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, 0m, 0.25m)	(+0.5m, 0m, 0.25m)
B	(0m, +0.5m, 0.5m)	(0m, -0.5m, 0.5m)
C	(+0.5m, 0m, 0.75m)	(-0.5m, 0m, 0.75m)
D	(0m, -0.5m, 0.75m) Ruotando (in prossimità di D) 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)	(0m, +0.5m, 0.75m) Ruotando (in prossimità di D) 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 (D) 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 D.

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 quattro punti indicati (A, B, C, D), 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 D, ecc.)

Il punteggio che in certe situazioni viene visualizzato durante la simulazione NON ha alcun valore, in quanto si riferisce ad altre gare.
Eventuali indicazioni di OUT OF BOUND non vengono considerate né valutate.

Soluzione di Lorenzo Saba: preselection2017 (parte 1)

```
//Begin page funzioni
float distanza (float t[3]){
  float v[3];
    mathVecSubtract(v,t,myState,3);
    return mathVecMagnitude(v,3);
}
void speedcontrol (float pos[]){
  bool sw=false;//-- velocità componente x
   if(myState[3]>= 0.038
   || myState[3]<=-0.038)  sw=true;
                          //-- velocità componente y
   if(myState[4]>= 0.038
   || myState[4]<=-0.038) sw=true;
                          //-- velocità componente z
   if(myState[5]>= 0.038
   || myState[5]<=-0.038) sw=true;
   if(sw){
      mathVecSubtract(punt, pos, myState,3);
      DEBUG(("speedcontrol"));
   }
} //End page funzioni
```

Soluzione (parte 2)

```
//Begin  page main
float pos [3];
float pos1[3];
float pos2[3];
float pos3[3];
float pos4[3];
float punt[3];
float stop[3];
float ang [3];
float myState[12];
float start[3];
float dist;
int   state;
int   count;
char  punto[4];
float DistChg;

void init(){
    punto[0] = 'A';
    punto[1] = 'B';
    punto[2] = 'C';
    punto[3] = 'D';

    api.getMyZRState(myState);

    for(int i=0;i<3;i++)
      start[i] = myState[i];

    state=0;
    count=0;

    pos1[0]= -0.5f;
    pos1[1]= 0;
    pos1[2]= 0.25f;

    pos2[0]= 0;
    pos2[1]= 0.5f;
    pos2[2]= 0.5f;

    pos3[0]= 0.5f;
    pos3[1]= 0;
    pos3[2]= 0.75f;

    pos4[0]= 0;
    pos4[1]= -0.5f;
    pos4[2]= 0.75f;

    pos [0]= 0;
    pos [1]= -0.45f;
    pos [2]= 0.70f;

    ang[0]=0;
    ang[1]=0;
    ang[2]=1;//per ruotare attorno asse z

    stop[0]=0;
    stop[1]=0;
    stop[2]=0;
    DistChg = 0.015f;    //--- 1,5 cm
```

Soluzione (parte 3)

```
    if (myState[1] > 0.0f){  //----RED
        pos1[0]= 0.5f;
        pos2[1]= -0.5f;
        pos3[0]= -0.5f;
        pos4[1]= 0.5f;
        pos [1]= 0.45f;
        DEBUG(("RED x=%5.3f y=%5.3f z=%5.3f dChg=%5.3f",
          myState[0], myState[1], myState[2], DistChg));
    } else                  //----BLUE
       DEBUG(("BLUE x=%5.3f y=%5.3f z=%5.3f dChg=%5.3f",
          myState[0], myState[1], myState[2], DistChg));
} //--- fine init

void loop(){
  api.getMyZRState(myState);
  switch (state){
  case 0:                                  //---  punto[0] = 'A';
      dist = distanza(pos1);
      if (dist <= DistChg){
            DEBUG(("Punto %c: d=%5.3f ",
                    punto[state], dist));
            state++;                       //---  andare al punto B !!
      } else
      if (myState[3]>=0.04
      || myState[4]>=0.04
      || myState[5]>=0.035){
          mathVecSubtract(stop,start, myState,3);
          api.setForces(stop);
      } else
      if (dist <= 0.15f){
          api.setPositionTarget(pos1);
      }
      else {
          mathVecSubtract(punt,pos1, myState,3);
          api.setForces(punt);
      }
      break;
  case 1:                                  //---  punto[1] = 'B';
      dist = distanza(pos2);
      if (count<4)
          api.setPositionTarget(pos2);
      else
      if (dist <= DistChg){
            DEBUG(("Punto %c: d=%5.3f ",
                    punto[state], dist));
            state++;                       //---  andare al punto C !!
            count=0;
      } else
      if (dist <= 0.21f){
          api.setPositionTarget(pos2);
      }
      else {
          mathVecSubtract(punt,pos2, myState,3);
          speedcontrol(pos1);              //-- può modificare punt
          api.setForces(punt);
      }
      count++;
      break;
```

Soluzione (parte 4)

```
  case 2:                                   //---  punto[2] = 'C';
      dist = distanza(pos3);

      if (count<3)
         api.setPositionTarget(pos3);

      else  if (dist <= DistChg){
         DEBUG(("Punto %c: d=%5.3f ", punto[state], dist));
         state++;                           //---  andare al punto D !!
         count=0;
      }
      else  if (dist <= 0.18f){
         api.setPositionTarget(pos3);
      }
      else {
         mathVecSubtract(punt,pos3, myState,3);
         speedcontrol(pos2);                //-- può modificare punt
         api.setForces(punt);
      }
      count++;
      break;

  case 3:                                   //---  punto[3] = 'D';
      dist =  distanza(pos4);

      if (count<3)
         api.setPositionTarget(pos4);

      else if (dist <= DistChg){
      DEBUG(("Punto %c: d=%5.3f x=%5.3f y=%5.3f z=%5.3f wx=%5.3f wy=%5.3f wz=%5.3f --- END",
            punto[state], dist,
            myState[0], myState[1],  myState[2],
            myState[9], myState[10], myState[11]));          //-- posizione e velocità angolare
//-------state++;                           ARRIVATI in D !!!!
         count=0;
      }
      else  if (dist <= 0.25f) {
         DEBUG(("verso %c: d=%5.3f x=%5.3f y=%5.3f z=%5.3f wx=%5.3f wy=%5.3f wz=%5.3f ",
            punto[state], dist,
            myState[0], myState[1],  myState[2],
            myState[9], myState[10], myState[11]));          //-- posizione e velocità angolare

         api.setPositionTarget(pos4);
      }
      else {
         mathVecSubtract(punt, pos, myState,3);
         speedcontrol(pos3);                //-- può modificare punt
         api.setForces(punt);
      }

      if (dist <= 0.5f)
         api.setAttRateTarget(ang);//--- per far ruotare lo SPHERE attorno ad un asse parallelo all'asse z

      count++;
      break;
  }     //--- fine switch
}       //--- fine loop
//End page main
```