```cpp
#pragma once

#include <vector>
#include "Math.hpp"
#include "RocketState.hpp"
#include "RocketDynamics.hpp"

inline RocketState eulerStep(const RocketState &x,
                             double dt,
                             double t,
                             const Vector3 &F_B,
                             const Vector3 &M_B,
                             const Vector3 &g_I,
                             double t_burn,
                             double mass_flow_rate,
                             double Cd,
                             double A_ref,
                             double mass_empty)
{

    RocketState xdot = computeRocketStateDot(x, F_B, M_B, g_I, t, t_burn, mass_flow_rate, Cd, A_ref, mass_


    RocketState x_next;

    x_next.r_I = x.r_I + dt * xdot.r_I;
    x_next.v_I = x.v_I + dt * xdot.v_I;
    x_next.euler   = x.euler   + dt * xdot.euler;
    x_next.omega_B = x.omega_B + dt * xdot.omega_B;
    x_next.mass = x.mass + dt * xdot.mass;
    x_next.I_B = x.I_B;

    return x_next;
}

inline void simulateEuler(const RocketState &x0,
                          double dt,
                          int numSteps,
                          const Vector3 &F_B,
                          const Vector3 &M_B,
                          const Vector3 &g_I,
                          double t_burn,
                          double mass_flow_rate,
                          double Cd,
                          double A_ref,
                          double mass_empty,
                          std::vector<RocketState> &states)
{
    states.clear();
    states.reserve(numSteps + 1);

    RocketState x = x0;
    states.push_back(x);

    for (int k = 0; k < numSteps; ++k)
    {
        double t = k * dt;

        // predict next state
        RocketState xNext = eulerStep(x, dt, t,
                                      F_B, M_B, g_I,
                                      t_burn, mass_flow_rate,
                                      Cd, A_ref, mass_empty);

        // ground check
        if (xNext.r_I.z <= 0.0 && k > 0)
            break;

        x = xNext;
        states.push_back(x);
    }
}
```