Condizioni iniziali e specifiche della gara:
Le simulazioni dovranno essere fatte in FreeMode.
La gara dovrà iniziare con lo SPHERES blu posizionato inizialmente nel punto X=0, Y=-0.5m, Z=0, e lo SPHERES rosso posizionato inizialmente nel punto X=0, Y=0.5m, Z=0.
I due satelliti dovranno muoversi ALL'INTERNO DELLA STESSA SIMULAZIONE lungo una traiettoria che tocchi i punti:

Punto	SPHERES blu	SPHERES rosso
A	(-0.5m, -0.5m, 0)	(0.5m, 0.5m, 0)
B	(-0.5m, 0, 0)	(0.5m, 0, 0)
C	(0, 0, -0.5m) Ruotando (in prossimità di C) su se stesso con una velocità angolare di 1 rad/s attorno ad un asse dello SPHERES parallelo all'asse Z (del sistema di riferimento fisso)	(0, 0, 0.5m) Ruotando (in prossimità di C) su se stesso con una velocità angolare di 1 rad/s attorno ad un asse dello SPHERES parallelo all'asse Z (del sistema di riferimento fisso)

Si considera terminata la traiettoria nel momento esatto in cui la distanza dal punto di arrivo (C) di almeno uno dei due SPERES scende al di sotto di 1cm E la velocità angolare sia +/- 0.05 rad/s dal valore atteso. Nel caso questo non accada durante tutta la simulazione, quando la traiettoria di almeno uno dei due SPERES raggiunge la distanza minima dal punto C.

Verranno assegnati:

- 10 punti ad ognuno;
- 0.1 punto in meno per ogni 1s di maggior tempo rispetto al minore assoluto;
- 0.1 punto in meno per ogni 1s di differenza fra i tempi in cui ciascuno dei due SPHERES termina la propria traiettoria;
- 0.2 punti in meno per ogni 1cm di distanza minima da ciascuno dei tre punti indicati (A, B, C), misurate entro 5 secondi oltre il termine della traiettoria.
- 0.2 punti in meno per ogni 0.01 rad/s di minima differenza rispetto alla velocità angolare attesa, misurati entro 5 secondi oltre il termine della traiettoria (ciò significa che se la velocità angolare si avvicina ulteriormente al valore nominale nei 5 secondi successivi al termine della traiettoria, si considera il valore più vicino raggiunto nei 5 secondi successivi).
- 1 punto in meno per ciascuna situazione anomala macroscopica (ad es. il movimento dello SPHERES sbagliato; traiettoria corretta ma sul piano sbagliato; scambio dei colori; simulazione interrotta prima di arrivare in C, ecc.)

BLU	RED
Partenza: (0 , -0.5, 0)	Partenza: (0 , +0.5, 0)
Punto **A** : (-0.5 , -0.5, 0)	Punto **A** : (+0.5, +0.5, 0)
Punto **B** : (-0.5 , 0, 0)	Punto **B** : (+0.5, 0, 0)
Punto **C** : (0, 0, -0.5)	Punto **C** : (0 , 0, +0.5)

Soluzione di Davide Ruggeri (2 programmi diversi per SPHERES **blue** e **red**)

preselection2016_A_**blue** (parte 1)

```
ZRState myState;
float fa1[3], fa2[3], fb1[3], fc1[3],
      fnul[3], v[3], targetRate[3];
int currentPoint;
float points[4][3], dist, dist2;
```

```
void init(){
//Punto A
    points[0][0] = -0.5;
    points[0][1] = -0.5;
    points[0][2] =  0.0;
//Punto B
    points[1][0] = -0.5;
    points[1][1] =  0.0;
    points[1][2] =  0.0;
//Punto C
    points[2][0] =  0.0;
    points[2][1] =  0.0;
    points[2][2] = -0.5;
    targetRate[0]=  0.0;
    targetRate[1]=  0.0;
    targetRate[2]=  1.0;
    v[0]=0.0;
    v[1]=0.0;
    v[2]=0.0;
```

```
//FORZE
    fa1[0]= -1.0f;
    fa1[1]=  0.0f;
    fa1[2]=  0.0f;
    fa2[0]=  1.0f;
    fa2[1]=  0.0f;
    fa2[2]=  0.0f;
    fb1[0]=  0.46f;
    fb1[1]=  1.0f;
    fb1[2]=  0.0f;
    fc1[0]=  0.945f;
    fc1[1]= -0.15f;
    fc1[2]= -1.0f;
    fnul[0]= 0.0f;
    fnul[1]= 0.0f;
    fnul[2]= 0.0f;
    dist2 = 1;
    currentPoint = 0;

}          //--- fine init
```

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

preselection2016_B_**red** (parte 1)

```
ZRState myState;
float fa1[3], fa2[3], fb1[3], fc1[3],
      fnul[3], v[3], targetRate[3];
int currentPoint;
float points[4][3], dist, dist2;
```

```
void init(){
//Punto A
    points[0][0] = 0.5;
    points[0][1] = 0.5;
    points[0][2] = 0.0;
//Punto B
    points[1][0] = 0.5;
    points[1][1] = 0.0;
    points[1][2] = 0.0;
//Punto C
    points[2][0] = 0.0;
    points[2][1] = 0.0;
    points[2][2] = 0.5;
    targetRate[0]= 0.0;
    targetRate[1]= 0.0;
    targetRate[2]= 1.0;
    v[0]=0.0;
    v[1]=0.0;
    v[2]=0.0;
```

```
//FORZE
    fa1[0]=  1.0f;
    fa1[1]=  0.0f;
    fa1[2]=  0.0f;
    fa2[0]= -1.0f;
    fa2[1]=  0.0f;
    fa2[2]=  0.0f;
    fb1[0]= -0.46f;
    fb1[1]= -1.0f;
    fb1[2]=  0.0f;
    fc1[0]= -0.945f;
    fc1[1]=  0.23f;
    fc1[2]=  1.0f;
    fnul[0]= 0.0f;
    fnul[1]= 0.0f;
    fnul[2]= 0.0f;
    dist2 = 1;
    currentPoint = 0;

}          //--- fine init
```

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

```
void loop(){                    //---preselection2016_A_blue (parte 2)
      api.getMyZRState(myState);
      dist =  distanceToPoint(currentPoint);
      if (currentPoint == 0){
          if (dist > 0.25)  api.setForces(fa1);
          if (dist < 0.25)  api.setForces(fa2);
          if (dist < 0.01) {
              DEBUG(("dist=%5.3f , Point=%d
                    wX=%5.3f wY=%5.3f wZ=%5.3f",
                    dist,currentPoint, myState[9],
                    myState[10], myState[11]));
              currentPoint++;    }
    }
    if (currentPoint == 1){
          if (dist > 0.3)  api.setForces(fb1);
          if (dist < 0.3)  api.setPositionTarget(points[1]);
          if ((dist < 0.016) &&  (dist2  < 0.016)) {
              api.setForces(fnul);
              DEBUG(("dist=%5.3f , Point=%d
                    wX=%5.3f wY=%5.3f wZ=%5.3f",
                    dist,currentPoint, myState[9],
                    myState[10], myState[11]));
              currentPoint=2;
          }
          dist2 = dist;      }
    if (currentPoint == 2){
        api.setAttRateTarget(targetRate);
        if (dist > 0.45){
              if ((myState[1]<0.003) && (myState[1]>-0.010)) {
                    v[0]= myState[3];
                    v[1]=-myState[1];
                    v[2]= myState[5];
                    api.setVelocityTarget(v);
                    fc1[1]=0.0;
              }
              else     fc1[1]=-0.2;
        api.setForces(fc1);
        }
        if (dist < 0.45) {
              api.setPositionTarget(points[2]);
              DEBUG(("dist=%5.3f , Point=%d
                    wX=%5.3f wY=%5.3f wZ=%5.3f",
                    dist,currentPoint, myState[9],
                    myState[10], myState[11]));
        }
    }          }                    //--- fine loop
```

```
void loop(){                    //---preselection2016_B_red (parte 2)
      api.getMyZRState(myState);
      dist =  distanceToPoint(currentPoint);
      if (currentPoint == 0){
          if (dist > 0.25)  api.setForces(fa1);
          if (dist < 0.25)  api.setForces(fa2);
          if (dist < 0.01) {
              DEBUG(("dist=%5.3f , Point=%d
                    wX=%5.3f wY=%5.3f wZ=%5.3f",
                    dist,currentPoint, myState[9],
                    myState[10], myState[11]));
              currentPoint++;    }
    }
    if (currentPoint == 1){
          if (dist > 0.3)  api.setForces(fb1);
          if (dist < 0.3)  api.setPositionTarget(points[1]);
          if ((dist < 0.016) &&  (dist2  < 0.016)) {
              api.setForces(fnul);
              DEBUG(("dist=%5.3f , Point=%d
                    wX=%5.3f wY=%5.3f wZ=%5.3f",
                    dist,currentPoint, myState[9],
                    myState[10], myState[11]));
              currentPoint=2;
          }
          dist2 = dist;      }
    if (currentPoint == 2){
        api.setAttRateTarget(targetRate);
        if (dist > 0.45){
              if ((myState[1]<0.010) && (myState[1]>-0.003)) {
                    v[0]= myState[3];
                    v[1]=-myState[1];
                    v[2]= myState[5];
                    api.setVelocityTarget(v);
                    fc1[1]=0.0;
              }
              else     fc1[1]= 0.2;
        api.setForces(fc1);
        }
        if (dist < 0.45) {
              api.setPositionTarget(points[2]);
              DEBUG(("dist=%5.3f , Point=%d
                    wX=%5.3f wY=%5.3f wZ=%5.3f",
                    dist,currentPoint, myState[9],
                    myState[10], myState[11]));
        }
    }          }                    //--- fine loop
```