# Orbital Space Defense
## Coordinate Systems and Relative Orbital Motion

March 18, 2011

## 1 Game Overview

The Orbital Space Defense iPhone app simulates the relative motion of a spacecraft (DEFENDER) with respect to a reference orbit. While playing the game, you can maneuver the DEFENDER by either tilting the phone or touching the screen.

The simulation also includes the following objects:

- ATTACKER - An enemy satellite that tries to intercept your DEFENDER.
- FUEL DEPOT - A canister of fuel (or propellant). Intercept this to replenish your fuel.
- DEBRIS - Any space junk that may be floating around. You do not want to collide with it.

The DEBRIS and FUEL DEPOT objects do not maneuver, but they do follow a pre-defined unforced trajectory. The ATTACKER objects are constantly maneuvering, with the goal of intercepting the DEFENDER.

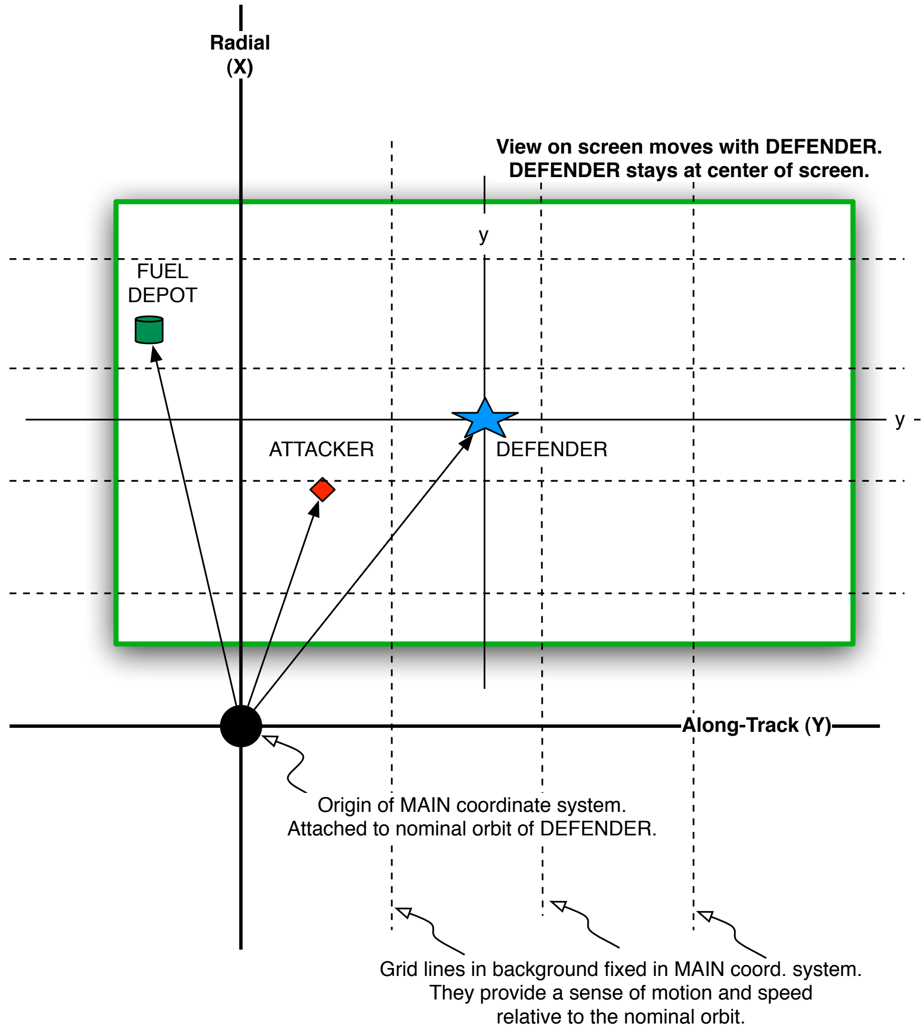
Your objective is to score points by staying close to the reference orbit for as long as possible. To do this you must maneuver to avoid the DEBRIS and the ATTACKERs, or shoot them down with your laser. However, the maneuvers eat up your fuel and the laser drains your battery. So occasionally you will need to intercept a FUEL DEPOT in order to replenish the spacecraft.

## 2 Coordinate Systems

Figure 2-1 on the following page provides a diagram of the coordinate systems used in the game. The game display is restricted to 2D, and so we only consider motion in the radial (X) and along-track (Y) directions.

The position and velocity of all objects are first defined in the MAIN coordinate system. The MAIN coordinate system is attached to the nominal orbit of the DEFENDER. As the DEFENDER maneuvers, it moves around in relation to the MAIN coordinate system. The manner in which the MAIN coordinate system moves around the Earth, following the nominal orbit, is illustrated in Figure 2-2 on page 3. The MAIN coordinate system is shown in two different example locations, at time T1 and then later at time T2. Note that the XY plane of the MAIN coordinate system is in the orbital plane. The cross-track direction (Z) is not shown in this 2D version of the game. Positive Z would point into the screen.

**Figure 2-1**. Coordinate Systems




The view on the screen is attached to and centered at the DEFENDER. Therefore, the positions drawn on the screen are all expressed in the DEFENDER's coordinate system.

Let the position of the DEFENDER in the MAIN coordinate system be $(y_D, x_D)$. Let the position of another object in the MAIN coordinate system be $(y, x)$. The position relative to the DEFENDER is:

$$\Delta y = y - y_D \tag{2-1}$$

$$\Delta x = x - x_D \tag{2-2}$$

# 3 Equations of Motion

## 3.1 Non-Maneuvering Objects

Non-maneuvering objects include the FUEL DEPOTs and DEBRIS objects. Their position in the MAIN coordinate system over time is conveniently expressed using the following Relative Orbit Elements (ROEs).

**Figure 2-2**. MAIN Coordinate System Rotates Around the Earth

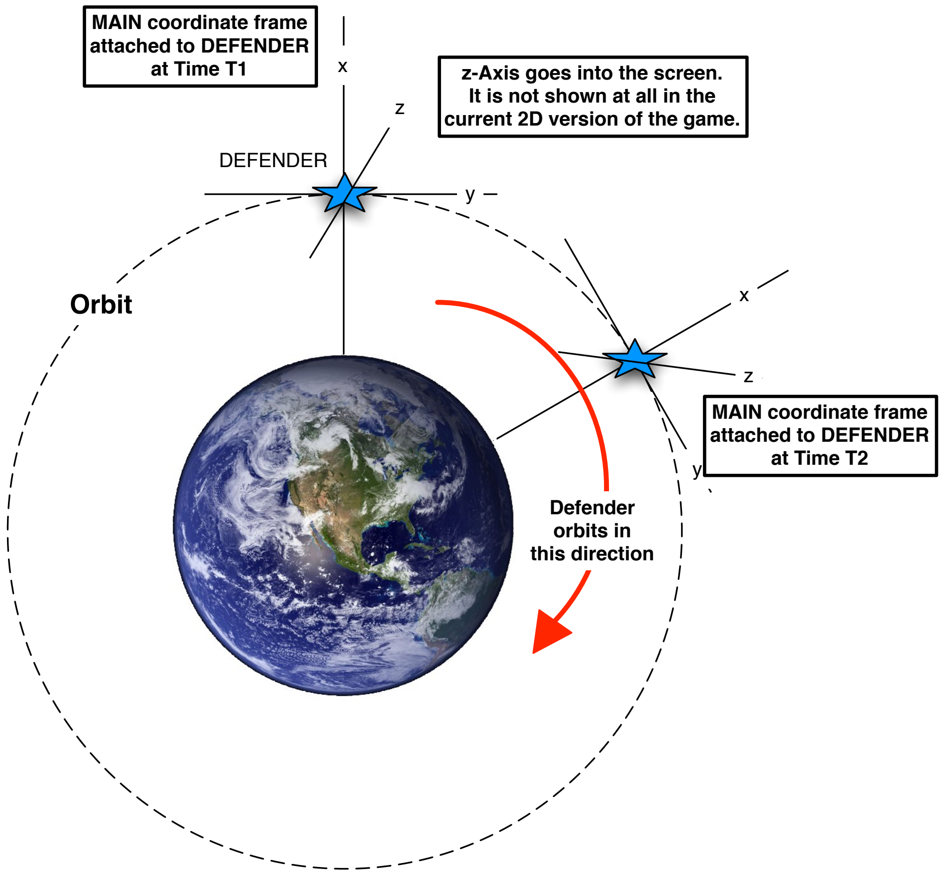



**Table 3-1**. Relative Orbit Elements

| ROE | Description |
|---|---|
| $x_D$ | Center of radial oscillatory motion |
| $y_D$ | Center of along-track oscillatory motion |
| $a_E$ | Semi-major axis of relative ellipse in orbital plane |
| $\beta_0$ | Phase angle of position on relative ellipse, measured towards $y$ axis from $x$ axis |
| $z_{\max}$ | Magnitude of cross-track oscillation |
| $\gamma$ | Phase angle of cross-track oscillation |

The position in the MAIN coordinate system at time $t$ is computed from the ROE set as follows:

$$\beta(t) = \beta_0 + nt \tag{3-3}$$
$$x(t) = x_D - \frac{a_E}{2}\cos\beta(t) \tag{3-4}$$
$$y(t) = y_D - \frac{3}{2}x_D nt + a_E \sin\beta(t) \tag{3-5}$$
$$z(t) = z_{\max}\sin\left(\gamma + \beta(t)\right) \tag{3-6}$$
$$\dot{x}(t) = \frac{1}{2}a_E n \sin\beta(t) \tag{3-7}$$
$$\dot{y}(t) = a_E n \cos\beta(t) - \frac{3}{2}n x_D \tag{3-8}$$
$$\dot{z}(t) = z_{\max} n \cos\left(\gamma + \beta(t)\right) \tag{3-9}$$

where $n$ is the orbit rate in units of rad/s. This is defined as:

$$n = \sqrt{\frac{\mu}{(R_e + h)^3}}$$

where $\mu = 398,600.436$ km$^3$/s$^2$ is the Earth gravitational constant, $R_e = 6,378.14$ is the mean equatorial radius of Earth, and $h$ is the orbit altitude.

## 3.2 Maneuvering Objects

The DEFENDER and ATTACKER objects each maneuver by applying thrusts. Let $\vec{F}$ be the applied thrust and let $m$ be the mass of the vehicle. The applied acceleration is:

$$\vec{a} = \vec{F}/m = \begin{bmatrix} F_x/m \\ F_y/m \\ F_z/m \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{3-10}$$

When playing the game, you create $F_x$ and $F_y$ inputs manually for the DEFENDER by either tilting or tapping the phone. Automatic control inputs may also be computed for the ATTACKERs. This is discussed in Section 3.3.

The total acceleration of the vehicle in the MAIN coordinate system is equal to this applied acceleration plus additional "virtual" acceleration terms that are functions of the vehicle's current position and velocity. The acceleration in each axis is:

$$\ddot{x} = 3n^2 x + 2n\dot{y} + a_x \tag{3-11}$$
$$\ddot{y} = -2n\dot{x} + a_y \tag{3-12}$$
$$\ddot{z} = -nz + a_z \tag{3-13}$$

The continuous dynamic system above can be discretized at a sample of $\Delta T$ seconds. In the discrete model, the state at point $k$ transitions to a new state at point $k+1$, as follows:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{z}_{k+1} \end{bmatrix} = \begin{bmatrix} 4-3c & 0 & 0 & s/n & 2(1-c)/n & 0 \\ 6(s-n\Delta T) & 1 & 0 & 2(c-1)/n & (4s-3n\Delta T)/n & 0 \\ 0 & 0 & c & 0 & 0 & s/n \\ 3ns & 0 & 0 & c & 2s & 0 \\ 6n(c-1) & 0 & 0 & -2s & 4c-3 & 0 \\ 0 & 0 & -ns & 0 & 0 & c \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ z_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{z}_k \end{bmatrix} \ldots \tag{3-14}$$

$$+ \begin{bmatrix} (1-c)/n^2 & 2(n\Delta T - s)/n^2 & 0 \\ -2(n\Delta T - s)/n^2 & 4(1-c)/n^2 - 1.5\Delta T^2 & 0 \\ 0 & 0 & (1-c)/n^2 \\ s/n & 2(1-c)/n & 0 \\ -2(1-c)/n & (4s - 3n\Delta T)/n & 0 \\ 0 & 0 & s/n \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

where $c = \cos(n\Delta T)$ and $s = \sin(n\Delta T)$.

## 3.3 Automatic Control

Each ATTACKER object uses automatic control to govern its applied acceleration vector. The target of each0 ATTACKER is the DEFNDER.

For the DEFENDER, we also compute the automatic control associated with returning to the nominal reference orbit. This is done so that we can display the direction and magnitude of the thrust vector as a guide.

Let $[x_T, y_T, z_T, \dot{x}_T, \dot{y}_T, \dot{z}_T]$ be the target state vector, which is defined differently for the DEFENDER and ATTACKERs. For each ATTACKER object, its target state is equal to the current state of the DEFENDER. For the DEFENDER, its target state is fixed at $[0, 0, 0, 0, 0, 0]$. In future versions, we will allow the user to specify the target state of the DEFENDER with a custom ROE set.

The applied acceleration for automatic control is computed as follows:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_k = \begin{bmatrix} -3n^2 - K_P & 0 & 0 & -K_V & -2n & 0 \\ 0 & -K_P & 0 & 2n & -K_V & 0 \\ 0 & 0 & n^2 - K_P & 0 & 0 & -K_V \end{bmatrix} \begin{bmatrix} x - x_T \\ y - y_T \\ z - z_T \\ \dot{x} - \dot{x}_T \\ \dot{y} - \dot{y}_T \\ \dot{z} - \dot{z}_T \end{bmatrix}_k \tag{3-15}$$

where the state $[x_T, y_T, z_T, \dot{x}_T, \dot{y}_T, \dot{z}_T]$ is the state of the object applying control, in the MAIN coordinate system.

This is full state linear feedback control. There are two control gain parameters, $K_P > 0$ and $K_V > 0$. The closed-loop system response is second order, with all of the poles having the same natural frequency $\omega_N$ and damping ratio $\zeta$:

$$\omega_N = \sqrt{K_P} \tag{3-16}$$

$$\zeta = 2\omega_N K_V \tag{3-17}$$

It should be noted that this method of full state feedback control is NOT an optimal control method for satellite maneuvers. In practice, the task is to plan a sequence of impulsive maneuvers that bring the satellite to some desired relative state at a future time, with the objective of minimizing the total delta-v (propellant). This approach is called model predictive control, or MPC. It is an optimal control problem that, when formulated over a finite sequence of time steps, leads to a linear system of constraint equations and a linear cost function – otherwise known as a linear program (LP). The full state feedback approach is much simpler and requires minimal computation, and is therefore more suitable for game play.