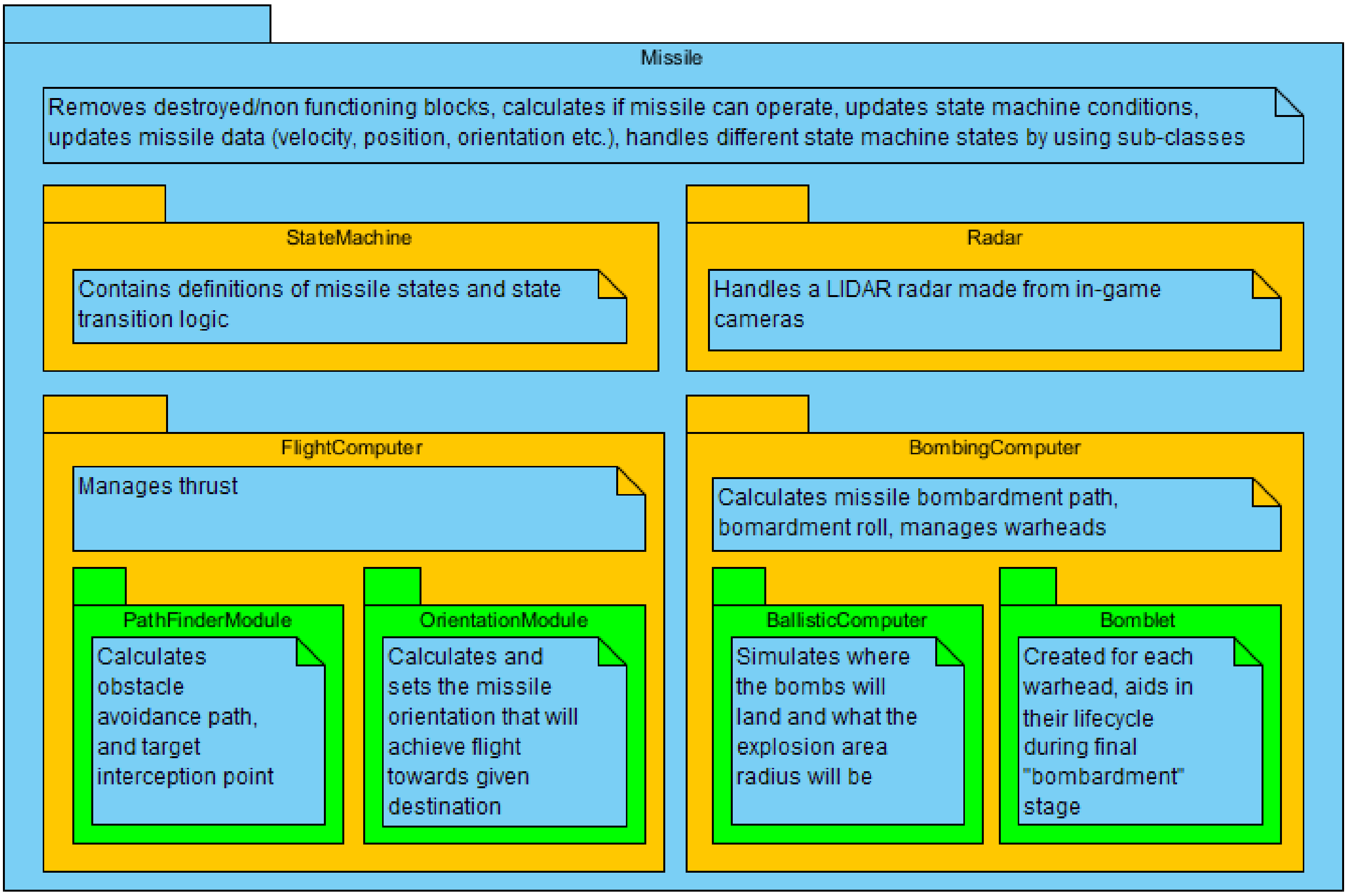
Code Philosophy and Editing Permissions

This script is written in a straightforward and easy-to-understand manner, prioritizing readability over strict optimization. Developers are encouraged to modify and improve the script as needed. If you make enhancements and release them within Space Engineers, I kindly ask for a credit mention as the original author.

Code Structure

The script is designed with a strong emphasis on encapsulation. The missile is implemented as a class containing multiple sub-classes, each responsible for a specific aspect of its flight. Below is a diagram illustrating the internal class structure:



Removes destroyed/non functioning blocks, calculates if missile can operate, updates state machine conditions, updates missile data (velocity, position, orientation etc.), handles different state machine states by using sub-classes

Contains definitions of missile states and state transition logic

Handles a LIDAR radar made from in-game cameras

Manages thrust

Calculates missile bombardment path, bomardment roll, manages warheads

Calculates obstacle avoidance path, and target interception point

Calculates and sets the missile orientation that will achieve flight towards given destination

Simulates where the bombs will land and what the explosion area radius will be

Created for each warhead, aids in their lifecycle during final "bombardment" stage

State Machine

The core of the missile's behavior is managed through a state machine, implemented as one of its sub-classes. The following states and transition conditions define the missile's operation:

Missile States:

```
public enum MissileStates
{
  Freefall,
  NoGuidanceFlight,
  GuidanceFlight,
  DetectedTargetFlight,
  TargetBombardment,
  BombHandling,
}
```

State Transition Conditions:

```
public struct StatesSwitchConditions
{
   public bool FreefallTimePassed;
   public bool NoGuidanceFlightTimePassed;
   public bool TargetWithMatchingCharacteristicsDetected;
   public bool TargetWithinBombardmentProximity;
   public bool BombsHaveBeenReleased;
   public bool AllBombsHaveExploded;
}
```

Possible Improvements

Here are some potential areas for improvement:

- **Side Camera Support:** Adding and handling additional cameras for better detection/tracking capabilities.
- **Persistent Storage:** Saving missile data to Storage to maintain functionality across world reloads.
- **Multi-Missile Handling:** Enhancing the script to control multiple missiles simultaneously.
- **Improved Target Interception Algorithm:** Refining the interception point calculations for more effective bombardment of moving targets.
- **Reduced Generalization:** Introducing specific handling for different missile flight scenarios to enhance overall performance and accuracy.

Contributions and enhancements are welcome and encouraged! If you implement any of these improvements, feel free to share your version within the community.