Rapid-I vision measuring System Project

Internship activity - <Day 30>

Author:Yeswanthram R

**Add s circle to the Rcad :**

void RCadApp::AddCircleToCurrentPCD(int id)

{

try

{

if(CurrentPCDShape == NULL) return;

((PCDCircle\*)CurrentPCDShape)->AddParentShapes(getShapesList().getList()[id]);

}

catch(...){ MAINDllOBJECT->SetAndRaiseErrorMessage("RCAD1226", \_\_FILE\_\_, \_\_FUNCSIG\_\_); }

}

**draw the current arc and Selected Points..**

void ArcHandler::draw(int windowno, double WPixelWidth)

{

try

{

drawFlexibleCh();

if(valid)

{

GRAFIX->SetGraphicsLineWidth(ShapeThickness);

if(getClicksDone() == 1)

GRAFIX->drawLine(getClicksValue(0).getX(), getClicksValue(0).getY(), getClicksValue(1).getX(), getClicksValue(1).getY());

else if(getClicksDone() > 1 && getClicksDone() < getMaxClicks())

GRAFIX->drawArc(center[0], center[1], radius, startangle, sweepangle, WPixelWidth);

GRAFIX->SetGraphicsLineWidth(1);

}

}

catch(...){ MAINDllOBJECT->SetAndRaiseErrorMessage("ARCMH0010", \_\_FILE\_\_, \_\_FUNCSIG\_\_); }

}

**Function to calculate the angle between two lines**

double RapidMath2D::Line2Line\_angle(double L1slope, double intercept1, double L2slope, double intercept2, double \*mousept)

{

try

{

double pt1[2], pt2[2], temp;

double angle = 0;

//take the slope range 0 t0 180//

Angle\_FirstScndQuad(&L1slope); Angle\_FirstScndQuad(&L2slope);

//If the two lines are parallel then return false//

if(IsEqual(L1slope, L2slope, MeasureDecValue)) return 0;

//new points on the line with repect to mouse move//

//X value remains same , y1 = m1\*x + c1 and y2 = m2\*x+c2//

if(L1slope == M\_PI\_2)

{

pt1[0] = intercept1;

pt1[1] = mousept[1];

temp = pt1[0];

}

else

{

pt1[0] = mousept[0];

pt1[1] = ((tan(L1slope)) \* mousept[0] + intercept1);

temp = pt1[1] ;

}

if(L2slope == M\_PI\_2)

{

pt2[0] = intercept2;

pt2[1] = mousept[1];

temp = pt2[1];

}

else

{

pt2[0] = mousept[0];

pt2[1] = ((tan(L2slope)) \* mousept[0] + intercept2);

temp = pt2[1];

}

**find the circle radius and center passing throu three points**

try

{

double m, n, o, p, q, r, s;

m = pow(point1[0], 2) + pow(point1[1], 2);

n = pow(point2[0], 2) + pow(point2[1], 2);

o = pow(point3[0], 2) + pow(point3[1], 2);

p = point3[0] - point2[0];

q = point2[0] - point1[0];

r = point3[1] - point2[1];

s = point2[1] - point1[1];

//Calculating the X and Y co-ods of the centre of circle

if((p \* s - q \* r)!= 0)

{

center[0] = -((n - o) \* s - (m - n) \* r) / (2 \* ((p \* s) - (q \* r)));

center[1] = -((n - o) \* q - (m - n) \* p) / (2 \* ((q \* r) - (p \* s)));

\*radius = Pt2Pt\_distance(center, point1);

return true;

}

return false;

}

**calculate the linear distance between two circles**

double RapidMath2D::LCircle2Circle\_distance\_Type(double \*center1, double \*center2, double radius1, double radius2, int type)

{

try

{

double distance = 0;

if(type == 0) //circumference to circumference distance (min)//

distance = abs(abs(center1[0] - center2[0]) - (radius1 + radius2));

else if(type == 1) //center to center distance//

distance = abs(center1[0] - center2[0]);

if(type == 2) //circumference to circumference distance (max)

distance = abs(center1[0] - center2[0]) + radius1 + radius2;

else if(type == 3) //circumference to circumference distance (min)//

distance = abs(abs(center1[1] - center2[1]) - (radius1 + radius2));

else if(type == 4) //center to center distance//

distance = abs(center1[1] - center2[1]);

else if(type == 5) //circumference to circumference distance (max)

distance = abs(center1[1] - center2[1]) + radius1 + radius2;

return distance;

}

catch(...){ SetAndRaiseErrorMessage("RMATH2D0155", \_\_FILE\_\_, \_\_FUNCSIG\_\_); return 0; }

}







