A Projectile Subsystem in a Flight Simulation System

1985

Robert F. Hodson

University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation


This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
A PROJECTILE SUBSYSTEM IN A FLIGHT SIMULATION SYSTEM

BY

ROBERT F. HODSON
B.S., University of Connecticut, 1981

RESEARCH REPORT

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering University of Central Florida Orlando, Florida

Spring Term 1985
ABSTRACT

An overview of a flight simulation system is presented with a discussion of the system components and the interaction between functional units. The functions of each unit in the visual system are described. Specifically, the projectile subsystem portion of the visual system is presented in detail. A projectile subsystem executive structure is presented with capability of controlling projectile activation and deletion. Mathematical models for missiles with linear projected impact and proportional guidance are discussed. Ballistic projectile models with and without wind and drag considerations are developed. The mathematical equations for position and attitude calculations are given. Design considerations and implementation of algorithms are also presented with other system design trade-offs.
TABLE OF CONTENTS

CHAPTER

I. FLIGHT SIMULATION SYSTEM .................. 1
II. PROJECTILE SUBSYSTEM ..................... 13
III. AAA MATH MODEL .......................... 24
IV. AIM9 MATH MODEL .......................... 31
V. GUN MATH MODEL ............................ 44
VI. BOMB MATH MODEL .......................... 48
VII. TRADE-OFFS AND CONCLUSIONS .......... 56
CHAPTER I

FLIGHT SIMULATION SYSTEM

A general block diagram of a Flight Simulation System is shown in Figure 1-1. The heart of a Flight Simulator can be divided into three subsystems:

1) the aircraft's cockpit,
2) the aircraft's motion system, and
3) the aircraft's visual system.

The cockpit of the flight simulator should be a replica of the cockpit of the aircraft being simulated. Aircraft controls and instrumentation should be an exact representation of the actual aircraft. The feel of the controls as well as the physical location and operation of the instrumentation should be the same as the actual aircraft. The windows of the cockpit will be visual displays. These displays are often cathode ray tubes (CRTs), which are viewed through an optical dome (or lens). Projection displays or light valves are also used in some simulator cockpits. The pilot will view the
Figure 1-1. Flight simulation system block diagram.
simulated scene through the visual display device which is chosen.

The cockpit is often mounted on a motion platform to simulate the actual motions that will occur during flight. Not all flight simulators use motion platforms. In some simulators, visual cues are used to create the illusion of motion. In a simulator that does use motion, the cockpit will roll, pitch, and turn with the pilot's controls. Effects such as air turbulence, crash and uncontrollable spins can be incorporated into the motion controls.

The visual system generates the video information which is sent to the cockpit for display. The visual system enables the pilot to fly through a digitized visual data base. The visual data base (VDB) typically contains a model of a portion of the real world. A typical data base may be an Air Force base with runways, buildings, trees, neighboring hills, mountains, rivers, valleys, and other landmarks. The visual data base stores the world through which the pilot flies. As the pilot makes flight maneuvers, the portion of the data base which the pilot sees through the cockpit displays is updated to give the illusion of flight and travel. In general, the visual
system updates the cockpit displays, using the visual database, to generate visual motion cues of flight through a realistic environment.

The cockpit, motion system, and visual system combined make up a flight simulator. In addition to those major components, there is a maintenance/instructor console. The purpose of this console is to provide instructor interface for training purposes and a maintenance interface for tasks such as system initialization, system backup, mission recording and others.

There are several reasons for flight simulation training rather than actual aircraft flight training. Flight simulators eliminate the risk involved with training inexperienced pilots. Potentially dangerous maneuvers, such as air-to-air refueling, can be practiced repeatedly with no risk to pilot or crew. Also, destructive exercises can be practiced, such as evasive action from weaponry or aerial dogfights. Flight simulators are also cost effective. The fuel costs associated with training pilots in actual aircraft can be excessive. Flight simulators offer an alternate training method with reduced cost.
A block diagram of a visual system is shown in Figure 1-2. The visual system shown in Figure 1-2 contains the following components:

1) Data Base and Data Base Management System,

2) Visual Computation System,

3) Frame I or Coordinate Set Allocation System,

4) Frame II or 3D to 2D Transformation System,

5) Frame III or Picture Display System, and

6) Maintenance/Instructor Console.

The visual data base contains digitized data representing the area in which the flight simulator will fly its missions. The VDB contains information that represents terrain and culture (buildings, roads, etc.) data in three dimensions. The Data Base Management System (DBMS) is used to load and update the VDB. The DBMS has functions which aid a modeller in the generation of new data base features. The functions performed by the DBMS are nonreal-time and are performed prior to the simulation of a flight mission. Once the VDB is updated and loaded, the VDB will be read in real-time as a training mission.
Figure 1-2. Visual system block diagram.
progresses. The VDB contains 360 degree field-of-view (FOV) around the aircraft.

The Visual Computation System (VCS) controls the position and attitude of the moving models in simulation. In a training mission, there may be moving threats, such as enemy aircraft or tanks. The movement of these models must be updated in real-time as the mission progresses. In addition to aircraft or vehicles, weaponry can also be used in a training mission. Anti-Aircraft Artillery (AAA) can be fired at the aircraft or the pilot may choose to fire a bomb, bullet or missile. The position of the projectiles must be tracked during their flight. The VCS performs this function. In general, the VCS generates attitude and position data for all moving models and projectiles in the training mission with the exception of the training aircraft itself, whose position is determined by the pilot's actions.

The Frame I System or Coordinate Set Allocation System has the function of assigning coordinate sets to all objects which can be displayed in the scene. The system must have a limit to the number of coordinate sets that can be processed. This limit may be due to timing
constraints, hardware constraints, software constraints or a combination of these. Due to the limit on the number of coordinate sets that can be processed, the Visual System must keep track of all active coordinate sets in the system, and delete and add coordinate sets as required. The bookkeeping associated with coordinate set allocation is performed in the Frame I System. Another function of the Frame I System is to generate a direction cosine matrix associated with each coordinate set. The direction cosine matrix is used to perform the coordinate set rotation of each object in the scene into a common coordinate system. This function is required since the position of each object in the simulation may be generated relative to different coordinate systems. For example, the position of a bomb may be relative to the aircraft position at the time the bomb is dropped, but the position of an anti-aircraft artillery may be relative to the missile site. The Frame I system may contain other functions along with its primary function of coordinate set allocation. Some additional functions which may or may not be incorporated into the Frame I System are fading, impact processing, cloud processing, flak and smoke marker processing and other special effects.
The Frame II System, or 3D to 2D Transformation System, performs the coordinate set rotations to put all objects potentially in the scene into a common coordinate system. Once the data base, moving models, projectiles and other objects are represented in a common coordinate system, potentially visible faces are selected by determining if they fall within the FOV of the pilot. The faces which are in the FOV are then projected into the view window. This is a 3D to 2D transformation of potentially visible faces. The faces are then prioritized with respect to the pilot's viewpoint and the ordered faces are sent to Frame III. Figure 1-3 depicts the Frame II processing.

The Frame III, or Picture Display System, maps the priority ordered faces from Frame II into the individual picture elements or pixels. In determining the color and intensity of each pixel, fading, blending, and texturing effects are added. The Frame III processes are performed on each pixel, and video information for each pixel is then output to the cockpit display.

The maintenance and instructor console is used for operator interface. Capabilities can be provided so the operator can record a training session for playback at a
Figure 1-3. Frame II processing.
later time or the operator can "freeze" a training mission to point out details that an inexperienced pilot may not be aware of. The maintenance and instructor console can also have diagnostic and debug capabilities to aid in troubleshooting system problems.

The processing components in the visual system can be implemented in hardware or software. The major constraint, which determines implementation, is time. In a flight simulator, the display must be updated at a rate which will produce smooth motion. There are simulators existing presently which operate at a 30Hz update rate. Newer simulators operate at a 50Hz or 60Hz rate. The higher update rate reduces display flicker and provides more accurate placement of moving models and projectiles in the scene. Traditionally, the VCS and Frame I functions have been implemented in software using a high speed general purpose computer and Frames II and III functions have been implemented in special purpose hardware. This division of hardware and software will undoubtedly change as technology advances and general purpose high speed hardware devices become available.
The Projectile Subsystem is the part of the VCS which controls the projectile allocation and the motion of the projectile during the time it is active in the simulation. The executive control, and the projectile flight algorithms will be discussed in the following chapters.
CHAPTER II

PROJECTILE SUBSYSTEM

The projectile subsystem is a part of the visual system that is located in the VCS. A high level block diagram of the Projectile Subsystem is shown in Figure 2-1.

![Figure 2-1. Projectile Subsystem Block Diagram](image-url)
The projectile subsystem calculates the trajectories of the projectiles in the simulation. The projectiles which will be modelled in later chapters are: bombs, bullets, anti-aircraft missiles, and AIM9 heat seeking missiles. The inputs to the projectile subsystem are: projectile activation inputs, initial position and attitude data, and aircraft position data. The projectile activation inputs are commands to begin the trajectory computation for a projectile. In the case of bombs, bullets and AIM9 missiles, the pilot would fire the aircraft's gun, drop a bomb or launch a AIM9 missile. The pilot's action would generate an activation input which would be sent to the projectile subsystem. In the case of an anti-aircraft missile, the activation signal would be generated by a launch site fire command. The fire command can be generated manually or some other algorithm, a range test from the aircraft to site location, can be used. Along with activation inputs, initial position and attitude data are required to define the starting points of the projectile's trajectory. In the case of a projectile which is launched from the aircraft, the aircraft centroid may be a reasonable initial position or more information may be required. The actual position of the projectile relative to the aircraft's position may be
needed. In the case of an AAA, the launch site position is needed as a projectile start position.

A projectile executive (PJEXEC) program is required to control the information flow in the projectile subsystem. The general hierarchy of the projectile executive relative to the other functional units in the projectile subsystem is shown in Figure 2-2.

![Diagram of Projectile Subsystem Hierarchy](image)

Figure 2-2. Projectile Subsystem Hierarchy

The Projectile Executive must perform several functions. The primary function of PJEXEC is to call the modules for trajectory calculations. The AAA, bomb, bullet and AIM9 are the projectiles which will be
discussed in later chapters. In general, the projectile's trajectory will be calculated by a mathematical model of the actual weapon's flight path.

The activation and deletion of projectiles is controlled by the projectile executive. For simplicity, the activation of AIM9 and AAA missiles will be assumed to be singular. Singular means that a unique decision is made to launch each missile. This implies that one missile can be launched at a given time and the minimum time between missile launches will be one system clock time. In actuality, this provides a launch repetition rate that is very fast since a typical system clock for a real-time flight simulator will run between 30Hz and 60Hz. In terms of inputs to the projectile executive, one fire signal must be provided for each AAA or AIM9 that is to be launched. The fire signal will cause the activation of one missile. For additional missiles to be activated, additional fire signals must be received.

The activation of a bomb is slightly more complex than the activation of the AAA or AIM9 missiles. The pilot has the capability of dropping several bombs with one command. Two methods of bomb release are considered, releasing all bombs simultaneously or a ripple release
method. There are trade-offs to both approaches, the first method burdens the bomb math model with starting several trajectory calculations in one clock time. This can become a critical problem if the initial trajectory computations are long and time consuming. The method of ripple release requires a time controlled release of each bomb. Ripple release relieves some of the time burden which may be placed on the bomb math model occurring in a simultaneous bomb release. Also, due to the relatively fast clock period, a ripple release method can be effectively made to act like a simultaneous bomb release by shortening the interval between dropping bombs to one clock period. A ripple release will be assumed through the rest of this paper.

The activation of bullets differs from that of bombs and missiles. The F-16 has a Gatling gun which is fired for a time interval dependent on the pilot's actions. The gun will fire as long as the pilot has the fire command button depressed. The signal received from the cockpit can cause the activation of several bullets. The pilot fire command will start and stop the activation of bullets and the projectile executive must control the activation of bullets at a specified time interval which is dependent on the Gatling gun's firing rate.
A flowchart for a simple activation routine is shown in Figure 2-3. There are several constraints that may be included in the activation criteria. The amount of ammunition (bombs, bullets and missiles) the aircraft can carry is a physical constraint. It may be desirable to include some bookkeeping in the projectile activation routine to insure the pilot doesn't activate more projectiles than his weapon store has. Another constraint is the number of projectiles that the simulation system can process at a given time. Trajectory calculations for each projectile require a fixed time interval and must be completed during each real-time clock period. Remember that a clock period is typically 16 to 33 milliseconds and there is a limited number of calculations that can be performed in that time interval. The number of calculations is dependent on the system used to perform calculations. In addition to time limitations, the cost associated with building system hardware to accommodate a large number of projectiles becomes excessive.

Due to the limited number of projectiles that can be processed at one time, an activation priority may be assigned to each projectile. The priority algorithm has the function of activating high priority projectiles before lower priority projectiles. It may also be
Figure 2-3. Projectile activation flow.
desirable to deactivate or delete a lower priority projectile to allow for the activation of a higher priority projectile. Priority algorithms can be as simple as "first come, first served," but more likely, a system designer would want to weigh the priorities relative to the effects they would have in the simulation. As an example, it is likely that a bomb would have a higher priority than a bullet since the visual effect of a bomb would be more apparent than that of a bullet.

Projectile deletion is a function performed by the projectile executive. Projectiles may be deleted for priority reasons or may be terminated by other means. A flight time may be associated with each projectile. After projectile activation, the flight time counter can be decremented until it becomes zero. At this time, the projectile can be deleted. The concept of using a flight timer is straightforward. The timekeeping can be performed in the projectile executive itself or by the individual projectile mathematical models. In the latter case, the projectile mathematical model must send a flag to the projectile executive requesting to be deleted from the simulation. In either case, the active list of projectiles is updated in the Projectile Executive.
Figure 2-4 shows the data flow interaction between the projectile executive and an individual mathematical model.

The deletion of projectiles may be more complex than time based deletion. For instance, it is desirable to delete a bomb projectile after it impacts the terrain so that an explosion effect can be generated. In a case like this, additional inputs may be required to the bomb mathematical model to determine if the position of the bomb is below the terrain. The determination of the bomb's position with respect to the terrain is performed by another part of the simulation system (typically Frame II), and received as an input to the bomb mathematical model and a deletion request is then passed on to the Projectile Executive. The bomb impacting the terrain is
just one example of conditions which merit deletion of a projectile. In the case of missiles and bullets, other termination conditions may exist. Sometimes the deletion request can be generated by the projectile mathematical model itself and sometimes external inputs from other parts of the system are required. Deletion criteria for the projectiles is will be covered in the mathematical models discussed in later chapters.

In summary, the basic flow of the projectile executive is shown in Figure 2-5. The functions performed by each processing block are listed. The functions which are performed in a given system are dependent on the complexity of the simulation system as well as time, hardware, software, and cost constraints.
Figure 2-5. Projectile Subsystem Functional Flow

- Interpretation of pilot's commands
- Weapon store bookkeeping
- Priority
- Activation rates
- Projectile trajectory calculations
- Deletion criteria tested
- Removal of active projectiles
CHAPTER III

AAA MATH MODEL

The AAA projectile is a ground-to-air missile. The AAA has no aircraft tracking capability, therefore the missile is aimed ahead of the target aircraft on an intersecting path. A straight line trajectory is simulated for the AAA. Provided the target aircraft is in range and doesn't take evasive action after the AAA launch, the missile and aircraft will impact. The launch trajectory of an AAA is depicted in Figure 3-1.

\[ (X_s, Y_s, Z_s) \]

**Figure 3-1. AAA Trajectory**
The vectors in Figure 3-1 are defined as follows:

- $\mathbf{V}_a$ Aircraft velocity
- $\mathbf{V}_m$ Missile velocity
- $\mathbf{R}_a (X_a, Y_a, Z_a)$ Aircraft position
- $\mathbf{R}_s (X_s, Y_s, Z_s)$ AAA Site position
- $\mathbf{R}_m (X_m, Y_m, Z_m)$ Missile position.

The line of sight, vector ($\mathbf{L}$), from the launch site to the aircraft position can be calculated. A conversion from polar coordinates to nadir is performed in the calculation

$$\mathbf{L} = (X_1, Y_1, Z_1)$$  \hspace{1cm} (3-1)

$$X_1 = (X_a - X_s) \cos(X_1) \times \mathbf{K}$$  \hspace{1cm} (3-2)

$$Y_1 = (Y_a - Y_s) \times \mathbf{K}$$  \hspace{1cm} (3-3)

$$Z_1 = (Z_a - Z_s)$$  \hspace{1cm} (3-4)

Where $\mathbf{K}$ is the conversion constant from degrees to feet.

$$\mathbf{K} = 2\times\pi\times\mathbf{Re}/360$$  \hspace{1cm} (3-5)

Where $\mathbf{Re}$ is the radius of the earth, assuming a spherical earth. The angle $\mathbf{B}$ can now be calculated by

$$\mathbf{B} = \cos \frac{-1 \mathbf{V}_a \cdot \mathbf{L}}{|\mathbf{V}_a| \times |\mathbf{L}|}$$  \hspace{1cm} (3-6)
If $T$ is the time it takes for the AAA to impact the aircraft, the following relationship can be written using the Law of Cosines.

$$2 \frac{(V_m \cdot T)^2}{2} + \frac{L}{2} = 2 \cdot V_a \cdot T \cdot L \cdot \cos(B) \quad (3-7)$$

Solving for $T$ yields,

$$T = \frac{2}{L \cdot \left[-V_a \cdot \cos(B) + (V_a \cdot \cos(B) + V_m - V_a) \right]^{1/2}}$$

$$\frac{2}{V_m - V_a}$$

The impact position can now be calculated by $V_a \cdot T$. This gives the impact position in nadir, to be consistent, the impact position is converted to polar coordinates.

$$(X_i, Y_i, Z_i) = \text{impact position} \quad (3-9)$$

$$X_i = V_a \cdot X/T \cdot \cos(X_i) \cdot K \quad (3-10)$$

$$Y_i = V_a \cdot Y/T \quad (3-11)$$

$$Z_i = V_a \cdot Z \quad (3-12)$$

The impact position is now in polar form. The launch site position is also in polar form. This is done to calculate incremental $X$, $Y$, and $Z$ AAA position change vectors.

$$\triangle X = (X_i - X_s) \cdot (FR/T) \quad (3-13)$$

$$\triangle Y = (Y_i - Y_s) \cdot (FR/T) \quad (3-14)$$

$$\triangle Z = (Z_i - Z_s) \cdot (FR/T) \quad (3-15)$$

The values of $\triangle X$, $\triangle Y$ and $\triangle Z$ are the incremental change in
missile latitude, longitude and altitude for one frame time (FR). The AAA position can then be calculated each frame time by

\[
X_m(t) = X_m(t-1) + \Delta X\quad (3-16)
\]
\[
Y_m(t) = Y_m(t-1) + \Delta Y\quad (3-17)
\]
\[
Z_m(t) = Z_m(t-1) + \Delta Z\quad (3-18)
\]

The (t) designation is to denote current and past frame times. The initial (t-1) position is the launch site position.

It may seem to be troublesome calculating incremental change vectors when the missile's velocity vector (\(\bar{V}_m\)) is known. The missile's position at any given time is simply

\[
\bar{R}_m = \bar{R}_s + \bar{V}_m t.\quad (3-19)
\]

The advantage of using incremental (\(\Delta X, \Delta Y, \Delta Z\)) position change vectors becomes apparent when you consider the processing required for equations 3-16 through 3-18 versus equation 3-19. In the first case, three additions are performed and in the second case three additions and three multiplications are performed. Since computation time is often critical, the first method is more desirable.
In addition to the missiles position, the attitude of the AAA in the scene may also be desired. The attitude of the projectile is defined by the missile's roll, pitch, and heading. The definition of roll, pitch, and heading can be found in Appendix A. Before presenting equations for calculating the roll, pitch and heading of a missile, it should be stated that the attitude of a projectile in a scene may not add greatly to scene realism and be costly in terms of processing time. If the missile is traveling very fast and is coming from behind the target aircraft, the pilot may never see the missile. Attitude calculations become even less relevant for small projectiles like bullets. In cases like these, a symmetric, attitude independent model of the projectile is used. The question of performing attitude calculations is again a trade-off between processing time, scene realism and pilot training benefits.

For the AAA missile, it is assumed to be symmetric about the X-axis, therefore, the roll can be set to zero (Figure 3-2).
Figure 3-2. AAA Modelling Coordinate System.

The missile is traveling in the direction of $V_m$ which is in nadir coordinates. The projection of the velocity vector in the $XY$ plane is given by

$$V_{xy} = (V_y^2 + V_x)^{1/2}$$

(3-20)

The pitch and heading of the missile can now be calculated by

$$P_m = \arctan(V_z/V_{xy})$$

(3-21)

$$H_m = \arccos(V_y/V_{xy})$$

(3-22)

The position and attitude of the missile have been determined. The calculations are based on a straight line trajectory. If the target aircraft takes evasive action after the missile's launch, the missile will not impact
the target. Assuming that impact processing is performed by some other part of the simulation system and that the impact processor will request missile deletion by the projectile executive if impact occurs, the case of no impact must still be considered. The AAA math model knows the time to impact. If a deletion request has not been processed by the impact time plus some delta time for a safety margin, the AAA math model can deduce that the target aircraft took evasive action and the AAA can be deleted from the simulation. If the simulation has the capability of producing weapon effects such as flak puffs, the AAA model may request a flak puff to be located at the missile's last location prior to termination.

The AAA math model presented in this chapter obviously does not incorporate many flight trajectory variables such as wind, gravity, drag, fuel consumption, and others. The model which is presented does give a reasonable cue to training pilots as to the conditions present when a AAA is launched. There are existing simulators which use a straight line trajectory approach for the simulation of AAA. The major advantage of the AAA math model discussed is the simplicity of the model and the simplicity of the real-time equations. This makes processing fast and efficient.
CHAPTER IV

AIM9 MATH MODEL

The AIM9 Missile is an air-to-air guided missile. The AIM9 is fired from the aggressive aircraft at a target aircraft. The missile locks on to the target and tracks it until impact occurs, unless the target aircraft takes effective evasive action. A math model with tracking capability and missile characteristics to provide realism during evasive action are required. A tracking algorithm based on the line-of-sight from the missile to the target is presented in this chapter. Control parameters are incorporated into the math model for simulation of missile response time, turn rate limiting and tracking resolution.

The initial position of the missile is the position of the aircraft which fires the missile. The line-of-sight vector from the missile to the target aircraft is given by

\[ \overline{L(n)} = (Lx, Ly, Lz) \]  \hspace{2cm} (4-1)

where,

\[ Lx(n) = (\text{LNGt} - \text{LNGm})(K)\cos(\text{LATm}) \]  \hspace{2cm} (4-2)

\[ Ly(n) = (\text{LATt} - \text{LATm})K \]  \hspace{2cm} (4-3)
\[ L_z(n) = ALT_t - ALT_m \]  
\( (4-4) \)

The designators \( t \) and \( m \) are for target and missile respectively. The \( n \) designates that the line-of-sight is calculated at a discrete point in time. The constant, \( K \), is used to convert degrees to feet.

An angular change, \( P \), between two line-of-sight vectors can be calculated by

\[ P = \cos(\overline{l(n)} \cdot \overline{l(n-1)}). \]  
\( (4-5) \)

The small "\( l \)" denotes the normalized line-of-sight vector. Figure 4-1 depicts the angular change.

---

**Figure 4-1. Angular Change.**
A direction vector, $A$, can be calculated to show the direction of the angular change. The vector can be calculated by

$$
\overline{A} = \overline{V_m(n)} \times (\overline{L(n)} \times \overline{L(n-1)}).
$$

(4-6)

The resulting vector, $\overline{A}$, can then be normalized to get $\overline{a}$. The normalized direction vector is in the plane defined by the line-of-sight vectors and is perpendicular to the missile's current velocity vector. The change vector will be used in the calculation of the missiles new velocity vector and then the new missile position. With the direction of the change vector known, the magnitude of the change must be determined. The angular change, $P$, given in equation 4-5 is the magnitude of the angular change prior to any control parameter effects. Three filters, shown in Figure 4-2, will be used to simulate missile effects.

$$
P(n) \xrightarrow{\text{deadband}} Q(n) \xrightarrow{\text{rate limit}} W(n) \xrightarrow{\frac{1-K}{1-Kz}} F(n)
$$

Figure 4-2. Angular change filters.
The deadband filter removes minute angular changes. If the line-of-sight vector has not changed significantly from the previous calculation time, the angular change is set to zero, otherwise it is output unchanged. The deadband transfer function is shown in Figure 4-3.

![Figure 4-3. Deadband Filter](image)

The breakaway points DB1 and DB2 can be selected based on the tracking resolution of the actual missile.

The rate limiting filter restricts the turn rate of the missile. This is an important control parameter since a common evasive action for a target aircraft is a high-G turn. If the maximum turn rate of the missile is exceeded, the missile cannot track the target aircraft.
The rate limiting transfer function is shown in Figure 4-4.

\[ W(n) = \tan(Q(n)) \] (4-7)

Figure 4-4. Rate Limiting Filter

When the angular change \( Q(n) \) exceeds the rate limits \( R_1 \) and \( R_2 \), the value is clamped. The function between the rate limits is actually a tangent function.

The tangent function converts an angle, \( Q(n) \), to a distance. With no rate limiting, it is desirable to add a change vector to the past line-of-sight vector which will produce a new line-of-sight vector. This magnitude is defined by the tangent function (Figure 4-5).
Figure 4-5. Rate Limit Tangent Function.

The rate limiting transfer functions in Figure 4-5 shows constant slope of one between rate limits. This is an approximation for small angular changes since

$$\theta = \tan \theta$$

(4-8)

for small angles when \( \theta \) is in radians. This is a good approximation if the update time is fast since the missile can only turn a small amount between consecutive updates. This approximation will also save processing time since trigonometric functions are often time consuming calculations.

A time lag is introduced into the angular change calculation. The transfer function described as a Z-transfer is
The output \( F(n) \) is a function of the past input and the present input. Equation 4-10 describes this difference equation.

\[
F(n) = K F(n-1) + (1 - K) W(n). \quad (4-10)
\]

The constant \( K \) is a control parameter between zero and one which weighs the present and past inputs. The time lag causes the missile to react to changes in the target position with some delay. The constant, \( K \), is a function of the actual missile's dynamic response time.

The value of \( F(n) \) is used as the magnitude of the angular change vector, \( F(n) \). The angular change vector is added to the past line-of-sight vector to determine the direction of the new missile velocity vector. When the angular change vector is added to the past line-of-sight vector, the resultant vector must be normalized as shown in equation 4-11.

\[
\overline{vm(n)} = \frac{l(n-1) + F(n) a(n)}{\sqrt{l(n-1) + F(n) a(n)^2}} \quad (4-11)
\]

where \( \overline{vm(n)} \) is the normalized missile velocity vector. Giving the missile an average missile velocity of \( V_m \), the new missile velocity vector becomes

\[
\overline{V_m(n)} = V_m \overline{vm(n)}. \quad (4-12)
\]
It should be noted that the past line-of-sight vector was used in the preceding calculations. In the initial velocity calculation, the missile starts to fly along the line-of-sight, but after the first processing time, the missile flies along a vector which may be altered from the line-of-sight by the angular change filtering process, refer to Figure 4-6.

\[
\begin{align*}
\mathbf{v}_m(n) & \quad \mathbf{l}(n-1) \\
\mathbf{l}(n) & \\
P(n) & \quad \mathbf{F}(n)
\end{align*}
\]

Figure 4-6. Effect of Filtering.

Since the missile will not fly along the line-of-sight, \( \mathbf{l}(n) \), the normalized velocity vector should be used as the past line of sight vector in the angular change calculation for successive processing cycles after the initial velocity calculation.
With the missile's velocity vector determined, the new missile position can be readily found by
\[
R_m(n) = R_m(n-1) + V_m(n)(F) \tag{4-13}
\]
where \( R_m \) is the missile position at specified times, \( V_m \) is the missile velocity vector and \( F \) is the frame time. The velocity vector was calculated in nadir coordinates. It should be remembered that the coordinate systems must be consistent, \( \overline{R_m} \) is also in nadir. The position vector \( \overline{R_m} \) must be converted to polar coordinates if the resulting missile position is in latitude, longitude and altitude form.

The attitude of the missile must also be calculated if the missile is modelled as a six degree of freedom moving model. The attitude calculations which were described for the AAA in Chapter Three are also for the AIM9 missile. The equations are repeated here for convenience.

\[
P_m = \arctan \left( \frac{V_z}{V_{xy}} \right) \tag{4-14}
\]
\[
H_m = \cos^{-1} \left( \frac{V_y}{V_{xy}} \right) \tag{4-15}
\]
where \( V_{xy} = (V_x + V_y)^{1/2} \tag{4-16} \)

The roll of the aircraft can be set to zero if the missile is modelled symmetrically about the X-axis.
In addition to the position and altitude calculations of the AIM9 missile, the target lock-on determination must also be simulated. There are differences in target lock-on determination for the AIM9J and AIM9L missiles. The AIM9J is a heat seeking missile and therefore the aspect angle of the target with respect to the missile's aim vector must be considered. The AIM9L is an all-aspect missile and therefore the target's attitude is not important in determining missile lock-on. But, in the case of either missile, the target aircraft must be in the sensor range of the missile for lock-on determination. A conical section is used to determine if the missile can track the target, refer to Figure 4-7.

![Missile Lock-on Determination Cone](image)

**Figure 4-7. Missile Lock-on Determination Cone**

The missile lock-on test consists of a range and angle test. The angle test is
A < B then pass,
B > A then fail.
The angle can be found by
\[
A = \cos \left( \frac{-1}{|\mathbf{R}_t \cdot \mathbf{V}_a|} \right) \quad (4-17)
\]
The range test is simply
\[
\mathbf{R}_t < \mathbf{R}_c \text{ then pass,} \\
\mathbf{R}_t > \mathbf{R}_c \text{ then fail.}
\]
If both lock-on tests are passed, the missile can track the target if it is a AIM9L. The AIM9J requires further testing for aspect angle. The aspect angle can be determined by the dot product between the missile and target aircraft velocity vectors.
\[
\mathbf{V}_t \cdot \mathbf{V}_a = \mathbf{V}_t \cdot \mathbf{V}_a \cdot \cos(\text{AS}) \quad (4-18)
\]
where AS is the aspect angle, or
\[
\text{AS} = \cos \left( \frac{-1}{|\mathbf{V}_t \cdot \mathbf{V}_a|} \right) \quad (4-19)
\]
If AS is less than the maximum acceptable aspect angle for the missile, and all other lock-on tests have been passed, then the AIM9J can track the target aircraft. If a missile is fired by the pilot and lock-on determination tests are failed, the missile should travel a straight line trajectory. This can be done by using a constant missile velocity vector.
There are several other considerations about the AIM9 missile a designer might want to incorporate into the simulation for the effect of realism. One such consideration is a thrust build-up delay. After missile activation, the missile requires several seconds before it actually leaves the rack. This can be simulated by delaying the missile position update until an initial thrust build-up time delay has lapsed. Other missile considerations are the means of evasive action which are commonly taken against a AIM9J heat-seeking missile. Two evasive maneuvers are (1) flying into the sun and veering off with a high-G turn, and (2) dropping a counter measure flare and then veering off with a high-G turn. These two types of evasive action will not be discussed in detail here, but can be implemented with the use of a sun direction vector and a counter measure flare position vector.

The missile can be terminated or removed from the simulation when it impacts the target. Other termination criteria may include range or time in flight. If range or time in flight are the reason for termination, it may be desirable to let the missile freefall. This may not always be desirable, since freefall processing requires
additional processing time and there is very little chance that the missile will still impact the target.
CHAPTER V
GUN MATH MODEL

The F-16 aircraft has a 20 mm gun mounted in the fuselage of the plane. The bullets fired from the gun can be simulated as point models with a position, but no attitude. The attitude of the projectile is not important since the visual effect of the bullets orientation adds little or no additional realism to the scene. The bullet will be modelled as a projectile with a ballistic trajectory, without drag. The bomb model incorporates drag and will be discussed in a later chapter. For now, it will be assumed that the drag on the bullet is negligible or can be approximated by using an average bullet velocity in the trajectory equations.

First, the initial position and velocity of the bullet must be determined. Once again, simplifying assumptions can be made, the initial bullet position is given the initial aircraft position and if the muzzle of the gun is assumed parallel to the X-axis of the aircraft, the initial velocity is given by

$$\vec{V}_b = \vec{V}_{ax} + \vec{V}_m$$  \hspace{1cm} (5-1)
where $\overline{V_b}$ is the bullet velocity, $\overline{V_{ac}}$ is the aircraft velocity, and $V_m$ is the average muzzle velocity. A more general projectile model which allows for a general offset vector to define the initial projectile position and an arbitrary velocity initial velocity vector is considered in a later chapter in the discussion of the bomb model.

Equation 5-1 is in moving model coordinates and can be rotated to nadir coordinates by

$$\begin{bmatrix} \overline{V_b} \\ \text{nadir} \end{bmatrix}_{\text{mm}} = \begin{bmatrix} \overline{D} \overline{C} \end{bmatrix}_{\text{nadir}} \begin{bmatrix} \overline{V_b} \end{bmatrix}_{\text{mm}}$$

(5-2)

where,

$$\begin{bmatrix} \overline{D} \overline{C} \end{bmatrix}_{\text{mm}}_{\text{nadir}} = \begin{bmatrix} \cos(p) \sin(h)\sin(p)\sin(r) & \sin(h)\sin(p)\cos(r) \\
*\sin(h) + \cos(h)\cos(r) & -\cos(h)\sin(r) \\
\sin(h)\sin(r) & \cos(h)\sin(p)\cos(r) \\
\sin(p) & -\cos(p)\sin(r) & -\cos(p)\cos(r) \end{bmatrix}$$

where, $r$ is the roll, $p$ is the pitch, and $h$ is the heading of the aircraft.

With the initial position $(\text{LAT}_o, \text{LNG}_o, \text{ALTo})$ and velocity $(V_{bx}, V_{by}, V_{bz})$ known, the new bullet position can be determined by

$$\text{LAT}(t) = \text{LAT}_o + V_{by} \ast t \ast K$$

(5-4)
LNG(t) = LNGo + Vbx * t * K/cos(LATo) (5-5)

ALT(t) = ALTo + Vbz * t + 1/2 * g * t^2 (5-6)

where t is the time in flight

LAT(t), LNG(t) and ALT(t) are the bullet position, K is the conversion constant from feet to degrees, and g is gravitational acceleration.

When the gun is initially fired, t=0 and the position of the bullet is at LATo, LNGo and ALTo which is the aircraft position. The flight time is accumulated each frame by

\[ t(n) = t(n-1) + \Delta t \] (5-7)

where n is the frame count from the time the bullet is fired and t is the time between frames.

The position of the bullet can now be calculated each frame time using equations 5-4, 5-5, and 5-6. But if these equations are implemented as shown, the projectile position will be determined in an inefficient manner. The implementation of the equations can be made more efficient by noting that 5-4 and 5-5 are linear and the LAT and LNG change the same amount each frame time. The new LAT and LNG can be determined by

\[ LAT(n) = LAT(n-1) + \Delta LAT \] (5-8)
LNG(n) = LNG(n-1) + \Delta LNG \quad (5-9)

where,

LAT = Vby * \Delta t * K \quad (5-10)

LNG = Vbx * \Delta t * K/cos(LATo). \quad (5-11)

By implementing equations 5-8 and 5-9, the time to perform several multiply calculations can be saved. The \Delta LAT and \Delta LNG need to be calculated only once when the gun is fired.

Bullet termination can be performed in several ways. Three potential criteria of determining whether or not to remove the bullet from the simulation are: time in flight, distance from aircraft, and impact. In the first method, a flight time is associated with the bullet, and when the flight time is exceeded the bullet model requests deletion. The second method is based on distance from the aircraft. When the distance is large, the pilot cannot see the bullet and therefore deletion can be requested. The third method is to test the bullet for impact with a target or terrain. Upon impact, a request for deletion can be made. All or some of the deletion criteria may be implemented, but as always, processing time is a major consideration in the algorithm selected.
CHAPTER VI

BOMB MATH MODEL

For a bomb to successfully hit a desired target, many factors must be considered by the pilot dropping the bomb. The type of bomb is a consideration, since its size and shape will affect its trajectory. The wind effects are also a consideration, along with the aircraft's attitude and velocity. These considerations and others must be incorporated into a realistic bomb simulation.

First, the initial position of the bomb must be determined. In the case of the bullet model, the initial position of the bullet was said to be the aircraft centroid. In the bomb model, a more general initial position will be calculated. The rack position for the bomb will be defined in moving model coordinates with the origin located at the aircraft's centroid. The bomb position offset vector will be called Pos, refer to Figure 6-1.
If the aircraft position is given in polar coordinates, the bomb's initial position can be determined by

\[
\begin{align*}
\text{LAT}_b &= \text{LAT}_a + \text{LAT}_b \\
\text{LNG}_b &= \text{LNG}_a + \text{LNG}_b \\
\text{ALT}_b &= \text{ALT}_a + \text{ALT}_b
\end{align*}
\]

where \( \text{LAT}_b, \text{LNG}_b \) and \( \text{ALT}_b \) describe the bomb position.
LATac, LNGac and ALTac describe the aircraft position, LATb, LNGb and ALTb describe the change in position due to the offset vector. The offset vector is given in moving model coordinates, therefore, the attitude of the aircraft will effect the LATb, LNGb and ALTb. First the position offset vector is rotated into nadir coordinates.

\[
\begin{bmatrix}
\text{Pos} \\
nadir
\end{bmatrix} = \begin{bmatrix}
\text{D.C.} \\
m\text{m}
\end{bmatrix} \begin{bmatrix}
\text{Pos} \\
nadir
\end{bmatrix} \text{m} \text{m}
\]

(6-4)

The direction cosine moving model to nadir coordinates incorporates the attitude of the aircraft and is given by

\[
\begin{bmatrix}
\text{DC} \\
nadir
\end{bmatrix} \text{m} \text{m} = \begin{bmatrix}
\cos(p) & \sin(h) & \sin(p) & \sin(r) & \sin(h) & \sin(p) & \cos(r) \\
\* & \* & \* & \* & \* & \* & \*
\end{bmatrix}
\]

where,

\[
\begin{bmatrix}
\cos(p) & \cos(h) & \sin(p) & \sin(r) & \sin(h) & \sin(p) & \cos(r) \\
\* & \* & \* & \* & \* & \* & \*
\end{bmatrix}
\]

\[
\begin{bmatrix}
\cos(p) & \cos(h) & \sin(p) & \sin(r) \\
\* & \* & \* & \*
\end{bmatrix}
\]

\[
\begin{bmatrix}
\sin(p) \\
\* & \* & \* & \*
\end{bmatrix}
\]

where, r is the roll, p is the pitch, and h is the heading of the aircraft.

The offset position vector, in nadir coordinates, can now be used to determine \(\Delta\)LATb, \(\Delta\)LNGb and \(\Delta\)ALTb.

\[
\Delta\text{LATb} = Xos \* \cos(\text{LATac}) \* K 
\]

(6-6)

\[
\Delta\text{LNGb} = Yos \* K 
\]

(6-7)
\( \Delta \text{ALT}_b = Zos \) \hspace{1cm} (6-8)

Where Xos, Yos, and Zos are the X, Y and Z components of the position offset vector in nadir, and K is the conversion from ft to degrees.

The initial velocity of the bomb must also be determined for the bomb's trajectory calculation. Three factors influencing the bomb's initial velocity are: the aircraft's velocity, the wind velocity, and the bomb ejection velocity. The wind is assumed to be constant and can be described by the nadir velocity vector, \( V_w \). The ejection velocity of the bomb will be given by a general velocity vector, \( V_e \), given in moving model coordinates. Figure 6-2 shows how the ejection velocity may change as a function of bomb rack position.

Figure 6-2. Bomb Rack Position.
The aircraft's velocity, $\overline{V_{ac}}$, is also given in moving model coordinates, the initial velocity vector of the bomb, in nadir coordinates can then be determined by

$$\overline{V_i} = \overline{V_w} + \left[ DC \right]_{\text{mm}}^{\text{nadir}} \left[ \overline{V_{ac}} + \overline{V_e} \right]$$ (6-9)

With the initial bomb position and velocity known, the bomb's trajectory can be determined. A general ballistic equation with drag is given by

$$d = d_0 + A \overline{V_0} t + 1/2(a-B \overline{V_0}) t^2$$ (6-10)

where $d$ is the final position,

$d_0$ is the initial position,

$A$ and $B$ are drag coefficients,

$\overline{V_0}$ is the initial velocity,

$a$ is the acceleration, and

$t$ is the time in flight.

Equation 6-10 is written in one dimension, but applies for all three dimensions. The equation can be rewritten for $X$, $Y$ and $Z$ directions in terms of a change in position defined in nadir coordinates given by $\triangle X$, $\triangle Y$ and $\triangle Z$.

$$\triangle X = A_x \overline{V_{ix}} t + B_x \overline{V_{ix}} t^2$$ (6-11)

$$\triangle Y = A_y \overline{V_{iy}} t + C_y \overline{V_{iy}} t^2$$ (6-12)

$$\triangle Z = A_z \overline{V_{iz}} t + 1/2(g-B_z \overline{V_{iz}}) t^2$$ (6-13)

Note that in the $\triangle X$ and $\triangle Y$ equation there is no
gravitational acceleration, \( g \), and the \( B \) drag coefficients incorporate the constant \( \frac{1}{2} \) and the sign.

The bomb drag coefficients must be determined next in order to calculate \( \Delta X \), \( \Delta Y \), and \( \Delta Z \). Bomb coefficients can be calculated using actual bombing data. Bombing tables for a particular bomb include ground range and fall time for a particular altitude, air speed, and dive angle. At a particular altitude and air speed, values can be taken from the bombing tables to solve for the bomb coefficients by solving simultaneous equations, refer to example 6-1.

The drag coefficients for the \( X \) and \( Y \) equations can be found similarly. Note that it is not practical to solve for each possible airspeed and altitude. Several sets of drag coefficients can be determined over ranges of airspeed and altitude. These values can be stored in a table that is available in real-time. A table lookup can be performed based on the aircraft's current altitude and airspeed. To keep the table size small, a piecewise linear approximation of the bomb drag coefficients can be stored in the table, and linear interpolation can be used to determine bomb drag coefficients between data points.
Altitude (K feet) | Dive Angle (Degrees) | Airspeed (Knots) | Ground Range (Feet) | Fall time (Sec) |
--- | --- | --- | --- | ---
1.5 | 10 | 520 | 5055 | 5.98 |
1.5 | 20 | 520 | 3309 | 4.08 |

\[ \text{Viz} = 1.68666 \times \text{Airspeed} \times \sin(\text{dive angle}) \]

\[ \text{Viz}_1 = 1.68666 \times 520 \times \sin(10) = 152.30027 \text{ ft/sec} \]

\[ \text{Viz}_2 = 1.68666 \times 520 \times \sin(20) = 299.973 \text{ ft/sec} \]

Write simultaneous equations.

\[ 1500 = Az \times 152.30027 \times (5.98)^2 + (16.1 - Bz(152.30027)) \times (5.98) \]

\[ 1500 = Az \times 299.973 \times 4.08^2 + (16.1 - Bz \times 299.973) \times (4.00) \]

Solve for Az and Bz.

\[ Az = 0.989002943 \]

\[ Bz = 0.004317886 \]

Example 6-1. Solving for Drag Coefficients (Suciu 1984).
With the drag coefficients known, the bomb's trajectory can be calculated using equations 6-11, 6-12 and 6-13. The bomb model must also request to be deleted from the simulation when its free fall is complete. Termination can be determined by detecting a terrain impact or by time in flight criteria.
CHAPTER VII

TRADE-OFFS AND CONCLUSIONS

In the design of a projectile subsystem, several trade-offs are made. Several techniques were used in the projectile math models described in the preceding chapter. Sometimes accuracy was sacrificed for ease of implementation. Often time constraints affected the trajectory algorithm or its implementation. Other trade-offs affecting system capacity, scene realism, and training effectiveness must also be weighed and incorporated into the system design.

Execution time in a real-time simulation system influences many of the decisions made. There is a small window of time available to perform the desired processing. The designer must choose between the implementation of a few complex projectile math models or several math models with less complexity that can be executed faster. As discussed throughout the earlier chapters, the implementation of an algorithm can have significant effects on the time required to perform the function. Looking at mathematical functions, although
there are differences between computers, some generalities in terms of execution time can be made. Trigonometric functions generally require much more time than simple multiplication and division functions, and additions and subtractions require even less time. Also, double precision mathematics typically requires more time than single precision mathematics. Time can be saved by implementing equations with the less time consuming functions whenever possible. For complex functions requiring a lot of time, lookup tables can often be implemented. The use of lookup tables extensively, although they can often save time, can use large amounts of memory. A careful balance must be kept between time and memory and their availability.

Ultimately, the trade-offs that are made in design and implementation of a projectile subsystem are made to obtain effective training of pilots. Some early studies show that "relatively low-fidelity simulation training can transfer effectively to air-to-ground weapons delivery skills" (Schacter 1983). A study with combat ready pilots with and without simulation training has indicated that simulation "training should improve pilots' survivability, particularly during their first few combat missions" (Schacter 1983).
A general overview of a F-16 flight simulation system has been presented with specific consideration given to a projectile subsystem. By presenting a general system overview and several math model algorithms, some of the design and implementation techniques used in a projectile subsystem have been presented. System trade-offs and other considerations have also been discussed to present the flavor for system design considerations.
APPENDIX A

COORDINATE SYSTEM DEFINITIONS

The moving model coordinate system is shown in Figure A-1.

Figure A-1. Moving model coordinate set.

The origin of the moving model system is at the centroid of the model. X is out the nose of the aircraft, Y is out
the right wing, and Z is directed down. Heading is defined as angular rotation of the positive X axis towards the positive Y axis, on the Z axis. Roll is defined as angular rotation of the positive Y axis towards the positive Z axis, on the X axis. Pitch is defined as angular rotation of the positive Z axis towards the positive X axis, on the Y axis.

The nadir coordinate system is shown in Figure A-2.

![Figure A-2. Nadir coordinate system.](image)

The origin of the nadir coordinate system is fixed to a point on the earth. X points East, Y points North, and Z points up.
REFERENCES


