FLYTBASE ASSIGNMENT

UAV Strategic Deconfliction System

- Sarvesh Rajkumar

Reflection & Justification Document

1. Design and Architecture

The system was designed with **modularity** and **scalability** in mind:

- **Waypoint Class**:
    - A **lambda-generated Waypoint class** is used to store spatial information and perform distance calculations (**distance_to** method) between waypoints.
- **DroneTrajectory Class**:
    - Utilizes the **Drone's waypoints** and **time duration** to identify the drone's position at a time instant. This accounts for the temporal domain.
    - Here, two functions are designed based on user choice. If each segment is to be covered in an equal duration, then **position_const_timesegment()** can be used. Else if a constant velocity is preferred over total distance, then **position_const_velocity()** can be used.
    - Based on the function used, the spatial-temporal relationship of the drone changes.
- **Conflict Detection**:
    - For a given **FPS** and **Duration**, the number of frames and **equal time steps** are computed from it. At each time instant then the drone trajectories are considered, and a safety distance threshold checks if there are any drones within a spherical radius around the primary drone.
    - If no conflicts are detected, then we return **clear**. Otherwise, each different **conflict pattern with the drone IDs** is reported. A different conflict pattern is encountered if the drones in conflict with the primary drones are different compared to the previous time instant.
- **Visualization**:
    - Here, both **3D visualization** and an **XZ** visualization are provided. The two views are used to justify that some views may show the drones in conflict or in the circle radius, but in 3D space, the drones are not in conflict.
    - The visualization updates every frame with the current drone positions and their trajectories, and also displays if the drone is **safe** at the moment, or the **conflicts that the drone is in**.
- **Scalability**:
    - The waypoint class and Drone trajectory class can be extended to store more information such as communication between drones, status, etc.

2. Spatial and Temporal Checks

- **Spatial Check**:
    - The **primary.distance_to(id2)** method calculates the distances between two waypoints **primary** and **id2.** If the distance is below the **safety distance**(15 units), then a conflict is detected and the drone id2 is noted. A blue sphere around the primary drone indicates the critical conflict zone.
- **Temporal Check**:
    - All drones' positions are compared at time **t** based on the current frame being updated. Only active drones (within their **t_start** to **t_end**) are evaluated. This time instant **t**, the spatial position of the drones is derived using the **position functions** in the DroneTrajectory Class.

3. AI Integration

- Here, AI hasn't been used. Based on the derived positions of drones at time instants, a safety check is performed to find conflicts.
- **Potential Use Cases**:
    - In the case of conflict detection, AI can be used through a Neural Network or an LSTM specialized for handling temporal information that can take the drone's position, velocities, and waypoints to predict. AI would be more useful in the case where the trajectories aren't just straight lines, a map is given, dynamic obstacles, and to handle uncertainties. AI can predict future trajectories to check for conflicts rather than just at the current moment.
    - Reinforcement Learning can be used for the drones to replan their paths based on learned data. This can help prevent such conflicts in the first place.
    - When there are lots of drones, if conflict between each drone has to be computed, then we can perform an Approximate Nearest Neighbour algorithm to save computation time.

---

4. Testing Strategy and Edge Cases

- **Scenario Tests**:
    - **Conflict-Free**: Drones on parallel paths or non-intersecting trajectories.
    - **Direct Conflict**: Intersecting Trajectories.
    - **Edge Cases**:
        - Drones entering/exiting airspace mid-mission. The drone is identified as active or considered for conflict only from the start time.
        - Near-misses (just outside safety_dist).
        - Multiple Drones conflict check.

5. Scaling to Real-World Deployment

- **GeoSpatial Sharding:** Divide space into voxels or regions. Conflict check is only done with nearby shards and saves computational time.
- **Bloom Filters:** A precomputational filter before expensive conflict calculations that eliminates those drone pairs that are guaranteed not to be in conflict.
- Select a leader drone between the drones within a region to communicate with other drones. Leader drones resolve conflicts. Proximity based communications saves resources

6. AI Help for the assignment

- Assisted in plotting safety sphere around primary drone
- Inspired to design the simplified Waypoint class
- Helped with the visualization style to combine 3D and XZ views.
- Helped to print combined conflict message of drones