programma ZR di esempio per **traiettoria circolare**

Prof.ssa Paola Grandillo

per far compiere una traiettoria circolare **sul piano z=0**
allo spheres BLU che parte da (0, 0.5, 0) ; durata 70secondi

```
//Controllo distanza
ZRState myState;
float v[3];

//Circonferenza
float raggio, omega, alpha, cosalpha, sinalpha, dist;
float w[3], c[3];
int currentPoint;
char vicino;

float points[4][3];

void init(){

//---  Punto A - point=0
    points[0][0] = 0.25;
    points[0][1] = 0.25;
    points[0][2] = 0.0;

//--- Punto B - point=1
    points[1][0] = 0.0;
    points[1][1] = 0.0;
    points[1][2] = 0.0;

//--- Punto C - point=2
    points[2][0] =-0.25;
    points[2][1] = 0.25;
    points[2][2] = 0.0;

//--- Punto D - point=3
    points[3][0] = 0.0;
    points[3][1] = 0.5;
    points[3][2] = 0.0;
```

```
//---  CIRCONFERENZA DA 'A' fino a 'D'
//---  x^2+y^2-0.5y=0
//---  raggio = 0.25
//---  centro(0.25 , 0.25)
    c[0] = 0.0;
    c[1] = 0.25;
    c[2] = 0.0;
    raggio = 0.25f;
    alpha = 0.15; // un decimo di Pgreco/2 radianti = circa 9 gradi
    omega = 0.15;
    cosalpha = cosf(alpha);
    sinalpha = -sinf(alpha);
    currentPoint=0;
    vicino='N';
}  //---  fine di init

void loop(){
       circumference2D();
}
```

```
//**************************************************************************
//Funzioni di servizio

void mathVecMultipl(float *v, float *a, float k, int lenght)
{
    for(int i= 0; i<lenght ; i++)
        v[i] = a[i]*k;
}

float distanceToPoint(int point)
{
    api.getMyZRState(myState);
    mathVecSubtract(v,points[point],myState,3);
    return mathVecMagnitude(v,3);
}
```

```
//*********************************************************************************
//Funzione Circonferenza

void circumference2D (){
    api.getMyZRState(myState);
    v[0] = myState[0];
    v[1] = myState[1];
    v[2] = myState[2] = 0.0;

    mathVecSubtract(v,v,c,3);       //--- Traslazione dal centro
    mathVecNormalize(v,3);

    w[0] = c[0] + raggio * (v[0] * cosalpha - v[1] * sinalpha);
    w[1] = c[1] + raggio * (v[0] * sinalpha + v[1] * cosalpha);
    w[2] = 0.0;

    mathVecSubtract(v,w,myState,3);
    mathVecNormalize(v,3);
    mathVecMultipl(v,v,omega*raggio,3) ;
    api.setVelocityTarget(v);

    dist = distanceToPoint(currentPoint);
    if(dist >= 0.01 && dist < 0.05){
        DEBUG(("point=%d dist=%5.3f x=%5.3f y=%5.3f",
                  currentPoint, dist, myState[0], myState[1]));
        vicino = 'S';
    }
    else
    if (vicino == 'S'){
        vicino = 'N';
        currentPoint = (currentPoint+1)%4;
    }
}
```

DEBUG a console :

```
Sphere 1, 14.0s, DBG: point=0 dist=0.035 x= 0.264 y= 0.283
Sphere 1, 15.0s, DBG: point=0 dist=0.017 x= 0.267 y= 0.250
Sphere 1, 16.0s, DBG: point=0 dist=0.031 x= 0.262 y= 0.221
Sphere 1, 27.0s, DBG: point=1 dist=0.035 x= 0.033 y=-0.011
Sphere 1, 28.0s, DBG: point=1 dist=0.012 x= 0.002 y=-0.012
Sphere 1, 29.0s, DBG: point=1 dist=0.030 x=-0.029 y=-0.010
Sphere 1, 40.0s, DBG: point=2 dist=0.030 x=-0.264 y= 0.223
Sphere 1, 41.0s, DBG: point=2 dist=0.015 x=-0.264 y= 0.255
Sphere 1, 42.0s, DBG: point=2 dist=0.039 x=-0.262 y= 0.287
Sphere 1, 53.0s, DBG: point=3 dist=0.029 x=-0.026 y= 0.513
Sphere 1, 54.0s, DBG: point=3 dist=0.015 x= 0.007 y= 0.513
Sphere 1, 55.0s, DBG: point=3 dist=0.039 x= 0.038 y= 0.508
```

http://static.zerorobotics.mit.edu/docs/tutorials/PolarCoordinates.pdf