20-3-2024

Look I wrote all of this with NO HELP FROM ANY AI so read until the end if you have any questions check the last page for help

This file is written in this date "20-3-2024" was last updated in "20-3-2024"
This is Version 0.1 of the integration points.

This will be updated regularly as we discover problems and mistakes ofc, but for now this seems reasonable I think.

Please Check out the documentation so you can better understand the concept
follow this link "Flowchart, Visual Workspace for Innovation (miro.com)"

This file is to help put the base of integration between the GNC Algorithms written in "**python**" with the environment built on unity in "**C#**"

The method of integrating the two programming environments is tasked to "**Lander Team.**"

The file's purpose is to define the main points of integration, what are the inputs that the python script takes and what it outputs.

Parts:

- **Python script or environment**
- **Unity C# code or environment**

So basically, the requirements consist of

- Inputs:
  The values the **python script** takes from the **Unity environment.**

- Outputs:
  The values the **python script** takes from the **Unity environment.**

- Practices:
  Tips or Asks that should be fulfilled.

1. Inputs:

As for now the inputs are given to the navigation to run the algorithms that estimate the **current state vector, + The current real state vector for comparison.**

The Inputs **“Where we are and how fast we are going…”** should be given to the python script so that it can choose an action to do from set of actions the spacecraft can do, use thrusters, us CG offset, deploy parachute…

NAVIGATION INPUT - Google Docs

- State vector: Real current values from the Environment
  [
  X position, Y position, Z position,
  X velocity, Y velocity, Z velocity,
  ROLL, PITCH, YAW,
  acceleration X, acceleration Y, acceleration Z
  ]

- The raw data from accelerometer sensor in the form of LIST
  AS [acceleration X, acceleration Y, acceleration Z]

- The data from gyroscope sensor in the form of list also
  AS [ROLL*, PITCH*, YAW*] in angel degree
  angle -> Change in angle*

- The data from magnetometer in the form of list
  AS [magnitude X, magnitude Y, magnitude Z]

- Mass of the spacecraft + fuel

- target point (x, y, z)

- control gains.

2. Outputs:

Control actions; which are called by the python GNC algorithm "**control**" algorithm to be exact to fix the trajectory of the spacecraft or the attitude or... for the beneficial of the mission.

The control commands as for today, are best thought of as an **"API"**

This basically means that we need to be able to call a function that will make the spacecraft preforms a certain action like deploying the parachute...

Each action that could be done will have a certain function with his description in the function
Please checkout the output Docs:
[CONTROL OUTPUT - Google Docs]

```
set_attitude[pitch, yaw, roll]:
    """
    Sets the desired pitch, yaw, and roll angles for spacecraft orientation.
    :param pitch: Pitch angle in degrees.
    :param yaw: Yaw angle in degrees.
    :param roll: Roll angle in degrees.
    """
set_bank_angle[angle]:
    """
    Sets the bank angle for spacecraft orientation.
    :param angle: The desired bank angle in degrees.
    """
adjust_cg_offset[offset]:
    """
    Adjusts the center of gravity (CG) offset to create aerodynamic lift.
    :param offset: The offset value indicating the shift in mass distribution.
    """
adjust_trajectory[correction_vector] #corrections involve firing thrusters or adjusting control surfaces
    """
    Makes small corrections to the spacecraft's trajectory during descent.
    :param correction_vector: Vector indicating the adjustment needed for trajectory correction.
    """
deploy_control_surfaces[ surface_type, position]:
    """
    Deploys and controls aerodynamic control surfaces.
    :param surface_type: Type of control surface (e.g., fins, flaps).
    :param position: Position or angle at which to deploy the control surface.
    """
fire_thrusters[ thrust_direction, thrust_power]:
    """
    Fires spacecraft thrusters.
    :param thrust_direction: Direction of thrust (e.g., x, y, z).
    :param thrust_power: Power or magnitude of thrust.
    """
deploy_parachute[]:
    """
    Deploys the parachute for descent.
    """
```

3. **Practices:**
    - The integration way should be **easy to understand and customize** as we test.
    - We will need a way to **run Fast random customizable** scenarios in the environment so that we can do “Monto Carlo” Algorithms *Search for it for more information.*
    - We should have a way to run python script and env on the **same PC** to reduce the barriers of testing.
    - The idea of **API** should be searched for and applied in a clean code way to be editable when we find the problems later.
    - If any other ideas were found, **PLEASE Document it and let us know so we can apply it if it’s feasible.**

IF anything isn’t clear please don’t give me a headache ask

Control concerns -> Rana.

Navigation concerns -> Seif.

Practices -> Allah الله

والسلام عليكم ورحمه الله وبركاته