

Graphics Engine

1.0

Generated by Doxygen 1.8.14

Contents

1	3802ICT	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	colour Struct Reference	5
3.1.1	Detailed Description	5
3.2	matrix Class Reference	5
3.2.1	Constructor & Destructor Documentation	6
3.2.1.1	matrix() [1/2]	6
3.2.1.2	matrix() [2/2]	6
3.2.2	Member Function Documentation	6
3.2.2.1	add_val()	6
3.2.2.2	get_cols()	7
3.2.2.3	get_rows()	7
3.2.2.4	get_val()	7
3.2.2.5	multiply()	7
3.2.2.6	print()	8
3.2.2.7	set_col()	8
3.2.2.8	set_row()	8
3.2.2.9	set_up_transformation()	8
3.2.2.10	set_val()	9
3.3	point Struct Reference	9

3.3.1	Detailed Description	9
3.4	Polygon Class Reference	9
3.4.1	Constructor & Destructor Documentation	10
3.4.1.1	Polygon() [1/3]	10
3.4.1.2	Polygon() [2/3]	10
3.4.1.3	Polygon() [3/3]	11
3.4.2	Member Function Documentation	11
3.4.2.1	additive_rotate()	11
3.4.2.2	additive_translate()	11
3.4.2.3	change_points()	11
3.4.2.4	draw()	12
3.4.2.5	find_bottom_right_point()	12
3.4.2.6	find_top_left_point()	12
3.4.2.7	rotate()	12
3.4.2.8	save_transformation()	12
3.4.2.9	scale()	13
3.4.2.10	set_colour()	13
3.4.2.11	translate()	13
3.4.2.12	undo_transformation()	13
3.5	rectangle Struct Reference	14
3.5.1	Detailed Description	14
3.6	Text Class Reference	14
3.6.1	Constructor & Destructor Documentation	15
3.6.1.1	Text()	15
3.6.2	Member Function Documentation	15
3.6.2.1	draw()	15
3.6.2.2	update_text()	15
3.7	velocity Struct Reference	16
3.7.1	Detailed Description	16

Chapter 1

3802ICT

Chapter 2

Class Index

2.1 Class List

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

colour	5
matrix	5
point	9
Polygon	9
rectangle	14
Text	14
velocity	16

Chapter 3

Class Documentation

3.1 colour Struct Reference

```
#include <structs.h>
```

Public Attributes

- double **R**
- double **G**
- double **B**

3.1.1 Detailed Description

Three doubles representing values for red, green and blue respectively

Parameters

<i>R</i>	red value
<i>G</i>	green value
<i>B</i>	blue value

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

- structs.h

3.2 matrix Class Reference

Public Member Functions

- [matrix](#) (int num_rows, int num_cols)
- [matrix](#) ()

- void [set_row](#) (int row_number, std::vector< double > row)
- void [set_col](#) (int col_number, std::vector< double > col)
- void [set_val](#) (int row_number, int col_number, double val)
- void [add_val](#) (int row_number, int col_number, double val)
- double [get_val](#) (int row_number, int col_number)
- int [get_rows](#) ()
- int [get_cols](#) ()
- void [print](#) ()
- [matrix multiply](#) ([matrix](#) other_matrix)
- void [set_up_transformation](#) ()

3.2.1 Constructor & Destructor Documentation

3.2.1.1 [matrix\(\)](#) [1/2]

```
matrix::matrix (
    int rows,
    int cols )
```

Constructor for a matrix when given and number of rows and columns

Parameters

<i>rows</i>	an integer representing the number of rows the matrix should have
<i>cols</i>	an integer representing the number of columns the matrix should have

3.2.1.2 [matrix\(\)](#) [2/2]

```
matrix::matrix ( )
```

Constructor for a matrix if not given a size. Makes a 4 by 4 matrix

3.2.2 Member Function Documentation

3.2.2.1 [add_val\(\)](#)

```
void matrix::add_val (
    int row_number,
    int col_number,
    double val )
```

Adds a value at the given row and column with the given value

Parameters

<i>row_number</i>	integer representing which row to replace
<i>col_number</i>	integer representing which column to replace
<i>val</i>	double representing the value to add in the matrix

3.2.2.2 `get_cols()`

```
int matrix::get_cols ( )
```

Returns the number of columns the matrix has

3.2.2.3 `get_rows()`

```
int matrix::get_rows ( )
```

Returns the number of rows the matrix has

3.2.2.4 `get_val()`

```
double matrix::get_val (
    int row_number,
    int col_number )
```

Returns a value at the given row and column

Parameters

<i>row_number</i>	integer representing which row to replace
<i>col_number</i>	integer representing which column to replace

3.2.2.5 `multiply()`

```
matrix matrix::multiply (
    matrix other_matrix )
```

Performs matrix dot multiplication between this matrix and a given matrix

Parameters

<i>other_matrix</i>	the matrix to multiply with
---------------------	-----------------------------

3.2.2.6 print()

```
void matrix::print ( )
```

Prints out the matrix to stdout in a formatted way

3.2.2.7 set_col()

```
void matrix::set_col (
    int col_number,
    std::vector< double > col )
```

Replaces a col in the matrix with the given vector

Parameters

<i>col_number</i>	integer representing which column to replace
<i>col</i>	vector of doubles representing the new col values

3.2.2.8 set_row()

```
void matrix::set_row (
    int row_number,
    std::vector< double > row )
```

Replaces a row in the matrix with the given vector

Parameters

<i>row_number</i>	integer representing which row to replace
<i>row</i>	vector of doubles representing the new row values

3.2.2.9 set_up_transformation()

```
void matrix::set_up_transformation ( )
```

Sets up the matrix as a blank transformation matrix

3.2.2.10 set_val()

```
void matrix::set_val (
    int row_number,
    int col_number,
    double val )
```

Replaces a value at the given row and column with the given value

Parameters

<i>row_number</i>	integer representing which row to replace
<i>col_number</i>	integer representing which column to replace
<i>val</i>	double representing the value to use in the matrix

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

- matrix.h
- matrix.cpp

3.3 point Struct Reference

```
#include <structs.h>
```

Public Attributes

- int **x**
- int **y**

3.3.1 Detailed Description

Two integers representing a pixel on the screen

Parameters

<i>x</i>	x co-ordinate of pixel
<i>y</i>	y co-ordinate of pixel

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

- structs.h

3.4 Polygon Class Reference

Public Member Functions

- [Polygon](#) (std::vector< [point](#) > points, [point](#) coordinates)
- [Polygon](#) (std::vector< [point](#) > points)
- [Polygon](#) ()
- void [change_points](#) (std::vector< [point](#) > points)
- void [set_colour](#) ([colour](#) RGB)
- void [draw](#) ()
- void [scale](#) (int x_scale, int y_scale)
- void [rotate](#) (double angle)
- void [additive_rotate](#) (double angle)
- void [translate](#) (double x_offset, double y_offset)
- void [additive_translate](#) (double x_offset, double y_offset)
- void [save_transformation](#) ()
- void [undo_transformation](#) ()
- [point](#) [find_top_left_point](#) ()
- [point](#) [find_bottom_right_point](#) ()

3.4.1 Constructor & Destructor Documentation

3.4.1.1 [Polygon\(\)](#) [1/3]

```
Polygon::Polygon (
    std::vector< point > points,
    point coordinates )
```

Constructor for a polygon when give the points and a starting point

Parameters

<i>points</i>	vector of points representing the points making up the polygon
<i>coordinates</i>	a point representing where the polygon should be drawn on the screen

3.4.1.2 [Polygon\(\)](#) [2/3]

```
Polygon::Polygon (
    std::vector< point > points )
```

Constructor for a polygon when give the points. Draws the polygon at 0, 0

Parameters

<i>points</i>	vector of points representing the points making up the polygon
---------------	----------------------------------------------------------------

3.4.1.3 Polygon() [3/3]

```
Polygon::Polygon ( )
```

Constructor for a blank [Polygon](#). Defines one point at 0, 0 and draws it at 0, 0

3.4.2 Member Function Documentation

3.4.2.1 additive_rotate()

```
void Polygon::additive_rotate (
    double angle )
```

Rotates the [Polygon](#) by the given angle (Adds to rotation)

Parameters

<i>angle</i>	double representing the angle in degrees
--------------	------------------------------------------

3.4.2.2 additive_translate()

```
void Polygon::additive_translate (
    double x_offset,
    double y_offset )
```

Translates the [Polygon](#) by the given dimensions (Adds to translation)

Parameters

<i>x_scale</i>	integer representing what to move the x value of the Polygon by
<i>y_scale</i>	integer representing what to move the y value of the Polygon by

3.4.2.3 change_points()

```
void Polygon::change_points (
    std::vector< point > points )
```

Replaces the vector the [Polygon](#)'s points with the given vector

Parameters

<i>points</i>	vector of points representing the points making up the polygon
---------------	----------------------------------------------------------------

3.4.2.4 draw()

```
void Polygon::draw ( )
```

Draws the [Polygon](#) on the screen

3.4.2.5 find_bottom_right_point()

```
point Polygon::find_bottom_right_point ( )
```

Finds the highest x value and the smallest y value

3.4.2.6 find_top_left_point()

```
point Polygon::find_top_left_point ( )
```

Finds the smallest x value and the highest y value

3.4.2.7 rotate()

```
void Polygon::rotate (
    double angle )
```

Rotates the [Polygon](#) to the given angle

Parameters

<i>angle</i>	double representing the angle in degrees
--------------	------------------------------------------

3.4.2.8 save_transformation()

```
void Polygon::save_transformation ( )
```

Saves the currently used matrix to a stack

3.4.2.9 scale()

```
void Polygon::scale (
    int x_scale,
    int y_scale )
```

Scales the [Polygon](#) by the given dimensions

Parameters

<i>x_scale</i>	integer representing what to multiply the x scale of the Polygon by
<i>y_scale</i>	integer representing what to multiply the y scale of the Polygon by

3.4.2.10 set_colour()

```
void Polygon::set_colour (
    colour RGB )
```

Replaces the colour the [Polygon](#) is filled with with the given colour

Parameters

<i>RGB</i>	struct representing the 3 double values for R, G and B
------------	--------------------------------------------------------

3.4.2.11 translate()

```
void Polygon::translate (
    double x_offset,
    double y_offset )
```

Translates the [Polygon](#) to the given dimensions

Parameters

<i>x_scale</i>	integer representing where to move the x value of the Polygon to
<i>y_scale</i>	integer representing where to move the y value of the Polygon to

3.4.2.12 undo_transformation()

```
void Polygon::undo_transformation ( )
```

Reverts the transformation matrix to the last saved matrix

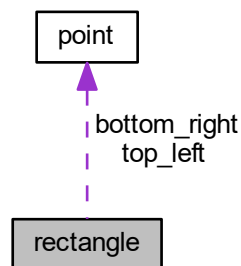
The documentation for this class was generated from the following files:

- polygon.h
- polygon.cpp

3.5 rectangle Struct Reference

```
#include <structs.h>
```

Collaboration diagram for rectangle:



Public Attributes

- [point](#) **top_left**
- [point](#) **bottom_right**

3.5.1 Detailed Description

Two points representing the top left and bottom right corners to form a rectangle

Parameters

<i>top_left</i>	point where the top left of the rectangle is located
<i>bottom_right</i>	point where the bottom right of the recatngle is located

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

- structs.h

3.6 Text Class Reference

Public Member Functions

- [Text](#) (std::string text_to_display, [point](#) bottom_left_position, [colour](#) RBG)

- void `draw` ()
- void `update_text` (std::string text_to_display)

3.6.1 Constructor & Destructor Documentation

3.6.1.1 Text()

```
Text::Text (
    std::string text_to_display,
    point bottom_left_position,
    colour RGB )
```

Constructor for text. Currently only supports numbers

Parameters

<i>text_to_display</i>	a string representing what text to display
<i>bottom_left_position</i>	a point representing the bottom left position to draw the text
<i>RGB</i>	a struct representing the RGB values of the colour to draw the text as

3.6.2 Member Function Documentation

3.6.2.1 draw()

```
void Text::draw ( )
```

Draws the numbers to the screen

3.6.2.2 update_text()

```
void Text::update_text (
    std::string text_to_display )
```

Replaces the text that will be replaced

Parameters

<i>update_text</i>	string representing the new text to display
--------------------	---------------------------------------------

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

- text.h
- text.cpp

3.7 velocity Struct Reference

```
#include <structs.h>
```

Public Attributes

- double **x**
- double **y**
- double **speed**

3.7.1 Detailed Description

Three doubles representing velocity as both x and y components and the magnitude of a vector

Parameters

<i>x</i>	x component of the velocity
<i>y</i>	y component of the velocity
<i>speed</i>	the magnitude of the velocity vector

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

- structs.h

Index

- add_val
 - matrix, [6](#)
- additive_rotate
 - Polygon, [11](#)
- additive_translate
 - Polygon, [11](#)
- change_points
 - Polygon, [11](#)
- colour, [5](#)
- draw
 - Polygon, [12](#)
 - Text, [15](#)
- find_bottom_right_point
 - Polygon, [12](#)
- find_top_left_point
 - Polygon, [12](#)
- get_cols
 - matrix, [7](#)
- get_rows
 - matrix, [7](#)
- get_val
 - matrix, [7](#)
- matrix, [5](#)
 - add_val, [6](#)
 - get_cols, [7](#)
 - get_rows, [7](#)
 - get_val, [7](#)
 - matrix, [6](#)
 - multiply, [7](#)
 - print, [8](#)
 - set_col, [8](#)
 - set_row, [8](#)
 - set_up_transformation, [8](#)
 - set_val, [8](#)
- multiply
 - matrix, [7](#)
- point, [9](#)
- Polygon, [9](#)
 - additive_rotate, [11](#)
 - additive_translate, [11](#)
 - change_points, [11](#)
 - draw, [12](#)
 - find_bottom_right_point, [12](#)
 - find_top_left_point, [12](#)
 - Polygon, [10](#)
 - rotate, [12](#)
 - save_transformation, [12](#)
 - scale, [12](#)
 - set_colour, [13](#)
 - translate, [13](#)
 - undo_transformation, [13](#)
- print
 - matrix, [8](#)
- rectangle, [14](#)
- rotate
 - Polygon, [12](#)
- save_transformation
 - Polygon, [12](#)
- scale
 - Polygon, [12](#)
- set_col
 - matrix, [8](#)
- set_colour
 - Polygon, [13](#)
- set_row
 - matrix, [8](#)
- set_up_transformation
 - matrix, [8](#)
- set_val
 - matrix, [8](#)
- Text, [14](#)
 - draw, [15](#)
 - Text, [15](#)
 - update_text, [15](#)
- translate
 - Polygon, [13](#)
- undo_transformation
 - Polygon, [13](#)
- update_text
 - Text, [15](#)
- velocity, [16](#)