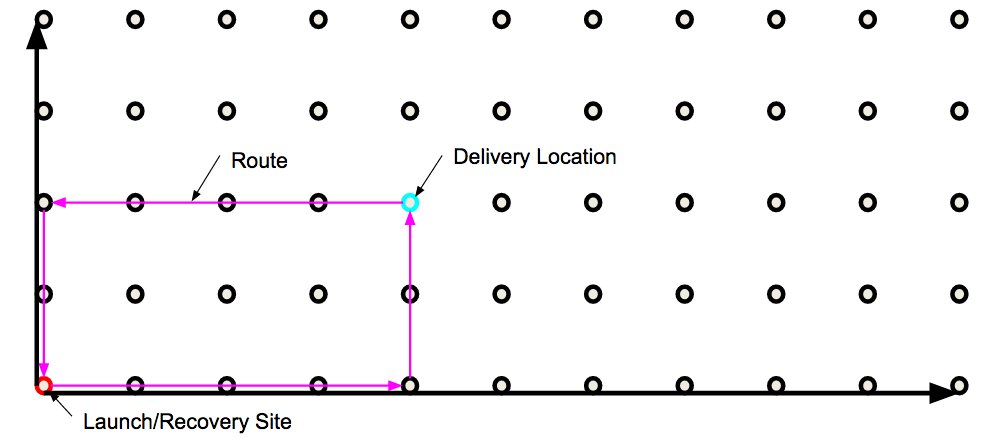
**16.35 Final Project: Delivery Drone Simulator**

**Requirements Doc**

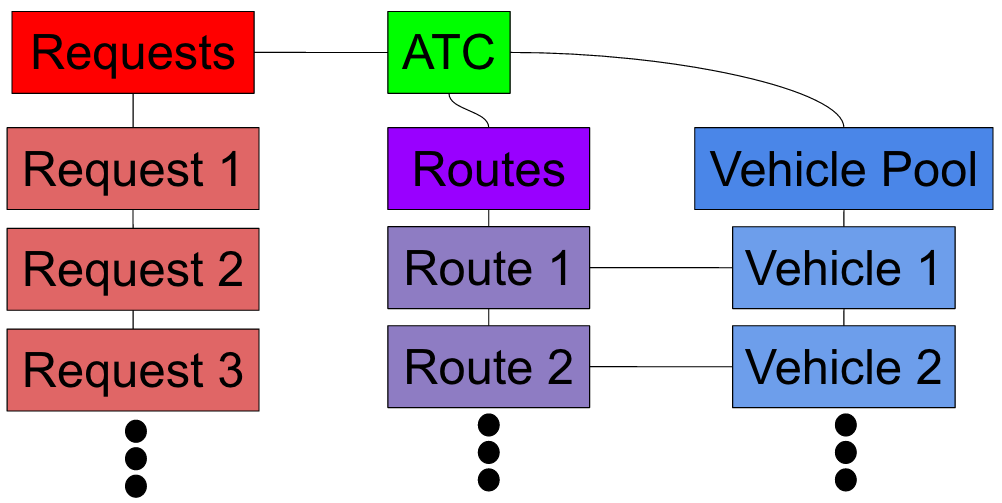
Our project is based on expanding on the GroundVehicle simulator to simulate a drone delivery system. As such, our primary focus has been to create an ATC class that can schedule the fastest possible routes for delivery requests while subject to time, space and other resource constraints. We have also created FlightVehicle, FlightController and Route classes to account for the drones’ differing operational capabilities compared to GroundVehicle objects, to control the drones in order to follow a designated route and to store the route from headquarters to delivery location and back, respectively.

For this project we have modeled our system drones on Amazon PrimeAir delivery drones which feature both rotor and fixed-wing flight. For simplicity, the modeled drones operate in a 2-1\2 dimensional space. While taking off or landing the drone can only move and rotate along the z-axis while in-flight it behaves like a GroundVehicle object with a fixed turn radius and minimum and maximum velocity requirements.

In terms of system architecture, the Simulator creates a set number of FlightVehicles located at the drone headquarters (0,0) at the start of its thread and periodically generates delivery requests for random locations in x and y coordinates and sends these requests to the ATC for scheduling. The ATC then finds the fastest feasible Route consisting of only vertical and horizontal segments from the headquarters to delivery location (Figure 1) and assigns this route to an available FlightVehicle and signals the FlightController associated with the specific vehicle to begin executing the route. Once the FlightVehicle has executed the delivery and returned to headquarters the FlightVehicle is cleared of its assigned Route and the FlightController clears its computed controller for that Route.



**Figure 1: Delivery route overview**



**Figure 2: ATC class architecture**

1. **ATC Class**
   1. Variables
      1. vehicles shall be an array of FlightVehicles that the ATC can direct
      2. numVehicles shall denote the total number of FlightVehicles that the ATC can direct
      3. startLoc shall be a tuple denoting the start location coordinates (x,y) of a delivery request
      4. vmax shall be the maximum velocity that the ATC will give to a FlightVehicle
      5. vmin shall be the minimum velocity that the ATC will give to a FlightVehicle
      6. Constraints
         1. vehicles shall be an array of FlightVehicle objects
         2. numVehicles shall be an integer
         3. numVehicles shall be equal to the number of FlightVehicle objects in vehicles
         4. The startLoc x-coordinate shall be a float
         5. The startLoc y-coordinate shall be a float
         6. vmax shall be a float
         7. vmax shall be non negative
         8. vmax shall be greater than or equal to vmin
         9. vmin shall be a float
         10. vmin shall be non negative
         11. vmin shall be less than or equal to vmax
   2. Constructor
      1. The constructor shall take no arguments
      2. The constructor shall instantiate numVehicles to 0
      3. The constructor shall instantiate vehicles to an empty array
      4. The constructor shall instantiate startLoc to (5,5)
      5. The constructor shall instantiate vmax to 5
      6. The constructor shall instantiate vmin to 1
   3. Methods
      1. canHandleMore() method
         1. Inputs
            1. The canHandleMore() method shall take no arguments
         2. Outputs
            1. The canHandleMore() method shall return a Boolean variable
            2. The returned Boolean shall be True if at least 1 FlightVehicle is available to execute a delivery route
            3. The returned Boolean shall be False if there are no FlightVehicles available to execute a delivery route
      2. handleRequest(destination, time)
         1. Inputs
            1. The handleRequest() method shall take in a tuple (destination) of the x-coordinate (x) and y-coordinate (y) of a new delivery request and the current time as an argument
            2. destination shall be a tuple of length 2
            3. x shall be a float
            4. x shall be in the range [6,100]
            5. y shall be a float
            6. y shall be in the range [6,100]
            7. Time shall be a float in seconds
         2. Outputs
            1. None, handleRequest() is a set method
         3. Invalid Inputs
            1. The handleRequest method shall raise an IllegalArgumentException if destination is not a tuple of length 2
            2. The handleRequest method shall raise an IllegalArgumentException if x is not within the range[1,250]
            3. The handleRequest method shall raise an IllegalArgumentException if y is not within the range [1,250]
            4. The handleRequest method shall raise an IllegalArgumentException if time is negative
         4. Routing of drone to delivery location (x,y)
            1. The handleRequest() method shall use the location of the requested delivery (x,y) to create a new piecewise route for a drone to follow in executing a delivery request
            2. The route shall be determined according to the shortest horizontal and vertical line segments from the drone location to the delivery location
            3. The route horizontal and vertical line segments shall intersect at a 90-degree angle
            4. The handleRequest() method shall add this new route to an available FlightVehicle if the route is compatible with all other current routes
            5. The handleRequest() method shall discard this route if it is not compatible with all other current routes
            6. The handleRequest() method shall create new routes by adjusting the velocity and time until one is found that is compatible with all other current routes
      3. addVehicles(vehicle) method
         1. Inputs
            1. addVehicles() shall take in a FlightVehicle object as input
         2. Outputs
            1. None, addVehicles() is a set method
         3. The addVehicles() method shall add the vehicle to the ATC vehicles list
      4. rmVehicle() method
         1. Inputs
            1. The rmVehicle() method shall take no arguments
         2. Outputs
            1. None, rmVehicle() method is a set method
         3. The rmVehicle() method shall raise an IllegalArgumentException if the ATC vehicle list is empty
         4. The rmVehicle() method shall raise an IllegalArgumentException if there are no available FlightVehicles in the ATC vehicle list
         5. If there is at least 1 available FlightVehicle in the ATC vehicle list the rmVehicle() method will remove the first instance of an available FlightVehicle from the ATC vehicle list
      5. checkRoutes(route) method
         1. Inputs
            1. The checkRoutes() method shall take a Route object (route) as an argument
         2. Outputs
            1. The checkRoutes() method shall return a Boolean variable
            2. The returned Boolean shall be True if the route is compatible with all other current routes
            3. The returned Boolean shall be False if the route is not compatible with all other current routes
         3. Invalid Inputs
            1. The checkRoutes() method shall raise an IllegalArgumentException if route if route is not a Route object
         4. Checking compatibility of a route
            1. The checkRoutes() method shall check that the route does not intersect the route of any other FlightVehicle in the ATC vehicle list in the 4 dimensions of space and time
      6. getVMax() method
         1. Inputs
            1. The getVMax() method shall take no arguments
         2. Outputs
            1. The getVMax() method shall return vmax
      7. getVMin() method
         1. Inputs
            1. The getVMin() method shall take no arguments
         2. Outputs
            1. The getVMin() method shall return vmin
      8. setVMax(v) method
         1. Inputs
            1. The setVMax() method shall take a float v as input
         2. Outputs
            1. None, setVMax() method is a set method
         3. The setVMax() method shall raise an IllegalArgumentException if v is negative
         4. The setVMax() method shall raise an IllegalArgumentException if v < vmin
         5. The setVMax() method shall set vmax to v
      9. setVMin() method
         1. Inputs
            1. The setVMin() method shall take a float v as input
         2. Outputs
            1. None, setVMin() method is a set method
         3. The setVMin() method shall raise an IllegalArgumentException if v is negative
         4. The setVMin() method shall raise an IllegalArgumentException if v > vmax
         5. The setVMin() method shall set vmin to v
2. **Route Class**
   1. Variables
      1. X1 shall denote the Route’s starting x-coordinate
      2. Y1 shall denote the Route’s starting y-coordinate
      3. X2 shall denote the Route’s horizontal and vertical segments intersection x-coordinate
      4. Y2 shall denote the Route’s horizontal and vertical segments intersection y-coordinate
      5. X3 shall denote the Route’s ending x-coordinate
      6. Y3 shall denote the Route’s ending y-coordinate
      7. T shall denote the Route’s starting time in milliseconds since simulation start time
      8. Constraints
         1. X1 shall be a float
         2. X1 shall be in the range [5,100]
         3. Y1 shall be a float
         4. Y1 shall be in the range [5,100]
         5. X2 shall be a float
         6. X2 shall be in the range [5,100]
         7. Y2 shall be a float
         8. Y2 shall be in the range [5,100]
         9. X3 shall be a float
         10. X3 shall be in the range [5,100]
         11. Y3 shall be a float
         12. Y3 shall be in the range [5,100]
         13. T shall be a float
         14. T shall be in the range [0,sys.float\_info.max)
   2. Constructor
      1. The internal representation of the route’s starting, segment intersection and ending coordinates shall be initialized according to the constructor’s arguments
      2. The constructor shall take in an array with 4 values
         1. The first value shall be a tuple of the route’s starting coordinates (X1,Y1)
         2. The second value shall be a tuple of the route’s Route horizontal and vertical segments intersection coordinates (X2,Y2)
         3. The third value shall be a tuple of the route’s ending coordinates (X3,Y3)
         4. The fourth value shall be the time that the route starts in milliseconds after the simulation start
      3. Invalid Inputs
         1. The constructor shall raise an IllegalArgumentException if the input array is not of length 4
         2. The constructor shall raise an IllegalArgumentException if the first value if not a tuple of length 2
         3. The constructor shall raise an IllegalArgumentException if the second value is not a tuple of length 2
         4. The constructor shall raise an IllegalArgumentException if the third value is not a tuple of length 2
         5. The constructor shall raise an IllegalArgumentException if X1 is not within the range [5,100]
         6. The constructor shall raise an IllegalArgumentException if Y1 is not within the range [5,100]
         7. The constructor shall raise an IllegalArgumentException if X2 is not within the range [5,100]
         8. The constructor shall raise an IllegalArgumentException if Y2 is not within the range [5,100]
         9. The constructor shall raise an IllegalArgumentException if X3 is not within the range [5,100]
         10. The constructor shall raise an IllegalArgumentException if Y3 is not within the range [5,100]
         11. The constructor shall raise an IllegalArgumentException if T is not within the range [5,sys.float\_info.max]
   3. Methods
      1. getStart() method
         1. Inputs
            1. The getStart() method shall take no inputs
         2. Outputs
            1. The getStart() method shall return an array of 2 elements [x,y]
            2. x shall be the x-coordinate of the start
            3. y shall be the y-coordinate of the start
      2. getIntersection() method
         1. Inputs
            1. The getintersection() method shall take no inputs
         2. Outputs
            1. The getIntersection() method shall return an array of 2 elements [x,y]
            2. x shall be the x-coordinate of the intersection
            3. y shall be the y-coordinate of the intersection
      3. getEnd() method
         1. Inputs
            1. The getEnd() method shall take no inputs
         2. Outputs
            1. The getEnd() method shall return an array of 2 elements [x,y]
            2. x shall be the x-coordinate of the end
            3. y shall be the y-coordinate of the end
      4. getStartTime() method
         1. Inputs
            1. The getStartTime() method shall take no inputs
         2. Outputs
            1. The getStartTime() method shall return the start time given at initialization
      5. getVelocity() method
         1. Inputs
            1. The getVelocity() method shall take no inputs
         2. Outputs
            1. The getVelocity() method shall return the velocity given at initialization
3. **FlightVehicle class (inherits from GroundVehicle class)**
   1. Variables
      1. availability shall denote whether a FlightVehicle is available for executing a route or not
      2. selfID shall denote the FlightVehicle’s assigned number
      3. numRotors shall denote the number of rotors the FlightVehicle has
      4. wingspan shall denote the wingspan of the FlightVehicle in meters
      5. maxThurst shall denote the maximum thrust that the FlightVehicle can provide
      6. x shall denote the FlightVehicle’s x-coordinate
      7. y shall denote the FlightVehicle’s y-coordinate
      8. z shall denote the FlightVehicle’s altitude
      9. dx shall denote the FlightVehicle’s x-translational velocity
      10. dy shall denote the FlightVehicle’s y-translational velocity
      11. dz shall denote the FlightVehicle’s z-translational velocity
      12. theta shall denote the FlightVehicle’s orientation
      13. dtheta shall denote the FlightVehicle’s rotational velocity
      14. route shall denote the FlightVehicle’s current executing route
   2. Constraints
      1. availability shall be a Boolean
      2. selfID shall be an integer
      3. selfID shall be within the range [0,numVehicles]
      4. numRotors shall be an integer
      5. numRotors shall be in the range [4,8]
      6. wingSpan shall be a float
      7. wingSpan shall be in the range [0,10]
      8. maxThrust shall be a float
      9. maxThrust shall be in the range [0,100]
      10. x shall be a float
      11. x shall be in the range [5,100]
      12. y shall be a float
      13. y shall be in the range [5,100]
      14. z shall be a float
      15. z shall be in the range [0,100]
      16. dx shall be a float
      17. dx shall be in the range [0,15]
      18. dy shall be a float
      19. dy shall be in the range [0,15]
      20. dz shall be a float
      21. dz shall be in the range [-10,10]
      22. theta shall be a float
      23. theta shall be in the range [-π,π]
      24. dtheta shall be a float
      25. dtheta shall be in the range [-π/4,π/4]
      26. route shall be a Route object
   3. Constructor
      1. The constructor shall take arguments for the FlightVehicle’s position and velocity
      2. Position shall be an array of length 4 consisting of the FlightVehicle’s x and y coordinates, theta orientation and z altitude (x,y,theta,z)
      3. Velocity shall be an array of length 4 consisting of the FlightVehicle’s x and y velocities, theta velocities and theta velocity (dx,dy,dtheta,dz)
      4. The constructor shall initialize the internal representation of the FlightVehicle’s position, orientation and velocities according to the input arguments
      5. The constructor shall initialize availability to False
      6. The constructor shall initialize selfID to the number of currently initialized FlightVehicles
      7. The constructor shall initialize numRotors to 4
      8. The constructor shall initialize wingSpan to 0.75
      9. The constructor shall initialize maxThrust to 10
      10. The constructor shall initialize route to None
      11. Invalid Inputs
          1. The constructor shall raise an IllegalArgumentException if the position array is not of length 4
          2. The constructor shall raise an IllegalArgumentException if the velocity array is not of length 4
          3. The constructor shall raise an IllegalArgumentException if x is not within the range [5,100]
          4. The constructor shall raise an IllegalArgumentException if y is not within the range [5,100]
          5. The constructor shall raise an IllegalArgumentException if z is not within the range [0,100]
          6. The constructor shall raise an IllegalArgumentException if theta is not within the range [-π,π]
          7. The constructor shall raise an IllegalArgumentException if dx is not within the range [0,15]
          8. The constructor shall raise an IllegalArgumentException if dy is not within the range [0,15]
          9. The constructor shall raise an IllegalArgumentException if dz is not within the range [-10,10]
          10. The constructor shall raise an IllegalArgumentException if dtheta is not with the range [-π/4,π/4]
   4. Methods
      1. isAvailable() method
         1. Inputs
            1. The isAvailable() method takes no arguments
         2. Outputs
            1. The isAvailable() method returns availability
      2. setAvailable(av) method
         1. Inputs
            1. The setAvailable() method shall take the FlightVehicle’s availability (av) as an argument
            2. av shall be a Boolean
         2. Outputs
            1. None, setAvailable() is a set method
         3. Invalid Inputs
            1. The setAvailable() method shall raise and IllegalArgumentException if av is not a Boolean
         4. Setting the FlightVehicle’s availability
            1. The setAvailable() method shall set the internal availability of the FlightVehicle to av
      3. setRoute(route) method
         1. Inputs
            1. The setRoute() method shall take a Route object (route) as an argument
         2. Outputs
            1. None, setRoute() is a set method
         3. Invalid Inputs
            1. The setRoute() method shall raise an IllegalArgumentException if route is not a Route object
         4. Setting the FlightVehicle’s Route
            1. The setRoute() method shall set the internal route of the FlightVehicle to route
      4. getRoute() method
         1. Inputs
            1. The getRoute() method shall take no arguments
         2. Outputs
            1. The getRoute() method shall return route
      5. getID() method
         1. Inputs
            1. The getID() method shall take no arguments
         2. Output
            1. The getID() method shall return selfID
      6. getPosition() method
         1. Inputs
            1. The getPosition() method shall take no arguments
         2. Outputs
            1. The getPosition() method shall return an array of 4 elements (x,y,theta,z)
            2. x shall be the FlightVehicle’s current x-coordinate (x)
            3. y shall be the FlightVehicle’s current y-coordinate (y)
            4. z shall be the FlightVehicle’s current z-altitude (z)
            5. theta shall be the FlightVehicle’s current orientation (theta)
      7. getVel() method
         1. Inputs
            1. The getVel() method shall take no arguments
         2. Outputs
            1. The getVel() method shall return an array of 4 elements (dx, dy, theta, dz)
            2. dx shall be the FlightVehicle’s current x-translational velocity
            3. dy shall be the FlightVehicle’s current y-translational velocity
            4. dz shall be the FlightVehicle’s current z-translational velocity
            5. dtheta shall be the FlightVehicle’s current rotational velocity
      8. advance(sec,msec) method
         1. Inputs
            1. The advance() method shall take the seconds (sec) and milliseconds (msec) of a time period t as arguments
            2. sec and msec shall be integers
         2. Outputs
            1. None, advance() is a set method
         3. Invalid Inputs
            1. The advance() method shall raise an IllegalArgumentException if sec is not an integer
            2. The advance() method shall raise an IllegalArgumentException if msec is not an integer
         4. Modifying the internal state of the FlightVehicle
            1. Modifying the internal state of the FlightVehicle while climbing or descending

The advance() method shall set the internal theta-orientation of the FlightVehicle to theta = theta + dtheta\*t

The advance() method shall set the internal z-altitude of the FlightVehicle to z = z + dz\*t

* + - * 1. Modifying the internal state of the FlightVehicle while turning

The advance() method shall calculate the kinematic and dynamic change of the FlightVehicle that occurs after time period t while the FlightVehicle is turning according to a circular dynamics model

The advance() method shall set the internal theta-orientation (theta) of the FlightVehicle to theta = theta + dtheta\*t

The advance() method shall set the internal x-position (x) of the FlightVehicle to x = xcirc + r\*sin(theta)where xcirc is the x-distance traveled by the FlightVehicle while in circular motion and r is the radius of the circle

The advance() method shall set the internal y-position (y) of the FlightVehicle to y = ycirc - r\*cos(theta) where ycirc is the y-distance traveled by the FlightVehicle while in circular motion and r is the radius of the circle

The advance() method shall set the internal x-velocity (dx) of the FlightVehicle to dx = speed\*cos(theta)

The advance() method shall set the internal y-velocity (dy) of the FlightVehicle to dy = speed\*sin(theta)

* + - * 1. Modifying the internal state of the FlightVehicle while the vehicle is not turning

The advance() method shall set the internal x-position (x) of the FlightVehicle to x = x + dx\*t

The advance() method shall set the internal y-position (y) of the FlightVehicle to y =y + dy\*t

* + - * 1. Constraints

The modified x-position (x) shall be within the range [5,100]

The modified y-position (y) shall be within the range [5,100]

The modified y-position (z) shall be within the range [0,100]

The modified theta-orientation (theta) shall be within the range [-π,π]

The modified x-velocity (dx) shall be within the range [0,15]

The modified y-velocity (dy) shall be within the range [0,15]

The modified z-velocity (dz) shall be within the range [-10,10]

1. **FlightController class**
   1. Variables
      1. maxForwardVel shall denote the maximum forward flight velocity of the FlightVehicle
      2. minForwardVel shall denote the minimum forward flight velocity of the FlightVehicle
      3. maxUpVel shall denote the maximum vertical flight velocity of the FlightVehicle
      4. minUpVel shall denote the minimum vertical flight velocity of the FlightVehicle
      5. oppertaingAlt shall denote the altitude at which the FlightVehicle shall operate
      6. Constraints
      7. maxForwardVel shall be a float
      8. minForwardVel shall be a float
      9. maxUpVel shall be a float
      10. minUpVel shall be a float
      11. opperatingAlt shall be a float
   2. Constructor
      1. The constructor shall take a Simulator object (sim) and FlightVehicle object (fv) as arguments
      2. The constructor shall initialize maxForwardVel to 5
      3. The constructor shall initialize minForwardVel to 1
      4. The constructor shall initialize maxUpVel to 5
      5. The constructor shall initialize minUpVel to -5
      6. The constructor shall initialize opperatingAlt to 30
   3. Invalid Inputs
      1. The constructor shall raise an IllegalArgumentException if sim is not a simulator object
      2. The constructor shall raise an IllegalArgumentException if fv is not a FlightVehicle object
   4. Methods
      1. initializeController() method
         1. Inputs
            1. The initializeController() method shall take no arguments.
         2. Outputs
            1. None
         3. Discretizing route into time segments
            1. The initializeController() method shall calculate the amount of time needed in seconds + milliseconds for the FlightVehicle to reach the operatingAlt.
            2. The initializeController() method shall calculate the amount of time needed in seconds + milliseconds for the FlightVehicle to fly a direct route from the starting location to the intersection location.
            3. The initializeController() method shall calculate the amount of time needed in seconds + milliseconds for the FlightVehicle to fly a direct route from the intersection to the ending location.
      2. getControl(sec,msec) method
         1. Inputs
            1. The getControl() method shall take the simulator time in seconds (sec) and milliseconds (msec) as arguments.
         2. Outputs
            1. The getControl() method shall return a Control based on the current Simulator time to apply the proper control for a FlightVehicle to complete a specified Route.
         3. Calculating new control
            1. The getControl() method shall calculate a new Control based on the Simulator time.