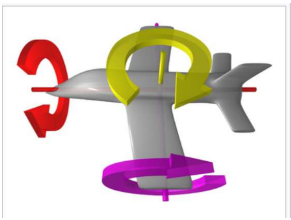
Angoli di Eulero in ZR *ECO-SPHERES*

Il programma che segue utilizza le funzioni **getOtherEulerState** , **getMyEulerState** per esporre mediante DEBUG gli angoli di Eulero con cui **RED** e **BLUE** sono ruotati rispetto al riferimento fisso XYZ; ad ogni istante, il programma modifica (mediante **setEulerTarget**) la rotazione_RPY (**Roll_Pitch_Yaw**) di **BLUE** per fargli assumere quella di **RED**. Il programma viene simulato con il Game *ECO-SPHERES AL-Hook&Tug* (senza detriti) per poter vedere meglio gli SPHERES.

Inizialmente **BLUE** e **RED** presentano l'uno all'altro la faccia con i ganci (sono fissati sulla ***+x face*** - p.14; i ganci sono già pre-orientati a **90°** - p.15 - per facilitare l'hooking).



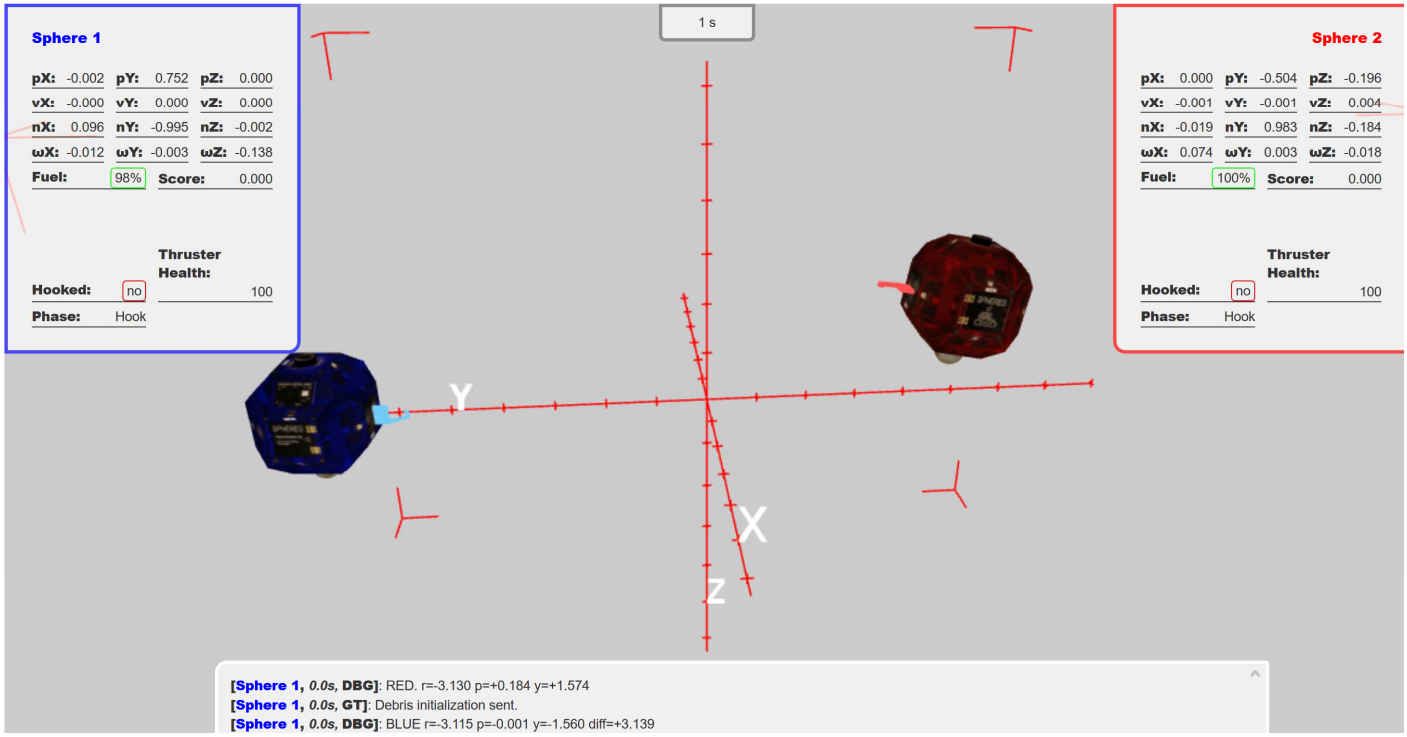
1 - **Roll**	Rotation about the **X** Reference Axis	**"rollio"**
2 - **Pitch**	Rotation about the **Y** Ref. Axis	**"beccheggio"**
3 - **Yaw**	Rotation about the **Z** Ref. Axis	**"imbardata"**

BLUE e **RED** hanno inizialmente **Yaw** (=rotazione attorno all'asse **Z**) opposti, mentre **Roll** (=rotazione attorno all'asse **X**) e **Pitch** (=rotazione attorno all'asse **Y**) uguali.

In pochissimi secondi **BLUE** si dispone come **RED** (**Roll_Pitch_Yaw** uguali) e ogni secondo ruota in modo simile rimanendo fermo nella posizione iniziale.

```
float myState[12];
float otherState[12];
float otherEulerState[12];
float myEulerState[12];
float target[3];

void init(){
}
void loop(){

  api.getMyZRState(myState);
  api.getOtherZRState(otherState);
  game.getOtherEulerState(otherEulerState);
  game.getMyEulerState(myEulerState);

  float diff = dist(otherEulerState, myEulerState);

  DEBUG(("RED. r=%+5.3f p=%+5.3f y=%+5.3f ",
     otherEulerState[6], otherEulerState[7], otherEulerState[8]));
  DEBUG(("BLUE r=%+5.3f p=%+5.3f y=%+5.3f diff=%+5.3f",
     myEulerState[6], myEulerState[7], myEulerState[8], diff));
target[0] = otherEulerState[6];
target[1] = otherEulerState[7];
target[2] = otherEulerState[8];
game.setEulerTarget(target);
}

float dist(float uno[], float due[])  {
    float distanceSquared = sqrtf((uno[6] - due[6]) * (uno[6] - due[6])
                                + (uno[7] - due[7]) * (uno[7] - due[7])
                                + (uno[8] - due[8]) * (uno[8] - due[8]));
    return distanceSquared;
}
```

ECO-SPHERES AL-Hook&Tug




```
Sphere 1, 1.0s, DBG: RED. r=-3.078 p=+0.192 y=+1.592
Sphere 1, 1.0s, DBG: BLUE r=-3.135 p=+0.001 y=-1.436 diff=+3.035

Sphere 1, 28.0s, DBG: RED. r=-1.542 p=-0.020 y=+1.749
Sphere 1, 28.0s, DBG: BLUE r=-1.568 p=-0.029 y=+1.741 diff=+0.028
```

ECO-SPHERES AL-Hook&Tug