```
//Steer.cpp : contains definition of Steer class

#include "Steer.h"
#include "math.h"
#include <iostream>

using namespace std;

Steer::Steer(double w, double x, double y, double a):
pi(3.14159), A(a), W(w), X(x), Y(y), thetaRC(0.0), FRRatio(0.0), FLRatio(0.0),
RRRatio(0.0), RLRatio(0.0)
{
	if(A > 1)
		A = 1;
	if(A < 0)
		A = 0;

}
void Steer::Calc4WheelTurn(double radian)
{
	thetaRC = pi - radian;  //convert steering angle to rear center wheel angle

	if(thetaRC != pi / 2)	//If we are not driving straight forward...
	{
		if(thetaRC < pi / 2)	//Right Turn
```

```
            {
                this->RightTurn4Wheels();
            }
            else if(thetaRC > pi / 2)  //Left Turn
            {
                this->LeftTurn4Wheels();
            }
        }
        else      //thetaRC = pi / 2
        {
            thetaFL = pi / 2;
            thetaFR = pi / 2;
            thetaRL = pi / 2;
            thetaRR = pi / 2;

            FLRatio = 1;
            FRRatio = 1;
            RLRatio = 1;
            RRRatio = 1;
        }
        //Solve for fastest wheel speed
        double speedarray[] = {fabs(FL), fabs(FR), fabs(RL), fabs(RR)};

         int length = 4;
    double maxspeed = speedarray[0];
```

```
    for(int i = 1; i < length; i++)
    {
        if(speedarray[i] > maxspeed)
            maxspeed = speedarray[i];
    }

            //Set ratios based on maximum wheel speed
            FLRatio = FL/maxspeed;
            FRRatio = FR/maxspeed;
            RLRatio = RL/maxspeed;
            RRRatio = RR/maxspeed;
}
void Steer::LeftTurn4Wheels()
{
            Z = ((A * X) * tan(pi - thetaRC));  //solve for robot turn raidus

            //solve for wheel angles
            thetaRL = pi - atan((Z - W) / (A * X));
            thetaRR = pi - atan((Z + W) / (A * X));
            thetaFR = pi / 2;  //default if A=1, front wheels drive straight forward
            thetaFL = pi / 2;  //default if A=1, front wheels drive straight forward

            if(A != 1)
            {
```

```
            thetaFL = atan((Z - Y) / ((1 - A) * X));        //These are identical for right and left turns
            thetaFR = atan((Z + Y) / ((1 - A) * X));        //These are identical for right and left turns
        }

        //Solve for turning radii (wheel speed)
        FL = (Z - Y) / sin(thetaFL);
        FR = (Z + Y) / sin(thetaFR);
        RL = (Z - W) / sin(pi - thetaRL);
        RR = (Z + W) / sin(pi - thetaRR);
}
void Steer::RightTurn4Wheels()
{
        Z = ((A * X) * tan(thetaRC));  //solve for robot turn raidus

        thetaRL = atan((Z + W) / (A * X));
        thetaRR = atan((Z - W) / (A * X));
        thetaFR = pi / 2;  //default if A=1, front wheels drive straight forward
        thetaFL = pi / 2;  //default if A=1, front wheels drive straight forward

        if(A != 1)
        {
            thetaFR = pi - atan((Z - Y) / ((1 - A) * X)); //These are identical for right and left turns
            thetaFL = pi - atan((Z + Y) / ((1 - A) * X)); //These are identical for right and left turns
        }
```

```
	//Solve for turning radii (wheel speed)
	FL = (Z + Y) / sin(pi - thetaFL);
	FR = (Z - Y) / sin(pi - thetaFR);
	RL = (Z + W) / sin(thetaRL);
	RR = (Z - W) / sin(thetaRR);
}

void Steer::SetA(double a)
{
	if(a > 1)
		a = 1;
	if(a < 0)
		a = 0;

	A = a;
}
const double Steer::GetFRRatio()
{
	return FRRatio;
}
const double Steer::GetFLRatio()
{
	return FLRatio;
}
const double Steer::GetRRRatio()
```

```
{
	return RRRatio;
}
const double Steer::GetRLRatio()
{
	return RLRatio;
}
const double Steer::GetThetaFL()
{
	return thetaFL;
}
const double Steer::GetThetaFR()
{
	return thetaFR;
}
const double Steer::GetThetaRC()
{
	return thetaRC;
}
const double Steer::GetThetaRL()
{
	return thetaRL;
}
const double Steer::GetThetaRR()
{
```

```
        return thetaRR;

}
```