JinksDraw

0.0

Generated by Doxygen 1.8.15

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 JinksDraw Namespace Reference

Classes

- class Circle
- class Line

  This class models a 2D line with a deque of points.
- class Point

  This class models a point in 2D space with an x and a y coordinate.
- class Primitive

  Empty class that all primitives inherit from. Useful for making lists.
- class Rectangle

Variables

- const Primitive PRIME_NULL = Primitive()

5.1.1 Variable Documentation

5.1.1.1 PRIME_NULL

```
const Primitive JinksDraw::PRIME_NULL = Primitive()
```

Chapter 6

Class Documentation

6.1 JinksDraw::Circle Class Reference

```
#include <primitives.h>
```

Inheritance diagram for JinksDraw::Circle:

class_jinks_draw_1_1_circle-eps-converted-to.pdf

Public Member Functions

- Circle (Point &origin, double radius)
- void reset ()
- Point getOrigin ()

  gets the origin
- double getRadius ()

  gets the radius
- void setOrigin (Point &newOrigin)

  sets the origin
- void setRadius (double newRadius)

  sets the radius
- std::vector< Point > intersection (Line &line)

  calculates the intersection of this circle and a line
- std::vector< Point > intersection (Circle &line)

  calculates the intersection of this circle and another circle

Private Attributes

- Point ∗ origin = new Point()
- double radius = 0.0

Friends

- std::ostream & operator<< (std::ostream &os, const Circle &cr)

  this allows Circle to have a stream representation

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Circle()

```
JinksDraw::Circle::Circle (
              Point & origin,
              double radius )
```

6.1.2 Member Function Documentation

6.1.2.1 getOrigin()

```
Point JinksDraw::Circle::getOrigin ( )
```

gets the origin

6.1.2.2 getRadius()

```
double JinksDraw::Circle::getRadius ( )
```

gets the radius

6.1.2.3 intersection() [1/2]

```
std::vector< Point > JinksDraw::Circle::intersection (
              Line & line )
```

calculates the intersection of this circle and a line

Parameters

Line&	line the line to intersect the circle

Returns

a vector containing the points of intersection if any

6.1.2.4 intersection() [2/2]

```
std::vector< Point > JinksDraw::Circle::intersection (
             Circle & line )
```

calculates the intersection of this circle and another circle

Parameters

Circle&	circle the circle to intersect the circle

Returns

a vector containing the points of intersection if any

6.1.2.5 reset()

```
void JinksDraw::Circle::reset ( )
```

6.1.2.6 setOrigin()

```
void JinksDraw::Circle::setOrigin (
             Point & newOrigin )
```

sets the origin

6.1.2.7 setRadius()

```
void JinksDraw::Circle::setRadius (
             double newRadius )
```

sets the radius

6.1.3 Friends And Related Function Documentation

6.1.3.1 operator<<

```
std::ostream & operator<< (
             std::ostream & os,
             const Circle & cr )  [friend]
```

this allows Circle to have a stream representation

```
Point origin = Point(1.0, 2.0);
double radius = 10.0;
cout « Circle(origin, radius) « endl; // Circle(O:Point(1.0, 2.0), R:10.0)
```

6.1.4 Member Data Documentation

6.1.4.1 origin

```
Point* JinksDraw::Circle::origin = new Point()  [private]
```

6.1.4.2 radius

```
double JinksDraw::Circle::radius = 0.0  [private]
```

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

- /home/kenjinks/Documents/JinksDraw/primitives.h

6.2 JinksDraw::Line Class Reference

This class models a 2D line with a deque of points.

`#include <primitives.h>`

Inheritance diagram for JinksDraw::Line:

class_jinks_draw_1_1_line-eps-converted-to.pdf

Public Member Functions

- Line (Point &newStart, Point &newEnd)

  the constructor for a 2D line
- void reset ()

  this method resets the class attributes to their default state
- Point & getStart ()

  returns a pointer to the start Point
- Point & getEnd ()

  returns a pointer to the end Point
- void setStart (Point &newStart)

  sets the start to a new Point
- void setEnd (Point &newEnd)

  sets the end to a new Point
- void setByPolar (Point origin, double radius, double angle)
- double calcSlope ()

  calculates the slope of the line
- std::vector< Point > intersection (Line &intersectingLine)

  returns the intersection of this line and another line if any
- std::vector< Point > subpoint (int divisions=2)

  returns a vector of points that subdivide the line,
- std::vector< Line > subline (int divisions=2)

  returns a vector of lines that subdivide the line
- double getLength ()

  calculates and returns the length of the line
- double getAngle ()

  calculates and returns the angle of the line in radians
- Line getUnitLine ()

  recalculates end point to be at 1 unit and same angle from start point

Private Attributes

- Point ∗ start = new Point()
- Point ∗ end = new Point()

Friends

- std::ostream & operator<< (std::ostream &os, const Line &ln)

  this allows Line to have a stream representation

6.2.1 Detailed Description

This class models a 2D line with a deque of points.

Methods include Intersection, Subpoint, Length, Angle... more methods may be created in the future Operator ostream is implemented

```
Point p1 = Point(1.0, 2.0)
Point p2 = Point(3.0, 4.0)
cout « Line(p1, p2) « endl; // ((1.0, 2.0), (3.0, 4.0))
```

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Line()

```
JinksDraw::Line::Line (
               Point & newStart,
               Point & newEnd )
```

the constructor for a 2D line

Parameters

const	Point& start a reference to a start Point
const	Point& end a reference to an end Point

6.2.3 Member Function Documentation

6.2.3.1 calcSlope()

```
double JinksDraw::Line::calcSlope ( )
```

calculates the slope of the line

Returns

a slope as a double

6.2.3.2 getAngle()

`double JinksDraw::Line::getAngle ( )`

calculates and returns the angle of the line in radians

Returns

the angle in radians

6.2.3.3 getEnd()

`Point * JinksDraw::Line::getEnd ( )`

returns a pointer to the end Point

Returns

a pointer to a Point object

6.2.3.4 getLength()

`double JinksDraw::Line::getLength ( )`

calculates and returns the length of the line

Returns

the length as a double

6.2.3.5 getStart()

`Point * JinksDraw::Line::getStart ( )`

returns a pointer to the start Point

Returns

a pointer to a Point object

6.2.3.6 getUnitLine()

```
Line JinksDraw::Line::getUnitLine ( )
```

recalculates end point to be at 1 unit and same angle from start point

6.2.3.7 intersection()

```
std::vector< Point > JinksDraw::Line::intersection (
             Line & intersectingLine )
```

returns the intersection of this line and another line if any

future plans to turn this into a template that will accept any primitive

Parameters

Line∗	intersectingLine the line intersecting this line

Returns

a vector of Point objects

6.2.3.8 reset()

```
void JinksDraw::Line::reset ( )
```

this method resets the class attributes to their default state

6.2.3.9 setByPolar()

```
void JinksDraw::Line::setByPolar (
             Point origin,
             double radius,
             double angle )
```

6.2.3.10 setEnd()

```
void JinksDraw::Line::setEnd (
             Point & newEnd )
```

sets the end to a new Point

6.2.3.11 setStart()

```
void JinksDraw::Line::setStart (
             Point & newStart )
```

sets the start to a new Point

6.2.3.12 subline()

```
std::vector< Line > JinksDraw::Line::subline (
             int divisions = 2 )
```

returns a vector of lines that subdivide the line

the default of 2 gives 2 equal halves

Parameters

int	divisions = 2 the number of divisions of the line

Returns

a vector of Line objects

6.2.3.13 subpoint()

```
std::vector< Point > JinksDraw::Line::subpoint (
             int divisions = 2 )
```

returns a vector of points that subdivide the line,

the default of 2 gives the midpoint

Parameters

int	divisions = 2 the number of divisions of the line

Returns

a vector of Point objects

6.2.4 Friends And Related Function Documentation

6.2.4.1 operator<<

```
std::ostream & operator<< (
             std::ostream & os,
             const Line & ln )  [friend]
```

this allows Line to have a stream representation

```
Point p1 = Point(1.0, 2.0);
Point p2 = Point(3.0, 4.0);
cout « Line(p1, p2) « endl; // Line(Point(1.0, 2.0), Point(3.0, 4.0))
```

6.2.5 Member Data Documentation

6.2.5.1 end

```
Point* JinksDraw::Line::end = new Point()  [private]
```

6.2.5.2 start

```
Point* JinksDraw::Line::start = new Point()  [private]
```

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

- /home/kenjinks/Documents/JinksDraw/primitives.h

6.3 JinksDraw::Point Class Reference

This class models a point in 2D space with an x and a y coordinate.

`#include <primitives.h>`

Inheritance diagram for JinksDraw::Point:

`class_jinks_draw_1_1_point-eps-converted-to.pdf`

Public Member Functions

- Point (double x=0.0, double y=0.0)

  the Point constructor
- double getX ()

  access the x coordinate
- double getY ()

  access the y coordinate
- void setX (double x)

  sets the x coordinate
- void setY (double y)

  sets the y coordinate
- void setByPolar (double radius, double angle)

  sets the x and y coordinates based on radius and angle from origin (0,0)

Private Attributes

- double x = 0.0

  the x coordinate
- double y = 0.0

  the y coordinate

Friends

- std::ostream & operator<< (std::ostream &os, const Point &pt)

  this allows Point to have a stream representation
- Point operator* (const Point &lhs, const double rhs)

  *The * is to scale the coordinates of the Point.*
- Point operator* (const double lhs, const Point &rhs)

  *The * is to scale the coordinates of the Point.*
- Point operator/ (const Point &lhs, const double rhs)
- Point operator/ (const double lhs, const Point &rhs)

  The / is to scale the coordinates of the Point.
- Point operator+ (const Point &lhs, const Point &rhs)

  Allows adding two Points.
- Point operator- (const Point &lhs, const Point &rhs)

  Allows subtracting two Points.

6.3.1 Detailed Description

This class models a point in 2D space with an x and a y coordinate.

operators on this class include:

ostream,

```
cout « Point(10.0, 10.0) « endl; // (10.0, 10.0)
```

scale,

```
double d = 10.0;
Point p1 = Point(1.0, 2.0) * d;
Point p2 = d * Point(3.0, 4.0);
cout « p1 « endl; // (10.0, 20.0)
cout « p2 « endl; // (30.0, 40.0)
```

add and subtract.

```
Point p1 = Point(1.0, 2.0)
Point p2 = Point(3.0, 4.0)
Point p3 = p1 + p2;
Point p4 = p2 - p1;
cout « p3 « endl; // (4.0, 6.0)
cout « p4 « endl; // (2.0, 2.0)
```

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Point()

```
JinksDraw::Point::Point (
              double x = 0.0,
              double y = 0.0 )
```

the Point constructor

Parameters

double	x the x coordinate
double	y the y coordinate

6.3.3 Member Function Documentation

6.3.3.1 getX()

```
double JinksDraw::Point::getX ( )
```

access the x coordinate

Returns

the x coordinate as a double

6.3.3.2 getY()

```
double JinksDraw::Point::getY ( )
```

access the y coordinate

Returns

the y coordinate as a double

6.3.3.3 setByPolar()

```
void JinksDraw::Point::setByPolar (
             double radius,
             double angle )
```

sets the x and y coordinates based on radius and angle from origin (0,0)

Parameters

double	radius distance from origin (0,0)
double	angle angle in radians from origin (0,0)

6.3.3.4 setX()

```
void JinksDraw::Point::setX (
             double x )
```

sets the x coordinate

Parameters

double	x the value to set the x coordinate

6.3.3.5 setY()

```
void JinksDraw::Point::setY (
              double y )
```

sets the y coordinate

Parameters

double	y the value to set the y coordinate

6.3.4 Friends And Related Function Documentation

6.3.4.1 operator* [1/2]

```
Point operator* (
              const Point & lhs,
              const double rhs )  [friend]
```

The * is to scale the coordinates of the Point.

```
double d = 10.0;
Point p1 = Point(1.0, 2.0) * d;
cout « p1 « endl; // (10.0, 20.0)
```

6.3.4.2 operator* [2/2]

```
Point operator* (
              const double lhs,
              const Point & rhs )  [friend]
```

The * is to scale the coordinates of the Point.

```
double d = 10.0;
Point p2 = d * Point(3.0, 4.0);
cout « p2 « endl; // (30.0, 40.0)
```

6.3.4.3 operator+

```
Point operator+ (
              const Point & lhs,
              const Point & rhs )  [friend]
```

Allows adding two Points.

```
Point p1 = Point(1.0, 2.0)
Point p2 = Point(3.0, 4.0)
Point p3 = p1 + p2;
cout « p3 « endl; // (4.0, 6.0)
```

6.3.4.4 operator-

```
Point operator- (
             const Point & lhs,
             const Point & rhs )  [friend]
```

Allows subtracting two Points.

```
Point p1 = Point(1.0, 2.0)
Point p2 = Point(3.0, 4.0)
Point p4 = p2 - p1;
cout « p4 « endl; // (2.0, 2.0)
```

6.3.4.5 operator/ [1/2]

```
Point operator/ (
             const Point & lhs,
             const double rhs )  [friend]
```

6.3.4.6 operator/ [2/2]

```
Point operator/ (
             const double lhs,
             const Point & rhs )  [friend]
```

The / is to scale the coordinates of the Point.

```
double d = 5.0;
Point p2 = Point(5.0, 25.0) / d;
cout « p2 « endl; // (1.0, 5.0)
double d = 50.0;
Point p2 = d / Point(5.0, 25.0);
cout « p2 « endl; // (10.0, 2.0)
```

6.3.4.7 operator<<

```
std::ostream & operator<< (
             std::ostream & os,
             const Point & pt )  [friend]
```

this allows Point to have a stream representation

```
cout « Point(10.0, 10.0) « endl; // (10.0, 10.0)
```

6.3.5 Member Data Documentation

6.3.5.1 x

```
double JinksDraw::Point::x = 0.0  [private]
```

the x coordinate

6.3.5.2 y

```
double JinksDraw::Point::y = 0.0  [private]
```

the y coordinate

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

- /home/kenjinks/Documents/JinksDraw/primitives.h

6.4 JinksDraw::Primitive Class Reference

Empty class that all primitives inherit from. Useful for making lists.

```
#include <primitives.h>
```

Inheritance diagram for JinksDraw::Primitive:

6.4.1 Detailed Description

Empty class that all primitives inherit from. Useful for making lists.

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

- /home/kenjinks/Documents/JinksDraw/primitives.h

6.5 JinksDraw::Rectangle Class Reference

```
#include <primitives.h>
```

Inheritance diagram for JinksDraw::Rectangle:

class_jinks_draw_1_1_rectangle-eps-converted-to.pdf

Public Member Functions

- Rectangle (const Point &lowerLeft, const Point &upperRight)

Private Attributes

- Point ∗ lowerLeft
- Point ∗ upperRight

6.5.1 Constructor & Destructor Documentation

6.5.1.1 Rectangle()

```
JinksDraw::Rectangle::Rectangle (
             const Point & lowerLeft,
             const Point & upperRight )
```

6.5.2 Member Data Documentation

6.5.2.1 lowerLeft

```
Point* JinksDraw::Rectangle::lowerLeft  [private]
```

6.5.2.2 upperRight

```
Point* JinksDraw::Rectangle::upperRight  [private]
```

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

- /home/kenjinks/Documents/JinksDraw/primitives.h

Chapter 7

File Documentation

7.1 /home/kenjinks/Documents/JinksDraw/primitives.h File Reference

This file contains the prototypes for primitives.cpp.

```
#include "jinks_math.h"
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class JinksDraw::Primitive

  Empty class that all primitives inherit from. Useful for making lists.
- class JinksDraw::Point

  This class models a point in 2D space with an x and a y coordinate.
- class JinksDraw::Line

  This class models a 2D line with a deque of points.
- class JinksDraw::Circle
- class JinksDraw::Rectangle

Namespaces

- JinksDraw

Variables

- const Primitive JinksDraw::PRIME_NULL = Primitive()

7.1.1 Detailed Description

This file contains the prototypes for primitives.cpp.

Author

Ken Jinks

Date

Aug 2018

and is documented using Doxygen markup