

CAD Functions

AutoCad Script Functions	
crf	Create AutoCad Script File
scrlnr	Write Line/Rectangle to AutoCad Script File
screir	Write Circle to AutoCad Script File
scararc	Write Arc to AutoCad Script File
Geometric Functions	
ptic	Determine if Point is in Circle
inter	Determine if circles intersect
enccr	Determine if circles are enclosed
quad	Determine quadrant(1-4) of point
arctan	Angle(radians) of line w/ +ive X-axis
dist	Distance between 2 points
slope	Slope of Line
yntr	Y-Intercept of Line
dolfo	Perpendicular Dist Of Line From Origin
rad	Angle in Radians
horz	Determine if Line is horizontal
vert	Determine if Line is vertical
eltd	Extend both ends of a line thru distance
cil	Determine if circle Intersects line
cip	Determine intersection points of Circle w/ Line
ptoc	Make point to circle tangents
ctoc	Make circle to circle(parallel/cross) tangents
blunt	Join the tangent points of two vectors to a circle w/ an Arc.

Common Function Inputs

p1 End point of Line. For a vector(line w/ angle wrt x-axis) it is the 'start'(tail) point.
p2 End point of Line. For a vector(line w/ angle wrt x-axis) it is the 'end' (head) point.
p Point(x and y co-ordinates in a POINTF or POINTI structure)
c1 Circle #1(center point and radius in a CIRCF structure)
c2 Circle #2(center point and radius in a CIRCF structure)
c Circle(center point and radius in a CIRCF structure)
clr Color string eg "RED", "GREEN", "BLUE", "MAGENTA", "BLACK", "WHITE" etc or the corresponding AutoCad number eg "251" etc.
hnd Handle of AutoCad script file created by crf().

Data Structures Used

```
struct POINTF{                               //(x,y) co-ordinates
    double x;
    double y;
};

struct POINTI{                               //(x,y) co-ordinates
    DWORD x;
    DWORD y;
};

struct CIPS{                                 //intersection points of line w/ circle
    POINTF i[2];                             //intersection point,actual[0],virtual[1]
    double d[2];                             //Dist->End of line to Actual[0],Virtual[1] Intersection
};

struct CIRCF{                                //circle
    POINTF c;                                 //center point
    double r;                                 //radius
};

struct PNTF{                                 //Structure for ApnDyn() for POINTF data
    POINTF *pnt;                              //Array of POINTF data types
    DWORD udp;                                //# of used elements in dynamic array pnt[]
    DWORD adp;                                //# of actual elements in dynamic array pnt[]
};
```

Geometric Functions

ptic

DWORD ptic(CIRCF c,POINTF p)

Description

If point(p) is inside or on the perimeter of circle(c), return 1, else return 0

intcr

DWORD intcr(CIRCF c1,CIRCF c2)

Description

If circles($c1$) and ($c2$) intersect return 1, else return 0

enccr

DWORD enccr(CIRCF c1,CIRCF c2)

Description

If circle($c1$) is enclosed in or coincident w/ circle ($c2$) or visa-versa return 1, else return 0

quad

DWORD quad(POINTF p)

Description

Return the quadrant(1-4) of point(p)

arctan

double arctan(POINTF p1,POINTF p2)

Description

Return the angle(in radians) made by vector($p1,p2$) with +ive X-axis

dist

double dist(POINTI p1,POINTI p2)
double dist(POINTF p1,POINTF p2)

Description

Return the distance between 2 points

slope

double slope(POINTF p1,POINTF p2)

Description

Return the slope of line, with endpoints ($p1,p2$).

yntr

double yntr(POINTF p1,POINTF p2)

Description

Return the Y-Intercept of line, with endpoints $(p1,p2)$.

dolfo

double dolfo(POINTF p1,POINTF p2)

Description

Return the Perpendicular Dist Of Line of line, with endpoints $(p1,p2)$, from origin.

rad

double rad (double deg)

Description

Convert the input angle, in radians, to degrees. Return the degree value

horz

DWORD horz (POINTF p1,POINTF p2)

Description

If line, with endpoints $(p1,p2)$, is horizontal(parallel to x-axis) return 1 else return 0.

vert

DWORD vert (POINTF p1,POINTF p2)

Description

If line, with endpoints $(p1,p2)$, is vertical(parallel to y-axis), return 1 else return 0.

cil

DWORD cil(POINTF p1,POINTF p2,CIRCF c)

Description

If circle(c) intersects Line, with endpoints $(p1,p2)$, return 1 else return 0.

cip

CIPS cip(POINTF p1,POINTF p2,CIRCF c)

Description

Determine intersection points of Line w/ endpoints($p1,p2$) with Circle(c).

Returns

Details of intersection points in a variable of type CIPS. The members of this variable contain: -
i[0].x, i[0].y : Co-ordinates of 'nearest' intersection point
d[0] : Distance between the $p2$ point of line and 'nearest' intersection point
i[1].x, i[1].y : Co-ordinates of 'farthest' intersection point
d[1] : Distance between the $p2$ point of line and 'farthest' intersection point

Note

'nearest' intersection point is the intersecting point on the circle nearest to the $p1$ point of line
'farthest' intersection point is the intersecting point on the circle farthest from the $p1$ point of line

eltd

void eltd(POINTF p1,POINTF p2,double d,POINTF *el)

Description

Extend the endpoints($p1,p2$), of a line outward through distance(d).

Returns

New endpoints in an array of points $el[]$: -
 $el[0]$: co-ords of the $p1$ point of the line extended outward through distance 'd'
 $el[1]$: co-ords of the $p2$ point of the line extended outward through distance 'd'

ptoc

void ptoc(POINTF p, CIRCF c,POINTF *tng)

Description

Calculate tangent points on circle(c,r) for tangents from point(p) to it. Use ptic() to ensure that the point is not enclosed in and not on the perimeter of the circle

Parameters

tng -Address of result array of two POINTF elements.

Result

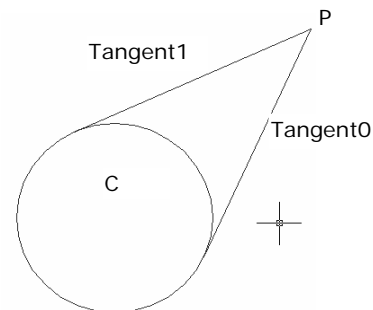
$tng[0]$ -End point of Tangent0 on the circle
 $tng[1]$ -End point of Tangent1 on the circle

Example

```
FILE *hnd=NULL;
POINTF p,tng[2];
CIRCF c;
```

```
p.x=15.0;    p.y=15.0;
c.c.x=5.0;   c.c.y=5.0;   c.r=5.0;
ptoc(p,c,tng);
```

```
hnd=crf(3,"d:\\Robocup ip\\sample",0);
scrcir(c,hnd,"251");
sclnr(p,tng[0],hnd,"251",0);
sclnr(p,tng[1],hnd,"251",0);
```



```
//Point co-ordinates
//Circle center co-ordinates & radius
//Calc tangent points on Circle
```

```
//Create AutoCad Script file
//Circle: (c) ->AutoCad script
//Line: p to tng[0] ->AutoCad script
//Line: p to tng[1] ->AutoCad script
```

Viewing the Output in AutoCad

Use procedure in sclnr() to run "sample3.scr" from "d:\\Robocup ip" folder, in AutoCad.

ctoc

DWORD ctoc(CIRCF c1,CIRCF c2, POINTF *tng, double *opn)

Description

Calculate source circle(*c1*) to destination circle(*c2*) parallel or cross tangent points.

Parameters

tng -Address of an array of four POINTF elements, in which tangent endpoints are returned.

opn -Make Parallel Tangents(+1).

-Make Cross Tangents(-1).

Result

tng[0] -Tangent0 start point

tng[1] -Tangent0 end point

tng[2] -Tangent1 start point

tng[3] -Tangent1 end point

Returns

1 -Valid Tangents exist. *tng[]* contains results

0 -Valid Tangents don't exist because circles are enclosed or they intersect and *opn*=+1. *tng[]* is unchanged.

Example

```
FILE *hnd=NULL;
```

```
POINTF tng[4];
```

```
CIRCF c1,c2;
```

```
c2.c.x=15.0; c2.c.y=15.0; c2.r=5.0;
```

```
c1.c.x=5.0; c1.c.y=5.0; c1.r=5.0;
```

```
hnd=crf(4,"d:\\Robocup ip\\sample",0);
```

```
scrcir(c1,hnd,"251");
```

```
scrcir(c2,hnd,"251");
```

```
ctoc(c1,c2,tng,-1);
```

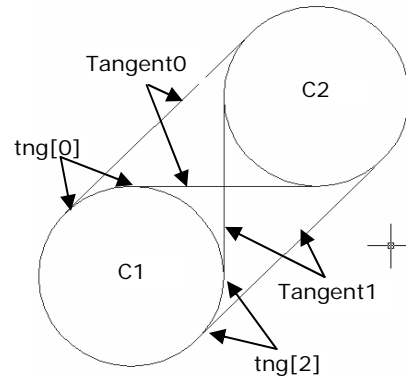
```
sclnr(tng[0],tng[1],hnd,"251",0);
```

```
sclnr(tng[2],tng[3],hnd,"251",0);
```

```
ctoc(c1,c2,tng,+1);
```

```
sclnr(tng[0],tng[1],hnd,"251",0);
```

```
sclnr(tng[2],tng[3],hnd,"251",0);
```



```
//Circle #1 center co-ordinates & radius
```

```
//Circle #2 center co-ordinates & radius
```

```
//Create AutoCad Script file
```

```
//Circle: (c1)->AutoCad script
```

```
//Circle: (c2)->AutoCad script
```

```
//Calc Cross Tangent points
```

```
//Line: tng[0] to tng[1]->AutoCad script
```

```
//Line: tng[2] to tng[3]->AutoCad script
```

```
//Calc Parallel points
```

```
//Line: tng[0] to tng[1]->AutoCad script
```

```
//Line: tng[2] to tng[3]->AutoCad script
```

Viewing the Output in AutoCad

Use procedure in sclnr() to run "sample4.scr" from "d:\\Robocup ip" folder, in AutoCad.

blunt

```
void blunt(POINTF p11,POINTF p12,POINTF p21,POINTF p22,CIRCF cf,
          PNTF *bln, double angl , DWORD f, POINTF p)
```

Description

Join the tangent points of two vectors to a circle w/ an Arc. The arc is made of line segments connecting the tangent point of vector1 to the tangent point of vector2. The number of segments in the Arc is determined by 'angl'. The approach of the vectors to the circle must be with their heads and convergent or parallel. Arc segment endpoints are stored in a dynamic array in the *bln* input.

Parameters

p11, p12 -Start, Tangent point point of vector1.
p21, p22 -Start, Tangent point point of vector2.
cf -Circle on which the i/p tangents converge.
bln -Address of result holder variable.
angl - $\cos(\text{Ang}/2)$. Angle between line segments of the arc will be more than specified by **Ang**.
f -For internal use, must always be 1.
p -The start & end points of the Arc aren't calculated. Start point(*p*), any point on vector1, is inserted in the 1st element of the result array as specified. The end point may be inserted externally.

Result

Stored in members of the structure variable pointed to by *bln*:-

pnt[] : Dynamic array of Arc segment endpoints.

Must be deleted in the calling program

cnt : Number of elements written to *pnt[]* by current call.

adp : Element count in *pnt[]*

Example

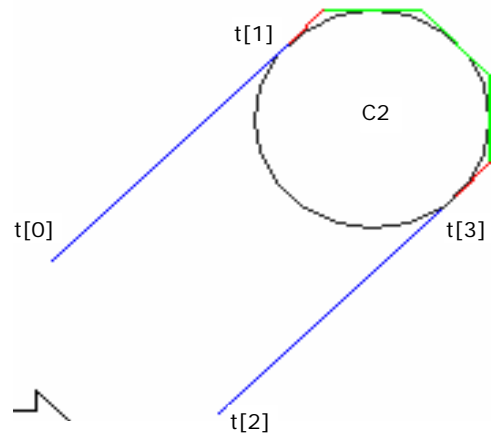
```
FILE *hnd=NULL;
POINTF t[4];
double ang;
PNTF bln={0};
DWORD i;
char *col;
CIRCF c1,c2;
```

```
c1.c.x=5.0; c1.c.y=5.0; c1.r=5.0;
c2.c.x=15.0; c2.c.y=15.0; c2.r=5.0;
```

```
hnd=crf(5,"d:\\Robocup ip\\sample",0);
ctoc(c1,c2,t,+1);
```

```
ang=90;
blunt(t[0],t[1],t[2],t[3],c2,&bln,cos(ang*PI/360),1,t[1]);
bln.pnt=ApnDyn(&bln.adp,&bln.udp,bln.pnt,t[3]);
```

```
scrlnr(t[0],t[1],hnd,"BLUE",0);
scrlnr(t[2],t[3],hnd,"BLUE",0);
scrcir(c2,hnd,"251");
for(i=0;i<bln.udp-1;i++){
    col=(i==0 || i==bln.udp-2 ? "RED" : "GREEN");
    scrlnr(bl.pnt[i], bln.pnt[i+1],hnd,col,0);
}
delete []bln.pnt;
```



```
//Circle #1 center co-ordinates & radius
//Circle #2 center co-ordinates & radius
```

```
//Create AutoCad Script file
//Calc Parallel Tangent points
```

```
//desired min angle between arc segs
//Arc for tangents converging on c1
//store end point of Arc
```

```
//Line from t[0] to t[1]
//Line from t[2] to t[3]
//Circle: (c1)->AutoCad script
```

```
//Arc segment color
//Arc segments
```

```
//delete dynamic array
```

Viewing the Output in AutoCad

Use procedure in *scrlnr()* to run "sample5.scr" from "d:\Robocup ip" folder, in AutoCad.

AutoCad Script Functions

crf

FILE *crf(DWORD no,char *nm,DWORD p)

Description

Create AutoCad Script File.

Parameters

no -File number(Suffixed to file name)
nm -File name & path string-eg "d:\\dir_name\\sub_dir1\\sub_dir2\\name_without_dot_and_suffix"
p -0-*nm*[] not dynamic or dynamic but not deleted.
1-*nm*[] is dynamic and deleted in function.

Returns

Creates a file with name specified and at the path specified with a ".scr" suffix.

If successfully created then a handle to the file otherwise zero.

Example

```
FILE *hnd=NULL;  
hnd=crf(0,"d:\\Robocup ip\\sample",0); //Create AutoCad Script file
```

scrarc

void scrarc(POINTF p1,POINTF p2,POINTF p3,FILE *hnd,char *clr)

Description

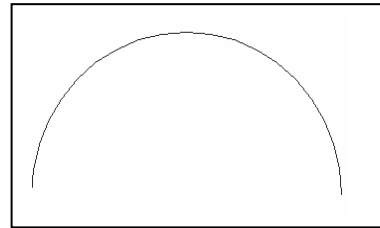
Write Arc to AutoCad Script File.

Parameters

p1,p2,p3 -Start, mid and end points of Arc.

Example

```
FILE *hnd=NULL;  
POINTF p1,p2,p3;  
  
p1.x=0.0; p1.y=0.0;  
p2.x=2.5; p2.y=2.5;  
p3.x=5.0; p3.y=0.0;  
hnd=crf(1,"d:\\Robocup ip\\sample",0);  
scrarc(p1,p2,p3,hnd,"251"); //Create AutoCad Script file
```



Viewing the Output in AutoCad

Use procedure in sclnr() to run "sample1.scr" from "d:\\Robocup ip" folder, in AutoCad.

scrcir

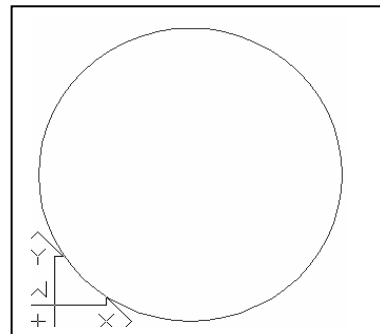
void scrcir(CIRCF c, FILE *hnd,char *clr)

Description

Write Circle to AutoCad Script File

Example

```
FILE *hnd=NULL;  
CIRCF c;  
  
c.c.x=5.0; c.c.y=5.0; c.r=5.0;  
hnd=crf(2,"d:\\Robocup ip\\sample",0);  
scrcir(c,hnd,"251"); //Circle center and radius  
//Create AutoCad Script file  
//Circle(c)
```



Viewing the Output in AutoCad

Use procedure in sclnr() to run "sample2.scr" from "d:\\Robocup ip" folder, in AutoCad.

scriNr

void scriNr(POINTF p1,POINTF p2,FILE *hnd,char *clr,DWORD opn)

Description

Write Line/Rectangle to AutoCad Script File.

Parameters

opn -Line(0),Rectangle(1).

Example

FILE *hnd=NULL;

POINTF p1,p2;

p1.x=0.0; p1.y=0.0;

p2.x=5.0; p2.y=5.0;

hnd=crf(0,"d:\\Robocup ip\\sample",0);

scriNr(p1,p2,hnd,"251",0);

scriNr(p1,p2,hnd,"251",1);

//Create AutoCad Script file

//Line from *p1* to *p2*

//Rectangle from *p1* to *p2*

Viewing the Output in AutoCad

In AutoCad enter "script" in the command window & select "sample0.scr" from "d:\\Robocup ip" folder.

