March 16, 2021

Contents

{There are definitely questions that will be asked throughout the documentation process. They may or may not be implemented.}

Chapter 1

Interface

1.1 GeoPlayground

Top level user interface elements Controls which geometry we are working in and what model of that geometry Saving and Loading contains the canvas that all graphics take place within. I see that Section 1.1, in its entirety, could be an instructional chapter in and of itself.

1.1.1 Tabs

Across the top left of the interface is a series of tabs. These enable us to switch from one geometry to another.

1.1.2 Radio Buttons

These buttons determine the behavior of how we interact with the canvas. Most, with exception of the Model, are used to figure out *whatToDo* in the code. This variable is in turn used extensively in many other contained objects.

Make Points

. Make a stand alone point

X Make an intersection point based on two circles/lines

M Make a midpoint between two points

Make Lines

i line segment

I line

T perpendicular line (line and a point)

B angle bisector

Make Circles

O Draw a circle based on two points

Measure

d Distance (based on two points)

∠ Angle (based on three points)

c Circumference (based on a circle)

a Area (based on a circle)

Move

⇌ Move a point or space

@ fix a point or an object

Display

? Hide object

! Unhide all (with button)

§ Show or hide label

∅ Erase all (with button)

Model

This area of radio buttons adjust to display the available models for each geometry.

1.1.3 Buttons

doItButton

This button is used to unhide or erase all, when appropriate above

Load

Load the objects of a geometry into an appropriately named file. The extension is geometry specific.

Save

Save the objects of a geometry into an appropriately named file. This button only becomes active when something has changes on the canvas.

1.1.4 Instruction Text

infoLabel[] Located at the bottom of each tabbed window. Text informing the user of what each function does is displayed here.

{Currently this is duplicated across all tabs. If *whatToDo* stays the same when switching geometries it is no use to have multiple instances of *infoLabel*}

Chapter 2

Foundation Classes

2.1 GeoPlayground

Top level user interface elements Controls which geometry we are working in and what model of that geometry Saving and Loading contains the canvas that all graphics take place within. I see that Section 1.1, in its entirety, could be an instructional chapter in and of itself.

2.1.1 declarations

eastPanel, cntrPanel, makePtsPanel, makeLn1Panel, makeLn2Panel, makeCrPanel, movePanel, meas1Panel, meas2Panel, display1Panel, display2Panel : JPanel These panels are for the layout of the graphical user interface.

tabbedPanePlane : JTabbedPane

A JTabbedPane that is used to switch between geometries. Each tab contains an instance of junkPanel, canvas, and infoLabel.

junkPanel : JPanel[]

a set of panels with the same number of elements as tabs. Each panel contains a canvas, below.

canvas : GeoCanvas[]

a set of GeoCanvases with the same number of elements as tabs

whatToDoCBG : ButtonGroup

a ButtonGroup associated with whatToDo

makePtsButton, makeLnsButton, makeIntButton, makePrpButton, makeCrsButton, moveButton, measDistButton, measAngleButton, measAreaButton, hideButton, unhideButton, makeSegButton, labelButton, clearButton, fixButton, makeMPButton, measCircButton, makeBSButton : JRadioButton

The set of radio buttons that select whatToDo above.

modelCBG : ButtonGroup

a ButtonGroup associated with what model of a geometry we are looking at.

modelButton : JRadioButton[]

The set of radio buttons that select the model above

doItButton : Button

Depending on the whatToDo this becomes active for deleting all or showing all constructs.

infoLabel : JLabel[]

a set of JLabels with the same number of elements as tabs. Why do we need this many? I don't know!!!!!!!!!!

whatToDo : int

This is the integer value for whatToDoCBG above

model : int

This is the integer value for the model above

MakePoints, MakeLines, MakeInt, MakePerps, MakeCircles, Move, MeasureDist, MeasureAngle, MeasureArea, HideObject, UnhideAll, MakeSegment, LabelObject, ClearAll, FixObject, MakeMdPt, MeasureCirc, MakeBisect : int

Integer enumerations for whatToDo

geoModel : int[]

a set of integers indicating the default model a geometry is viewed by.

CANVASSIZE : int

Default Canvas Size

textString : String

text string used to update infoLabel

saveButton : Button

Used to save a geometry specific set of constructs to a file.

loadButton : Button

Used to load a set of geometry specific constructs from a file.

2.1.2 methods

init()

Initialize the GUI

setInfoTA()

Change the infoLabel based on whatToDo

itemStateChanged(ItemEvent)

This and the following manage the behaviour of various objects

stateChanged(ChangeEvent)

actionPerformed(ActionEvent)

createAndShowGUI()

What it says

main(String)

This is the main loop of the Java WebStart application

getHght(), getWdth()

Get the global size parameters of the application, these are used to adjust the size of the canvas and the size of the points on the canvas.

2.2 GeoCanvas

Is the canvas that all graphical constructs are seen and interacted with. Interprets all of the mouse interactions and keyboard commands associated with geometry visualization Instead of replicating specific geometry specific elements that are necessary for rendering, we use a geoMetry object.

2.2.1 declarations

SZ : int

This is the size of the canvas. It can change, as well as the size of the points, with the size of the GeoPlayground object.

BISECTOR, SEGMENT, PERP, LINE, CIRCLE, POINT, PTonLINE, PTonCIRC, LINEintLINE0,LINEintLINE1, CIRCintLINE0, CIRCintLINE1, CIRCintCIRC00, CIRCintCIRC01, CIRCintCIRC10, CIRCintCIRC11, MIDPT, FIXedPT, DISTANCE, AREA, CIRCUMF, ANGLE: int

These are enumerated integers denoting the type of construct that we are working with. This copies from the same integers in GeoConstruct.

typedKey: int

The key that is typed to control the zoom

whereAtInList: int

Not used.....yet.

alreadyMoving : boolean

???????????

geometryListener : ChangeListener

????????

geometry : GeoMetry

This encapsulates the geometry specific behaviour. All of the other behaviors are not specific to any geometry.

list : LinkedList<GeoConstruct>

This is a list of all of the GeoConstructs that are being managed by the canvas.

clickedList : LinkedList<GeoConstruct>

This is a list of all the clicked objects on a canvas. Different composit objects are based on different numbers of other objects. More on this later.

potentialClick : GeoConstruct

An object that is clickable. Depends on whatToDo.

firstClicked : GeoConstruct

*******Obselete*********

fixedObject : GeoConstruct

A fixed object that geometric transforms are performed around.

vector1 : double[]

??????????

vector2 : double[]

??????????

norm : double[]

??????????

binorm : double[]

??????????

dragStart : double[]

When we move a construct or the space we want to know where we dragged it from and where it is now.

dragNow : double[]

2.2.2 methods

GeoCanvas()

Constructor that initializes the canvas as multiple listeners (**obselete???? could be migrated down)

GeoCanvas(GeoMetry)

Constructor that sets the geometry

clearFixedObject()

What it says.

paint(Graphics)

Paints the geometry specific space followed by all objects in list. Objects in the list are in different colors depending on object type and what is a potentialClick or clickedObject. If we are moving an object or space the new positions of the objects will be lighter.

colorSet(Graphics, GeoConstruct, boolean)

Set the color of what is to be drawn

drawAllConstructs(Graphics, boolean)

Used by paint to draw all constructs

translateAll()

Translates the space. Happens when we grab the space and move it.

getPotentialAny(double[])

getPotential methods take a set of coordinates and returns the first object that are within range.

getPotentialComposite(double[])

getPotentialLine(double[])

getPotentialCircle(double[])

getPotentialPointOrMeasure(double[])

getPotentialPointOrInt(double[])

getPotentialPointOrIntOrMeasure(double[])

addChangeListener(ChangeListener)

Don't quite know why this is here.

mousePressed(MouseEvent)

Used when moving constructs or the space around. I am not sure that we really need the switch statement here.

mouseMoved(MouseEvent)

Depending on GeoPlayground.whatToDo set potentialClick when appropriate

mouseDragged(MouseEvent)

Used for drawing the new position of constructs relative to where they were. Again, does not seem like we need a switch statement.

mouseReleased(MouseEvent)

Used to set the current position of a construct to the new position of a construct and update all. Again, the switch statement.

mouseEntered(MouseEvent)

Not used. But still needed to fullfill the confines of the MouseListener interface.

mouseExited(MouseEvent)

Used to detect the mouse exiting the valid space defined by the geometry. If we drag a construct out of the valid space, reset the new position to the old position.

mouseClicked(MouseEvent)

This is the real engine of GeoCanvas. It controls the creation of all constructs depending on GeoPlayground.whatToDo. I have a lot of questions about how and why different things are in here.

FixObject Select the fixed object to do transformations of the rest of the objects around

MakePoints I am wondering, for efficiency(?) if the creation of the actual point should take place after we check to see if there is another one in its place. Also, it seems that the last CreatePoint in the first if is needless. We will never have a potentialClick that is a point when creating a point.

MakeMdPt create a midpoint object. clickedList.addFirst(potentialClick) is all that is needed here. Not the temp switcharoo. It also seems that we want the clickedList in ID order to be able to efficiently compare when we are searching to see if it already exists. Another way of doing this may be to add potClicked to clickedList if size¡? and then sort the list based on ID. Get it in the correct order no effort. When creating points would it be prudent to set both XYZ and newXYZ at the same time?

MeasureCirc Create a Measure Circle Object. "Special Point". All of these CreateObject Routines check to see if the object already exists. It may be prudent to create a "AlreadyExists" routine that goes through the "list" and checks for elements in clickedList. It would clean up this part of the code a little bit.

MeasureArea Create a Measure Area Object. "Special Point". Only for circles...so far

MeasureDist Measure Distance object. "Special Point" on a hidden line between two points. clickedList.addFirst(potentialClick) is all that is needed here. Not the temp switcharoo. I am seeing that extending the class hierarchy would make it possible to hide some of the extra stuff in the constructors of those classes. Stuff such as setLabelShown and setXYZ....

MeasureAngle "Special Point" on an Angle Bisector object that displays the angle between three points (A,B,C) where B is the vertex. It goes through and checks for the existence of both objects before creating them as needed. Though the absence of the first should indicate the necessity of the latter.

MakeBisect "Special Line" that is created to bisect the angle between three points.

MakeLines,MakeSegment Lines and segements are created between two points. We have an elaborate code to check both Lines and Segments and convert between the two of them if the object exists as the other.

MakeCircles Circles are created from two points {Center, Point on circle}

MakeInt Intersection between two composit objects (Circles or Lines inclusive)

MakePerps Use a line and a point (in that order) to create a perpendicular line. I am not sure of why the code is put like it is at first glance... Must refer to deej. It is not in the usual: gatherClicks-¿check AlreadyExists-¿create if not order.

HideObject Hide the clicked object and its label (if shown)

LabelObject Label the clicked object.

mouseWheelMoved(MouseWheelEvent)

Used to zoom in and out. Not applicable to all models of all geometries.

keyTyped(KeyEvent)

keyPressed(KeyEvent)

keyReleased(KeyEvent)

2.3 GeoMetry

This class of objects relegates all Geometry specific behaviours, such as drawing on the screen, creating constructs.... This class, in particular, is an abstract class that can never be instantiated by itself

2.3.1 declarations

methods

getCurvature()

Deej needs to explain this more.

drawModel(Graphics, int)

Provides the text for the model choice radio buttons and draws the appropriate grid or bounding space for that model.

convertMousetoCoord(MouseEvent, int)

Converts the mouse coordinates on screen to the coordinates in the appropriate model. I am seeing the possiblity of a model class.

mouseIsValid(double[])

Determins if the mouse is valid in the particular model.

createPoint(int, LinkedList<GeoConstruct> , double[])

This and the next three regulate the creation of different kinds of objects for each geometry.

createLine(int, LinkedList<GeoConstruct>)

createCircle(GeoConstruct, GeoConstruct)

createIntersections(GeoConstruct, GeoConstruct, LinkedList<GeoConstruct>)

dotProduct(double[], double[])

Mostly consistent measurements, Hyperbolic overloads

crossProduct(double[], double[])

acos(double)

distance(GeoConstruct, GeoConstruct)

Geometry specific measurements

angle(GeoConstruct, GeoConstruct)

area(GeoConstruct, boolean)

getScale()

For zoom capable models these do what they say

setScale(double)

extension()

Geometry specific Strings to assist in file management.

getName()

getFileFilter()

2.4 GeoConstruct

These are the individual graphics classes representing points, lines, and circles They are individualized for each geometry and model of that geometry. This particular class is abstract.

2.4.1 declarations

BISECTOR, SEGMENT, PERP, LINE, CIRCLE, POINT, PTonLINE, PTonCIRC, LINEintLINE0, LINEintLINE1, CIRCintLINE0, CIRCintLINE1, CIRCintCIRC00, CIRCintCIRC01, CIRCintCIRC10, CIRCintCIRC11, MIDPT, FIXedPT, DISTANCE, ANGLE, CIRCUMF, AREA : int

The type of object that something is.

x : double

3D vector representing coordinates in various geometries.

y : double

z : double

newX : double

3D vector representing a new coordinate in those geometry

newY : double

newZ : double

constList : LinkedList<GeoConstruct>

This is the list that this object is constructed from. Empty if none.

type : int

The type that an object is. Changable under certain conditions.

ID : int

The ID of the object, depending on its placement in the list. Could be gotten directly from the list, but why bother?

shown : boolean

Is the object visible

labelShown : boolean

Is the label visible

isReal : boolean

Is the object Viewable?

isRealNew : boolean

Is the new position of the object Viewable? Viewable meaning is its position able to be calculated and viewed in the space provided.

2.4.2 methods

GeoConstruct()

Really not used... ever. Could be gotten rid of.

GeoConstruct(int, double[])

These could be utilized more in subclasses.

GeoConstruct(int, double[], double[])

GeoConstruct(int, GeoConstruct, GeoConstruct)

GeoConstruct(int, double[], GeoConstruct)

getType()

What they say

getID()

getShown()

getLabelShown()

getX()

getNewX()

getY()

getNewY()

getZ()

getNewZ()

getValid()

Hey it's better than getReal!!!

getValidNew()

getSize()

Returns the size of the constList.

get(int)

Gets the i'th entry in the list. If i¿size, return null.

getXYZ(double[])

What they Say.

getNewXYZ(double[])

setID(int)

setType(int)

setShown(boolean)

setValid(boolean)

setValidNew(boolean)

setLabelShown(boolean)

setXYZ(double[])

setXYZ(double[], double[])

I don't know why we have this?

setNewXYZ(double[])

setNewXYZ(double[], double[])

Or this

draw(Graphics, int, boolean)

Draw the specific object

mouseIsOver(double[], int)

Take a coordinate and the size of the space and indicate whether the mouse is over the object.

update()

update the coordinates of the object

translate(double[], double[])

Need to ask Deej

transform(GeoConstruct, double[], double[])

Need to ask Deej

getScale()

I am going to have to review how all of these work.

setScale(double)

resetScale()

Chapter 3

Utility Classes

These are classes that aid the functionality of the program as a whole,

3.1 MathEqns

Deej is going to have to document these

3.1.1 methods

min(double, double)

max(double, double)

round(double)

chop(double)

chop(double, int)

norm(double[])

norm(double[], double[])

crossProduct(double[], double[], double[])

hypCrossProduct(double[], double[], double[])

normalize(double[])

hypNormalize(double[])

scalarProduct(double, double[])

addVec(double[], double[])

subVec(double[], double[])

dotProduct(double[], double[])

hypProduct(double[], double[])

rotate(double, double, double, double, ProjectiveConstruct)

rotate(double, double, double, double, SphericalConstruct)

transform(GeoConstruct, ProjectiveConstruct, double[], double[])

transform(GeoConstruct, SphericalConstruct, double[], double[])

transform(GeoConstruct, ToroidalConstruct, double[], double[])

transform(GeoConstruct, EuclideanConstruct, double[], double[])

transform(GeoConstruct, HyperConstruct, double[], double[])

hypPerp(double[], double[], double[])

hypLineIntLine(double[], double[], double[])

hypTranslate(double[], double[], double[])

sphTranslate(double[], double[], double[])

makeStandard(double[])

acosh(double)

eucAngle(double[], double[], double[])

hypAngle(double[], double[], double[])

sphAngle(double[], double[], double[])

3.2 CircleEqns

Deej is going to have to document these

3.2.1 methods

calculateCL(double[], double[], double[], double[], boolean)

calculateCC0(double[], double[], double[], double[], double[], boolean)

calculateCC1(double[], double[], double[], double[], double[], boolean)

calculateHypCL(double[], double[], double[], double[], boolean)

calculateHypCC(double[], double[], double[], double[], double[], boolean)

calculateEucCL(double[], double[], double[], double[], double[], boolean)

calculateEucCC(double[], double[], double[], double[], double[], boolean)

3.3 SaveLoad

3.3.1 methods

save(LinkedList <GeoConstruct>, GeoCanvas)

Saves the linkedList into a geometry specific file, using an extension, with the name of the geometry in the first line. The rest of the file consist of a line for each object in the list
`!KEY:ID,Type,<x,y,z>,{parent-1,parent-2,...},isShown,hasLabel,isReal`
It tries to save the file with a geometry specific extension.

load(LinkedList <GeoConstruct>, GeoCanvas)

This procedure parses a geometry specific file that we select into the linkedList. It has a couple of measures to make sure that we are loading the file into the appropriate geometry (file extension and name parameter in the first line). Each line is parsed and the appropriate construct references are made (what construct depends on others).**********I have been thinking about asking the user if they would like to switch to the appropriate geometry if the wrong one is chosen**********

processGeometry(String, String)

Is used to make sure the geometry name matches with the geometry name in the first line of the file.

processLine(String, LinkedList <GeoConstruct> , GeoMetry)

Uses a Scanner object to parse through the line and set the appropriate attributes of each construct.

StringToType(String)

Used to convert between the string and integer representations of the types of constructs.

TypeToString(int)

Chapter 4

Derived Classes

4.1 EuclideanGeometry extends GeoMetry

This is the specific class for Euclidean Geometry

4.1.1 declarations

4.1.2 methods

4.1.3 getCurvature()

drawModel(Graphics, int)

convertMousetoCoord(MouseEvent, int)

mouseIsValid(double[])

createPoint(int, LinkedList<GeoConstruct>, double[])

createLine(int, LinkedList<GeoConstruct>)

createCircle(GeoConstruct, GeoConstruct)

createIntersections(GeoConstruct, GeoConstruct, LinkedList<GeoConstruct>)

distance(GeoConstruct, GeoConstruct)

angle(GeoConstruct, GeoConstruct)

area(GeoConstruct, boolean)

getScale()

setScale(double)

extension()

getName()

getFileFilter()

4.2 ToroidalGeometry extends GeoMetry

This is the specific class for Euclidean Geometry

4.2.1 declarations

4.2.2 methods

getCurvature()

drawModel(Graphics, int)

convertMousetoCoord(MouseEvent, int)

mouseIsValid(double[])

createPoint(int, LinkedList<GeoConstruct>, double[])

createLine(int, LinkedList<GeoConstruct>)

createCircle(GeoConstruct, GeoConstruct)

createIntersections(GeoConstruct, GeoConstruct, LinkedList<GeoConstruct>)

distance(GeoConstruct, GeoConstruct)

angle(GeoConstruct, GeoConstruct)

area(GeoConstruct, boolean)

getScale()

setScale(double)

extension()

getName()

getFileFilter()

4.3 ProjectiveGeometry extends GeoMetry

This is the specific class for Euclidean Geometry

4.3.1 declarations

4.3.2 methods

getCurvature()

drawModel(Graphics, int)

convertMousetoCoord(MouseEvent, int)

mouseIsValid(double[])

createPoint(int, LinkedList<GeoConstruct>, double[])

createLine(int, LinkedList<GeoConstruct>)

createCircle(GeoConstruct, GeoConstruct)

createIntersections(GeoConstruct, GeoConstruct, LinkedList<GeoConstruct>)

distance(GeoConstruct, GeoConstruct)

angle(GeoConstruct, GeoConstruct)

area(GeoConstruct, boolean)

getScale()

setScale(double)

extension()

getName()

getFileFilter()

SphericalGeometry extends GeoMetry

This is the specific class for Euclidean Geometry

getCurvature()

drawModel(Graphics, int)

convertMousetoCoord(MouseEvent, int)

mouseIsValid(double[])

createPoint(int, LinkedList<GeoConstruct>, double[])

createLine(int, LinkedList<GeoConstruct>)

createCircle(GeoConstruct, GeoConstruct)

createIntersections(GeoConstruct, GeoConstruct, LinkedList<GeoConstruct>)

distance(GeoConstruct, GeoConstruct)

angle(GeoConstruct, GeoConstruct)

area(GeoConstruct, boolean)

getScale()

setScale(double)

extension()

getName()

getFileFilter()

4.4 HyperbolicGeometry extends GeoMetry

This is the specific class for Euclidean Geometry

4.4.1 declarations

4.4.2 methods

getCurvature()

drawModel(Graphics, int)

convertMousetoCoord(MouseEvent, int)

mouseIsValid(double[])

createPoint(int, LinkedList<GeoConstruct>, double[])

createLine(int, LinkedList<GeoConstruct>)

createCircle(GeoConstruct, GeoConstruct)

createIntersections(GeoConstruct, GeoConstruct, LinkedList<GeoConstruct>)

dotProduct(double[], double[])

acos(double)

distance(GeoConstruct, GeoConstruct)

angle(GeoConstruct, GeoConstruct)

area(GeoConstruct, boolean)

getScale()

setScale(double)

extension()

getName()

getFileFilter()

4.5 EuclideanConstruct extends GeoConstruct

This is the specific class for Euclidean Geometry

4.5.1 declarations

scale : double

scaleLimit : int

4.5.2 methods

EuclideanConstruct()

EuclideanConstruct(int, double[])

EuclideanConstruct(int, double[], double[])

EuclideanConstruct(int, EuclideanConstruct, EuclideanConstruct)

EuclideanConstruct(int, double[], EuclideanConstruct)

EuclideanConstruct(int, LinkedList<GeoConstruct>)

EuclideanConstruct(int, LinkedList<GeoConstruct>, double[])

intersect(int, EuclideanConstruct)

setXYZ(double[])

setXYZ(double[], double[])

setNewXYZ(double[])

setNewXYZ(double[], double[])

translate(double[], double[])

getScale()

setScale(double)

resetScale()

rescale(double[])

unscale(double[])

transform(GeoConstruct, double[], double[])

4.6 EuclideanPoint extends EuclideanConstruct

This is the specific class for Euclidean Geometry

4.6.1 declarations

4.6.2 methods

EuclideanPoint(int, double[])

EuclideanPoint(int, double[], double[])

EuclideanPoint(int, double[], EuclideanConstruct)

EuclideanPoint(int, EuclideanConstruct, EuclideanConstruct)

EuclideanPoint(int, LinkedList<GeoConstruct>, double[])

intersect(int, EuclideanConstruct)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.6.3 EuclideanLine extends EuclideanConstruct

This is the specific class for Euclidean Geometry

4.6.4 declarations

vec1 : double[]

vec2 : double[]

norm : double[]

binorm : double[]

4.6.5 methods

EuclideanLine(int, LinkedList<GeoConstruct>)

intersect(int, EuclideanConstruct)

intersect(int, EuclideanLine)

intersect(int, EuclideanCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.7 EuclideanCircle extends EuclideanConstruct

This is the specific class for Euclidean Geometry

4.7.1 declarations

4.7.2 methods

EuclideanCircle(int, EuclideanConstruct, EuclideanConstruct)

intersect(int, EuclideanConstruct)

intersect(int, EuclideanLine)

intersect(int, EuclideanCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

getCLxyz(EuclideanPoint, EuclideanConstruct, int)

getNewCLxyz(EuclideanPoint, EuclideanConstruct, int)

getCCxyz(EuclideanPoint, EuclideanConstruct, int)

getNewCCxyz(EuclideanPoint, EuclideanConstruct, int)

4.8 ToroidalConstruct extends GeoConstruct

4.8.1 declarations

scale : double

scaleLimit : int

4.8.2 methods

ToroidalConstruct()

ToroidalConstruct(int, double[])

ToroidalConstruct(int, double[], double[])

ToroidalConstruct(int, ToroidalConstruct, ToroidalConstruct)

ToroidalConstruct(int, double[], ToroidalConstruct)

ToroidalConstruct(int, LinkedList<GeoConstruct>)

ToroidalConstruct(int, LinkedList<GeoConstruct>, double[])

intersect(int, ToroidalConstruct)

setXYZ(double[])

setXYZ(double[], double[])

setNewXYZ(double[])

setNewXYZ(double[], double[])

translate(double[], double[])

getScale()

setScale(double)

resetScale()

rescale(double[])

unscale(double[])

transform(GeoConstruct, double[], double[])

4.9.1 declarations

4.9.2 methods

ToroidalPoint(int, double[])

ToroidalPoint(int, double[], double[])

ToroidalPoint(int, double[], ToroidalConstruct)

ToroidalPoint(int, ToroidalConstruct, ToroidalConstruct)

ToroidalPoint(int, LinkedList<GeoConstruct>, double[])

intersect(int, ToroidalConstruct)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.10 ToroidalLine extends ToroidalConstruct

This

4.10.1 declarations

vec1 : double[]

vec2 : double[]

norm : double[]

binorm : double[]

4.10.2 methods

ToroidalLine(int, LinkedList<GeoConstruct>)

intersect(int, ToroidalConstruct)

intersect(int, ToroidalLine)

intersect(int, ToroidalCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.11 ToroidalCircle extends ToroidalConstruct

This

declarations

4.11.1 methods

ToroidalCircle(int, ToroidalConstruct, ToroidalConstruct)

intersect(int, ToroidalConstruct)

intersect(int, ToroidalLine)

intersect(int, ToroidalCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

getCLxyz(ToroidalPoint, ToroidalConstruct, int)

getNewCLxyz(ToroidalPoint, ToroidalConstruct, int)

getCCxyz(ToroidalPoint, ToroidalConstruct, int)

getNewCCxyz(ToroidalPoint, ToroidalConstruct, int)

4.12 SphericalConstruct extends GeoConstruct

This

4.12.1 declarations

4.12.2 methods

SphericalConstruct()

SphericalConstruct(int, double[])

SphericalConstruct(int, double[], double[])

SphericalConstruct(int, SphericalConstruct, SphericalConstruct)

SphericalConstruct(int, double[], SphericalConstruct)

SphericalConstruct(int, LinkedList<GeoConstruct>)

SphericalConstruct(int, LinkedList<GeoConstruct>, double[])

intersect(int, SphericalConstruct)

setXYZ(double[])

setXYZ(double[], double[])

setNewXYZ(double[])

setNewXYZ(double[], double[])

translate(double[], double[])

transform(GeoConstruct, double[], double[])

4.13 SphericalPoint extends SphericalConstruct

This

4.13.1 declarations

4.13.2 methods

SphericalPoint(int, double[])

SphericalPoint(int, double[], double[])

SphericalPoint(int, double[], SphericalConstruct)

SphericalPoint(int, SphericalConstruct, SphericalConstruct)

SphericalPoint(int, LinkedList<GeoConstruct>, double[])

intersect(int, SphericalConstruct)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.14 SphericalLine extends SphericalConstruct

This

4.14.1 declarations

vec1 : double[]

vec2 : double[]

norm : double[]

binorm : double[]

4.14.2 methods

SphericalLine(int, LinkedList<GeoConstruct>)

intersect(int, SphericalConstruct)

intersect(int, SphericalLine)

intersect(int, SphericalCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.15 SphericalCircle extends SphericalConstruct

This

4.15.1 declarations

vec1 : double[]

vec2 : double[]

norm : double[]

binorm : double[]

4.15.2 methods

SphericalCircle(int, SphericalConstruct, SphericalConstruct)

intersect(int, SphericalConstruct)

intersect(int, SphericalLine)

intersect(int, SphericalCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

getCLxyz(SphericalPoint, SphericalConstruct, int)

getNewCLxyz(SphericalPoint, SphericalConstruct, int)

getCCxyz(SphericalPoint, SphericalConstruct, int)

getNewCCxyz(SphericalPoint, SphericalConstruct, int)

4.16 ProjectiveConstruct extends GeoConstruct

This

4.16.1 declarations

4.16.2 methods

ProjectiveConstruct()

ProjectiveConstruct(int, double[])

ProjectiveConstruct(int, double[], double[])

ProjectiveConstruct(int, ProjectiveConstruct, ProjectiveConstruct)

ProjectiveConstruct(int, double[], ProjectiveConstruct)

ProjectiveConstruct(int, LinkedList<GeoConstruct>)

ProjectiveConstruct(int, LinkedList<GeoConstruct>, double[])

intersect(int, ProjectiveConstruct)

setXYZ(double[])

setXYZ(double[], double[])

setNewXYZ(double[])

setNewXYZ(double[], double[])

translate(double[], double[])

transform(GeoConstruct, double[], double[])

4.17 ProjectivePoint extends ProjectiveConstruct

This

4.17.1 declarations

4.17.2 methods

ProjectivePoint(int, double[])

ProjectivePoint(int, double[], double[])

ProjectivePoint(int, double[], ProjectiveConstruct)

ProjectivePoint(int, ProjectiveConstruct, ProjectiveConstruct)

ProjectivePoint(int, LinkedList<GeoConstruct>, double[])

intersect(int, GeoConstruct)

intersect(int, ProjectiveConstruct)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.18 ProjectiveLine extends ProjectiveConstruct

This

4.18.1 declarations

vec1 : double[]

vec2 : double[]

norm : double[]

binorm : double[]

4.18.2 methods

ProjectiveLine(int, LinkedList<GeoConstruct>)

intersect(int, ProjectiveConstruct)

intersect(int, ProjectiveLine)

intersect(int, ProjectiveCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.19 ProjectiveCircle extends ProjectiveConstruct

This

4.19.1 declarations

vec1 : double[]

vec2 : double[]

norm : double[]

binorm : double[]

4.19.2 methods

ProjectiveCircle(int, ProjectiveConstruct, ProjectiveConstruct)

intersect(int, ProjectiveConstruct)

intersect(int, ProjectiveLine)

intersect(int, ProjectiveCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

getCLxyz(ProjectivePoint, ProjectiveConstruct, int)

getNewCLxyz(ProjectivePoint, ProjectiveConstruct, int)

getCCxyz(ProjectivePoint, ProjectiveConstruct, int)

getNewCCxyz(ProjectivePoint, ProjectiveConstruct, int)

4.20 HyperConstruct extends GeoConstruct

This

4.20.1 declarations

scale : double

scaleLimit : int

4.20.2 methods

HyperConstruct()

HyperConstruct(int, double[])

HyperConstruct(int, double[], double[])

HyperConstruct(int, HyperConstruct, HyperConstruct)

HyperConstruct(int, double[], HyperConstruct)

HyperConstruct(int, LinkedList<GeoConstruct>)

HyperConstruct(int, LinkedList<GeoConstruct>, double[])

intersect(int, HyperConstruct)

setXYZ(double[])

setXYZ(double[], double[])

setNewXYZ(double[])

setNewXYZ(double[], double[])

translate(double[], double[])

getScale()

setScale(double)

resetScale()

rescale(double[])

unscale(double[])

transform(GeoConstruct, double[], double[])

4.21 HyperPoint extends HyperConstruct

This

4.21.1 declarations

4.21.2 methods

HyperPoint(int, double[])

HyperPoint(int, double[], double[])

HyperPoint(int, double[], HyperConstruct)

HyperPoint(int, HyperConstruct, HyperConstruct)

HyperPoint(int, LinkedList<GeoConstruct>, double[])

intersect(int, HyperConstruct)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.22 HyperLine extends HyperConstruct

This

4.22.1 declarations

vec1 : double[]

u2 : double[]

norm : double[]

binorm : double[]

4.22.2 methods

HyperLine(int, LinkedList<GeoConstruct>)

intersect(int, HyperConstruct)

intersect(int, HyperLine)

intersect(int, HyperCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

4.23 HyperCircle extends HyperConstruct

This

4.23.1 declarations

4.23.2 methods

HyperCircle(int, HyperConstruct, HyperConstruct)

intersect(int, HyperConstruct)

intersect(int, HyperLine)

intersect(int, HyperCircle)

draw(Graphics, int, boolean)

mouseIsOver(double[], int)

update()

getCLxyz(HyperPoint, HyperConstruct, int)

getNewCLxyz(HyperPoint, HyperConstruct, int)

getCCxyz(HyperPoint, HyperConstruct, int)

getNewCCxyz(HyperPoint, HyperConstruct, int)