com._604robotics.robot2012.vision

Class Target

java.lang.Object
    com._604robotics.robot2012.vision.Target

All Implemented Interfaces:

java.lang.Comparable<Target>

```
public class Target
extends java.lang.Object
implements java.lang.Comparable<Target>
```

This class represents a physical vision Target with four main attributes (x, y, z, angle). As well, there are estimated uncertainties attached to all of these numbers.

To get the position of the hoop, use the DistanceCalculations class.

Field Summary

Fields

Modifier and Type	Field and Description
double	**angle** This is the angle of the target, relative to the camera.
double	**angleUncertainty** This is the uncertainty of the angle of the target.
static double	**RelHoopY** The distance from the center of the target to the Y (vertical) value of the hoop.
static double	**RelHoopZ** The distance from the center of the target to the Z (depth) value of the hoop.
double	**x** x, y, and z represent the 3-d position of the target x will be positive when the target appears to be right of the center of the camera.
double	**xUncertainty** These are the uncertainties of the x, y, and z positions of the target.
double	**y** x, y, and z represent the 3-d position of the target x will be positive when the target appears to be right of the center of the camera.
double	**yUncertainty** These are the uncertainties of the x, y, and z positions of the target.
double	**z** x, y, and z represent the 3-d position of the target x will be positive when the target appears to be right of the center of the camera.
double	**zUncertainty** These are the uncertainties of the x, y, and z positions of the target.

Constructor Summary

Constructors

Constructor and Description
Target() A blank constructor to easily make a Target
Target(double x, double y, double z, double angle)
Target(double x, double y, double z, double xUncertainty, double yUncertainty, double zUncertainty, double angle, double angleUncertainty)
Target(**Point3d** point, double angle)

Method Summary

Methods

Modifier and Type	Method and Description
int	**compareTo**(**Target** that)
double	**getAngle**()
double	**getAngleUncertainty**()
Point3d	**getHoopPosition**()
Point3d	**getReflectedHoopPosition**()
Point3d	**getReflectedHoopPosition**(double bounceFactor)
double	**getX**()
double	**getXUncertainty**()
double	**getY**()
double	**getYUncertainty**()
double	**getZ**()
double	**getZUncertainty**()
void	**setAngle**(double angle)
void	**setAngleUncertainty**(double angleUncertainty)
void	**setPoint**(**Point3d** point)
void	**setX**(double x)
void	**setXUncertainty**(double xUncertainty)
void	**setY**(double y)
void	**setYUncertainty**(double yUncertainty)
void	**setZ**(double z)
void	**setZUncertainty**(double zUncertainty)
java.lang.String	**toString**()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

RelHoopY

```
public static final double RelHoopY
```

The distance from the center of the target to the Y (vertical) value of the hoop.

See Also:

Constant Field Values

RelHoopZ

```
public static final double RelHoopZ
```

The distance from the center of the target to the Z (depth) value of the hoop.

See Also:

Constant Field Values

angle

```
public double angle
```

This is the angle of the target, relative to the camera.

(angle)
.....(Target)
...../
...../
...../
..../ - - - - - - - |> (Camera)
.../
../
./
/
this value is expressed in radians.

angleUncertainty

`public double angleUncertainty`

This is the uncertainty of the angle of the target. This is interpreted as a plus or minus to the angle. Again, this is expressed in radians

x

`public double x`

x, y, and z represent the 3-d position of the target x will be positive when the target appears to be right of the center of the camera. y will be positive when the target appears to be above of the center of the camera. z will always be negative (see Wikipedia: Right-hand rule). As the absolute value of z increases, so does the distance from the camera to the target. To determine the approximate accuracy of these values, check [x, y, z]_accuracy. The units of these measures are in inches.

y

`public double y`

x, y, and z represent the 3-d position of the target x will be positive when the target appears to be right of the center of the camera. y will be positive when the target appears to be above of the center of the camera. z will always be negative (see Wikipedia: Right-hand rule). As the absolute value of z increases, so does the distance from the camera to the target. To determine the approximate accuracy of these values, check [x, y, z]_accuracy. The units of these measures are in inches.

z

`public double z`

x, y, and z represent the 3-d position of the target x will be positive when the target appears to be right of the center of the camera. y will be positive when the target appears to be above of the center of the camera. z will always be negative (see Wikipedia: Right-hand rule). As the absolute value of z increases, so does the distance from the camera to the target. To determine the approximate accuracy of these values, check [x, y, z]_accuracy. The units of these measures are in inches.

xUncertainty

`public double xUncertainty`

These are the uncertainties of the x, y, and z positions of the target. These are interpreted as pluses and minuses to the x, y, and z values. Again, these are in inches.

yUncertainty

`public double yUncertainty`

These are the uncertainties of the x, y, and z positions of the target. These are interpreted as pluses and minuses to the x, y, and z values. Again, these are in inches.

zUncertainty

`public double zUncertainty`

These are the uncertainties of the x, y, and z positions of the target. These are interpreted as pluses and minuses to the x, y, and z values. Again, these are in inches.

Constructor Detail

Target

`public Target()`

A blank constructor to easily make a Target

Target

```
public Target(double x,
      double y,
      double z,
      double angle)
```

Parameters:

`x` - the X coordinate of the center of the vision target

`y` - the Y coordinate of the center of the vision target

`z` - the Z coordinate of the center of the vision target

`angle` -

Target

```
public Target(double x,
      double y,
      double z,
      double xUncertainty,
      double yUncertainty,
      double zUncertainty,
      double angle,
      double angleUncertainty)
```

Parameters:

`x` - the X coordinate of the center of the vision target

`y` - the Y coordinate of the center of the vision target

`z` - the Z coordinate of the center of the vision target

`xUncertainty` - the X Uncertainty

`yUncertainty` - the Y Uncertainty

`zUncertainty` - the Z Uncertainty

`angle` - the Angle

`angleUncertainty` - the Angle Uncertainty

Target

```
public Target(Point3d point,
      double angle)
```

Parameters:

`point` - the Point

`angle` - the Angle

Method Detail

compareTo

```
public int compareTo(Target that)
```

Specified by:

`compareTo` in interface `java.lang.Comparable<Target>`

getAngle

```
public double getAngle()
```

Returns:

the angle that the vision target faces

getAngleUncertainty

```
public double getAngleUncertainty()
```

Returns:

the uncertainty of the Angle

getHoopPosition

```
public Point3d getHoopPosition()
```

Returns:

the position of the hoop accounting for the fact that the center of the hoop is not at the center of the target

getReflectedHoopPosition

```
public Point3d getReflectedHoopPosition()
```

Returns:

the reflected position of the hoop accounting for the fact that the center of the hoop is not at the center of the target. This is useful bounces

getReflectedHoopPosition

```
public Point3d getReflectedHoopPosition(double bounceFactor)
```

Parameters:

`bounceFactor` - a number that scales the changes in the x and z distances due to correction for hoop position. In a idealized collision, this is equal to the inverse of its coefficient of restitution. However, with spin, this number should be less.

Returns:

the reflected position of the hoop accounting for the fact that the center of the hoop is not at the center of the target. This is useful bounces

getX

```
public double getX()
```

Returns:

the X coordinate of the center of the vision target

getXUncertainty

```
public double getXUncertainty()
```

Returns:

the Uncertainty of the X coordinate

getY

```
public double getY()
```

Returns:

the Y coordinate of the center of the vision target

getYUncertainty

```
public double getYUncertainty()
```

Returns:

the Uncertainty of the Y coordinate

getZ

```
public double getZ()
```

Returns:

the Z coordinate of the center of the vision target

getZUncertainty

```
public double getZUncertainty()
```

Returns:

the Uncertainty of the Z coordinate of the vision target

setAngle

```
public void setAngle(double angle)
```

Parameters:

`angle` - the Angle to set

setAngleUncertainty

```
public void setAngleUncertainty(double angleUncertainty)
```

Parameters:

`angleUncertainty` - the angleUncertainty to set

setPoint

```
public void setPoint(Point3d point)
```

Parameters:

`point` - the point to set the center of this target

setX

```
public void setX(double x)
```

Parameters:

`x` - the X to set

setXUncertainty

```
public void setXUncertainty(double xUncertainty)
```

Parameters:

`xUncertainty` - the xUncertainty to set

setY

```
public void setY(double y)
```

Parameters:

`y` - the Y to set

setYUncertainty

```
public void setYUncertainty(double yUncertainty)
```

Parameters:

`yUncertainty` - the yUncertainty to set

setZ

```
public void setZ(double z)
```

Parameters:

`z` - the Z to set

setZUncertainty

```
public void setZUncertainty(double zUncertainty)
```

Parameters:

`zUncertainty` - the zUncertainty to set

toString

```
public java.lang.String toString()
```

public java.lang.String toString()

Overrides:

`toString` in class `java.lang.Object`