinWidth**
- int **WinX**
- int **WinY**
- float **Camera** [3]
- float **Centre** [3]
- float **UpVector** [3]

3.1.1 Detailed Description

Structure to hold all config variables

The following information is held in this structure:

- Window Information
  1. Window Dimentions
  2. Window Position
- Camera Information Used by gluLookAt()

Definition at line 31 of file config.h.

3.1.2 Field Documentation

3.1.2.1 float config::Camera[3]

X,Y,Z Coordinates of Camera

Definition at line 37 of file config.h.

3.1.2.2 float config::Centre[3]

X,Y,Z Coordinates of Centre Focus Point of Camera

Definition at line 38 of file config.h.

3.1.2.3 float config::UpVector[3]

X,Y,Z Values for Up Vector of the Camera

Definition at line 39 of file config.h.

3.1.2.4 int config::WinHeight

Window Height

Definition at line 32 of file config.h.

3.1.2.5 int config::WinWidth

Window Width

Definition at line 33 of file config.h.

3.1.2.6 int config::WinX

Window X Position

Definition at line 34 of file config.h.

3.1.2.7 int config::WinY

Window Y Position

Definition at line 35 of file config.h.

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

- **config.h**

3.2 dataset Struct Reference

`#include <main.h>`

Data Fields

- long **X**
- long **Y**
- double ∗ **x**
- double ∗ **y**
- double ∗ **z**

3.2.1 Detailed Description

Structure to hold the current data being worked on

Definition at line 15 of file main.h.

3.2.2 Field Documentation

3.2.2.1 double∗ dataset::x

x values

Definition at line 19 of file main.h.

3.2.2.2 long dataset::X

Number of x values in dataset

Definition at line 17 of file main.h.

3.2.2.3 double∗ dataset::y

y values

Definition at line 20 of file main.h.

3.2.2.4 long dataset::Y

Number of y values in dataset

Definition at line 18 of file main.h.

3.2.2.5 double∗ dataset::z

z values

Definition at line 21 of file main.h.

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

- **main.h**

3.3 frame Struct Reference

`#include <movie.h>`

Data Fields

- int **num**
- char(∗ **framepath**)[BUFSIZE]

3.3.1 Detailed Description

Holds frame locations

Definition at line 15 of file movie.h.

3.3.2 Field Documentation

3.3.2.1 char(∗ frame::framepath)[BUFSIZE]

Holds full path to frame data

Definition at line 17 of file movie.h.

3.3.2.2 int frame::num

Number of frames

Definition at line 16 of file movie.h.

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

- **movie.h**

3.4 light Struct Reference

`#include <lights.h>`

Data Fields

- GLenum **lightid**
- float ∗ **position**
- float ∗ **diffuse**
- float ∗ **specular**

3.4.1 Detailed Description

Holds Light infomation

Definition at line 18 of file lights.h.

3.4.2 Field Documentation

3.4.2.1 float∗ light::diffuse

Diffusion of light used by GL_DIFFUSE

Definition at line 21 of file lights.h.

3.4.2.2 GLenum light::lightid

Set to LIGHT[0|1|...] !FIXME: needs to be implemented

Definition at line 19 of file lights.h.

3.4.2.3 float∗ light::position

position of light

Definition at line 20 of file lights.h.

3.4.2.4 float∗ light::specular

Specular of light used by GL_SPECULAR

Definition at line 22 of file lights.h.

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

- **lights.h**

Chapter 4

3dplot File Documentation

4.1 config.h File Reference

Routines for reading and writing the config files.

`#include "lights.h"`

Data Structures

- struct **config**

Functions

- int **readConfigFile** (FILE ∗fh)

4.1.1 Detailed Description

Routines for reading and writing the config files.

Config files are used as the main input to 3dplot. They contain all the variables to recreate the plot, viewport, lights etc In general the extension .3dp should be used. Format of config file is as follows:

- One line per variable set
- Comments defined using # and continue to EOL
- Variables are set by <variable name> = <setting>
- Free formating in terms of whitespaces
- Invalid lines are ignored

Definition in file **config.h**.

4.1.2 Function Documentation

4.1.2.1 int readConfigFile (FILE ∗ *fh*)

Reads configuration file

Parameters:

fh file pointer to config file

Returns:

0 if file read ok, -1 otherwise

Definition at line 15 of file config.c.

4.2 control.h File Reference

Routines for processing key presses.

Functions

- void **updateSphericalCoords** (void)
- void **updateCartesianCoords** (void)
- void **updateCamera** (void)
- void **processNormalKeys** (unsigned char key, int x, int y)
- void **processSpecialKeys** (int key, int x, int y)

4.2.1 Detailed Description

Routines for processing key presses.

Handles all key presses as well as any camera movement related functions. Also defined are the funtions used by OpenGL as the key handling functions.

Definition in file **control.h**.

4.2.2 Function Documentation

4.2.2.1 void processNormalKeys (unsigned char *key*, int *x*, int *y*)

Function used by glutKeyboardFunc()

Handles all alphanumeric key presses and mouse movement

Parameters:

key Key pressed

x Mouse X coordinate

y Mouse Y coordinate

Definition at line 73 of file control.c.

4.2.2.2 void processSpecialKeys (int *key*, int *x*, int *y*)

Function used by glutSpecialFunc()

Handles non-alphanumeric key presses and mouse movement

Parameters:

key Key pressed

x Mouse X coordinate

y Mouse Y coordinate

Definition at line 98 of file control.c.

4.2.2.3 void updateCamera (void)

Moves the Camera to its new position

Definition at line 62 of file control.c.

4.2.2.4 void updateCartesianCoords (void)

Updates Cartesian Corrdinates of the Camera

Definition at line 52 of file control.c.

4.2.2.5 void updateSphericalCoords (void)

Updates Spherical Coordinates of the Camera

Definition at line 33 of file control.c.

4.3 convert.h File Reference

Routines for converting plot data.

Defines

- #define **STRIPWIDTH** 0.1

Functions

- **dataset** ∗ **funcToPara** (struct **dataset** ∗ds)
- **dataset** ∗ **kurvToPara** (struct **dataset** ∗ds)
- **dataset** ∗ **convertToPara** (char type[5], struct **dataset** ∗ds)

4.3.1 Detailed Description

Routines for converting plot data.

3dplot deals with parametric data (PARA). Other formats are supported by 3dplot but are converted to parametric. The following data types are supported:

- PARA Parametric data, this is the default
- FUNC Function plot data
- KURV Kurv plot data See the FILE_SPEC doc for full details

Definition in file **convert.h**.

4.3.2 Define Documentation

4.3.2.1 #define STRIPWIDTH 0.1

Width of line used to plot KURV Plots

Definition at line 21 of file convert.h.

4.3.3 Function Documentation

4.3.3.1 struct dataset∗ convertToPara (char *type*[5], struct dataset ∗ *ds*)

Converting data to parametric

This is the function that should be called. It detects what type the data is and then calls one of specific converting functions.

Parameters:

type Sting containing plot type as defined in the data file 5 chars includs NULL terminator

ds Pointer to data set to convert

Returns:

Pointer to converted dataset

Definition at line 84 of file convert.c.

4.3.3.2 struct dataset∗ funcToPara (struct dataset ∗ *ds*)

Converts Function data to Parametric

Parameters:

ds Pointer to data set to convert

Returns:

Pointer to converted dataset

Definition at line 14 of file convert.c.

4.3.3.3 struct dataset∗ kurvToPara (struct dataset ∗ *ds*)

Converts Kurv (Curve) data to Parametric

Parameters:

ds Pointer to data set to convert

Returns:

Pointer to converted dataset

Definition at line 39 of file convert.c.

4.4 display.h File Reference

Routines for handling OpenGL Display functions.

```
#include "main.h"
```

Enumerations

- enum **window_defaults** { **DEF_WIN_WIDTH** = 200, **DEF_WIN_HEIGHT** = 200, **DEF_WIN_X_POS** = 100, **DEF_WIN_Y_POS** = 100 }
- enum **camera_defaults** {
  CAMX = 5, **CAMY** = 0, **CAMZ** = 0, **CENTREX** = 0,
  CENTREY = 0, **CENTREZ** = 0, **UPVECTORX** = 0, **UPVECTORY** = 0,
  UPVECTORZ = 1 }

Functions

- void **initDisplay** (void)
- void **setCameraDefaults** (void)
- void **reshape** (int w, int h)
- void **drawAxis** (void)
- void **renderGraph** (void)
- void **drawGraph** (struct **dataset** ∗ds)

4.4.1 Detailed Description

Routines for handling OpenGL Display functions.

Definition in file **display.h**.

4.4.2 Enumeration Type Documentation

4.4.2.1 enum camera_defaults

Define camera defaults

Used by gluLookAt.

Enumeration values:

CAMX	x position coordinate
CAMY	y position coordinate
CAMZ	z position coordinate
CENTREX	x centre point coordinate
CENTREY	y centre point coordinate
CENTREZ	z centre point coordinate
UPVECTORX	x up vector value
UPVECTORY	y up vector value
UPVECTORZ	z up vector value

Definition at line 22 of file display.h.

4.4.2.2 enum window_defaults

Defines window defaults

Enumeration values:

DEF_ WIN_ WIDTH	Window width in pixels
DEF_ WIN_ HEIGHT	Window height in pixels
DEF_ WIN_ X_ POS	Window x position from left in pixels
DEF_ WIN_ Y_ POS	Window y position from top in pixels

Definition at line 11 of file display.h.

4.4.3 Function Documentation

4.4.3.1 void drawAxis (void)

Draw the graph axes

Each axis from -1 to 1, all data will be scaled.

Definition at line 100 of file display.c.

4.4.3.2 void drawGraph (struct dataset ∗ *ds*)

Function to do the initial drawing of the graph

Definition at line 142 of file display.c.

4.4.3.3 void initDisplay (void)

Initialise the display

- Initalise Window
- Clear Background
- Set up camera

Definition at line 22 of file display.c.

4.4.3.4 void renderGraph (void)

The glutDisplayFunc() display function.

- Call the **drawAxis()**(p. 18) function
- Plot the graph (held as a GL List)

Definition at line 182 of file display.c.

4.4.3.5 void reshape (int *w*, int *h*)

Function to be called when window is resized

Parameters:

- ***w*** New width of window
- ***h*** New height of window

Definition at line 155 of file display.c.

4.4.3.6 void setCameraDefaults (void)

Sets the camera defaults

Definition at line 62 of file display.c.

4.5 file.h File Reference

Routines for dealing with data files.

`#include "main.h"`

Functions

- long **getxNum** (struct **dataset** ∗ds)
- long **getyNum** (struct **dataset** ∗ds)
- long **getzNum** (struct **dataset** ∗ds)
- void **destroyDataSet** (struct **dataset** ∗ds)
- **dataset** ∗ **readDataFile** (FILE ∗fh)

4.5.1 Detailed Description

Routines for dealing with data files.

Definition in file **file.h**.

4.5.2 Function Documentation

4.5.2.1 void destroyDataSet (struct dataset ∗ *ds*)

Frees up memory used by unused dataset

Parameters:
ds Dataset to free up

Definition at line 112 of file file.c.

4.5.2.2 long getxNum (struct dataset ∗ *ds*)

Gets the number of X values in the dataset

Parameters:
ds Pointer to dataset

Returns:
Number of values.

Definition at line 18 of file file.c.

4.5.2.3 long getyNum (struct dataset ∗ *ds*)

Gets the number of Y values in the dataset

Parameters:
ds Pointer to dataset

Returns:

Number of values.

Definition at line 23 of file file.c.

4.5.2.4 long getzNum (struct dataset * *ds*)

Gets the number of Z values in the dataset

Parameters:

ds Pointer to dataset

Returns:

Number of values.

Definition at line 28 of file file.c.

4.5.2.5 struct dataset* readDataFile (FILE * *fh*)

Reads the data file into a data set

Parameters:

fh Pointer to data file

Returns:

Dataset containing data from file.

Definition at line 126 of file file.c.

4.6 graph.h File Reference

Routines for plotting the graph itself.

```
#include "main.h"
```

Functions

- void **defineMaterial** (void)
- float **redMap** (float height)
- float **greenMap** (float height)
- float **blueMap** (float height)
- void **plotGraph** (struct **dataset** *ds)

4.6.1 Detailed Description

Routines for plotting the graph itself.

Definition in file **graph.h**.

4.6.2 Function Documentation

4.6.2.1 float blueMap (float *height*)

Computers the Blue value for a height on the graph

This function assumes the height to be between -1 and 1

Parameters:
height Height of point.

Returns:
Blue colour value as used by OpenGL (i.e. between 0.0 and 1.0)

Definition at line 43 of file graph.c.

4.6.2.2 void defineMaterial (void)

Defines the material used on the graph

Definition at line 17 of file graph.c.

4.6.2.3 float greenMap (float *height*)

Computers the Green value for a height on the graph

This function assumes the height to be between -1 and 1

Parameters:
height Height of point.

Returns:
Green colour value as used by OpenGL (i.e. between 0.0 and 1.0)

Definition at line 36 of file graph.c.

4.6.2.4 void plotGraph (struct dataset * *ds*)

Plots the graph itself

This parses the dataset and plots the point. It assumes that the dataset is scaled so that all points lie between -1 and 1

Parameters:

ds dataset to plot

Definition at line 50 of file graph.c.

4.6.2.5 float redMap (float *height*)

Computers the Red value for a height on the graph

This function assumes the height to be between -1 and 1

Parameters:

height Height of point.

Returns:

Red colour value as used by OpenGL (i.e. between 0.0 and 1.0)

Definition at line 28 of file graph.c.

4.7 lights.h File Reference

Routines for dealing with lights.

`#include <GL/glut.h>`

Data Structures

- struct **light**

Functions

- int **addLight** (float x, float y, float z)
- void **destroyLight** (int lightID)
- void **delLight** (int lightID)
- void **moveLight** (int lightid, float x, float y, float z, float w)

4.7.1 Detailed Description

Routines for dealing with lights.

Definition in file **lights.h**.

4.7.2 Function Documentation

4.7.2.1 int addLight (float *x*, float *y*, float *z*)

Adds a light

Parameters:

x x coordinate of light

y y coordinate of light

z z coordinate of light

Returns:

lightid of light FIXME: needs to be implemented

Definition at line 21 of file lights.c.

4.7.2.2 void delLight (int *lightID*)

Delets a light

Parameters:

lightID OpenGL light ID

Definition at line 89 of file lights.c.

4.7.2.3 void destroyLight (int *lightID*)

Frees up memory for unused light

Parameters:

lightID OpenGL light ID

Definition at line 84 of file lights.c.

4.7.2.4 void moveLight (int *lightid*, float *x*, float *y*, float *z*, float *w*)

Moves a light relative to its current position

Parameters:

lightid ID of light to move TODO: Not supported yet

x New relative x position

y New relative y position

z New relative z position

w New relative w position - if 0 then light is positional.

Definition at line 94 of file lights.c.

4.8 main.h File Reference

Data Structures

- struct **dataset**

Functions

- int **expandFloatOption** (float ∗values, char ∗string)
- int **expandIntOption** (int ∗values, char ∗string)

4.8.1 Detailed Description

main.h(p. 26) 3dplot

Created by Nathan on Thu Oct 23 2003. Copyright (c) 2003 __MyCompanyName__. All rights reserved.

Definition in file **main.h**.

4.9 movie.h File Reference

Routines for production of movies.

Data Structures

- struct **frame**

Enumerations

- enum **buffer** { **BUFSIZE** = 1025 }

Functions

- void **destroyFrames** (struct **frame** ∗frames)
- **frame** ∗ **readFrames** (char ∗fn)
- void **renderFrames** (struct **frame** ∗frames)
- void **displayMovie** (void)
- void **saveMovie** (void)

4.9.1 Detailed Description

Routines for production of movies.

Definition in file **movie.h**.

4.9.2 Enumeration Type Documentation

4.9.2.1 enum buffer

Defines buffer size

Definition at line 10 of file movie.h.

4.9.3 Function Documentation

4.9.3.1 void destroyFrames (struct frame ∗ *frames*)

Frees memory from unused frames

Parameters:
  frames Frames to destroy

Definition at line 21 of file movie.c.

4.9.3.2 void displayMovie (void)

Display function used by glutDisplayFunc

Definition at line 161 of file movie.c.

4.9.3.3 struct frame∗ readFrames (char ∗ *fn*)

Loads frame names into an array

Parameters:

fh File containing data file names

Returns:

array containing list of files

Definition at line 39 of file movie.c.

4.9.3.4 void renderFrames (struct frame ∗ *frames*)

Renders the frams to display lists

Parameters:

frames Frames to render

path Path to save frames to

Definition at line 99 of file movie.c.

4.9.3.5 void saveMovie (void)

Function to display and save the movie to a file

again used by glutDisplayFunc

Definition at line 179 of file movie.c.

4.10 savegraph.h File Reference

Routines for saving the output to image files.

Functions

- int **TGAShot** (unsigned char ∗image, int width, int height, FILE ∗fh)
- int **takeScreenshot** (char ∗fn)
- int **saveGraph** (char ∗fn)

4.10.1 Detailed Description

Routines for saving the output to image files.

These routines will save the image to any desired format using the ImageMagick routines.

Definition in file **savegraph.h**.

4.10.2 Function Documentation

4.10.2.1 int saveGraph (char ∗ *fn*)

Saves current screen to a file

The output format is detected from the filename extension

Parameters:

fn Filename to save to

Returns:

0 if sucessful -1 if not

Definition at line 96 of file savegraph.c.

4.10.2.2 int takeScreenshot (char ∗ *fn*)

Takes a screenshot and calls **TGAShot()**(p. 33)

Parameters:

fn File name to wrte to.

Returns:

0 if successful -1 if not.

Definition at line 57 of file savegraph.c.

4.10.2.3 int TGAShot (unsigned char ∗ *image*, int *width*, int *height*, FILE ∗ *fh*)

Saves cuurent screen to TGA file

Used so then **saveGraph()**(p. 29) can convert to desired format

Parameters:

image Contents image via of glReadPixels(x, y, width, height, GL_BGR, GL_-UNSIGNED_BYTE, image)

width Width of image

height Height of image

fh File to save to

Returns:

0 if succesful

Definition at line 14 of file savegraph.c.

4.11 screenshot.h File Reference

Built in funtions for creating screenshots.

Enumerations

- enum **SCREENSHOT_FORMAT** { **SCREENSHOT_PPM**, **SCREENSHOT_-TGA**, **SCREENSHOT_BMP**, **SCREENSHOT_RAW** }

Functions

- int **takeScreenshot** (char *fn, **SCREENSHOT_FORMAT** format)
- int **BMPShot** (unsigned char *image, int width, int height, FILE *fh)
- int **TGAShot** (unsigned char *image, int width, int height, FILE *fh)
- int **PPMShot** (unsigned char *image, int width, int height, FILE *fh)
- int **RAWShot** (unsigned char *image, int width, int height, FILE *fh)

4.11.1 Detailed Description

Built in funtions for creating screenshots.

Now mostly redundant due to using ImageMagicK libs

Definition in file **screenshot.h**.

4.11.2 Enumeration Type Documentation

4.11.2.1 enum SCREENSHOT_FORMAT

Defines the built in types of screenshot avaliable

Enumeration values:

SCREENSHOT_PPM	PPM - `http://netghost.narod.ru/gff/vendspec/pbm/ppm.txt`
SCREENSHOT_TGA	TGA - `http://netghost.narod.ru/gff/vendspec/tga/tga.txt`
SCREENSHOT_BMP	BMP - `http://netghost.narod.ru/gff/vendspec/micbmp/bmp.txt`
SCREENSHOT_RAW	RAW glReadPixel dump

Definition at line 12 of file screenshot.h.

4.11.3 Function Documentation

4.11.3.1 int BMPShot (unsigned char * *image*, int *width*, int *height*, FILE * *fh*)

Writes BMP screenshot

Parameters:

image Image to save - dump of glReadPixels

width Width of image

height Height of image

fh File to save to

Returns:

0 if successful

Definition at line 17 of file screenshot.c.

4.11.3.2 int PPMShot (unsigned char ∗ *image*, int *width*, int *height*, FILE ∗ *fh*)

Writes PPM screenshot

Parameters:

image Image to save - dump of glReadPixels

width Width of image

height Height of image

fh File to save to

Returns:

0 if successful

Definition at line 109 of file screenshot.c.

4.11.3.3 int RAWShot (unsigned char ∗ *image*, int *width*, int *height*, FILE ∗ *fh*)

Writes RAW screenshot

Parameters:

image Image to save - dump of glReadPixels

width Width of image

height Height of image

fh File to save to

Returns:

0 if successful

Definition at line 126 of file screenshot.c.

4.11.3.4 int takeScreenshot (char ∗ *fn*, SCREENSHOT_FORMAT *format*)

Takes the screenshot using one of the built in functions

Parameters:

fn File name to save to

format Format to save to

Returns:

0 if successful -1 of not

4.11.3.5 int TGAShot (unsigned char * *image*, int *width*, int *height*, FILE * *fh*)

Saves cuurent screen to TGA file

Used so then **saveGraph()**(p. 29) can convert to desired format

Parameters:

- ***image*** Contents image via of glReadPixels(x, y, width, height, GL_BGR, GL_-UNSIGNED_BYTE, image)
- ***width*** Width of image
- ***height*** Height of image
- ***fh*** File to save to

Returns:

0 if succesful

Definition at line 14 of file savegraph.c.

Index