12-1993

Developing Realistic Cooperative Behaviors for Autonomous Agents in Air Combat Simulation

Dean P. Hipwell

AFIT/GCE/ENG/93D-05

DEVELOPING REALISTIC COOPERATIVE BEHAVIORS FOR AUTONOMOUS AGENTS IN AIR COMBAT SIMULATION

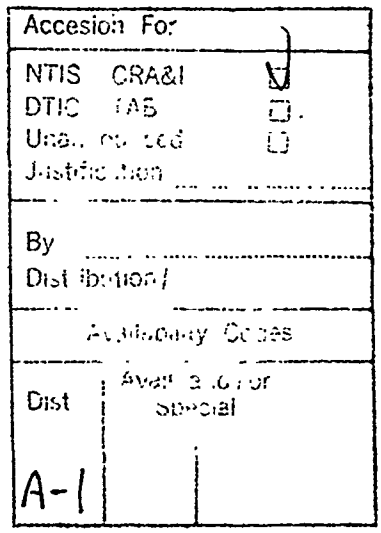
THESIS

Dean P. Hipwell
Captain, USAF

AFIT/GCE/ENG/93D-05

Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States government.



AFIT/GCE/ENG/93D-05

DEVELOPING REALISTIC COOPERATIVE BEHAVIORS FOR SEMI-AUTONOMOUS AGENTS IN AIR COMBAT SIMULATION

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of

the Requirements for the Degree of

Master of Science in Computer Engineering

Dean P. Hipwell, B.S.

Captain, USAF

December 1993

Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

Preface

The purpose of this study was to investigate, develop, and implement cooperative decision-making behaviors in autonomous agents of an air combat simulation by using a knowledge-based, expert system. If you thought that declaration had a lot of information, you're right. The idea of mixing human behaviors, air combat scenarios and computer programming seemed daunting to me when I started this project and reading it again still gives me the "willies". I wrote this thesis with three people in mind. To the computer engineer -- this document is primarily for you, so it contains complex descriptions of how the system works. To the engineering student -- in particular to those of you at AFIT who might develop this simulator further, I include pictures. To the air combat pilot -- I wrote this document so that some correlation could be made to textbook descriptions of air combat. Not being a pilot myself gave me an objective viewpoint of tactics and strategy, but I'd still prefer the experience instead of the study.

My thanks to Captain George Hluck who partnered me in developing a combat flight simulator. George developed the underlying architecture for a one versus one decision-making model. I developed multi-aircraft scenarios and cooperative decision models. Together we made a combat flight simulator capable of operating in many scenarios.

Thanks to my advisor Major Gregg Gunsch who kept me motivated and focused toward my goal. A word of thanks to Lieutenant Dan Gisselquist, who helped me work through the mathematical formulas in our simulator and led the way in developing an interface to the Red Flag terrain simulator used in the Graphics Lab. Also, to Steve Sheasby and Captain Brian Stoltz who connected us into a Red Flag simulator and helped identify improvements for our simulator. Having a visual display of my efforts was inspiring and gratifying.

And special thanks to my wife Lynne and son Gareth for putting up with my long hours at work. I'll make up for the time we've lost.

Dean P. Hipwell

Table of Contents

List of Figures

List of Tables

Abstract

The purpose of this study was to investigate, develop, and implement cooperative decision-making behaviors in an air combat simulation by using a knowledge based system. The specific aim was to model pilot decision processes during fighter aircraft combat scenarios. Since fighter combat typically involves many aircraft and not simply one versus a single enemy, cooperative decision-making is a critical part of realistic air combat simulation. Knowledge-based systems seemed well suited for this task because of built in features such as inference engines and rule based constructs. Writing an aircraft simulator, however, required definition of aircraft, basic flight maneuvers, and many other parameters including one versus one decision-making procedures. Cooperative decision-making behaviors were developed as extensions to the one versus one model.

This thesis addresses the specific problem of generating autonomous forces for inclusion in the Advanced Research Projects Agency (ARPA) Distributed Interactive Simulation (DIS) system. A 1993 survey by the Defense Modeling and Simulation Office noted the lack of implementations addressing Air Force platforms and the heavy emphasis on individual platform behaviors. The survey highlights problem areas including flexible and realistic behaviors within entities, real-time planning, and arbitration control schemes, as drawbacks to current implementations. The simulation system presented in this thesis addresses the problem of realistic behaviors and offers a method of modeling pilot decision processes rather than platform behaviors

The simulation system is based on phased control of a blackboard architecture in an object oriented design. Modular knowledge bases contain rules to process information used by fighter pilots in conducting aerial combat. Data is stored within agent class

definitions and shared between agents. Message passing coordinates the actions of cooperative agents. Phase control divides the entire rule base into smaller rule groupings to concentrate processing on essential calculations. Rule bases offer a selection of strategies, tactics, and maneuvers from which agents choose to fit specific engagements. The result is a highly flexible architecture that supports cooperative problem solving.

Cooperative decision-making in this simulation is leader-follower based. A designated team leader makes decisions and commands team members depending on circumstances. Cooperative agents share the workload in assessing threats and planning routes within the simulation environment. Leaders make an initial decision, but followers pick up leader responsibilities when the leader falls out of position or is destroyed altogether.

The simulation system was developed using the C Language Integrated Production System (CLIPS) Object Oriented Language. The system operates on UNIX workstations, DOS compatibles, and Macintosh computers. Currently, the system generates location data files which can be used to drive simulations on a Red Flag Terrain simulator, but there is potential for interactive DIS compatibility.

The simulator described in this thesis provides an architecture and design for modeling combat pilot decision processes. It includes methods for agent coordination and cooperative decision-making. The result is an air combat simulator called Pilot Decision Phases in CLIPS, or PDPC.

DEVELOPING REALISTIC COOPERATIVE BEHAVIORS FOR AUTONOMOUS AGENTS IN AIR COMBAT SIMULATION

1. Introduction

The purpose of this study was to investigate pilot decision-making processes and behavior patterns and implement those behaviors in air combat simulation. Research at the Air Force Institute of Technology (AFIT) led to an implementation of a framework for air combat simulation (Dyer and Gunsch, 1993). However, the actions and movements of adversary agents within the simulation did not realistically depict pilot tactics and strategies. Turns occurred instantaneously and at the maximum turning rate defined for each agent rather than gradually and smoothly. Weapons delivery took place at the earliest opportunity rather than at optimal points in an attack scenario. Planning was minimal and "deep" reasoning in decision-making was non-existent. The simulator mimicked basic flight motions, but could not conduct complex maneuvers. This thesis presents an air combat simulation which uses models of pilot behavior to realistically portray air combat scenarios. The simulation was developed using the C Language Integrated Production System (CLIPS) and is called Pilot Decision Phases in CLIPS (PDPC).

1.1 Background

1.1.1 Distributed Interactive Simulation. At present, the Advanced Research Projects Agency (ARPA) is developing a war game simulation in a program called

Distributed Interactive Simulation, or DIS (IST, 1992a). The DIS program will create synthetic battlefields by connecting computers and simulators through a nationwide communications network. Operators at diverse locations will climb into simulators and engage each other in mock combat. Through training in various scenarios, military personnel will gain battlefield experience in a safe and cost effective way.

A problem in scheduling mock combat exercises is in coordinating the needs of participants, or players. A scenario pitting tank battalions from Fort Bragg against fighter wings from Shaw Air Force Base would require detailed plans stating who, what, when, where, and why. More importantly, battle simulations will need an administrator to dictate when to stop and evaluate training progress. Simulations are controlled exercises that demand occasional stopping points for trainees to benefit from an analysis of their actions, or mistakes, made during battle. Coordinating adversaries in order to achieve some benefit may detract from the training value of DIS.

A second problem is a matter of scale: there may not be enough tank or fighter aircraft simulators to accurately depict realistic battlefield engagements. Reenacting battles of the Persian Gulf war would require hundreds of tank and aircraft simulators. Both friendly and enemy forces would need to be created in each scenario. Even if sufficient numbers of players were available, sufficient numbers of simulators may not be.

A third problem addresses the practical issue of communicating simulation data. To achieve realistic scenarios, massive amounts of computer data must be transferred to keep all players up to date. Even with state of the art communications technology, transmission delays of one-tenth of a second could mean the difference between hitting and missing a target. Subsequent arguments between players about who "shot" whom first would reduce the training effectiveness of DIS.

1.1.2 Autonomous Forces. Autonomous forces, also called computer generated forces, or CGF, are computer depictions of human organizations and human-controlled

machines. They are created using computer images and manipulated using computer algorithms to imitate the appearance, actions and behaviors of human-controlled machines. Autonomous forces within DIS are analogous to computer games where a single user competes against images presented on a computer screen. The concept of CGF goes farther than simple target shooting exercises. CGF will employ the tactics and strategies of human players without being under human control. CGF will eliminate scheduling difficulties because players will create battlefield engagements at any time. CGF will also provide a rich and large scale battlefield environment because players can depict large numbers of aircraft or tanks (Downes-Martin, 1992). By consolidating message traffic, CGF could reduce data transmissions requirements. CGF conveniently solves some DIS problems, but to enhance training experiences CGF must fool players into thinking they are competing against *human* players.

1.1.3 Realistic Behaviors in Autonomous Forces. Within DIS, forces generated by computer must behave realistically. One of the prime goals of CGF is to fool human players into thinking they are competing against, or with, other human players and not computers (Downes-Martin, 1992). Adversary aircraft should select tactics and perform maneuvers as if human pilots were at the controls. Collections of adversaries should move in unison across battlefields in coordinated efforts to attain strategic or tactical advantage. Friendly computer forces should provide battalion commanders the experience of seeing orders carried out in terms of massed movements. Sudden or erratic changes may reveal a computer generated player to an adversary. Human players could then label aircraft that make sharp turns as CGF and either defeat or avoid them. Ultimately, transitions between human and computer generated players should be transparent. If a human player decided to quit and walk away right in the middle of a fight, a computer generated player should be able to take control so that other players do not suspect a change occurred. Realistic depiction of human action is a key factor to successfully implementing CGF.

1.1.4 Cooperative Behaviors. Beyond achieving realism, human players must perceive CGF as cooperative teammates. Military operations depend on units acting as a cohesive team. Fighter pilots usually have a wing man. Tanks operate in battalions, or at the very least, in patrols. Current training simulators provide training for solitary warriors: a single pilot competing against adversaries as if no other friendly forces exist. In real battlefield environments, teamwork may dictate the success or failure of a mission. The actions of team members cooperating with each other can prove more effective than team members acting individually. To complete the image of realism, computer generated players must demonstrate cooperative behaviors and play team roles.

1.2 Problem Statement

The intent of this thesis was to investigate, develop, and implement cooperative behaviors in autonomous air combat agents for operation within the ARPA Distributed Interactive Simulation program. The scope of this thesis is stated in the following hypothesis.

Cooperative behaviors in autonomous agents of an air combat simulation can be implemented using universal reactive plans in a modularized knowledge-base system.

1.3 Scope

1.3.1 Fighter Combat Simulation. There are many aerial combat simulation systems, both military and commercial. Military versions are extensive, offering real-time graphics and requiring heavy duty computers. Commercial products are dominated by

"video" games. This thesis does not constitute a plan to build aerial combat simulators nor interfaces to these systems. Only the decision process was investigated and implemented. Parameters of this decision process include strategy and tactics, threat assessment, maneuver selection and other considerations.

1.3.2 Primary Sources. This thesis presents an architecture and a design for a fighter pilot model. The model is based primarily on two documents. *Fighter Combat, Tactics and Maneuvering*, by Robert L. Shaw , defines and illustrates key concepts of air combat (Shaw, 1985). *Pilot's Decision Definition and Analysis*, a report prepared by Titan Systems, Inc. of La Jolla, Calif. for the Pilot's Associate Program Office at the Air Force Wright Aeronautical Laboratories, defines key parameters and categorizes decision stages of air combat (Titan, 1986).

1.3.3 Limits of This Thesis. The task was one of encoding the decision process of fighter pilots cooperatively engaged in aerial combat. The PDPC simulator partially implements the concepts described in source documents. Combat simulation scenarios included one versus one, two versus one, and two versus two engagements. The one versus one scenario contrasts agent use of cooperative rules in two versus one and two versus two scenarios. This thesis does not illustrate engagements consisting of more than four agents.

1.4 Approach

1.4.1 Hierarchical Phase Control. The basic approach was to model the sequence of fighter pilot decisions as a series of phases. This approach is suggested by Giarratano and Riley in *Expert Systems, Principles and Programming* (Giarratano and Riley, 1989:455). Small rule-bases define events and transitions in each phase. A simple architecture connects consecutive phases as a sequential set of small rule-bases. The

overall goal is to achieve some phase of operation and make decisions based on that phase. Once an agent achieves a phase state, the only decision left is how to get out of that phase. Agents take actions and make plans during transitions between phases rather than while in the phase.

1.4.2 Rule-Based System. A rule-based system seemed appropriate since much of aerial combat can be explained in terms of rules. There are rules of flight, rules of engagement, rules of combat, and even the behaviors of a chain of command can be laid out in rules. The PDPC uses rules to capture the knowledge and experience needed in combat pilot decision processes.

1.4.3 Blackboard Model of Software Design. The basic design approach used the blackboard concept. Information used by fighter pilots during aerial combat is extensive and competes for attention (Funk and Lind, 1992). Pilots learn to ignore less important information in favor of critical information. Information remains accessible when a pilot needs it, but does not, or should not, interfere with the ability to fly a plane. A blackboard approach models pilot information flow by keeping data available when needed. From an implementation standpoint, data flowing through the PDPC could be monitored and newly encoded rules could easily be tied to necessary data as domain expertise was added.

1.4.4 Object-Oriented Design. The final design approach used object-oriented techniques (Rumbaugh, 1991). Initial designs required massive amounts of memory. Tests using a strict blackboard design required over 50 megabytes of storage to complete a single loop of a simple planned route. Converting to an object-oriented scheme eliminated memory requirement problems, simplified data access structures, and allowed more agents as the PDPC simulator expanded.

1.4.5 The C Language Integrated Production System. The simulation described in this thesis is written in the C Language Integrated Production System, or CLIPS. A rule-based system, CLIPS has a built-in inference engine to deliberate over rules and a

data structure to implement a blackboard. This saved development time in building an inference engine and blackboard. CLIPS has the capability of calling system and foreign language algorithms. Integrating a final product into another language would be simplified. CLIPS constructs can be embedded into other applications. Constructs compile into C language constructs, so porting versions into other applications could be accomplished. CLIPS constructs can also be embedded into Ada programs, so future developments could be tested with Ada language programs. CLIPS has an object-oriented capability which allows future modification and expansion. Having the "look and feel" of the artificial intelligence language, Lisp, CLIPS is a powerful tool for rapid prototype software development. (CLIPS, 1993)

Key features of CLIPS suitable for this thesis included a fact list, agenda, and modular development capability. The fact list provided a built-in blackboard system. Statements about conditions within the PDPC simulator were stored on the fact list for later reference. The agenda listed rules eligible for firing. Rules having satisfied predicate conditions were placed on the agenda and executed one at a time. The agenda allowed different forms of conflict resolution including depth first, breadth first, or even means-ends analysis. Prioritized rules caused the agenda to behave as a priority queue while rules of equal priority used the agenda as a stack. The latest version of CLIPS, version 6.0, allowed modularized rule bases. Rules that applied to specific situations in one case would not be considered in other cases. For example, takeoff and landing rules would not apply during engagements. A modular rule base quickly responded to simulation events since unnecessary rules were not even considered. Key features of CLIPS made it an ideal choice for this research project.

1.4.6 Hardware Development System. The PDPC simulator was developed on several platforms including Sun Sparcstations, 386 and 486 PCs, and Macintosh systems. Performance varied on each system, but PDPC transported without major code revision.

1.5 Assumptions

1.5.1 Fighter Combat Scenarios. This thesis is an exploration of pilot strategies and tactics used in two versus one and two versus two aerial combat. Efforts concentrate on team concepts such as role playing, role switching, and planning in multiple agent scenarios. A thesis by Hluck, explores one versus one strategy and tactics (Hluck, 1993). Cooperative behaviors among larger groups would need to be fully explored before realistic agents could operate autonomously on a system such as DIS.

1.5.2 Simulation Environment. Players operate in an arbitrary three-space environment as defined in Section 3.2.2.1. Agents within the simulation have no concept of terrain or obstacles.

1.5.3 Network Support. Although the agents are intended for use on DIS, an interface to DIS broadcasting media was not implemented. Fully interoperable network capability has been left for follow-on development.

1.6 Thesis Overview

This thesis contains six chapters and four appendices. Chapter Two is a historical development of research in the areas of planning, machine-learning, and cooperative systems as applied to autonomous agents. Chapter Three discusses the methodology used to develop the PDPC simulator. Chapter Four details implementation of the PDPC. Chapter Five reviews results of tests and simulation runs. Chapter Six presents conclusions and recommendations.

Four appendices define parameters and procedures used to implement the PDPC. Appendix A defines mathematical formulae used in the PDPC flight model. Appendix B identifies key parameters and terms used in PDPC rule modules. Appendix C elaborates

the pilot decision process by using a questionnaire measuring pilot situation awareness. Appendix D provides class slot definition for objects within the PDPC simulator.

A copy of the code for the PDPC is available from the following office:

AFIT/ENG
Artificial Intelligence Program
Wright-Patterson AFB, OH 45433

The CLIPS development environment can be obtained through the LinCom Corporation, Software Technology Branch Help Desk at (713) 286-8919 or by writing:

STB Products Help Desk
LinCom Corporation
1020 Bay Area Boulevard, #200
Houston, Texas 77058-2628

CLIPS is free to NASA, USAF, and their contractors for use on NASA and USAF projects.

1.7 Summary

This thesis presents a prototype expert system capable of executing cooperative behaviors in strategy and tactics during simulated aerial fighter combat scenarios. The system implements some of the decision processes used by combat fighter pilots. The basic approach divides decision processes into sequential phases with modular knowledge-bases defining the events and transitions within each phase. The simulation system was written in the rule-based language called CLIPS and was used primarily on PC and Sparcstation machines. For these reasons, I titled the simulator *Pilot Decision Phases in CLIPS*, or PDPC. The term PDPC is used throughout the remainder of this thesis.

2. Historical Development

Under the DIS program, computer generated agents will act as enemy, friendly, or even neutral participants (Downes-Martin, 1992). Since a range of scenarios will be executed under DIS, agents will function either singly or as part of a team. Individual agents need to know basic flight maneuvers as a baseline of performance. Agents then need to know how to put maneuvers together to execute combat tactics. Beyond tactics, an agent will need to know how to select tactics to fulfill a strategy. These layers of knowledge form a hierarchy that can be used to build an architecture. All three layers change when considering a team of aircraft as opposed to a single fighter. Although basic maneuvers stay the same, the choice of maneuvers and the choice of which agent performs a maneuver becomes critical to achieving a given goal. Tactics also gain variety in multi-aircraft scenarios since opposing aircraft may react differently to two attackers rather than to a single attacker (Shaw, 1985:Ch 5). Strategy, in the global sense, does not vary between combat aircraft on the same team since the goals of lead pilots are inherited by their partners. In an individual sense, strategy can be thought of as a prepared estimate of expected tactics and in this sense could vary from single to multiple aircraft engagements. Cooperative behaviors allow independent agents to act as a team in solving problems.

This chapter explores the extent of research in developing cooperative behaviors for computer generated agents. Computer generated agents are computer programs that communicate with each other and analyze data without human intervention. Research in cooperative behaviors centers on artificial intelligence solutions. One avenue of research proposes planning systems as a means of eliciting realistic behaviors. Planning systems include knowledge-based planning, universal or reactive plans, case-based planning, and

task management. A second avenue proposes learning systems as a means of modifying programmed behaviors to achieve performance improvements. Learning systems include explanation-based, neural network based, and discovery-based learning machines. A third approach explores interaction between independent computer systems. Blackboard and bulletin board models indicate potential development approaches for autonomous agents. Each area offers insight into developing cooperative behaviors in agents.

2.1 The Planning Approach

2.1.1 Knowledge Based Planning. V. V. S. Sarma and Savithri Raju present the concept of data fusion with knowledge-based processing to help battlefield commanders make decisions (Sarma and Raju, 1991). Data fusion is the collection and processing of information collected from multiple, geographically separate sensor systems. Collection sensors could include radar and sonar systems, infra-red detectors, seismometers, and even human sources such as photo intelligence and espionage. With such a cacophony on a battlefield, commanders must weigh each source before deciding a course of action. Data fusion systems process all information and evaluate conditions based on both current circumstances and previous knowledge. Summarized information is presented to field commanders as an aid in decision making.

In the PDPC simulator, each agent uses a filtered view of perfect information to gauge a response to an event. For example, each agent has knowledge of the location of every player in the simulation. CLIPS processes data held by all of the agents in the simulation, but PDPC rules limit action to adversaries within a specific radius.

2.1.2 Universal Plans. A universal plan is a preset list of actions that execute once a specific condition, or set of conditions, is recognized. Also called reactive plans, a computer generated agent collects and processes data, then selects a predetermined plan

of attack. In other words the computer agent reacts to specific scenarios, hence the term reactive plans. A simple system depending exclusively on a single universal plan does not demonstrate adaptability, a desired attribute of realism (Dyer and Gunsch, 1993:255). A universal plan system cannot improve performance since action lists are predetermined.

PDPC uses universal plans on several levels. The overall dynamic model is a linear system of "phases" which each agent must pass through during a mission. An agent receives a new mission, launches itself when ready, cruises around on patrol, and returns to land when recalled. This sequence of events constitutes a static universal plan. Within this plan, an agent can follow one of several route plans depending on the mission assigned. Agents can also calculate intercept trajectories and build a plan based on circumstances. Agents do not necessarily follow routes all the way to the goal. They may be interrupted enroute in which case a different set of rules apply and a new plan generated. This mechanism allows flexibility in static universal plans. The underlying system is a set of rules that must fire for any activity at all to take place. The PDPC simulator has three levels of universal plan: an overall strategy plan, an underlying procedural plan, and an intermediate flexible plan.

2.1.3 Case Based Planning. Research by Dyer and Gunsch indicates that a variation on universal planning, called case-based planning, could solve the problem of static performance and knowledge capturing inherent with universal plans (Dyer and Gunsch, 1993:256). Case-based planners begin with a universal plan, but modify planned actions based on current situations. A computer generated agent need not store large numbers of static universal plans and need not spend computational resources to evaluate multiple sensor sources. Cased-based planning starts by selecting a plan that closely fits a given set of conditions and then modifying that plan to exactly fit conditions. Case-based planners offer incremental performance improvement and adaptability.

The PDPC system uses the idea of case-based as a selector rather than a planner. Individual rules act as self-contained reactive plans which execute depending on conditions. This proved effective in design since the Shaw text and Titan report reduce air combat to a series of maneuver selections based on aircraft states (Shaw, 1985) (Titan, 1986). Selected rules provide incremental increases in positional advantage as an engagement progresses. Unlike a case-based planner, PDPC does not modify each rule when selected. Like a case-based planner, PDPC selects a plan that closely fits a given set of conditions.

2.1.4 Task Scheduling Approach. A thesis by Whelan investigates the feasibility of developing intelligent systems within a dynamic "real-time" architecture (Whelan, 1992). Task scheduling systems exhibit intelligent behavior by choosing a set of procedures from several procedure bases. Selection depends on internal and external conditions and the nature of threats in the environment. The architecture achieves a "real-time" nature by scheduling high priority tasks before low priority tasks and selecting short procedure sets to meet short response times. The effect is that the system addresses high threat events first by quickly offering a reactive plan. Then, when time permits, the system analyzes conditions and provides a more effective solution. The primary focus in task scheduling is dynamic task creation while meeting time criteria.

The PDPC system does not address the "real-time" issue since the primary focus was on pilot decision processes, but performance measurement data indicates the system could be modified to meet time criteria requirements (See Chapter 5). PDPC chooses a set of rules from a selection of rule bases. Salience rather than priority dictates rule firing order, but the system does not distinguish between short and long sets. With a slight modification to rule base access rules, longer procedures can be bypassed in favor of immediate action commands. For example, there is no need to build an intercept plan if a target is within missile firing range. For time based calculations, such as distance versus

velocity, PDPC assumes a time increment of one second. Before using PDPC in a "real-time" system, the time base would have to be connected to a "real-time" clock. "Real-time" implementation has been left for future development.

2.1.5 An Agent Based Pilot-Vehicle Interface. Task management is the focus of a system designed to help human pilots fly real airplanes (Funk and Lind, 1992). The Task Support System (TSS) employs object-oriented software design techniques to configure (and re-configure) the limited resources found within a real cockpit. The TSS is a knowledge-based system with a "real-time" task manager. The system completes tasks automatically, offers pre-sorted checklists when needed, and uses information displays to present information on current tasks rather than background tasks. The TSS uses a task scheduling approach, but is a pilot-vehicle interface rather than an autonomous agent.

The PDPC simulator employs object-oriented techniques in a knowledge-base system geared to accomplish tasks related to air combat. The simulator is modularized to conserve limited computing resources when building decision trees. Participants use a subset of the total rule base rather than pre-sorted checklists. The simulator also includes memory management utilities to release memory that is no longer needed. Information display is limited to textual output to either CLIPS development environment windows or to a log file. Position data for each agent is also captured in log files for later display using the Gnuplot interactive plotting program. Unlike the TSS and other "in-cockpit" programs such as the Pilot's Associate (PA) (Banks and Lizza, undated), PDPC was designed to close the loop of pilot decision processes and function autonomously.

2.2 The Learning Approach

2.2.1 Explanation Based Learning. Explanation-based learning is a way of improving performance based on inputs from a human controlled simulator . Like case-

based planning, explanation based learning modifies a universal plan. Unlike case-based planning, changes in agent behavior reflect actions taken by human pilots rather than modifications of a universal plan. One application of explanation based learning modifies the tactical plans knowledge base of a pilot-vehicle interface called the Pilot's Associate (PA) (Levi, 1992). A human pilot "flies" a simulator according to a predetermined script that includes a tactic for which PA does not have a plan. A knowledge engineer applies a record of the flight to a learning system shell. This shell generates an explanation-based learning plan to incorporate into the tactical plans knowledge base of PA. The resulting program applies the new plan when the learned situation occurs.

In PDPC, new tactics and maneuvers must be coded into rules and the simulator applies new rules whenever pre-conditions are satisfied. Like the explanation-based approach, PDPC gains capability when rules are added to the rule base. Unlike the explanation-based approach, a knowledge engineer must generate code based on text or verbal descriptions of tactics and maneuvers. An automated learning mechanism for PDPC has been left for future research.

2.2.2 Neural Network Based Learning. Research at the University of New Mexico resulted in a neural network model that selected aircraft maneuvers for different air combat scenarios (McMahon, 1990). The network was trained to select from 38 production rules to satisfy goals in air combat maneuvering. Results were compared to the decisions made by expert fighter pilots and contrasted with results from an equivalent rule-based production system. The neural network agreed with human pilot choices in 67 percent of the scenarios. The rule-based system agreed in only 25 percent of the scenarios. In this case, a neural network model showed improved selection performance when considering air combat maneuvering domains.

In contrast to the neural network model, PDPC contains over 400 production rules in 28 modules, however, a study comparing human pilot decisions to PDPC decisions has

been left for future research. The advantage of a rule-based simulation lies in coding of new knowledge. Where a rule-based system gains capability with the addition of rules, a neural network must be "trained" by creating and adjusting the weights of connections between nodes (Rich and Knight, 1991:Ch. 18). A rule-based system also maps more easily to a recognize-act inference cycle (Giarratano and Riley, 1989) which is characteristic of a reactive plan approach.

2.2.3 Discovery Based Learning. A thesis by Kilpatrick investigated discovery-based learning as a method of improving performance in route planning (Kilpatrick, 1992). Discovery-based learning uses heuristically-guided generation and test algorithms to create knowledge. In other words, a trial and error system was built into a free-running simulation environment. Computer generated agents incrementally changed sets of input parameters to determine which offered the greatest benefit in return. The most promising sets were stored for later trial. The final selection was the input parameter set having the highest heuristic value. Over time, agents improved performance in a variety of route planning scenarios.

In PDPC, agents incrementally change output parameters on the expectation of improving a benefit, e.g. a positional advantage over an adversary. Agents reassess positions on each move and select rules that meet predicate conditions. Predicate satisfaction depends on input parameter values which change as a simulation progresses. Subsequent actions move agents toward a goal and the process repeats. In this respect, PDPC acts as a trial and error system within a dynamic environment, but heuristic value is not retained for later trial. Instead, agents improve progress by using different rules that result in increased, or decreased output values.

Learning systems were not a primary focus of this thesis, but provide insight into the issues surrounding autonomous agents. Cooperative systems employ mechanisms used by autonomous agents, but also coordinate information and actions of cooperating agents.

2.3 Cooperative Systems

2.3.1 The Blackboard Model. The blackboard model uses modularized knowledge sources, a common data structure called a blackboard, and a control system (Barr, Cohen, and Feigenbaum, 1989:Vol. 4, Ch. 16). Knowledge sources are domain specific and operate on information contained on the blackboard. If a knowledge source finds information related to its particular part of the domain, then it either generates new information or modifies information already on the blackboard. The control system governs which knowledge source has access to the blackboard. The process of moving from source to source proceeds sequentially until no source can add or modify current blackboard information. The final state is presented as a solution.

The PDPC simulator follows the blackboard model with some exceptions. First, both common and private data structures store information. The common data structure is called a fact list (CLIPS, 1993). Information passed between rules is temporarily stored on the fact list. Private data structures consist of the class slots of object instances. Data manipulation results are stored in object slots for later use by knowledge sources. Second, not all knowledge sources have equal access to blackboard data. The PDPC rule base is modularized, but some modules must be specifically called by an agent before rules activate. For example, to check aircraft radar status, agents must place a fact on the fact list to call a module containing radar testing rules. Agents need not check radar status if conditions dictate more imperative actions, such as evading an adversary already within radar range. Third, the system is not designed to reach a final state. Scenarios continue as long as there are agents to process. In a narrow sense, the simulation reaches a final state every time an agent moves. Simulation control rules return control to executing rule modules until all agents return to their home base. In other areas, the PDPC follows the blackboard model.

The initial architectural design for PDPC stored all information in a common access data structure; the fact list. Every agent in the simulation had full access to all data. In effect, the fact list acted as a single layer blackboard. After loading agent definitions, environment definitions, and reactive plans, the number of facts exceed three hundred. Although this may not seem a lot of information, it was sufficient to require over 50 megabytes of dynamic memory during test runs. A single layer blackboard model did not meet the information capacity requirements of the PDPC simulator.

2.3.2 The Bulletin Board Model. Vernun Lun and Ian MacLeod present strategies for real-time dialogue and interaction in multi-agent systems (Lun and MacLeod, 1992). Specifically, they proposed a bulletin board model as an alternative to integrative and blackboard systems. Integrative systems allow the existence of multiple computer agents in an integrated knowledge-based system. A disadvantage of integrative systems is that each agent has its own knowledge base. Multiple knowledge bases expand storage requirements and require updating algorithms that cut across agent boundaries. Blackboard systems use a central knowledge base. Although storage space is conserved, blackboards act as bottlenecks when many agents simultaneously demand information. Bulletin boards provide an intermediate form of data pool, combining the best attributes of integrative and blackboard models. The result is that many agents can access a common information source without bottlenecks.

Lun and MacLeod go on to describe dialogue and interaction strategies between agents. By "posting" bulletins on a bulletin board, not only can agents pass messages to other agents, but information can be broadcast to all agents and time sensitive information can be monitored. Since cooperative systems depend on shared information, bulletin board models offer mechanisms to implement cooperative agents.

2.3.2 Hybrid Blackboard. The PDPC incorporates aspects of simple blackboards and bulletin boards into a hybrid structure. Information that must be passed between rules

is stored on a blackboard structure. Information passed between objects is directly posted to object slots, bypassing the blackboard. Temporary data is stored on the blackboard and removed after several time steps, implementing "event" bulletins on a bulletin board (Lun and MacLeod, 1992:672). These combined features reduced dynamic memory requirements and demand for blackboard access.

2.4 Survey of Semi-Autonomous Forces (SAF)

2.4.1 Current Implementations Several semi-autonomous agent based systems are reviewed in the *1993 Survey of Semi-Autonomous Forces*, a report prepared by the Defense Modeling and Simulation Office (DMSO, 1993). Typically in current implementations, SAF units and equipment portray ground combat missions including ground-to-air scenarios. One implementation, ARPA's Intelligent Forces program, addresses air-to-air combat, but is limited to beyond-visual-range scenarios. The Modular Semi-Automated Forces system uses a modular software structure to easily include customized behavior modules. The BDS-D Computer Generated Forces system provides each entity with "robust" behavior sets that dynamically invoked during scenarios. And the Institute for Simulation and Training developed a low-cost system based on state actions and transitions. The PDPC simulator contains features of several implementations and uses these features to implement close-in, air-to-air combat scenarios.

2.4.2 Deficiencies and Problem Areas. The DMSO Survey points out several deficiencies in current implementations. Few systems addressed Air Force platforms. Entity representations concentrated on individual platforms and provided little definition of group behaviors. Scenarios focused on close-in, ground combat with little attention paid to air-to-air combat. The PDPC simulator provides an architecture for solving all of these problems, but the main focus was in modeling cooperative decision processes.

2.5 Summary

This chapter discussed research and current implementations in the area of autonomous agents, also called computer generated forces. Agents participating in air combat simulations must process large amounts of data and prosecute a solution in a realistic manner. With the exception of reactive plans, planning systems require large storage areas to maintain pre-planned solutions. Learning systems require long lead times to compute solutions. Blackboard systems constrict information flow when processing large amounts of information. The PDPC simulator models pilot decision processes using reactive plans in a hybrid blackboard architecture. Automated learning systems have been left for future research.

3. Methodology

This chapter outlines the methods and procedures used in developing the PDPC simulator. Included are mathematical calculations required for aerial combat, an overall discussion of the CLIPS development environment and the modularized design of the PDPC simulator. The chapter finishes with a discussion of how the PDPC progresses through air combat phases and pilot decision processes. Implementation details are given in Chapter Four.

3.1 Air Combat Maneuvering

3.1.1 Introduction. The goal of air combat maneuvering is to bring a weapons system into a firing position (Shaw, 1985:1). Air combat pilots take one of two basic approaches: the "angles" fight or the "energy" fight. In the angles fight, fighter pilots seek a position advantage over an opponent. Characteristically, a position advantage means the aircraft is situated behind, and pointed at, the target. In the energy fight, pilots seek an energy advantage over an opponent. An energy advantage means the aircraft is situated above, or has a higher velocity than, the target. Pilots use basic flight maneuvers to achieve either a position or energy advantage.

Fighter pilots manipulate at least 40 quantitative parameters for each potential target during aerial combat (Titan, 1986:71). For example, combat pilots must weigh a target's potential weapons and maneuvering capabilities against his own capabilities combined with that of his partner. Each parameter is prioritized according to engagement circumstances. Combat pilots use subsets of parameters to decide strategy, tactics, target assignments,

and role of the wingman. Pilots assess each situation before, during, and after engagements. New threats are compared to existing threats to decide if or when to break-off from an engagement.

Combined with the mechanics of flight control, the domain of air combat maneuvering presents a complex and challenging coding task.

3.1.2 Vector Mathematics. Appendix A details the mathematical basis for the PDPC flight model. Overall concepts such as coordinate axis frames, forward, beam, and rear quarters, and targeting vector terminology are defined in this appendix. Mathematical functions based on the definitions in Appendix A were coded into the PDPC simulation.

3.1.3 Key Parameters. Appendix B describes the characteristics of key parameters in air combat maneuvering. Most of the parameters listed are derived from Shaw's book, *Fighter Combat, Tactics and Maneuvering* (Shaw, 1985) In air combat maneuvering there are several different kinds of parameters. Low-level detail parameters include aircraft specifications, such as maximum speed and altitude. These low-level parameters can be static or dynamic. Static means the value of the parameter should not change during the life of the platform. Dynamic means the pilot controls a parameter's value within a specific range. Some parameters can be associated with others and these associations are pointed out where necessary. Appendix B describes parameters and their characteristics.

Shaw identifies two critical parameters: specific energy and excess power. Specific energy indicates maneuver capabilities and is key to determining whether an aircraft should adopt the "energy" or "angles" fighter strategy. Specific energy depends on altitude, speed, gravity, and radial G-forces applied to the airframe. The second of Shaw's key parameters, excess power indicates climb performance. Excess power depends on velocity, thrust, drag, and aircraft weight. Although a fighter may not have an explicit energy advantage, excess power indicates the ability to gain an advantage.

Specific energy and excess power not only suggest what tactics a combat pilot should apply, they indicate whether a pilot is currently in an advantageous or disadvantageous position. A third party, such as a partner, could monitor progress in an engagement and decide to engage based on Shaw's parameters.

3.1.4 Pilot Situation Awareness. Appendix C lists a series of questions geared to measuring a pilot's awareness of combat situations. I developed these questions from an idea by Amburn who conducted a study measuring pilot situation assessment (Amburn, 1993). His study attempted to identify the merits of using virtual reality goggles versus computer monitor displays. Based on this suggestion, I incorporated key decision areas identified in the Titan report (Titan, 1986:110) with strategies and concepts defined by Shaw and generated a survey of pilot situation awareness. Survey questions range from checking radar signals, to selecting a formation and cooperative attack strategy, to planning a retreat. I used this survey to help build rule bases in the PDPC simulator.

3.1.5 Basic and Composite Maneuvers. The PDPC simulation uses two different methods to select and perform maneuvers. The first method is a "fly-to-point" calculation. Agents move toward a goal point which is defined in terms of x, y, and z coordinates. Angular and linear changes are controlled by vector mathematics functions defined in Appendix A. All an agent needs to know is the location of a goal and key parameters such as velocity and orientation change automatically. The second method is a planned version of "fly-to-point". Agents decide on a target and prepare an intercept trajectory plan. This method is used to calculate both intercept trajectories and flight routes over long distances. Maneuver trajectories over short distances use the first method. The choice of method depends on a player's phase in the mission.

3.1.6 Solo Versus Team Actions. The difference between solo and team actions appears when team players engage a target. Typically, the team leader engages threatening targets when within firing range. Depending on the strategy selected,

followers could select one of several actions. Followers could stay with the leader throughout an engagement. Followers could orbit an engagement and warn the leader of impending dangers or share the attack, trading orbit for attack maneuvers. Leader and follower could execute section tactics (Shaw, 1985:Ch. 5) in an attempt to surround a target. In any case, the strategy dictates maneuvers of follower aircraft.

3.2 System Construction

3.2.1 The Basic CLIPS Process. The primary constructs in CLIPS are facts, rules, and objects. A fact is a collection of words and symbols put into a sequential list. A rule is set of actions that will be executed if a specific set of preconditions occurs. Objects store data in slots defined by their class. CLIPS tests rule preconditions against existing facts and object slot data. If all preconditions for a rule are satisfied, CLIPS executes the actions specified by the rule.

3.2.1.1 Fact List. The fact list is an unordered list of statements indicating the current conditions in a program. Facts could be a simple statement of truth such as "radar on" or a complex list of parameters such as "target location 10.123 25.621 16.354." Facts could also mimic intelligible speech such as "message from pilot to partner is to proceed back to base". Facts placed on the fact list so that CLIPS can test them against rule preconditions. Facts are removed from the fact list if rule actions include retraction commands. Otherwise, facts remain on the fact list to test against more rules.

3.2.1.2 Agenda. The agenda is basically a waiting list containing rules that are ready to fire. At some point CLIPS may find all the facts needed to meet preconditions required by a rule. Rules that have their conditions met are placed on the agenda. When a rule comes to the top of the agenda CLIPS checks the fact list to see if conditions are still true and, if so, executes the actions specified by the rule.

3.2.1.3 Rete Algorithm. Since complex rules may require many pattern matches, CLIPS keeps track of the patterns matches it finds by using the Rete algorithm. CLIPS determines the truth, or falsity, of a rule by matching the pattern of each fact to similar patterns in each rule. Unless wild card patterns are allowed, patterns must match identically for CLIPS to put a rule on the agenda. When only a few facts are available rules may have part of their conditions met, but not enough to be put on the agenda. CLIPS keeps track of rules that only have part of their conditions met by partially instantiating each rule; incomplete matches generate a decision tree for later use. When more facts enter the fact list, partially instantiated rules progress toward completion until a complete set of matches exists. Then each fully instantiated rule is placed on the agenda for firing. While waiting for more facts, partially instantiated rules occupy memory. Many complex rules may use significant amounts of memory; enough to slow processing efficiency.

3.2.1.4 Modules. The PDPC uses modularized rule bases to improve performance and reduce dynamic memory requirements. Version 6.0 of CLIPS can be modularized, meaning rules can be grouped according to function or circumstance. Rules activate only when an agent specifically accesses a module. The key to performance improvement lies in the fact that rules not pertaining to a situation are not considered by the Rete algorithm. Effectively, CLIPS processes many small rule bases rather than one large rule base.

3.2.2 Simulation Environment. We arbitrarily defined the simulation environment to allow flexibility. Units are not assigned to all measures. Instead, units were scaled to fit the simulation once pilot decision processes were defined and tested.

3.2.2.1 Airspace and Terrain Definition. No specific airspace and terrain definitions were designed into the system An arbitrary playing grid of 1000 by 1000 by 1000 defined parameters for Gnuplot files, but agents were not restricted to specific

boundaries. Using the ratio of velocity to time increment, one unit of measure in the PDPC simulation is approximately 30 meters. Later interface to a Red Flag Terrain graphics displays allowed a playing field of 25,000 by 50,000 meters with an unlimited third component. Adapting PDPC location data output files required minor code modifications before agents remained within boundaries set by the Red Flag playing field.

3.2.2.2 Players and Stations. Initially four aircraft and four fixed stations were defined. As developments proceeded, these numbers varied to fit different scenarios. Currently, there are definitions for three playing sides: friendly, enemy, and neutral. Both friendly and enemy sides have two aircraft, one airfield, one refueling station, and plans outlining patrol routes through controlled airspace. Neutral agents use either friendly or enemy resources. In the PDPC simulator, friendly forces control southern airspace and enemy forces occupy northern airspace. Neutral forces control no airspace, but are free to move throughout the simulated environment. The term "player" defines a moveable agent within the simulation.

3.2.2.3 Movement Within the Environment. As simulation runs progress, each player is given one time increment to move in any direction. Movement is defined by a three-space velocity vector. During any maneuver, players change locations by an amount equal to the magnitude of the velocity vector. The flight model was kept simple to distinguish between decision processes and system delays. Players that did not return to the playing field, for example, revealed coding errors since the behavior could not be attributed to long system timing delays.

3.2.2.4 Simulation Timing. A fixed time increment of one second determines distance traveled based on player velocity. The PDPC simulator is not real-time since it is not coordinated with a real-time clock. Real-time measurement data reveals each iteration through rule modules varies in duration. Real-time increments depend on the number and state of players in the simulation, the number of rules in accessed rule modules, and the

system hardware used. Results in Section 5.9 indicate promise in terms of speed of operation, but rules will have to be modified to serve the purposes of CGF. A fixed time system helped concentrate design efforts on pilot decision processes rather than simulation control schemes.

3.3 Phase Control Architecture

3.3.1 Modular Rule Base System. The PDPC simulator uses a rule base that has been divided into small several rule groupings called modules (CLIPS, 1993). There are three levels of modules: main, phase control, and decision modules. Each level provides specific capabilities within the overall system architecture.

3.3.1.1 Main Module. At the top level is the main module. The main module contains procedural functions needed to produce vectored movement within the simulated environment and control rules needed to cycle simulations through time increments. Class and object definitions as well as utilities to generate plot files or Red Flag display data are also included in the main module. CLIPS begins and ends each pass through PDPC rule modules at the main module. When a player's phase slot does not contain "MAIN", control passes to a phase control module. This characteristic is used to initiate and control an infinite loop.

3.3.1.2 Phase Control Modules. Once a simulation begins, control focuses on different phase modules. Phase control modules are defined for each possible stage of a player's mission. The initial phase is called "New-Mission" since each player is given a mission to prosecute. The final phase is called "Landing" because the end of a mission typically follows a landing at some airfield. From New-Mission to Landing phases, players could traverse up to twenty different phase control modules. Phase control modules are listed in Table 3.1.

Table 3.1. Phase Control Modules.

Phase	Description
New-Mission	Receive a mission to prosecute
Launch	Take off from an airfield
Cruise	Move from point to point
Search	Look for targets
Identify	Characterize targets found
Intercept	Move between target and victim
Chase	Pursue without intent of engaging
Pursuit	Pursue with intent of engaging
Engage	Maneuver to a firing position
Acquire	Select and arm a weapon
Fire	Launch selected weapon
Analyze	Gauge effectiveness of weapon
Breakoff	Abort an acquire-fire-analyze sequence
Disengage	Abort an engagement
Avoid	Maneuver against a missile attack
Evade	Maneuver against a possible engagement
Retreat	Escape from a possible engagement
Refuel	Rendezvous with a tanker
Recall	Return to home base
Landing	Land at home base

Phase control modules establish a linear sequence of rule base groupings. Each module contains only those rules needed to execute actions in that phase of a mission. For example, in Launch phase players concentrate on taking off from an airfield, but in Search phase players concentrate on finding a target. While in Launch phase, players do not search for a target, and while in Search phase, players do not try to take off from an airfield. A single rule within each phase allows transition into another phase. For example, if a player finds a target while in Search phase a transition to Identify phase allows a player to determine whether that target is friendly, enemy, or neutral. If the player and target are of opposing sides, friendly versus enemy, the player transitions to Intercept phase and executes rules needed to prosecute an intercept trajectory. Once the target is intercepted, the player transitions to Pursuit phase and then to Engage phase if

the target comes within striking distance. Phase transitions progress, or regress, until mission time limits expire when players return to their home base.

3.3.1.3 Decision Modules. Planning and decision-making beyond the scope of prosecuting an established plan is conducted within decision modules. Each module is based on pilot decision areas identified in the Titan report (Titan, 1986:110-114). There are six decision modules containing rules ranging in scope from pre-engagement posturing to post-engagement consideration. Decision modules are itemized in Table 3.2

Table 3.2. Decision Modules.

PHASE	Description
Pre-Engagement	Checks radar, correlates target data, estimates threats, and begins commitment
Engagement Strategy	Selects a strategy, attack formation and approach to the target
Intercept Geometry	Calculates routes for intercept and bingo plans
Weapons Employment	Assigns targets, sets engage/abort criteria, selects and assigns weapons
Counter Action	Selects maneuvers, phase, and actions based on target activities
Post-Engagement	Decides whether to re-engage or disengage based on success of an attack

3.3.2. Simulation Control. Simulation runs proceed based on state information contained in player class definitions and global information stored on the fact list. Rules fire provided data is continuously updated. If new data is not generated or old data remains unmodified, CLIPS stops rule execution and simulations stop. Utility rules ensure simulation runs continue after each iteration.

3.3.2.1. Data Structures. There are three data storage mechanisms. First, data can be stored on the fact list as a simple fact. Permanent data and iteration control facts are defined at initialization and stored on the fact list. Second, temporary data is

stored on the fact list in a template. Templates are defined at initialization, but are not placed on the fact list unless specifically required during rule processing. The distinction between simple facts and templates is the order of placement on the fact list. Fact data stores old information that is replaced with modified data. Template data stores new information that is removed when consumed. The third data storage mechanism uses object slots to hold information. Slots hold initial data defining each player state and store modified data as simulations progress. Facts hold global information, templates allow message passing, and object slots allow multiple players in different stages of processing.

3.3.2.2. Focusing on a Module. Two mechanisms change the current rule module. First, the CLIPS "focus" function call changes the currently executing rule module. The focus function allows players to progress through phase control stages and call decision modules as needed. The second mechanism changes phases based on a rule in the main module which compares player phase states to the currently executing module. If the player's phase is not "MAIN", control is focused on the phase module required by the player. This allows automatic control transfers and players in different phases of missions. Switching between rule bases reduces hardware resource requirements by limiting the number of rules under consideration. It also greatly improves pattern matching speed under the Rete algorithm and results in improved overall performance.

3.3.2.3. Simulation Control. The entire simulation is controlled in the main module. On each iteration, or time step, "moveable" players analyze their situation and decide on direction, velocity, orientation and other parameters in preparation to "move." Once all players in a particular phase are ready, a single rule in the main module completes all movement calculations and stores modified data in each player's object slots. In other words, each player moves. CLIPS is focused on phase control modules until all players have been "moved." Utility rules monitor when every moveable player has moved which prompts simulation control rules to reset every player's state to "moveable." History

objects, plot files, and other outputs are updated on each iteration. Control is returned to phase modules by activating the focus control rule for each player and the cycle repeats.

3.3.2.4. Input and Output. Input and output remained simple since we concentrated on pilot decision models rather than interfacing. The primary output is text based and describes player states and decisions as a simulation progresses. A format statement embedded in each rule and function is switched on or off depending on the setting of global boolean variables defined for each module. When a variable is set to "t", a statement describing each rule firing within a module is sent to the standard output terminal, usually the display screen. When a variable is set to "nil", no statements appear. If a variable is set to "log", statements are sent to a log file recording the simulation. PDPC can also generate data files compatible with the Gnuplot interactive plotting program. Figures presented in Chapter Five were generated using Gnuplot and PDPC generated data files. An output available on "windows" based systems, the CLIPS development environment displays the current state of CLIPS programs. In PDPC, module names are equated with phase states, so simulation progress can be monitored on the fact list, agenda, and focus stack windows.

A complete I/O interface is available, but not fully implemented. CLIPS provides a function to read and write class structures to a file. When used, simulation progress can be monitored by reading and interpreting state data held by each object. In the UNIX version of PDPC, file-based class structure I/O can be changed to pipe-based. Using two C language utilities and a semaphore control algorithm, PDPC can support players originating in external programs. The interface is rudimentary, but provides complete access to aircraft class structure definitions while simulations execute.

3.3.3. Scenario Definition. Given a player definition and an initial plan to follow, the PDPC can enact different engagement scenarios. The scenarios are not scripted, but unfold with dynamic interaction of autonomous agents. Currently only testing and defense

missions are defined in detail. Air superiority and offensive rules are defined, but not fully tested. Testing and defensive mission assignments provided PDPC with sufficient capability to investigate cooperative behaviors.

3.3.3.1. Player Definition. Players are defined as class objects. Each object has slot to hold state data during simulations. At start-up, players must have a name, a side, a role, and a mission. Default names in PDPC are pilot, partner, bogey1, and bogey2. Pilot and partner are on the "friendly" side while bogey1 and bogey2 are on the "enemy" side. Neutral players are not pre-defined. A player's role can be one of leader, follower, or solo. A player's mission can be one of testing (for drone aircraft), defense, offense, or superiority. Each player's initial location must match an airfield location because the first movement rules execute a launch sequence from a base. All other object slots are updated dynamically as each simulation unfolds.

3.3.3.2. Mission Assignment. Initially, mission statements are part of each player's definition. The simulator makes mission assignments depending on initial player mission purposes. For example, if a player's mission is "defense," the initial mission assignment will be "CAP-station." A player's mission assignment changes during simulation runs. Player's switch from "CAP-station" to "patrol" depending on how long a simulation has run. A CAP station plan follows a closely spaced series of waypoints while a patrol plan follows a dispersed series. A player's mission assignment is "completed" upon return to a home base.

3.3.3.3. Team Assignment. Initially, players on the same side are not associated. Once the simulation begins. followers are assigned to a leader on the same side. Leaders dictate the mission purpose, assignment, and plan. Cooperative behavior rules ensure followers recognize the leader's role and act accordingly.

3.3.3.4. Plan Assignment. Once a player has a mission purpose, mission assignment and team assignments, a plan appropriate to the mission is selected. Currently,

only "patrol" and "CAP-station" plans are initially defined. Intercept and bingo plans are created and defined when needed. Behaviors in other phases, including plan selection, are invoked as players react to detected adversary players.

3.3.4. General Sequencing. The general sequence of events follows the phase control sequence. Once all assignments are complete, players proceed to Launch phase for take-off. When a launch is successfully executed, players cruise to their first waypoint. Players change to Search phase when they reach a waypoint. If no targets are detected, players cruise to the next waypoint. The simulation continues until scheduled mission time expires, the aircraft runs low on fuel, or a potential target is detected. If mission time expires, players return to their home base. When aircraft run low on fuel, they proceed to the closest refueling station. If a potential target is detected, players begin a tactical decision sequence leading to an engagement. When a threat is eliminated (or runs away), players return to their original mission assignment.

3.4. Summary.

This chapter defined the PDPC simulator architecture and implementation methods. A phase control architecture provided an easy means of coding air combat maneuvering, tactics, and strategy as well as build a simulation control system. The simulation environment was arbitrarily defined to allow flexible development. Vector mathematics formulae, data definitions, and pilot situation awareness are detailed in appendices.

4. Implementation

This chapter details the architecture and design of the PDPC simulator. Except where stated, notation conventions follow object-oriented design techniques (Rumbaugh, 1991). Section 4.1 begins the chapter with a description of the PDPC object model. Definitions of each class and subclass include summaries of slot definitions. A summary of data slots is contained here to illustrate relationships between objects and rules. Comprehensive slot definitions are contained in Appendix D.

Sections 4.2 through 4.3 describe the PDPC dynamic model. Section 4.2 provides a high level description of the PDPC dynamic model. Interaction between main, phase control, and decision modules is illustrated using state diagrams. Section 4.3 describes the dynamic model of each phase control module. Each dynamic model illustrates which rules cause transitions to other modules, which rules interact with other rules, and which rules result in player behaviors. Accompanying discussions describe the purpose of each module and the expected behaviors caused by key rules. Section 4.4 describes the dynamic model of each decision module. Decision modules are partitioned into sub-modules which are called depending on the requirements of players in the simulation. Players access more extensive decision processes in decision modules than offered in phase control modules. Cooperative behaviors are also treated more in depth in decision modules.

Section 4.5 provides a brief discussion of the PDPC functional model. The functional model of individual rules is nearly identical, so only player movement and simulation driver rules are modeled in depth.

The chapter concludes with discussions on implementing specific cooperative behaviors. Changing missile goals, swapping player roles, and executing cooperative maneuvers illustrate how PDPC elicits cooperative behaviors from players.

4.1 Object Model

The object model consists of a hierarchical class structure of players and plans. The overall class structure is shown in Figure 4.1. The "user" class marks the end of CLIPS standard classes and the beginning of the PDPC hierarchy.

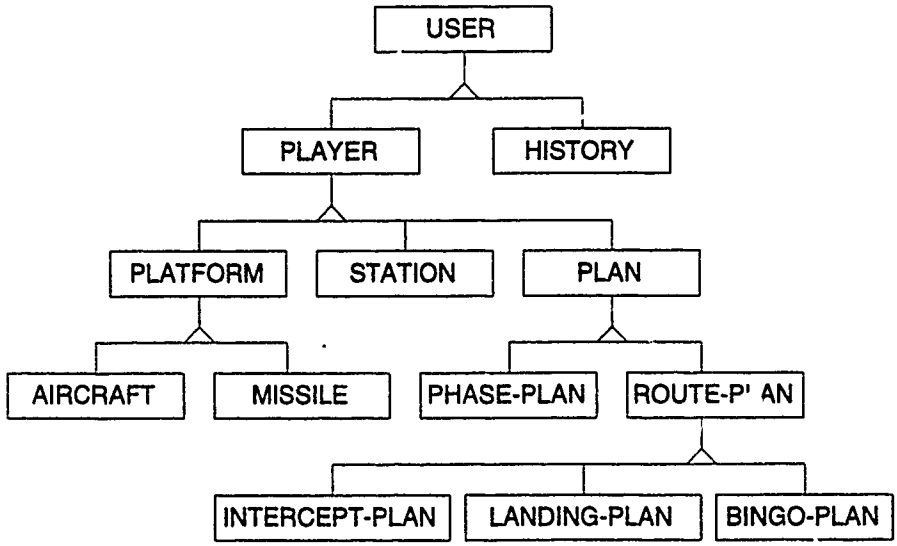


Figure 4.1. Class Structure.

4.1.1 Class Hierarchy. The class structure consists of twelve classes and subclasses arranged in four layers. Each subclass layer inherits slots from its superclass. The "player" class declares two slots: name-of and side. The name-of slot holds a symbol identifying a player object. The side slot indicates to which team a player belongs, one of friendly, enemy, or neutral. History objects are associated with each moveable player and record the last five locations of a player. History data is used to project paths and predict

player locations during pursuit and intercept phases. Subclasses inherit player class slots, but not history class slots.

4.1.2. Platform Definitions. The platform class defines moveable players such as an aircraft or missile. Each platform has location, velocity, and orientation vectors to define position within the environment. Moveable players are further characterized by phase, goal and control information. The phase slot indicates which phase module the agent requires. Goal data includes the name and location currently targeted. Control fields indicate player changes to aircraft velocity, acceleration, thrust, etc. within the aircraft body coordinate frame. Platform data slots are modified by rules dictating movement through the simulation.

4.1.2.1. Aircraft Definitions. The aircraft class defines the primary vehicle in the simulation. Aircraft class slots fall into one of several categories. State data includes the phase slot and a state slot which indicates whether the aircraft is moveable or not. Mission data describes which aircraft belong to which side, who are leaders and who are followers, the mission, current assignment and plan of action. Navigation and platform data slots are inherited from the platform class. Strategy data slots describe the approach, formation, and tactical coordination method one team uses against an opponent. Tactical data reflects the maneuver currently being executed as well as target threat information including prioritized target lists. Avionics slots indicate the current status and operating mode for radio, radar, and electronic counter-measure systems. Initial aircraft definitions are shown in Figure 4.2

4.1.2.2. Missile Definitions. The missile class defines the second vehicle in the simulation. Missiles inherit platform class slots upon instantiation. Missile class slots define missile capabilities, status, and ownership. Missiles are dynamically instantiable players. Aircraft objects create missile objects to fire at opponents. Other players must be defined at simulation start-up. Missile class slots are shown in Figure 4.3.

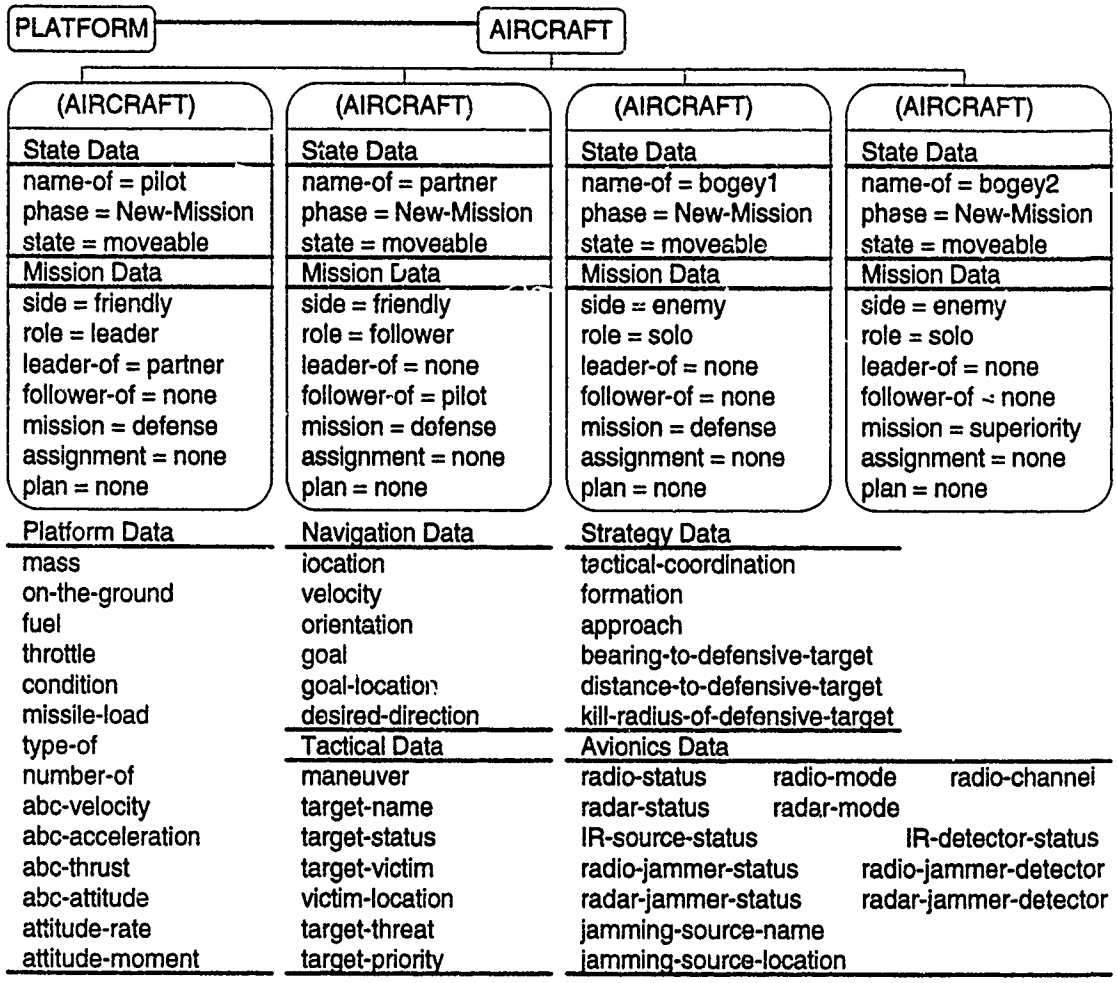


Figure 4.2. Aircraft Instances and Slots.

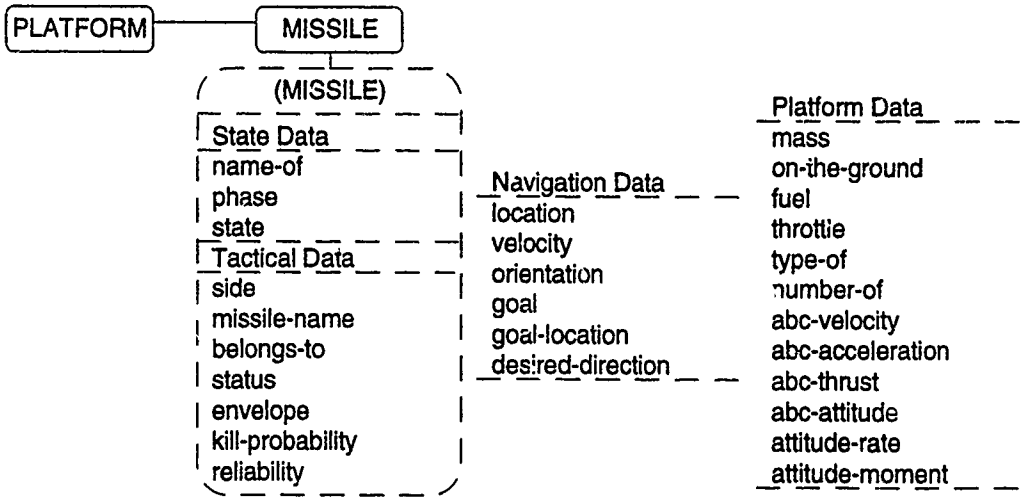


Figure 4.3. Missile Instance and Slots.

4.1.3. Plan Definitions. Two plan classes and three subclasses define the extent of planning in the PDPC simulation. Each plan is basically a template for dynamic planning. The main plan classes are shown in Figure 4.4.

4.1.3.1. Phase Sequence Plans. Phase plans control the sequence of phase modules. Some rules do not contain coded phase transition data. Instead, the next phase is looked up in the phase plan. Phase sequencing can be changed by changing the order of phases in the phase plan. Each side in the simulation has one phase plan.

4.1.3.2. Route Plans. A route plan is a pre-defined description of a sequence of waypoints. The location of each waypoint and the sequence to be followed is contained in each route plan. Aircraft select waypoint locations based on their current location. Route plans must be instantiated at start-up.

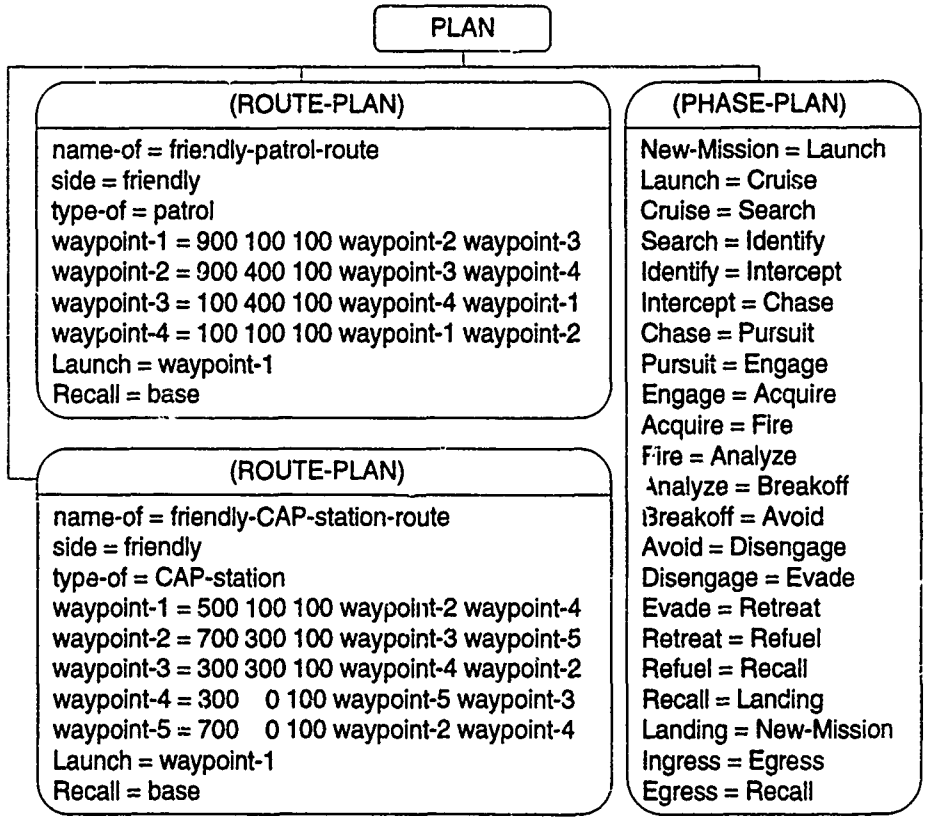


Figure 4.4. Initial Route and Phase Plan Instances.

4.1.3.3. Landing Plans. A landing plan is defined for each airfield. Instead of generic waypoint definitions, landing plans contain specific data geared to lead a player to the end of a runway. A landing plan is depicted in Figure 4.5.

4.1.3.4. Bingo Plans. Bingo plans are route plans that lead to a refueling station. The difference between bingo plans and route plans is that bingo plans can be modified depending on the location of aircraft. During combat, aircraft that are low on fuel return to the closest bingo field to refuel. Bingo plans allow calculation of the shortest distance to fuel. Before entering an engagement, players update their bingo plan with an new entry point and declaration of primary and secondary routes. The entry-point is the current location when the plan is updated. Waypoints are chosen to provide optional routes out of an engagement and back to refueling stations. The primary bingo field is the one with the shortest overall route.

4.1.3.5. Intercept Plans. Intercept plans are route plans that lead a player to a target even if the target choses a third party as its victim. Intercept plans calculate a path to a target while accounting for the fact that the target is moving. If the target is moving toward a third party, such as a base or tanker, intrcept trajectories are modified to move a player toward the victim instead of the target. Intercept plans are dynamically created to fit each potential intercept. Up to six waypoints can be used to define the intercept trajectory. Two more waypoints are reserved for orbit point definitions.

Orbit points provide waypoints for follower aircraft once a leader engages a target. Followers modify orbit points to stay close to an engagement in case lead aircraft get into trouble. If trouble occurs, follower aircraft engage the target and leader aircraft follow the orbit points. A typical intercept plan is shown in Figure 4.5.

4.1.4. Environment Definition. Environment definitions are stored in one of two ways: as global facts on the fact list or as station objects. Currently, four fixed stations are defined in PDPC simulations: two airfields and two refueling tankers. Each side is

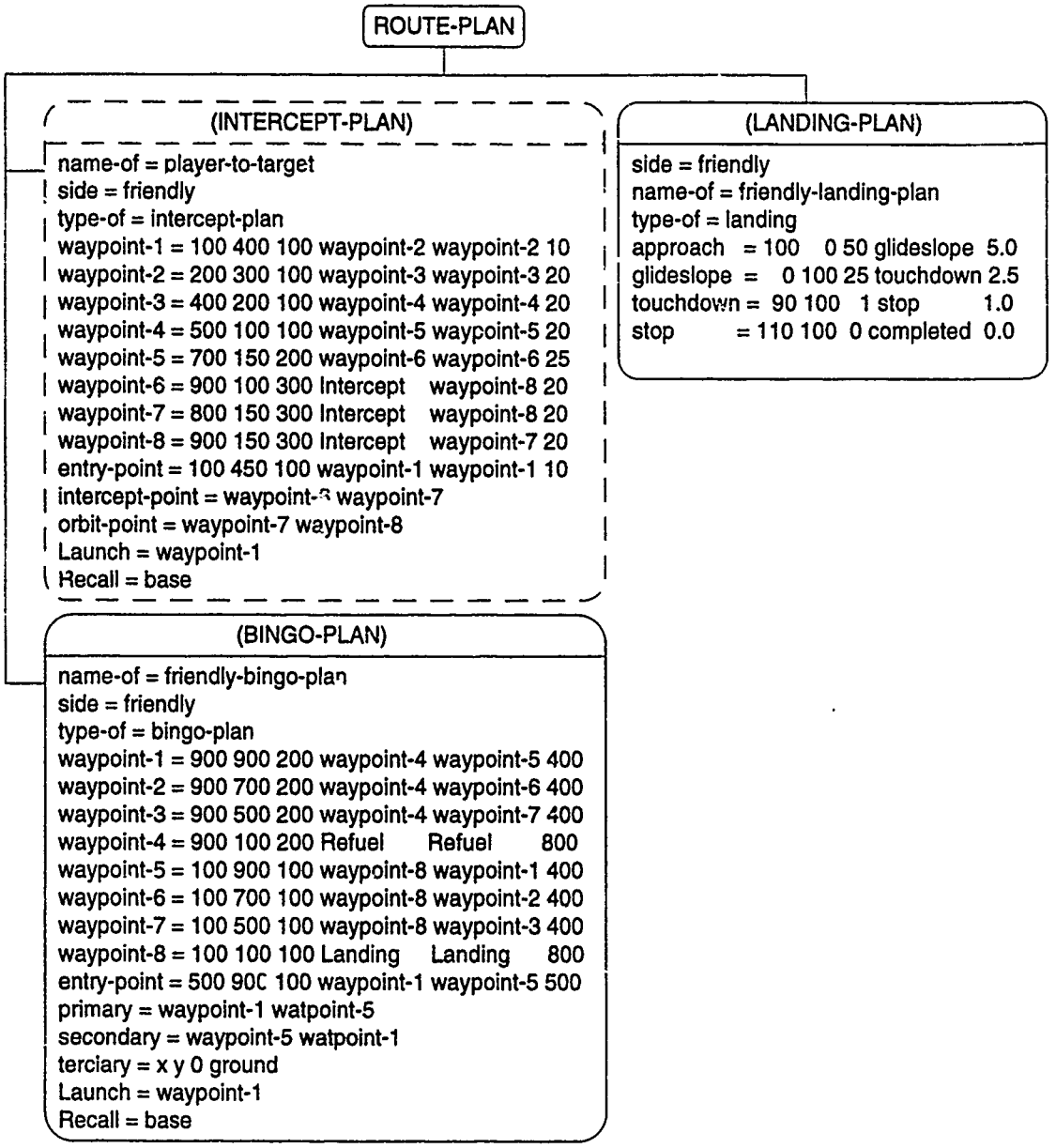


Figure 4.5. Intercept, Bingo, and Landing Plan Instances.

assigned one airfield and one tanker apiece. Tankers are defined as stationary to simplify scenarios while developing pilot decision modules. The boundary between opposing sides and wind velocity are defined as global facts. Station objects are shown in Figure 4.6.

4.1.5. History Objects. Each platform class object is associated with a history object. History objects store short term location data. Up to five consecutive locations

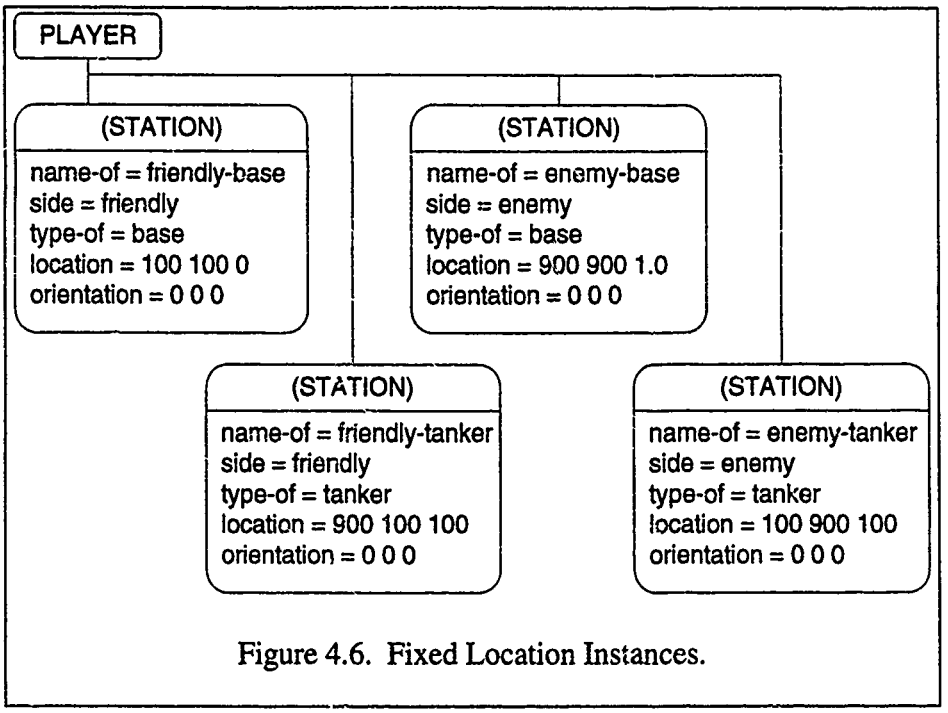


Figure 4.6. Fixed Location Instances.

can be recorded by a history object. Location data is used to compute both short distance and long distance intercept points. In a short distance intercept, the first three locations are used to predict the next location. In a long distance intercept, the first, third and fifth locations are used to predict a location after ten time steps. Short distance intercepts are used in engagement maneuvering while long distance intercept are used in planning intercept trajectories. Figure 4.7 depicts history object instatnces for each default player.

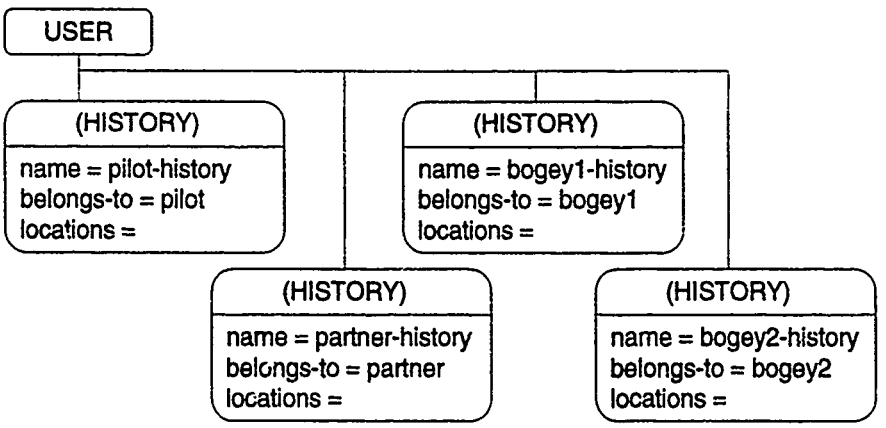


Figure 4.7. History Instances.

4.2 Dynamic Model.

The dynamic model spans 27 different modules, but basically there are three types: main, phase, and decision modules. This section begins with an overall dynamic model of the phase control sequence followed by a discussion of the relationship between phase modules and decision modules. Phase control modules comprise the largest part of the PDPC architecture, but decision modules provide deeper reasoning capabilities and model cooperative decision-making processes. This section concludes with a discussion of the main module and its role in PDPC simulations. Detailed discussion of individual phase and decision modules is contained in Sections 4.3 and 4.4.

4.2.1. Phase Control Sequence. The phase control sequence begins with the New-Mission module and concludes with the Landing module. As a simulation progresses, players change phase modules to meet different situations. For example, once mission assignments have been made in the New-Mission phase, players transition to the Launch phase and take-off from the airfield. Rules within each phase detect conditions warranting phase changes, execute actions if a phase change is not warranted, or access decision modules to determine appropriate actions. Figure 4.8 maps PDPC phase transition paths.

4.2.2. Decision Modules. Decision modules can be called at any point in the phase sequence. The ordering shown in Figure 4.9 indicates a rough parallel to phase sequencing. Pre-Engagement rules typically activate during Launch through Intercept phases. Engagement-Strategy rules initially activate during Intercept phase, but can be called during chase, pursuit or engage phases. Intercept-Trajectory rules are also initially activated in Intercept phase, but can be called at any time. Weapons-Employment, Counter-Action and Post-Engagement rules activate during or after Engage phase. Decision rules provide a more detailed analysis and planning mechanism than phase sequence rules.

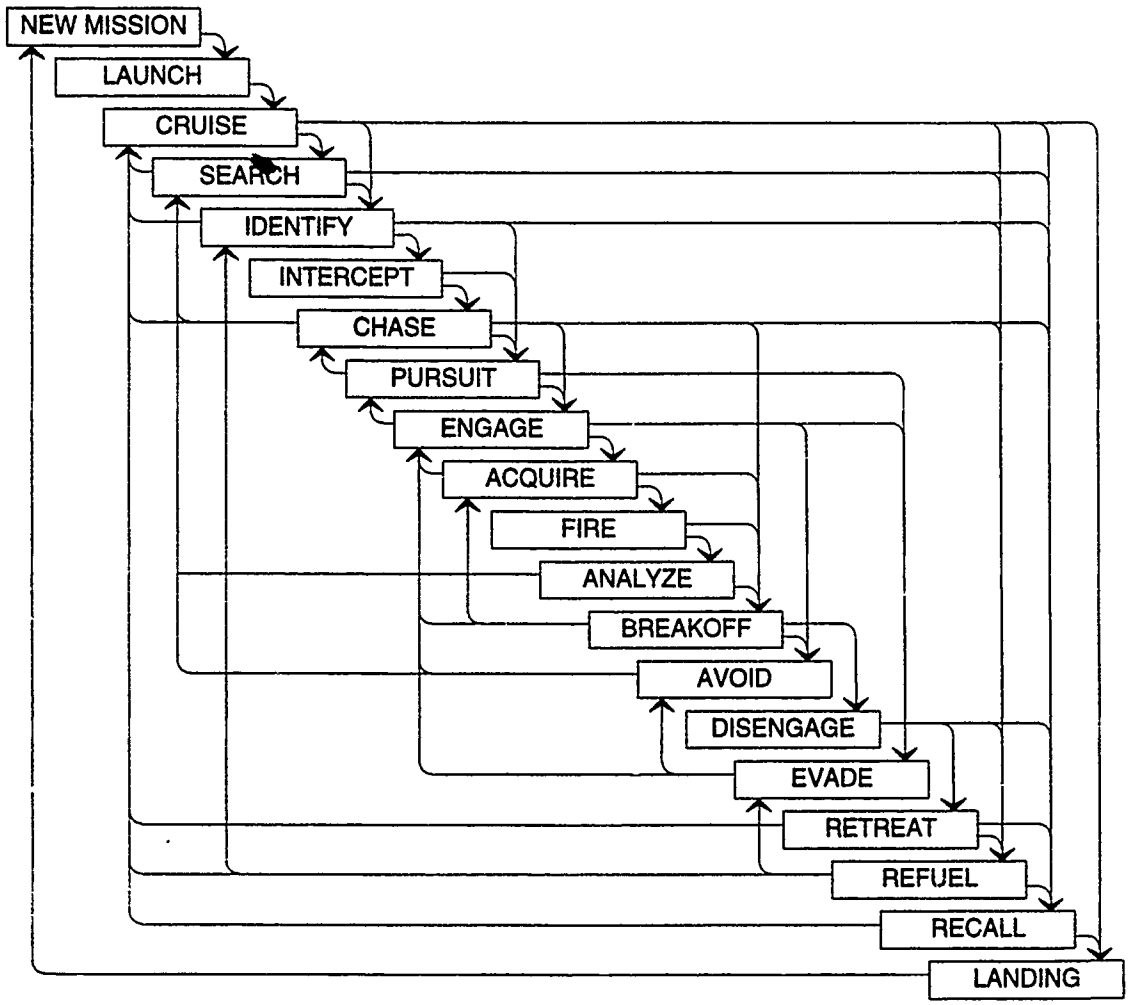


Figure 4.8. Phase Control Sequence Map.

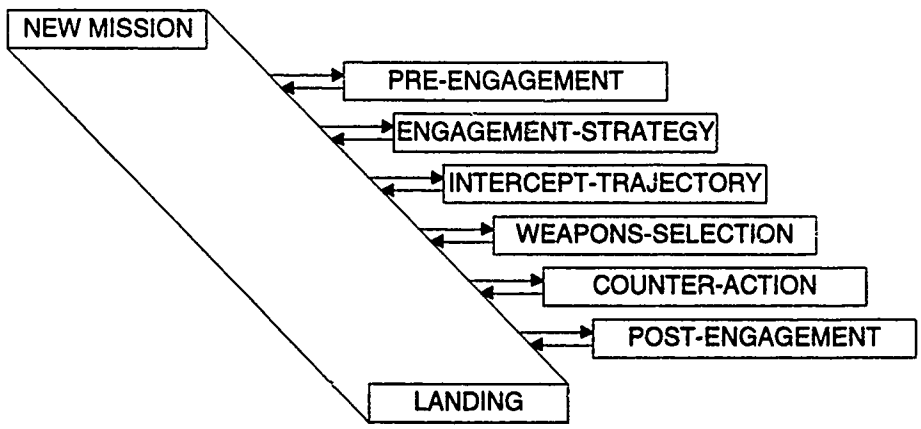


Figure 4.9. Decision Rule Modules.

4.2.3. Role of the Main Module. The Main module is the CLIPS definition for no module at all. Once all rules in a particular module have had a chance to fire, control returns to the Main module. Utility and control rules in the Main module provide simulation updates and generate output data files. The relationship between the Main module and all other modules is shown in Figure 4.10. Note that decision modules return control to phase sequence modules before control returns to the main module. This ensures the most current data is in place before movement rules within the Main module change player location, orientation, velocity and other parameters.

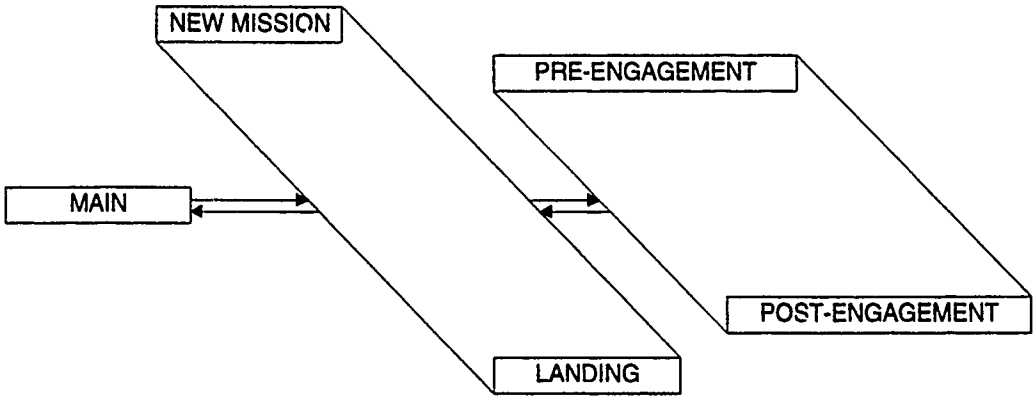


Figure 4.10. Main Module Control.

4.2.4. Main Module. Rules in the main module are shown in Figure 4.11. Each parallelogram represents a rule that fires during simulations. Arcs within the main module roundangle represent direct control. For example, the "operator" rule must fire before the "make-the-move" rule can fire. Data dependence ensures rules fire in a specific order, but rules can also fire independently. For example, "move-player" is the main rule of the simulation. Each player ready to move causes "move-player" to fire which, in turn, changes that player's location, speed and orientation. Once all players have moved, the "iterate" rule fires which leads to a new move for each player. The simulation is focused on the phase module dictated by the phase of each player and the cycle repeats.

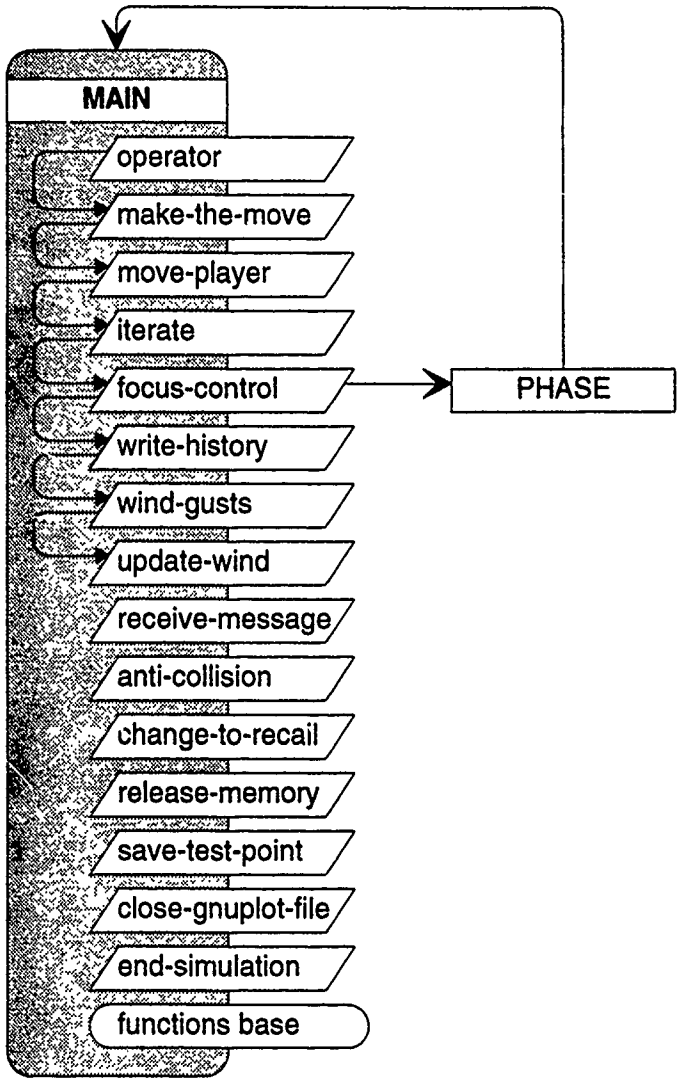


Figure 4.11. Main Module Rules.

Except for the "receive-message" rule, remaining rules provide simulation control and system level input/output. For example, the "anti-collision" rule ensures two objects do not occupy identical location coordinates. The "save-test-point" rule saves current object structures, including slot data values, to a file for later analysis. The exception is "receive-message" rule. This rule focuses players into a rule-base that processes communication between cooperating players. Messages such as "Follow me!" and "Missile warning!" trigger automatic responses in the receiving player. The typical response is a change oi phase with cooperating players also changing roles.

The final element of tne main module is the functions base. All functions, including basic math functions reside in the Main module. Putting functions in the Main module allowed easy access to common numerical procedures such as calculating target distance, relative velocity, and bearing. Each phase and decision module has visibility to all functions in the Main module.

4.3 Phase Module Dynamic Models.

This section illustrates the dynamic model of individual phase control modules. Detailed discussion of the pursuit through evade phases is provided by Hluck (Hluck, 1993). All other phase control modules are discussed in this section. Decision modules are discussed in Section 4.4.

4.3.1. New-Mission Phase. New-Mission rules give mission assignments to available aircraft. A new mission is selected based on mission purpose. Each mission is married to a default route plan. Once an aircraft has a mission assignment, the phase changes to Launch phase and scenarios begin. Upon return to a base, aircraft stand idle until all team players have landed at the base. Then a new assignment is given and the simulation restarts. Figure 4.12 shows the dynamic model of the New-Mission phase.

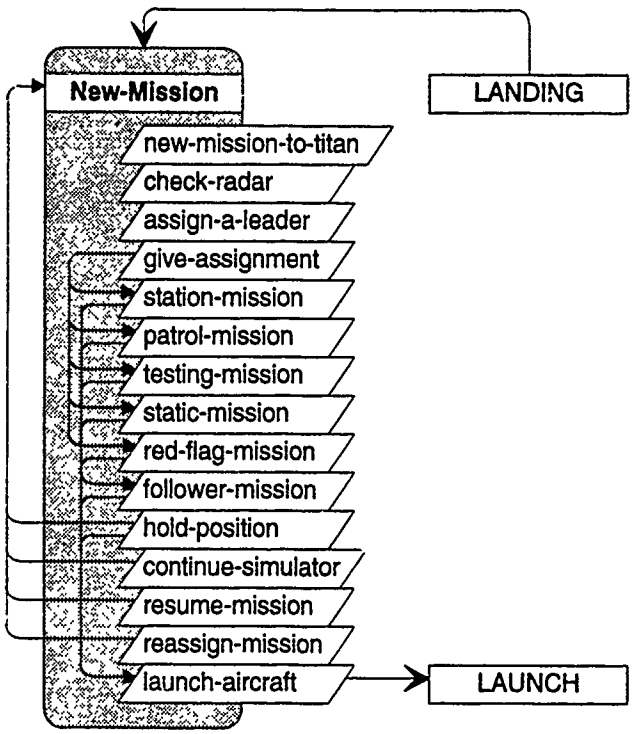


Figure 4.12. New Mission Phase.

4.3.2. Launch Phase. Launch phase follows new-mission phase as depicted in Figure 4.13. Aircraft depart their home base by accelerating to take-off speed and increasing pitch angle to gain altitude. For simplicity, we assigned either an west-to-east or north-to-south orientation on runways. Aircraft depart on an initial orientation of (0, 0, 0) or (0, 0, -pi/2) depending on which airfield is home base. Leader aircraft depart first if there is an assigned leader. Follower aircraft remain stationary until leaders have accelerated.

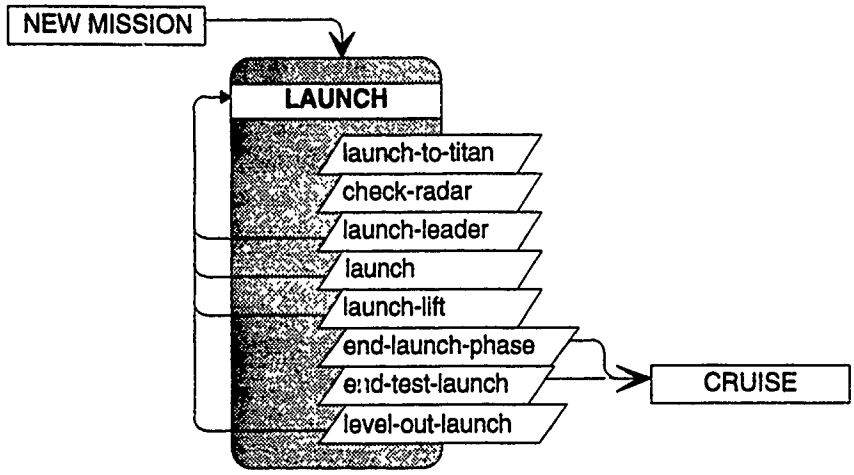


Figure 4.13. Launch Phase.

While within 50 units of the base, aircraft accelerate to cruising speed and continue gaining altitude. Once an aircraft is more than five units away and at least one unit up, the climb is reduced toward level, upright flight. When players are more than 50 units from the base, the Launch phase ends and Cruise phase begins.

4.3.3. Cruise Phase. Once in cruise phase, the assigned mission is prosecuted according to an assigned plan. Aircraft move from waypoint to waypoint while checking for potential targets. Radar status and operating mode is checked against rules in the Pre-Engagement decision module (discussed in paragraph 4.4.1). If a target is detected before reaching a waypoint, aircraft transition to Identify phase. Otherwise, a long range search is conducted upon reaching a waypoint. Low fuel warnings, expired mission time, or approach to home base are handled by transition rules to Refuel, Recall, and Landing phases, respectively.

Four rules specifically control the actions of follower aircraft. A "follow-the-leader" rule gives follower aircraft a goal of their leader so that team aircraft fly in formation. "Follower-solo" changes a follower's goal from a leader to a waypoint so that follower aircraft fly independently of the leader. "Follower-reformation ' resets the follower's goal to the leader so that formation flying can be resumed. "Follower-assignment" coordinates

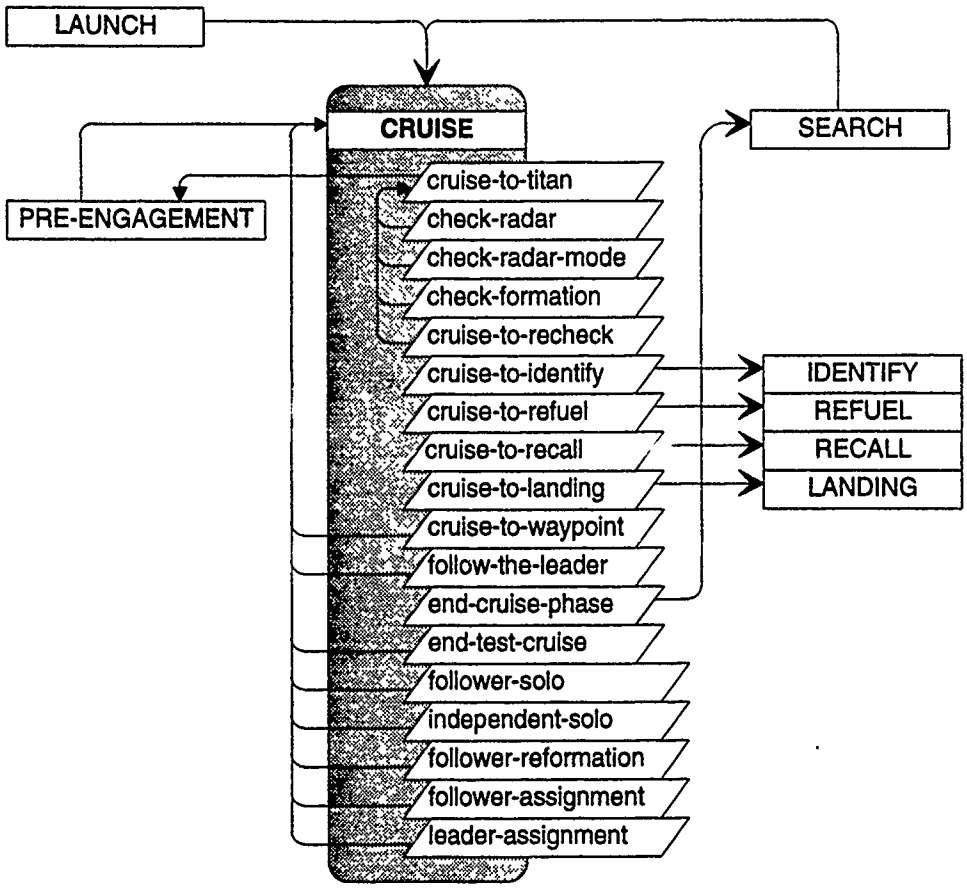


Figure 4.14. Cruise Phase.

leader and follower mission assignments. With these rules, team players move in unison toward a patrol station, separate to fly opposite sides of a patrol area, and rejoin to address threats or prosecute new missions.

4.3.4. Search Phase. The Search phase is depicted in Figure 4.15. Aircraft enter Search phase upon reaching a waypoint and conduct a long range search. Radar range in Search phase is double that of any other phase to simulate pilot concentration on a search task. If no targets are detected, aircraft erase old target data from their target list, return to Cruise phase, and proceed to the next waypoint. The "follower-search" rule ensures follower aircraft fly waypoints in a different order than leader aircraft during CAP-station assignments. The "follower-message" rule generates a message template from leader to

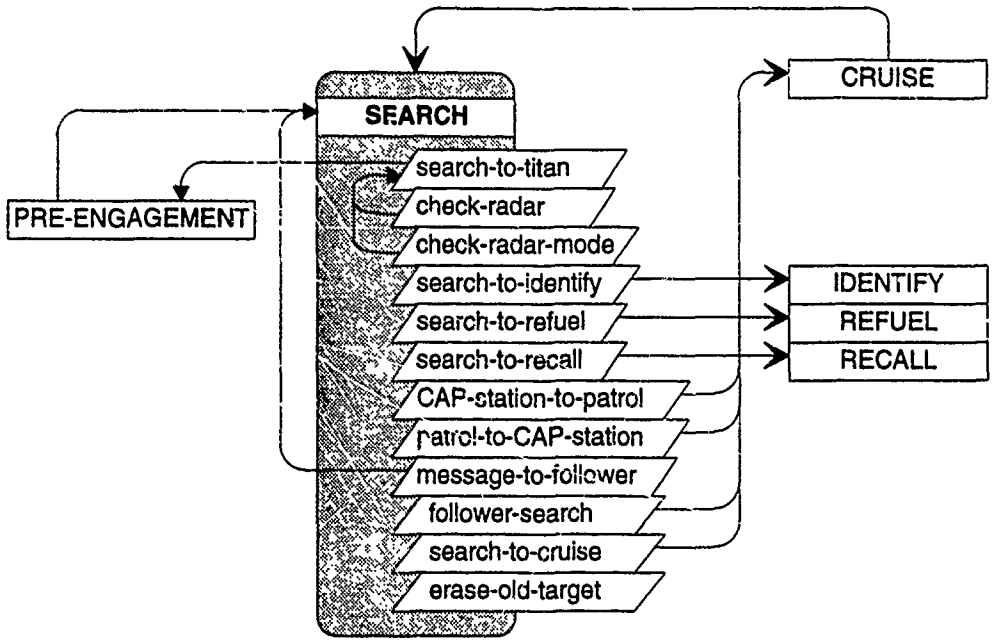


Figure 4.15. Search Phase.

follower aircraft so that cruise reformation rules activate. If a target is detected while in Search phase, a player's phase changes to Identify phase.

4.3.5. Identify Phase. If a potential target is detected, aircraft enter the Identify phase. The Identify phase is depicted in Figure 4.16. Targets not currently on the target-

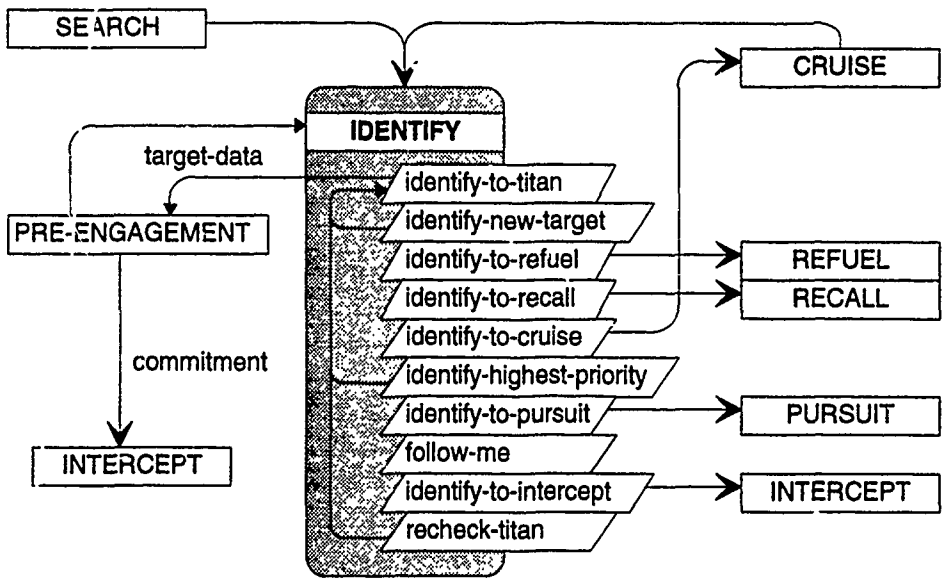


Figure 4.16. Identify Phase.

name list cause change into the Pre-Engagement module to reassess target data and gauge whether to enter an intercept trajectory. If the potential target turns out to be either neutral or on the same side, control returns to the Cruise phase. If commitment to a target is deemed necessary by Pre-Engagement rules, phase changes to Intercept phase.

4.3.6. Intercept Phase. Once in Intercept phase, both the Engagement-Strategy and Intercept-Geometry decision modules become active. Players select a strategy and plan an intercept trajectory. Intercept trajectories are based on three elements: aircraft location, target location, and victim location (or target's intended target). Players begin an intercept route once an intercept plan is created. Intercept phase ends when the aircraft is within 50 units of the potential target, at which point phase changes either to Chase or Pursuit phases. Before a transition out of Intercept phase, players update a bingo plan that guides them to a refueling point. The Intercept phase is shown in Figure 4.17.

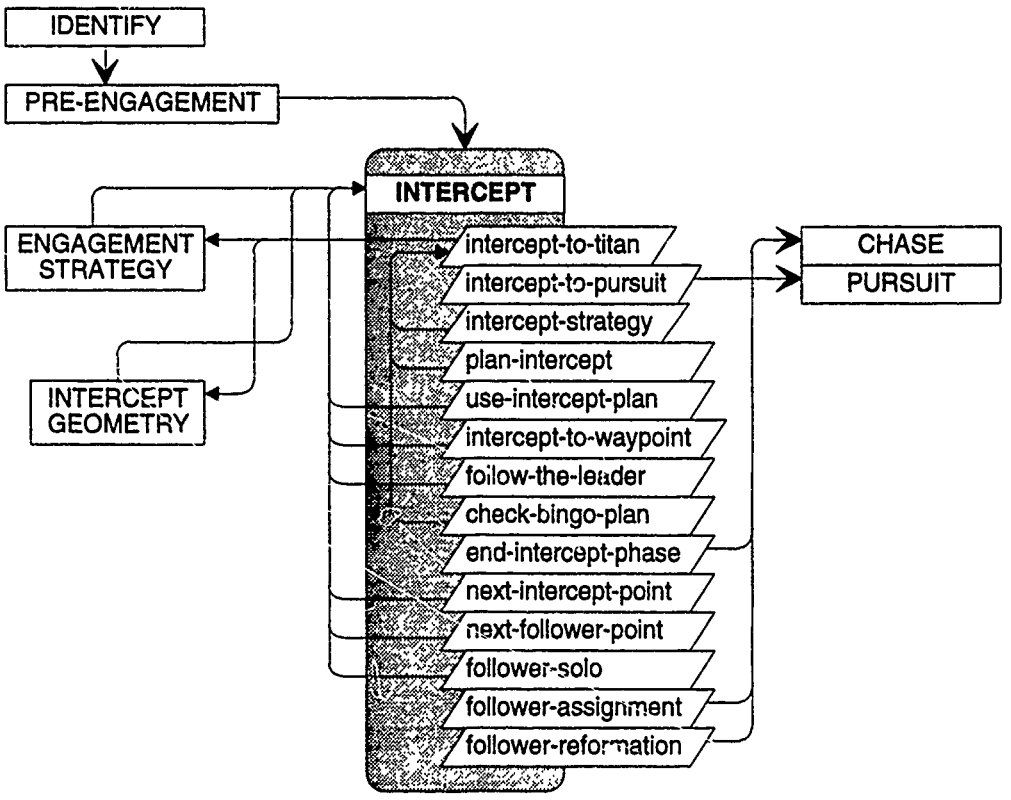


Figure 4.17. Intercept Phase.

4.3.7. Chase Phase. If a potential target turns and runs, aircraft enter the Chase phase. In Chase phase, aircraft pursue targets to the border and then return to a patrol mission in the Cruise phase. If the target turns toward the aircraft, phase changes to Pursuit or Engage phase depending on target distance. If a higher priority target enters controlled airspace, aircraft break off the chase and address the new target. The Chase phase is modeled in Figure 4.18.

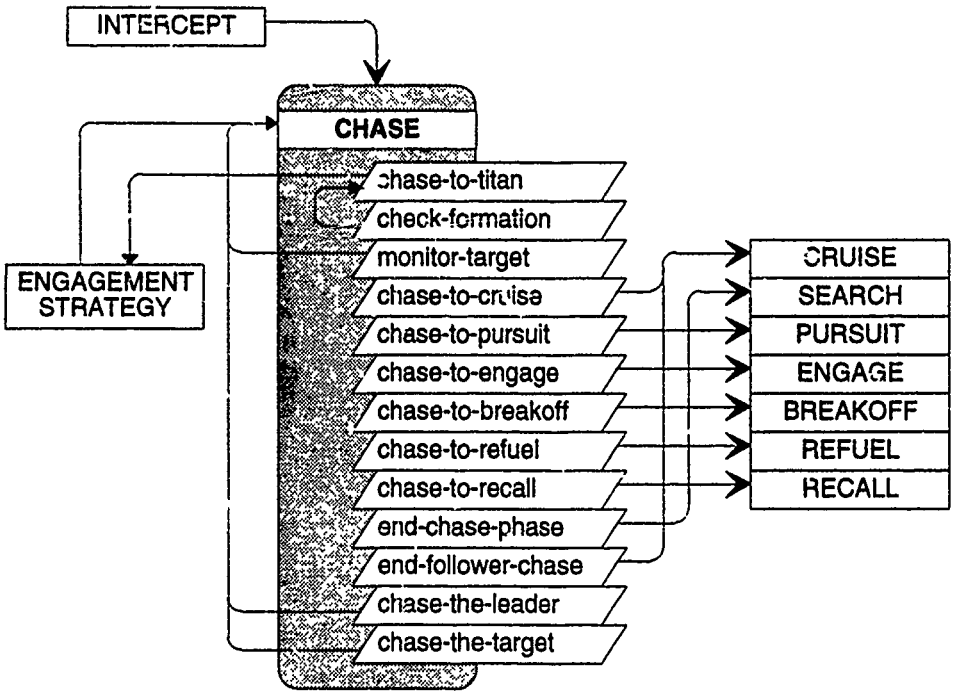


Figure 4.18. Chase Phase.

Three cooperative rules define the leader/follower relationship in the Chase phase. The "check-formation" rule queries the Engagement-Strategy decision module for a formation the follower should use. Once a formation is assigned, followers "chase-the-leader" as the leader executes an attack sequence. When the leader returns to Cruise phase, followers end their chase and also return to Cruise phase. The Chase phase allows followers to more aggressively conduct basic flight maneuvers than when in Cruise phase.

Followers can also switch to Engage phase if leader aircraft get into trouble or new threats come within detection range. Staying close to Engage phase obviates the need for a Search-Identify-Intercept sequence on the part of followers.

4.3.8. Pursuit Phase. Once within range of the target, leader aircraft enter pursuit phase while follower aircraft chase along in formation. The Pursuit phase is shown in Figure 4.19. Pursuit rules check the pursuit method and calculate an appropriate fly-to point, speed, and altitude. Aircraft move from Pursuit to either Engage or Evade phase and back again as aircraft "jockey" for position. Transition from Pursuit phase depends primarily on distance from the target. Detailed discussions of Pursuit phase rules are given by Hluck (Hluck, 1993).

4.3.9. Engage Phase. The engage phase is a highly dynamic and complex module. Aircraft consider lead, pure, and lag pursuit as a precursor to gaining a position advantage. Complex maneuvers are executed depending on approach and response of

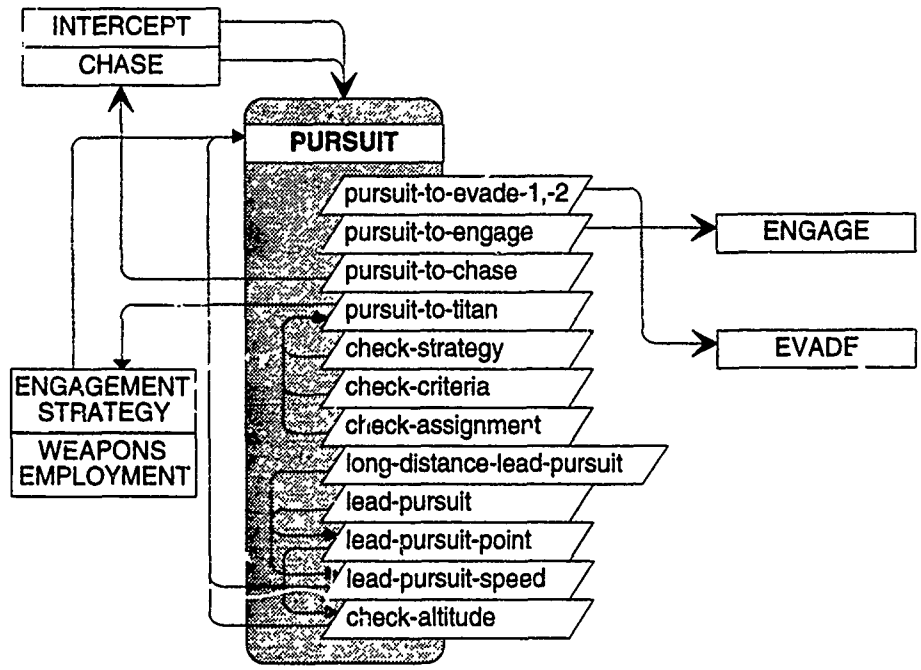


Figure 4.19. Pursuit Phase.

target aircraft. Optimal speed and altitude is calculated for best advantage. In depth discussion and analysis of the Engage phase is detailed by Hluck (Hluck, 1993).

Follower aircraft, in the mean time, execute one of three strategies. In the fighting-wing strategy, follower aircraft try to follow the leader through all maneuvers. In the dual-attack strategy, follower aircraft remain in orbit around the engagement and warn the leader when new targets approach. In the loose-deuce strategy, follower and leader take turns engaging a target until one gains a position advantage.

Leader aircraft can call for a cooperative maneuver from a follower. If both leader and follower are in position relative to a target, the leader can call for a "split-turn" maneuver, also called an "offensive split" (Shaw, 1985:204). The aircraft gaining a position advantage becomes leader and changes phase to the Acquire phase.

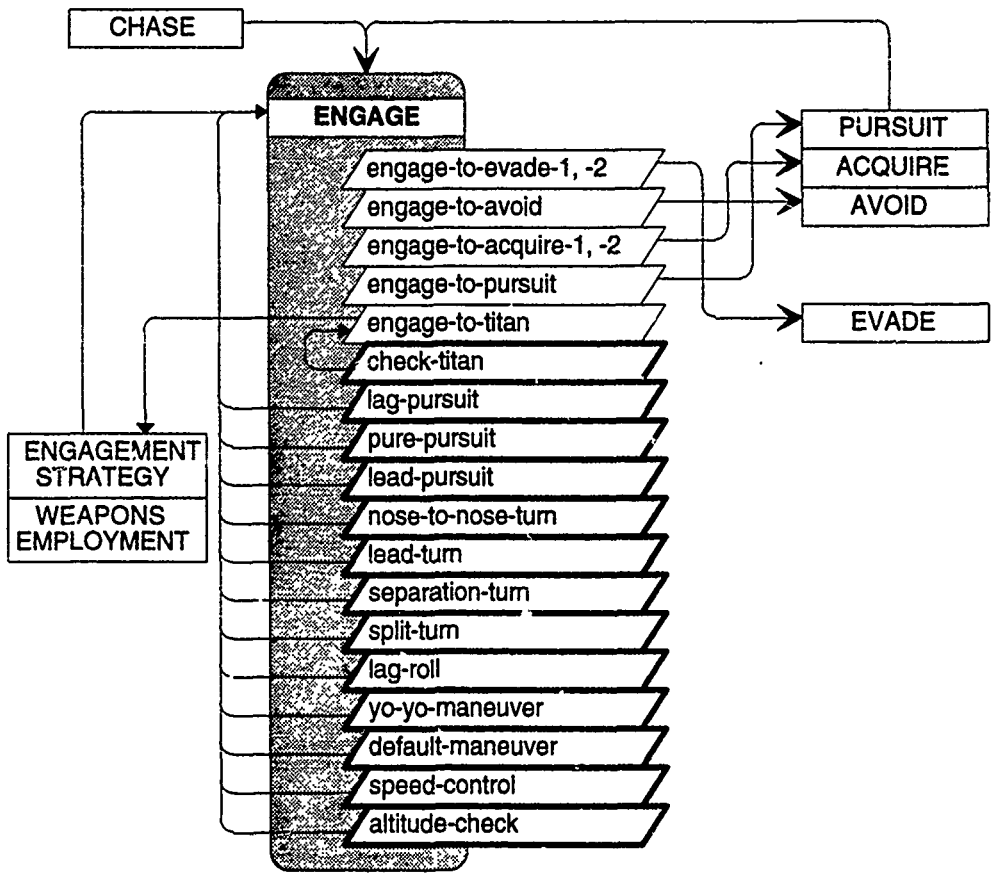


Figure 4.20. Engage Phase.

4.3.10. Acquire Phase. The primary purpose of the Acquire phase is to lock a missile onto a target (Hluck, 1993). First, a missile is instantiated and begins a launching sequence. Second, the aircraft must keep the target within the missile's launch envelope. And third, the aircraft must break away if either the target moves out of the launch envelope or a new threat is detected. Figure 4.21 shows the Acquire phase. The normal transition is to the Fire phase where missiles are launched.

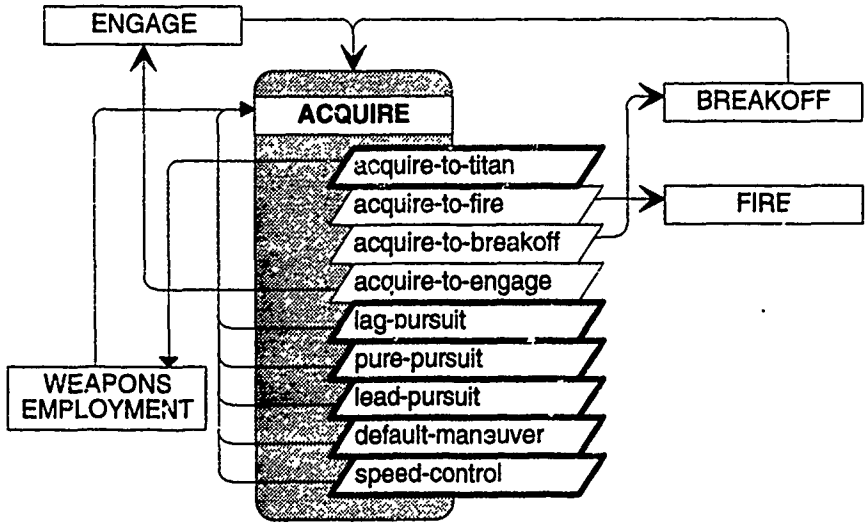


Figure 4.21. Acquire Phase.

4.3.11. Fire Phase. The primary purpose of the Fire phase is to release the missile aimed at a target (Hluck, 1993). Once released, the missile becomes an independent entity and follows basic flight rules to the target. The Fire phase is shown in Figure 4.22. Missiles remain in Fire phase throughout their flight. The aircraft switches to Analyze phase and monitors the missile as it pursues the target.

Missiles may run out of fuel and expire, detonate when they come within a designated kill radius of the target, or switch goals if another aircraft moves across its flight path. If a missile runs out of fuel, it is deleted from the simulation. If a missile detonates near the target, both the missile and target are deleted from the simulation. If a second aircraft

comes within the launch envelope of the missile and is closer than the original target, the missile will switch goals and pursue the closer aircraft. In any case, the missile accesses Weapons-Employment rules so that the originating aircraft can decide what to do.

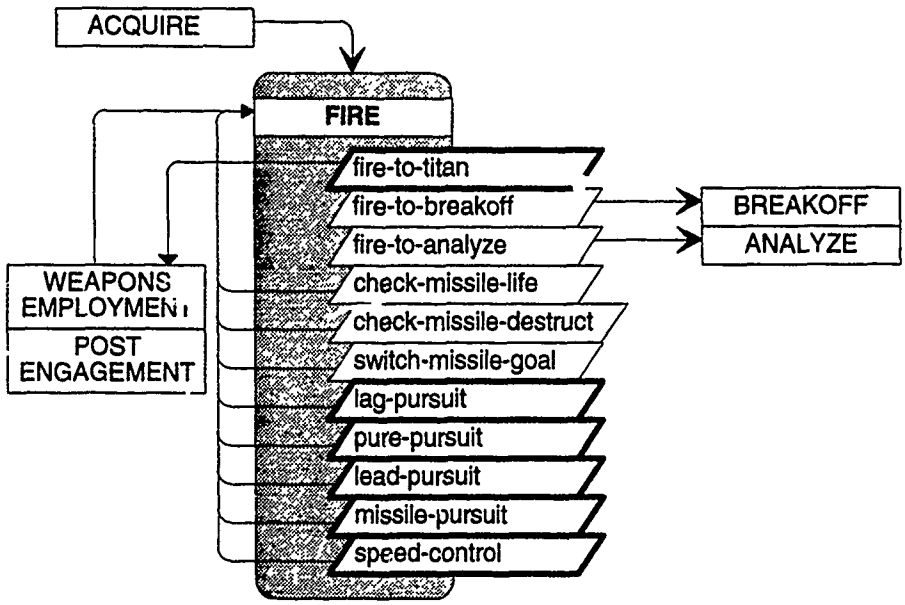


Figure 4.22. Fire Phase.

4.3.12. Analyze Phase. The Analyze phase performs two functions (Hluck, 1993). First, the aircraft monitors missile progress toward a target. Missile effectiveness is gauged by judging whether the missile hits or misses the target. If the missile hits the target, then the aircraft returns to Search phase to seek a new target. If the missile misses, aircraft return to the Engage phase and re-engage the target. If a new threat appears, aircraft change to the Breakoff phase and address the new target. The Analyze phase is depicted in Figure 4.23.

4.3.13. Breakoff Phase. Once in the Breakoff phase, aircraft need only determine whether to return to the engage phase or disengage completely (Hluck, 1993). Figure 4.24 shows the Breakoff phase. Aircraft re-engage if a missile fails to destroy a known target, but disengage if a new threat appears. If the threat is a missile, aircraft change

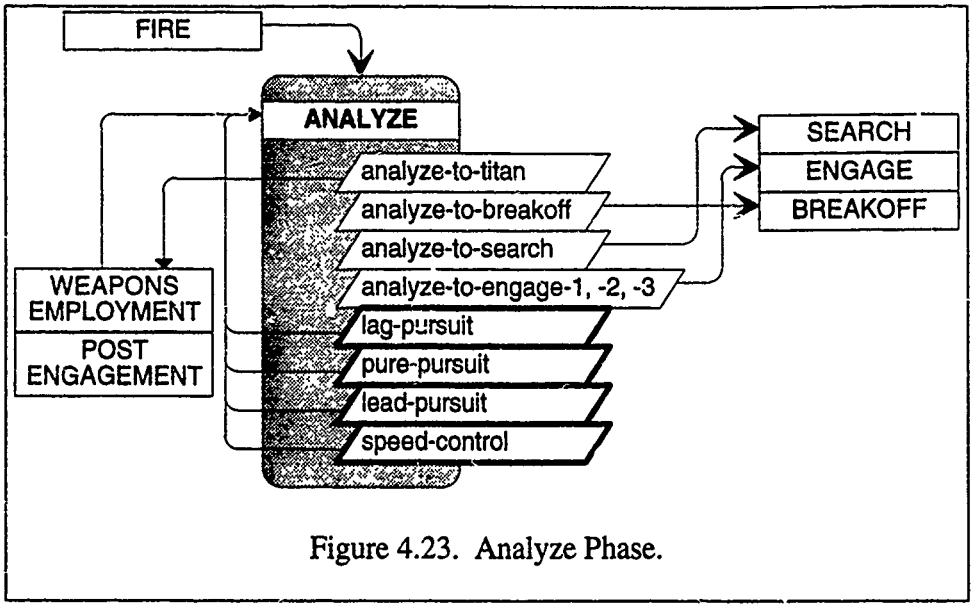


Figure 4.23. Analyze Phase.

phase to the Avoid phase. If the threat is composed of opposing aircraft, then a phase change to Evade phase occurs. Conditions causing the change to Breakoff phase guide further transitions in both Avoid and Evade phases.

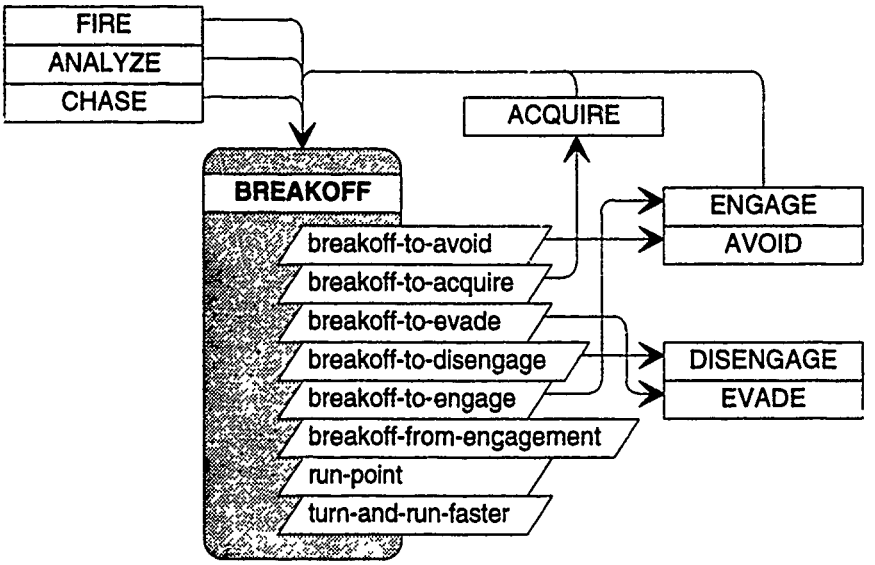


Figure 4.24. Breakoff Phase.

4.3.14. Avoid Phase. The Avoid phase is designed to activate missile avoidance rules (Hluck, 1993). The difference between aircraft evasion and missile avoidance lies in maneuvering tactics. While aircraft evasion tactics involve turning away from a potential

threat, missile avoidance tactics call for a turn toward the threat (Shaw, 1985:59). The Avoid phase executes rules designed to cut across the path of a missile. This method increases maneuvering requirements on the missile since it travels much faster than an aircraft. Using crossing maneuvers is analogous to adopting the "angles" fighter strategy in aircraft combat maneuvering. Once the missile is avoided, aircraft return to either Search or Engage phase to address the source of the missile. The Avoid phase is illustrated in Figure 4.25

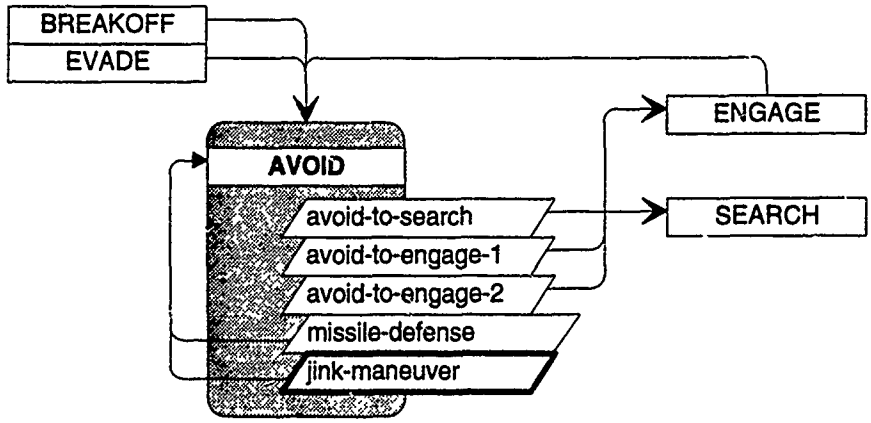


Figure 4.25. Avoid Phase.

4.3.15. Disengage Phase. The Disengage phase prosecutes a complete withdrawal from an engagement (Hluck, 1993). One of several reasons can prompt a disengagement. Low on fuel, out of missiles, mission time expired and other criteria dictate escape from the immediate combat area. Currently, only a bingo plan can be used to egress an engagement. Future simulators would need to complete an egress plan to properly execute a retreat. The Disengage phase is shown in Figure 4.26.

Cooperative rules in the Disengage phase coordinate leader and follower aircraft. If the leader disengages, followers resume flight formation and follow the leader out of an engagement area. If a follower must disengage, e.g. due to low fuel, the follower signals the leader who, in turn, disengages and returns with the follower provided disengagement

criteria in decision modules are satisfied. If criteria are not satisfied, leaders and followers begin retreats independently and rejoin either at the refueling point or home base.

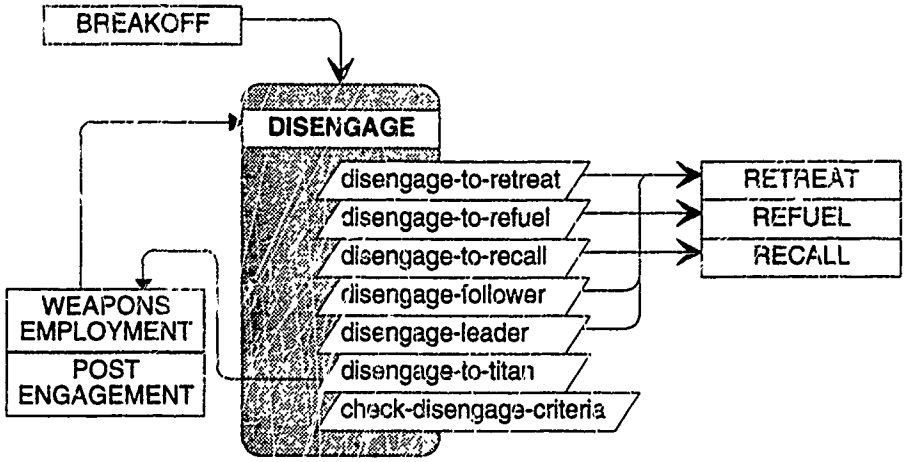


Figure 4.26. Disengage Phase.

4.3.16. Evade Phase. The Evade phase is designed for aircraft v[illegible]ying not to be detected or who are trying to reverse a position disadvantage. Evade phase is the complement of Pursuit phase. Where Pursuit phase attempts a pursuit offense, Evade phase attempts a pursuit defense. The Evade phase is shown in Figure 4.27 and is discussed in depth by Hluck (Hluck, 1993).

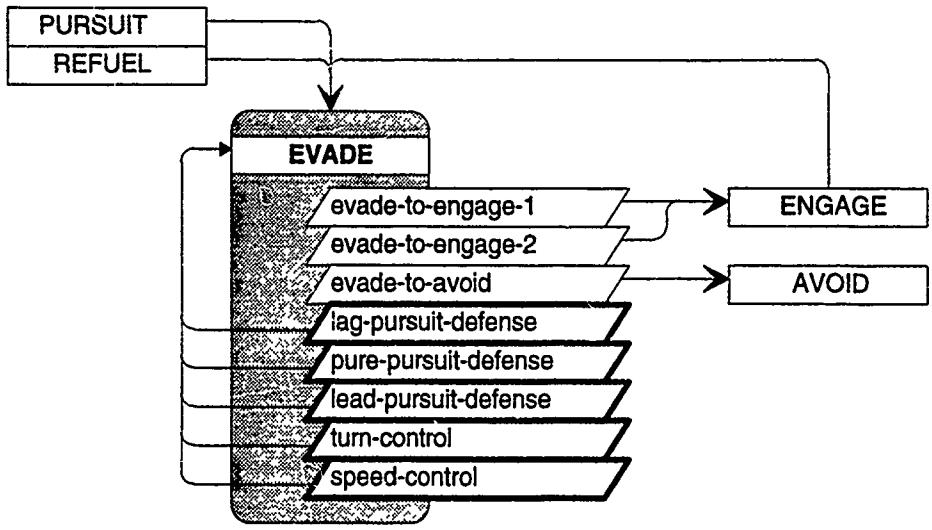


Figure 4.27. Evade Phase.

4.3.17. Retreat Phase. The Retreat phase heralds the end of a pursuit beyond the border. Aircraft entering retreat phase have one of three options. If an opponent leaves controlled airspace, players return to Cruise phase and resume the mission they had at the beginning of an engagement. Aircraft low on fuel change to Refuel phase and proceed to the nearest bingo field. The normal transition takes players to the Recall phase. Figure 4.28 illustrates the retreat phase.

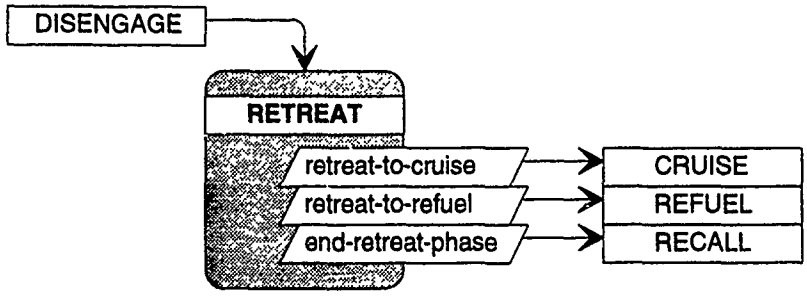


Figure 4.28. Retreat Phase.

4.3.18. Refuel Phase. Players in the Refuel phase create and execute a bingo plan according to the rules shown in Figure 4.29. The bingo plan is a route plan similar to patrol or CAP station plans, but waypoints are calculated based on location within the grid. Players have knowledge of two possible refueling points: the tanker and the base. The bingo planner plots a two-stage route back to both locations. The first stage takes players to waypoints due north or south of the refueling point. The second stage takes players back to the refueling location. Players select the shortest route since by this time they are low on fuel. Once refueled, players return to their original mission.

4.3.19. Recall Phase. Players in recall phase return to the base and land unless they have fuel to resume a cruising mission. When aircraft have more than 50 units of fuel and there are more than 100 time steps remaining in a simulation run, aircraft cruise toward the CAP-station and resume a CAP-station plan. Otherwise, aircraft return to their home base by changing to the Landing phase. The Recall phase is shown in Figure 4.30.

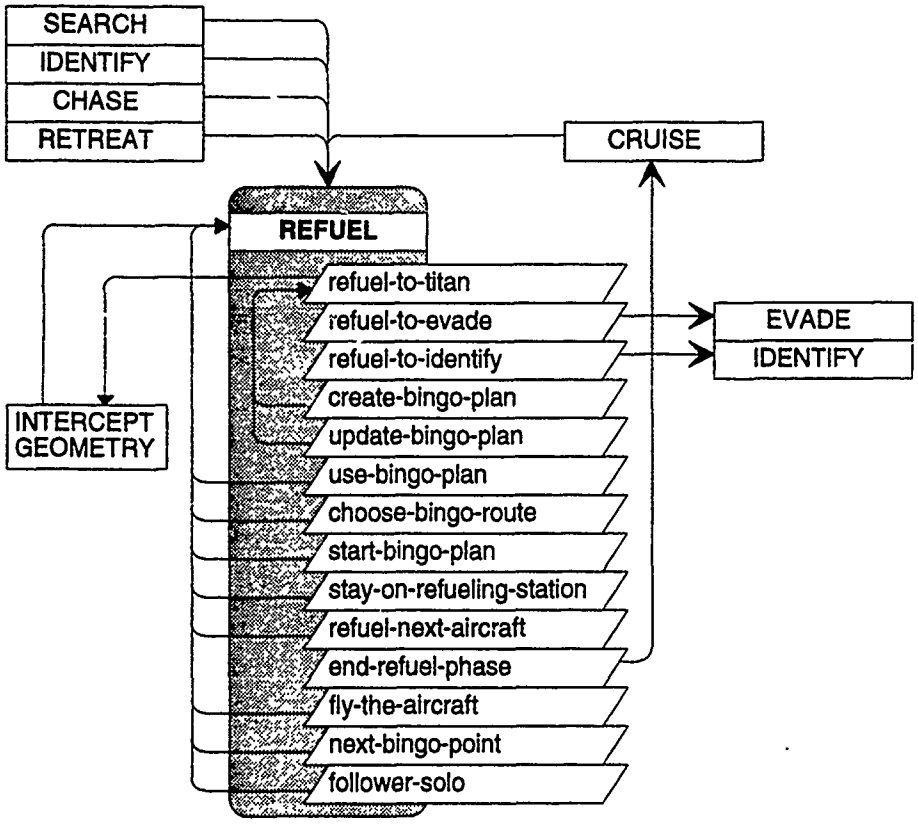


Figure 4.29. Refuel Phase.

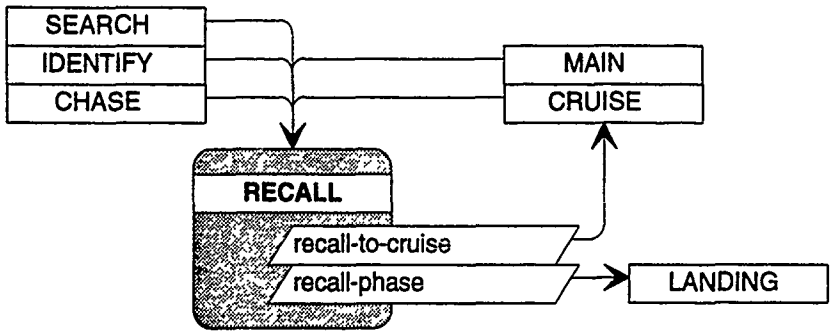


Figure 4.30. Recall Phase.

4.3.20. Landing Phase. Players in the landing phase follow a pre-defined plan to put them on the runway. Landing plans are route plans with special waypoints. Each landing plan has an approach, glideslope, and touchdown waypoint to guide players down to the runway. The "separate-landing-goals" rule changes follower aircraft into solo

aircraft so that cooperating players land independently of leaders. Once on the runway, players slow to a stop. When stopped, the mission assignment slot is labeled "completed" and players return to the New-Mission phase to get new mission assignments. When a simulation run has less than 100 time steps remaining, all aircraft in the simulation return to land at their home base and the simulation ends.

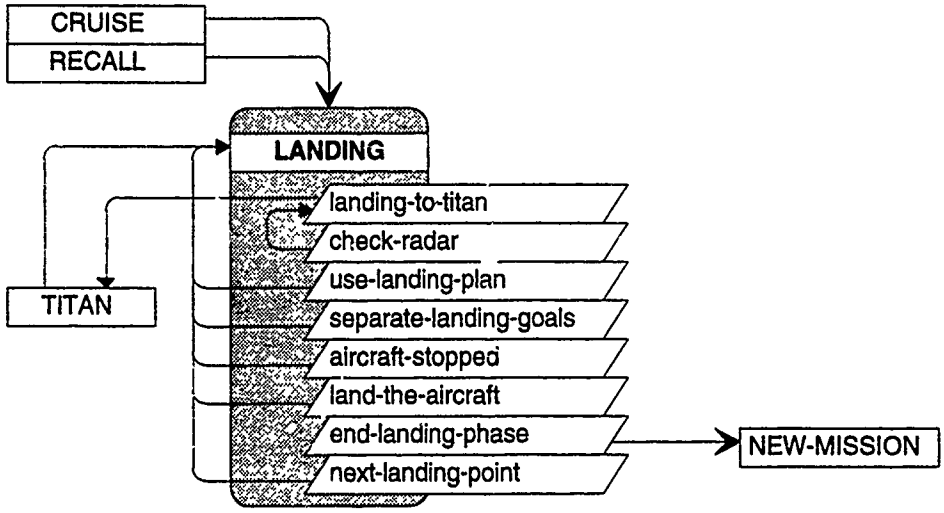


Figure 4.31. Landing Phase.

4.4 Decision Module Dynamic Models.

Decision modules implement key decisions made by pilots in aerial combat situations. Each module addresses one of six decision areas defined in the Titan report (Titan, 1986). Decision areas range from pre-engagement to post-engagement determinations. Key parameters needed to make decisions are coded as object slots in PDPC. Module names reflect the decision area corresponding to the Titan report. Module names and their connection to Titan report decision areas are summarized in Table 3.1, but are repeated in Table 4.1 for convenience.

Table 4.1. Decision Modules.

Decision Module	Titan Report Decision Area
Pre-Engagement	Pre-Engagement Decisions
Engagement-Strategy	Engagement Strategy Determination
Intercept-Geometry	Intercept Geometry Determination
Weapons-Employment	Weapons Employment Strategy Determination
Counter-Action	Response to Hostile Counter Action
Post-Engagement	Disengage/Re-Engage Strategy

4.4.1. General Description. Decision modules are called from phase modules when a player requires deeper reasoning than offered in each phase. For example, when a player detects a target, but does not have an intercept plan, the Intercept-Geometry module is called to make an intercept plan. If a plan already exists, that plan is implemented which avoids the time cost of re-planning. Players call a decision module by asserting a fact on the fact list to "check" the module needed. A rule in each phase module detects the fact and fires to invoke a focus onto the decision module.

Each decision module is composed of several sub-modules depending on the specific decision needed by a player. If a player has the wrong radar settings for a phase, then a fact to check radar is placed on the factlist. The syntax for a radar check fact is:

```
(check (type radar) (player name) (module Pre-Engagement))
```

which fires the rule opening the Pre-Engagement module. Rules in the decision module key on the type of check and the player's name to determine which decisions to make. The type of check dictates which sub-module rules are used. Rules in other sub-modules are ignored. Check facts act to restrict rule firing just as object phase slots control phase module selection.

4.4.1. Pre-Engagement Module. The Pre-Engagement module makes decisions needed before engaging a potential threat. Figure 4.32 shows the dynamic model for the Pre-Engagement module. Three sub-modules check radar settings, target data and commitment decisions.

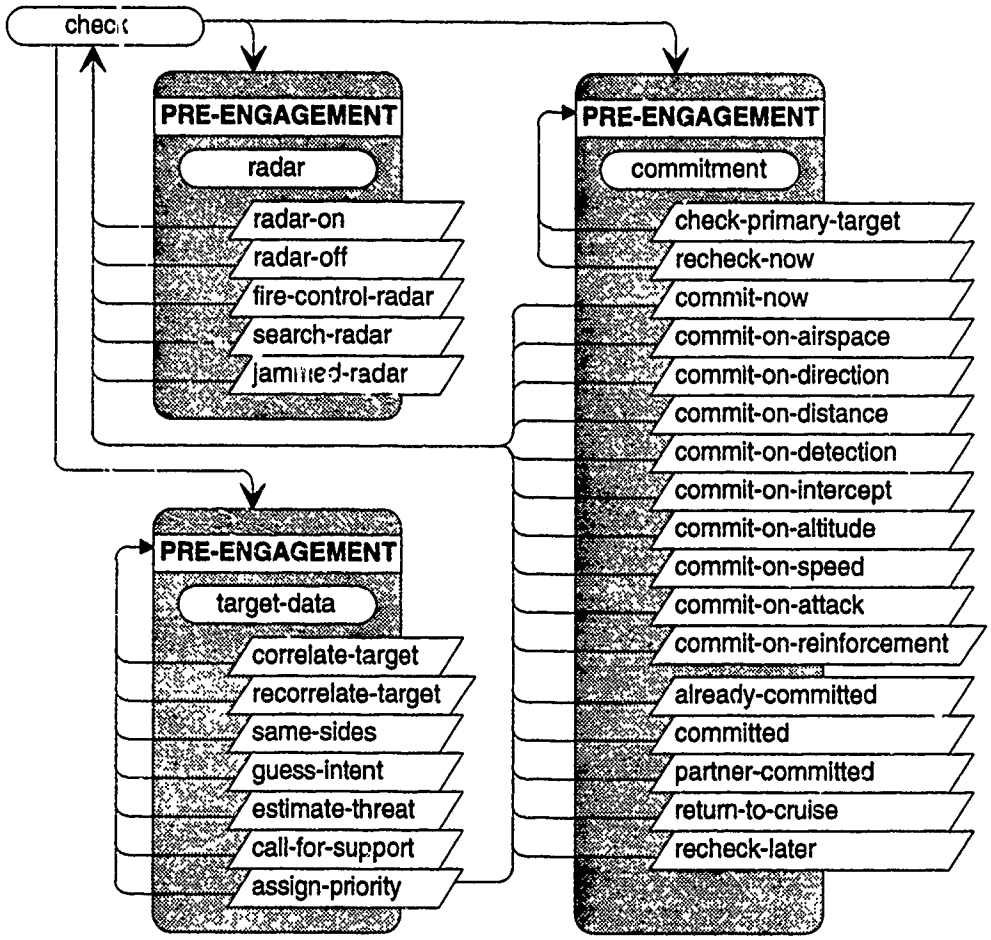


Figure 4.32. Pre-Engagement Module.

4.4.1.1 Radar Sub-module. The radar sub-module turns radar capabilities on or off depending on the phase of a player. If a player's phase is not New-Mission, Launch, or Landing, then the radar should be on and in search mode. Players also switch the radar to fire control or clear a jammed radar by returning to Pre-Engagement rules. Although players should be well passed the pre-engagement stage when needing fire control, return to the Pre-Engagement module consolidated similar rules, prevented rule duplication and allowed dual usage of rules such as the "search-radar" rule.

4.4.1.2 Target Data Sub-module. The target-data sub-module assesses the threat level of any moving target within radar range. Target data is stored in object slots

as lists for later recall. For each potential target, a player records the target's name, potential victim, threat level, and priority. When all targets have been assessed, the player sorts target data according to priority and addresses the first threat on the list. If the player is a member of a team, target data is shared so that team-mates avoid the cost of calculating threat levels.

The process begins with the "same-sides" rule where a player determines whether a potential target is on an opposing side. Then the player determines the intent of the target by assuming that the potential victim is the object on the player's side most directly in the target's path. The threat is estimated by assigning value to parameters such as the target's intention, distance, speed, altitude and location. The player associates a threat level of "high", "medium", or "low" with the target. Priority is calculated based on the target's threat level, type (e.g. fighter, tanker, bomber, etc.), range and direction with respect to the player. A fighter that is nearby, approaching, and has a high threat level receives a high priority.

Team members correlate target data with other team members. If the leader has no knowledge of the target, the follower's target data is passed to the leader. Otherwise, the leader's target data is selected over the follower's.

A player can call for additional support from the home base. When the number of targets exceeds the number of missiles carried, players assert a "Send help!" message onto the factlist. The form of the message is:

```
(message (from player) (to base) (channel #) (content SEND HELP!))
```

where *player* is the name of the sender, *home-base* is the name of the player's base, "channel" indicates which radio channel is used, and "content" holds the message text. Message facts cause the simulator to focus into a communications module where each message is processed. In the case of the "Send help!" message, the home base acknowledges the message by retracting the message fact. Future PDPC versions could

launch aircraft in response to calls for help. The current version uses the message to coordinate players on the same side. Regardless of leader or follower status, if a player calls for help and another player on the same side "hears" it, then the receiving player is assigned to be follower and the sender becomes leader. The new follower proceeds toward the leader's location to provide support.

4.4.1.3 Commitment Sub-module. The commitment sub-module controls player response to a threat. A key decision identified in the Titan report determined when a pilot should commit to pursuing a potential target. Air combat pilots used several parameters to judge the commitment point that would result in the greatest advantage. Rules in the Commitment module currently operate at a simpler level. Instead of judging the greatest advantage, commitment rules fire when a condition is met. For example, the commit-on-distance rule fires when a target reaches a range equal to five times the closing velocity. Commitment rules fire when opposing aircraft violate airspace boundaries, move toward or fire a missile at the player making the commitment decision. A player can fire a commitment rule when reinforcements arrive, or by reaching an intercept point, desired altitude or speed. When a rule fires, the player's goal changes to the target and the player moves into Pursuit phase.

Commitment rules can be delayed if a potential target does not satisfy one of the commitment rules, or short-circuited if a player has already engaged the target. The "recheck-now" and "recheck-later" rules work to delay a commitment decision for five iterations after an initial commitment check. If the initial check results in no commit rule firing, a fact to recheck the target later is placed on the fact-list. If the player re-enters the Pre-Engagement module, commitment rules will be clobbered for five iterations and the player will continue executing in the current phase. Commitment decisions can also be made in phase control modules under certain conditions. For example, if a player identifies a new target that has a high threat level, the player's phase changes to Pursuit

phase without checking commitment rules. Otherwise, players check commitment rules before entering an engagement.

4.4.2. Engagement Strategy Module. The Engagement-Strategy module selects a formation, approach and strategy to use against an adversary. Figure 4.33 shows the dynamic model for the Engagement-Strategy module. Three sub-modules check approach, tactical coordination and formation. Rule firings result in a selection written to an object slot rather than a phase change as described in the Pre-Engagement module. Resulting slot data is used in phase sequence modules to direct or coordinate player actions.

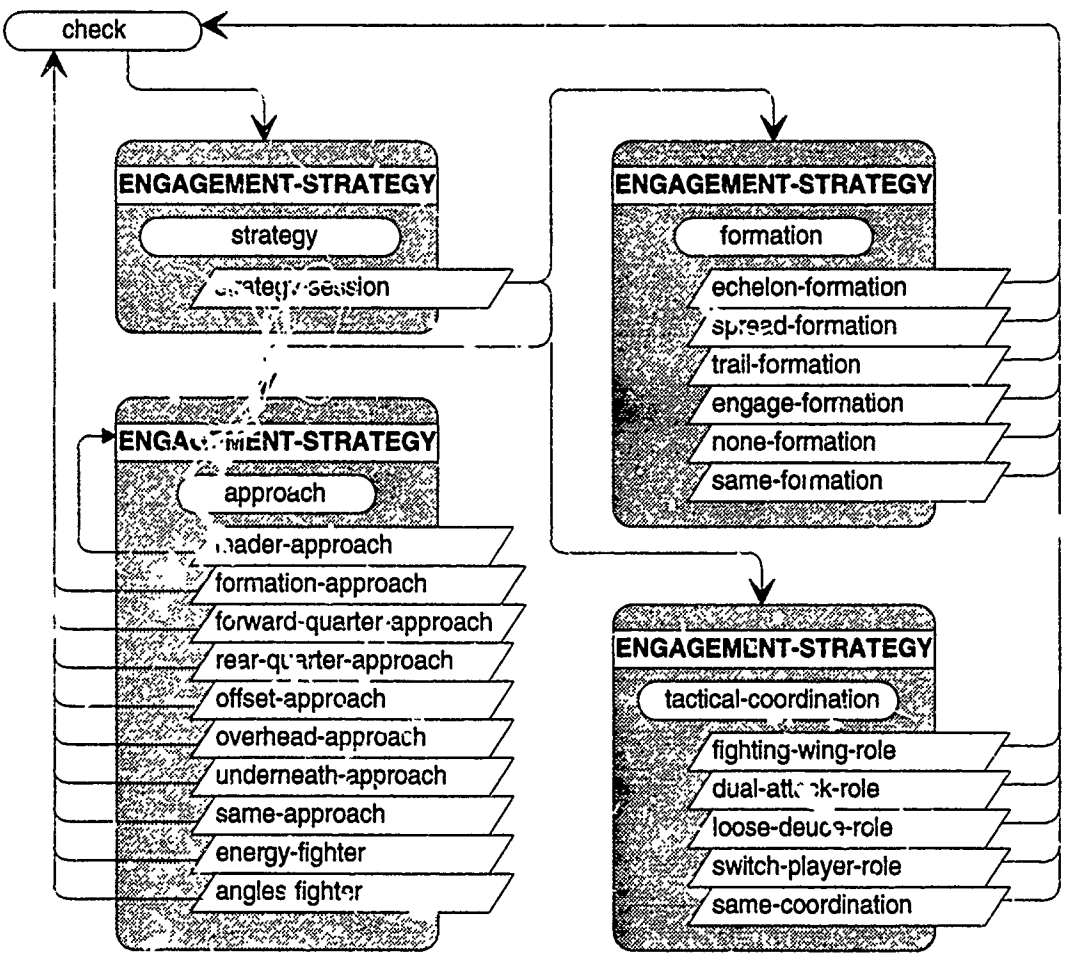


Figure 4.33. Engagement Strategy Module.

4.4.2.1 Strategy Sub-module. The strategy sub-module determines which of the other sub-modules will be invoked. If the player calling for engagement strategy decisions is a solo fighter, then only the approach sub-module is used. If the player has a leader role, then all three sub-modules are called. If a player is a follower, then the first call to Engagement-Strategy prompts the leader to dictate an approach, formation and tactical-coordination. Subsequent calls by the follower do not prompt the leader unless the leader changes phase. Until a leader change occurs, the follower performs the same approach, formation and coordination as initially assigned.

4.4.2.2 Approach Sub-module. The approach sub-module determines the direction a player is using to pursue a goal or the energy state a player has relative to an engaged target. For leaders and solo players, the approach identifies whether the player is in front of, behind, off to one side, above, or below the target. Followers use a default rear quarter approach to get into a flight formation, prompt the leader for an approach, or use the same approach as initially given. Players in Engage phase also consider current energy relative to their assigned target. The "energy" or "angles" fighter strategy identifies whether a player should use high speed or high turn-rate tactics. Determination is based on the specific energy and excess power equations defined by Shaw. (See Appendix B)

4.4.2.3 Tactical Coordination Sub-module. The tactical coordination sub-module selects one of the dual-player strategies defined by Shaw (Shaw, 1985). The fighting wing role dictates that follower aircraft stay in formation on the leader. Dual attack and loose deuce strategies use the follower as an additional attacker. Each strategy determines what phase a follower should use while the leader engages a target. With a fighting wing role, followers stay in Chase phase and chase the leader. With a dual attack role, followers stay in Intercept phase and travel between orbit points of an intercept plan. In the loose deuce role, both leader and follower engage the same target with the player having the greatest advantage taking the lead role. The "switch-player-role" rule changes

leader and follower roles when loose deuce coordination is used and the follower has a better position advantage on the target.

4.4.2.4 Formation Sub-module. The formation sub-module dictates how a follower should fly in relation to a leader. Selected formations depend on the phase of the leader. For example, when the leader is in Cruise phase, the follower flies in echelon formation. The "engage-formation" rule selects a formation depending on tactical coordination as well as leader phase. For example, a leader in Engage phase declaring a tactical coordination of fighting-wing puts the follower in trail formation. Loose deuce strategy takes the follower out of formation (assigns a "none" formation). Given the leader does not change phase or tactical coordination, the follower uses the same formation as initially assigned.

4.4.3. Intercept Geometry Module. The Intercept-Geometry module plans a path from a player to a goal. Four types of plans were anticipated, but only two were fully implemented: intercept and bingo plans. Ingress and egress plans are primarily designed for offensive missions which were not needed at this stage in the research. The Intercept-Geometry module is depicted in Figure 4.34.

4.4.3.1 Intercept Plan Sub-module. The intercept planner computes a series of waypoints from a player's current position to a target's predicted position. The intercept planner uses history object location data to project an intercept point along the target's path. Players move toward the intercept point as a goal rather than the target's location. Along the intercept route, the planner identifies "free fire areas" a player should avoid. A free fire area is a ground station that has permission to fire at any flying object. The intercept planner defines intermediate waypoints directing players around designated free fire areas. Starting, intermediate, and intercept points are stored as slot data in a intercept plan object. Players follow the intercept plan point-by-point until either reaching the intercept point or a commitment rule fires. Back in the Intercept phase module, if a

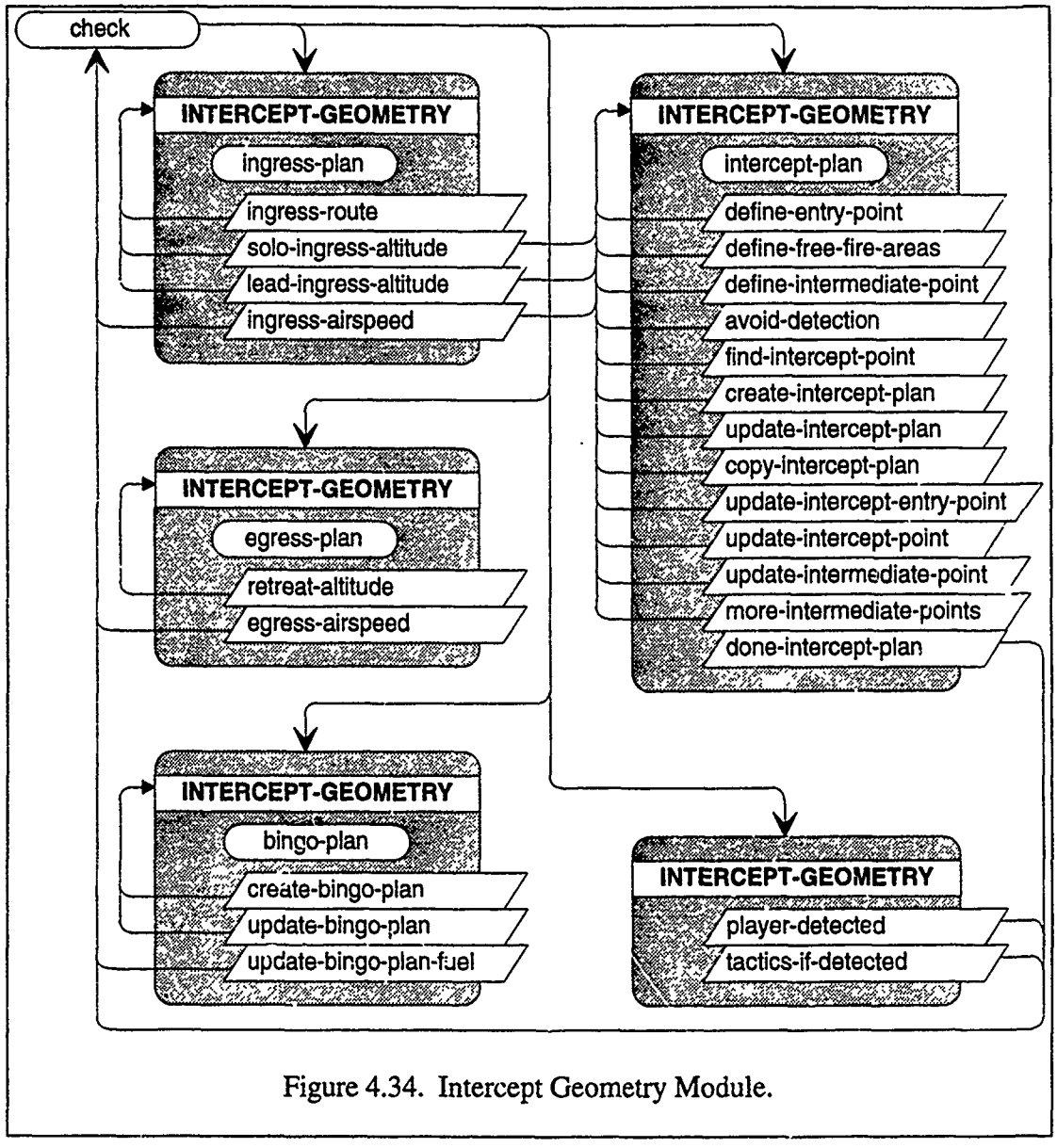


Figure 4.34. Intercept Geometry Module.

player arrives at an intercept point and is near the target, phase changes to Pursuit phase. If the target moved out of radar range, the player returns to Search phase.

4.4.3.2 Bingo Plan Sub-module. The bingo planner computes a series of waypoints from a player's current position to a refueling location. This planner plots six intermediate points based on the location of the player, base and tanker. Each point lies on a line stretching due north or south of a refueling point depending on whether the player is north or south. Three waypoints are placed on each refueling line, one level with

the player's north-south location, one closer to the refueling point an one farther away. The player selects the shortest path to a refueling point unless an opponent player lies in that path. Then the player chooses an alternate waypoint or alternate refueling point. Waypoint data is stored in a bingo-plan object.

4.4.4. Weapons Employment Module. The Weapons-Employment module assigns targets, selects weapons, and considers engage/disengage criteria. Three sub-modules compartmentalize decision rules: target assignment, engage criteria and weapons selection. Figure 4.35 illustrates the Weapons Employment module.

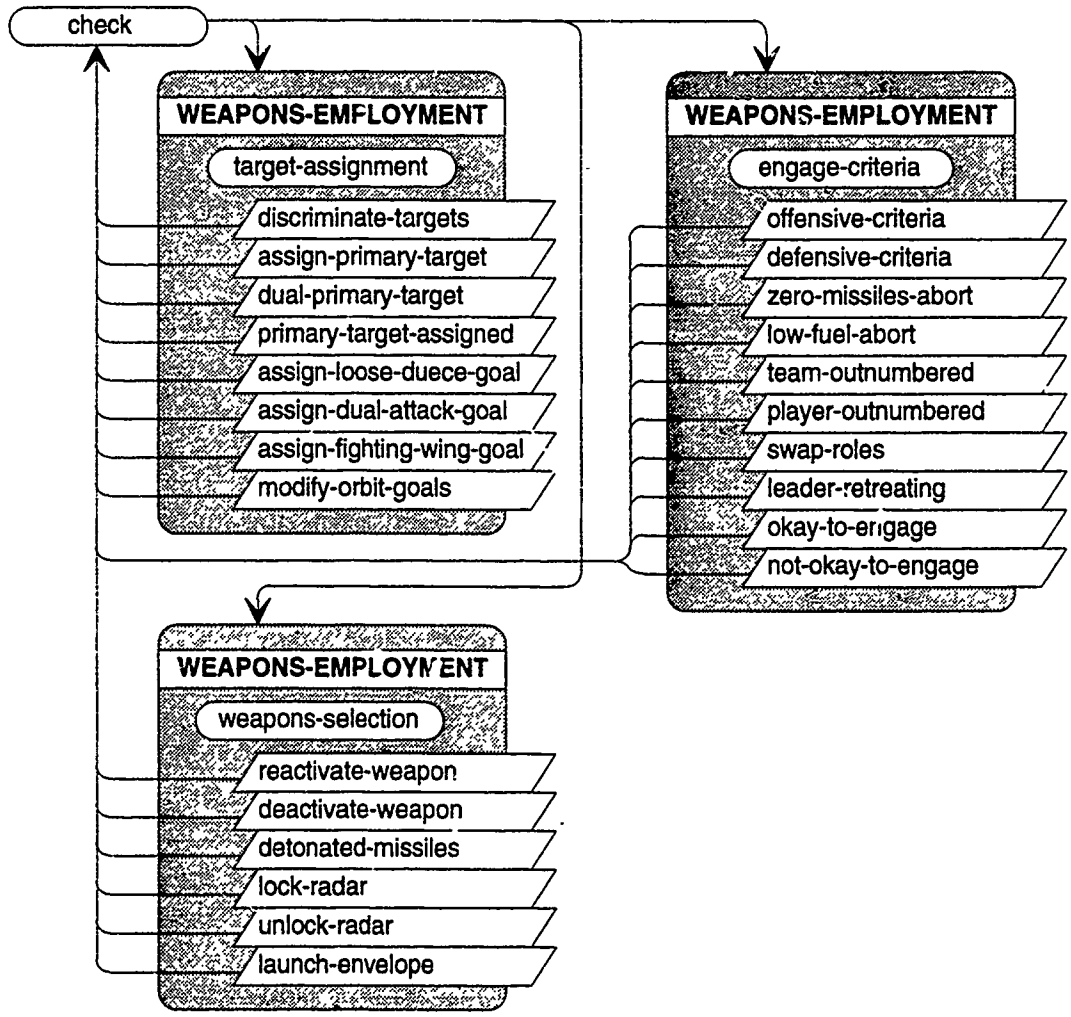


Figure 4.35. Weapons Employment Module.

4.4.4.1 Target Assignment Sub-module. The target assignment sub-module identifies and assigns primary and secondary targets and assigns team strategy goals. In a multiple target scenario, the discriminate-targets rule determines which target should be pursued. A lower priority target could be pursued if it is closer, approaching, and in front of the player discriminating targets. Otherwise, the player is assigned and pursues the higher priority target. A dual primary target assignment occurs when leader and follower aircraft engage two opponents. The leader takes on the primary target while the follower addresses the secondary target. Followers could also receive goals depending on tactical coordination. Currently, followers assume an orbit around a leader's engagement when the tactical coordination is either loose-deuce or dual-attack. When the tactical coordination is fighting-wing, followers chase the leader regardless of the leader's phase even when the leader engages a target.

4.4.4.2 Engage Criteria Sub-module. The engage criteria sub-module determines whether a player should engage a target or disengage and retreat. Players consider both engage and disengage criteria to prevent searching two modules for a single rule. Disengage criteria are also considered in the Post Engagement module. Players engage a target if in pursuit, within range, and no disengage rule fires. Criteria for disengaging include stopping at the border, running out of missile or fuel, or being outnumbered by opposing forces. The "swap-roles" rule fires if a leader has disengaged and the follower is available to attack the arget. In this final case, the follower becomes leader and the leader becomes follower.

4.4.4.3 Weapons Selection Sub-module. The weapons selection sub-module activates and deactivates missile objects during an engagement. Missile objects are instantiated when a player changes from engage to acquire phase. Players must maintain target acquisition for five iterations before a missile actually fires. During that time the target may move out of the launch envelope and the missile should not be fired. Weapons

selection rules either reactivate missile objects if a player tries to re-acquire a target, or delete missile objects if a player disengages. Missile objects also delete when detonated.

The launch envelope rule adjusts missile launch range and angle depending on the target's relative location and velocity. When a player closes on a target, missile launch range increases simulating an early launch opportunity. Launch range decreases when the target is escaping. Launch angle varies depending on the target's crossing velocity. The crossing velocity estimates the angular rate of change in the line of sight. If the target is crossing the player's field of view from left to right, the launch angle adjusts to the left. A target crossing from right to left adjusts the angle to the right. Adjustments to launch range and angle simulate early launch opportunities and prevent late launches.

4.4.5. Counter Action Module. The Counter-Action module contains instructions on how to avert an attack. Currently this module is not fully developed. Development efforts concentrated on engagement maneuvers and tactics rather than evasive maneuvers. Simple rules employ electronic countermeasures, assert facts that dispense chaff or flares during a missile attack, or accelerate players in close range of a missile. The Counter Action module is shown in Figure 4.36.

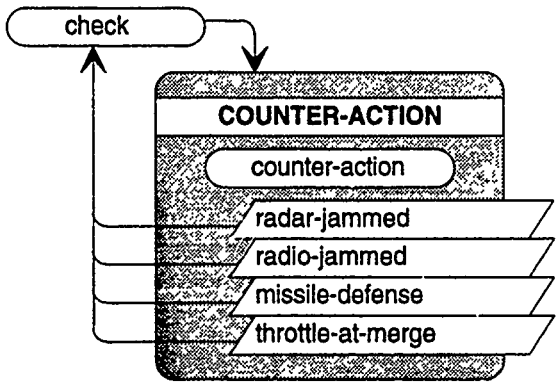


Figure 4.36. Counter Action Module.

4.4.6. Post Engagement Module. The Post-Engagement module, shown in Figure 4.37, contains rules guiding a player to a new target assignment. The "target-destroyed"

rules clean up a player's target list and prevent a re-engagement. "Missile-missed" rules determine whether a players re-engage a target or retreat. The "team-missile-missed" rule swaps leader and follower roles when the follower has more missiles left than the leader. "Check-known" and "check-unknown" return players to Identify and Search phases, respectively. Detailed disengage criteria is also contained in the Weapons Employment module and was not duplicated in the Counter-Action module.

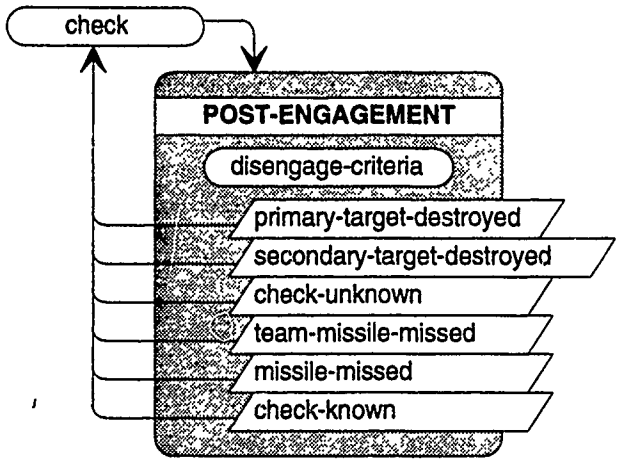


Figure 4.37. Post Engagement Module.

4.5 Functional Model.

This section describes the functional model of PDPC. Since the simulator is extensive (over 400 rules), and since CLIPS processes each rule the same way, this section discusses only a general model. In the following diagrams, italicized text represents CLIPS constructs while non-italicized text represents PDPC constructs.

4.5.1. Typical Rule Model. The functional model of a typical CLIPS rule is depicted in Figure 4.38. A rule construct basically consists of a predicate statement to determine rule validity and a consequent statement to perform some action or actions.

The predicate statement is called the "left-hand side" of a rule and constitutes the "if" portion of an "if-then" construct. The consequent is called the "right-hand side" and constitutes the "then" portion. CLIPS pattern-matches each conditional element in the predicate statement to data elements drawn from either the factlist or object slots. When a conditional element requires processed data values, CLIPS calls appropriate functions and compares the results. If all conditional elements are satisfied, the rule "fires" and executes the actions in the consequent statement.

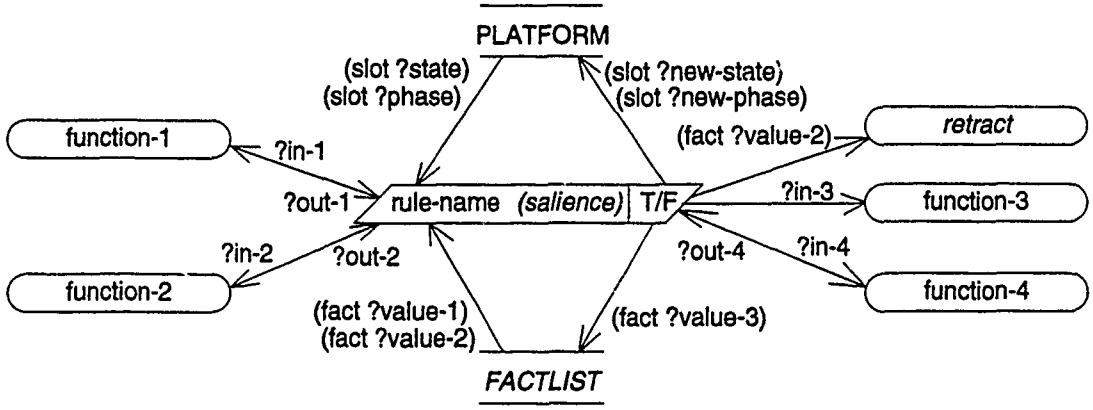


Figure 4.38. CLIPS Rule Functional Model.

Actions in the consequent statement modify data elements or execute functions depending on the requirements of the rule. When a rule fires, object slot data is changed and facts are added to (or retracted from) the factlist. Functions on the right-hand side execute system calls and simulation updates, or collect more data to process the rule. The predominant method in PDPC is to modify object slot data.

4.5.2. Main Module. The main module controls the simulator's progression by resetting player states to "moveable" once every player has moved. Each player object can have a state of "moveable", "move", or "moved" (refer to Figure 4.39.) A player in the moveable state has access to every rule in PDPC depending on phase. When rules fire, the

player's state changes to "move" which prevents further phase rule firings and makes the player eligible to fire the move-player rule. Once a player moves, the player's state changes to "moved". If the "move-player" rule detects that every player in the simulation has moved, the fact "(everybody moved)" is placed on the factlist.

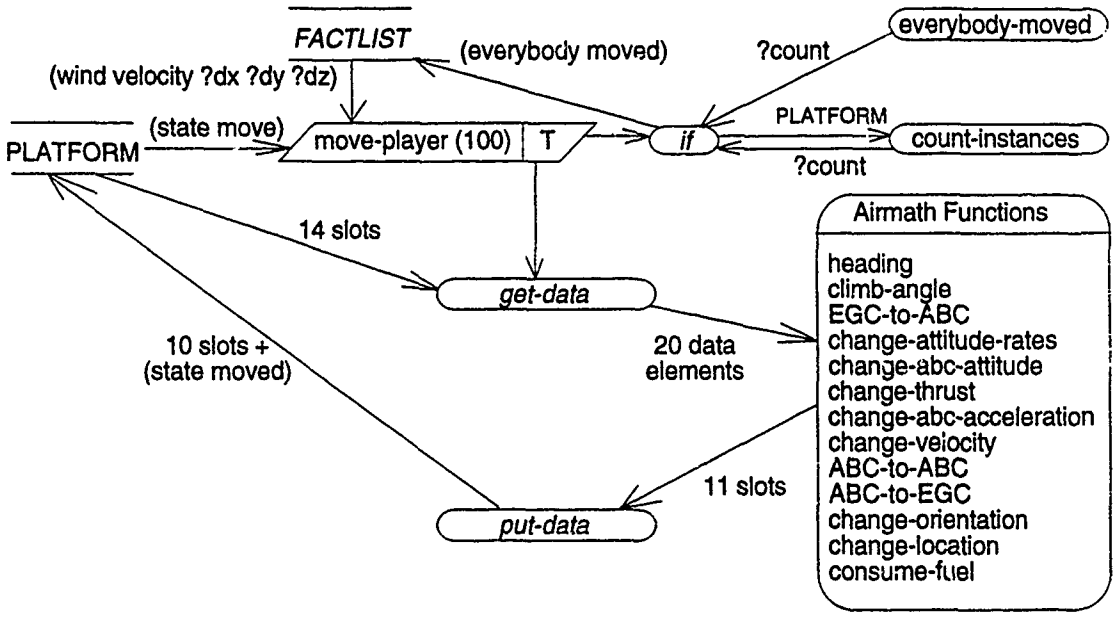


Figure 4.39. Main Player Movement Rule.

When "(everybody moved)" appears on the factlist, the "iterate" rule (see Figure 4.40) begins the process of resetting the simulator for another move. The fact that everybody moved is retracted, the iterate step is decremented to *n-1*, every platform receives a state of "moveable", and the "focus-control" rule is refreshed. The "focus-control" rule will fire once for every moveable platform in the simulation. When fired, CLIPS accesses a module corresponding to a player's phase and pattern matches on the module's rules. If "focus-control" is not refreshed, CLIPS halts the simulation after passing once through PDPC. The combination of "move-player", "iterate", and "focus-control" drive PDPC simulation runs through many iterations.

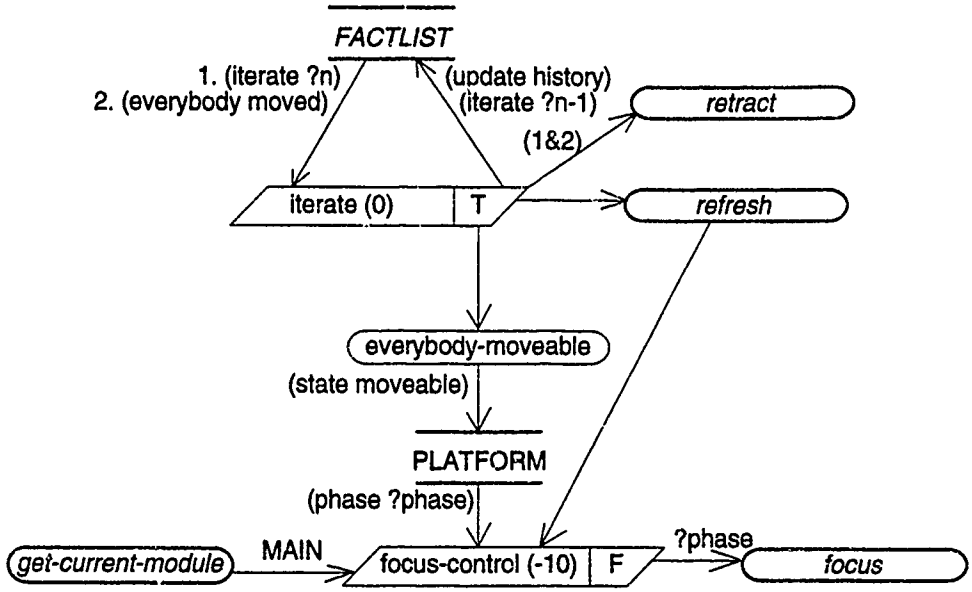


Figure 4.40. Main Simulation Driver Functional Model.

4.5.3. Phase Modules. Rules in each phase module follow the functional flow of typical rules as shown in Figure 4.41. Rules draw on platform slot data to calculate predicate values on the left-hand side. For rules to fire, specific aircraft must be moveable or appropriate facts must appear on the factlist. Each phase module contains three types of rules. First, rules that change a player's state to "move" prevent further rule firing and return control to the Main module. Second, rules that act in concert either change object slot data or assert facts onto the factlist, but control is retained by the phase module. Third, rules requiring further decision use the CLIPS focus function to focus on decision modules and fire more rules. In each case, modified data is returned to object slots so that other aircraft can use the rule.

4.5.4. Decision Modules. Rules in each decision module follow the functional flow of typical rules and each module contains the three rule types as described in Paragraph 4.5.3, but control flow is slightly modified. When a decision module relinquishes control because no further rules are eligible to fire, control returns to the phase module that called

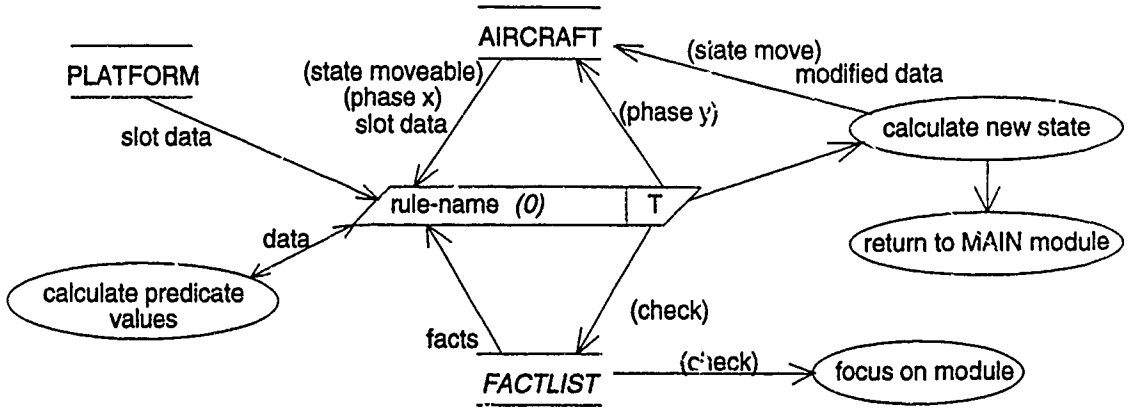


Figure 4.41. Typical Rule Functional Model.

it rather than to the Main module. Rule firings in a decision module may change a player's state so that phase module rules can fire when previously they could not. Decision rules that change a player's state, however, cause control to pass through the phase module and back to the Main module.

4.5.5. Concurrency in Modules. CLIPS uses all platform objects to test predicate constraints on rules. When using a phase or decision modules, CLIPS fires all rules having satisfied constraints regardless of a player's state. So, for example, if four aircraft have a phase of Cruise, CLIPS will fire four "cruise-to-waypoint" rules before returning control to the Main module. Operation proceeds sequentially through phases, but concurrently through objects. Concurrent operation conserves execution time of rules such as "focus-control" since modules are not accessed individually for each object.

4.6 Cooperative Player Implementation Issues.

4.6.1 Missile Goal Swapping. When developing missile scoring rules I found a model of cooperative response. In the Fire phase, missiles are given the name of a target to pursue. Although this approach used perfect information, giving the missile the name of the goal saved time in developing complex sensor rules and procedures. To explore

cooperative capabilities, modified rules gave missiles the ability to change goals depending on which player was closest to its nose. The same approach was used to consider when cooperative players should swap leadership roles.

4.6.2 Detecting Disadvantage. The key to initiating a role change was in detecting when a role change was warranted. Missiles swap goals depending on which player is closest to its nose. Cooperative players swap roles using a similar mechanism. In one rule, players swap roles depending on which player is farthest from an opposing player's nose. In another, players swap roles depending on which player has the target closest to its own nose. The former case detects a disadvantage while the latter detects an advantage. Either case warrants a role change depending on the strategy employed.

4.6.3 Aircraft Role Swapping. Once a role change was deemed necessary, players swapped leader for follower roles depending on the strategy employed. Except for the fighting wing strategy, cooperative followers assumed the leader role and leaders assumed follower roles. The task of changing roles consisted of writing data into object slots.

4.7 Summary.

The PDPC uses object-oriented design techniques in a knowledge-based system to implement an air combat simulator. The object design contains 13 classes and sub-classes describing aircraft, missiles, air bases, and flight plans. The overall architecture follows a linear sequence of phase modules with each module representing one stage in an air combat mission. Detailed planning and decision-making continues in a battery of decision modules. Rules within each module simulate the decision-making process combat fighter pilots use during air combat engagements. Both solo and team tactics and strategies are built into rules within each module. The architecture and design allows PDPC to simulate various types of air combat scenarios.

5. Results

This chapter details results of using rules designed to elicit cooperative behaviors from team players. Simulation test runs used various mission scenarios to demonstrate the capabilities of PDPC from single aircraft test flights to two versus two combat engagements. This approach provided a basis on which to analyze the effects of cooperative rules and compare the performance of cooperating versus non-cooperating agents. This chapter presents and analyzes some of those test runs.

Cooperative player rules changed the character of multi-aircraft scenarios. Players having a leader-follower relationship traded roles depending on advantages perceived against an adversary. The cooperative player having the greatest position advantage assumed the leadership role. Scenarios changed in character because cooperative team players did not react identically to the same threat. Cooperative players demonstrated different team behaviors compared to that of independent players.

Sections 5.1 through 5.4 describe the performance of "friendly" aircraft when presented with various threat scenarios. In each series of tests, performance of different combinations of team players are compared to different combinations of threats. Section 5.1 introduces the "playing field" and describes what players do under non-adversarial conditions. Section 5.2 defines the combat scenarios used to demonstrate the differences between cooperative and independent players. Section 5.3 presents results of aircraft attempting to target and destroy drone aircraft. Section 5.4 illustrates performance of aircraft versus a single defensive target instead of a drone. The variety of engagement scenarios highlights the effects of rules designed to produce cooperative behaviors in autonomous agents.

Section 5.5 spotlights the performance of cooperating players in executing team maneuvers. In this section a "split-turn" is analyzed in detail to illustrate the requirements (and problem areas) of implementing cooperative behaviors in PDPC agents.

The final section of this chapter analyzes system loading data collected during simulation tests. Although preliminary, results indicate the potential for developing PDPC into a "real-time" air combat simulation.

5.1 Non-Adversarial Scenario

5.1.1 Purpose. Non-adversarial scenarios illustrated player actions under non-combat conditions and demonstrated the capabilities of the PDPC flight model.

5.1.1 General Description. One of two different grid definitions identified aircraft location, speed and other environment relationships. The first grid arbitrarily defined a 1000 by 1000 unit grid over which aircraft move. Aircraft were not restricted to the grid and occasionally strayed outside defined limits. The intent was to study interaction between players rather than between player and environment. A second grid defined paths over terrain maps used by programs in the AFIT Graphics Laboratory. This second definition attempted to map aircraft movements in relation to terrain features. For consistency reasons, scenarios in this thesis are illustrated using the first grid definition.

Up to four aircraft, two to a side, were flown in each simulation. Each side, friendly and enemy, consisted of two aircraft, a home base, and a refueling point. Aircraft moved through points in the environment. Bases and refueling points were stationed near the four corners of the grid. Depending on mission scenario, aircraft could move to a combat air patrol (CAP) station, follow a pre-defined patrol route, proceed to the refueling point, or return to land at the home base. For clarity, aircraft were labeled "pilot", "partner",

"bogey1", and "bogey2". The pilot and partner belonged to friendly forces while the two bogeys belonged to enemy forces.

Tests were run using different scenarios. The key to which mission was simulated lies in the mission slot of each aircraft object. If a player had a testing mission, it followed a prescribed flight test route. The predominant scenario was the defense mission. A player having a defense mission followed CAP station route plans, but could deviate from plans when encountering an opposing force. Two forms of defensive mission defined different types of combat operations: superiority and defense. A superiority mission allowed aircraft to operate anywhere on the grid. The defense mission restricted aircraft to one half of the playing grid; the one containing the home base.

The basic scenario for each simulation started and ended at the home base. Upon startup, aircraft received a route plan based on mission assignment. Leader and follower assignments applied only to friendly forces for purposes of comparison. Once a mission, assignment, and plan were given, aircraft proceeded to launch from their home base and the prescribed route. If a member from an opposing team was detected, players considered options to address the threat. High threats were addressed by a player moving to intercept the opposing force. Low threats were assessed, but not acted upon. If no threats were detected, players moved along a route until either time or fuel expired. When fuel ran low, players moved to either the refueling station or their home base to refuel. Players resumed their mission once refueled. When time ran out, all players returned to their home station.

5.1.2 Testing Missions. In testing missions, aircraft move along pre-defined routes and then return to their home base. Opposing forces do not interact. In fact, the only players that interact are those with a leader-follower relationship. As a flight leader follows a route plan, a follower flies behind trying to match the leader's speed and direction. Solo players follow route plans independently which gives the impression of a

leader-follower relationship, but none exists. Testing missions prosecute without interruption until players return to their home base.

During tests, players followed planned routes to "patrol" an area of the grid. The graphs in Figures 5.1 and 5.2 show the paths taken by players during flight tests. Figure 5.1 illustrates the patrol route used by "enemy" forces while Figure 5.2 illustrates the route used by "friendly" forces. The inconsistent nature of the loops reflects efforts to incorporate wind and gravity effects. The trajectories of bogey1 and bogey2, however, are indistinguishable since, as independent aircraft, they fire the same rules in an identical sequence. In contrast, the erratic path in the "friendly" patrol loop was made by follower aircraft adjusting to changes made by its leader. Followers fire different rules than leaders which results in distinctive flight trajectories.

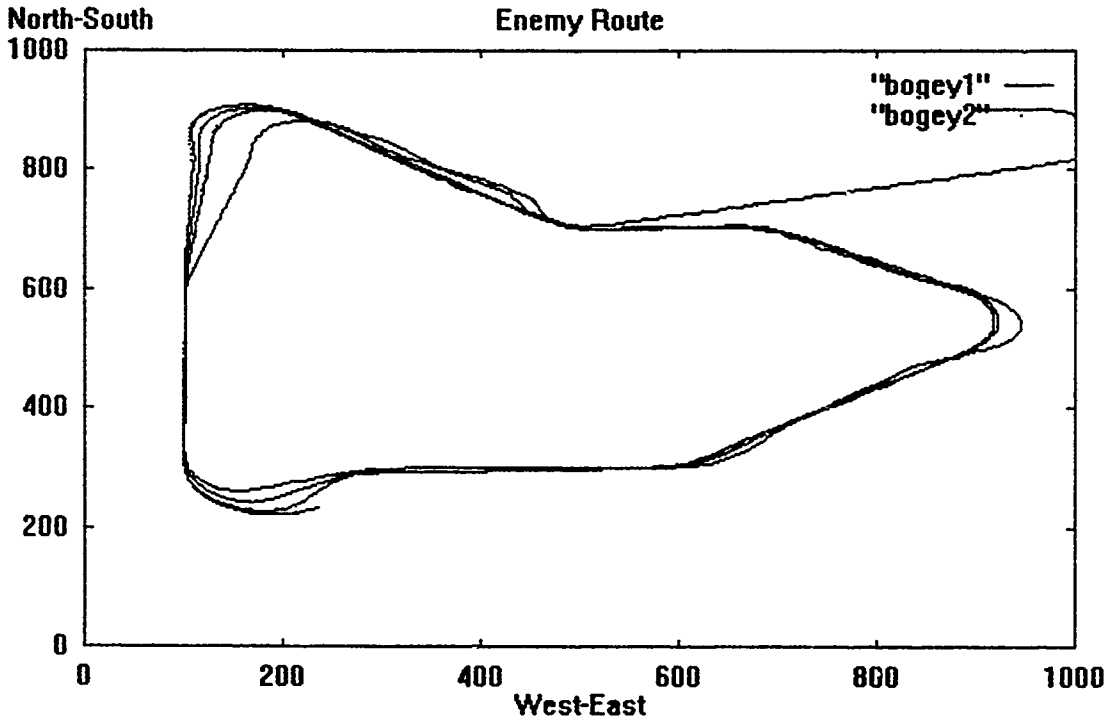


Figure 5.1. Planned Path of "Enemy" Forces Testing Mission.

The testing mission primarily provided a means of assessing flight control rules, but also produced a convenient way to define drone aircraft. Initial tests used drone aircraft to assess approach and attack maneuvers. In developing algorithms to score missile effectiveness against drones, I found a means of coordinating cooperative players. Details are discussed in Section 4.8.

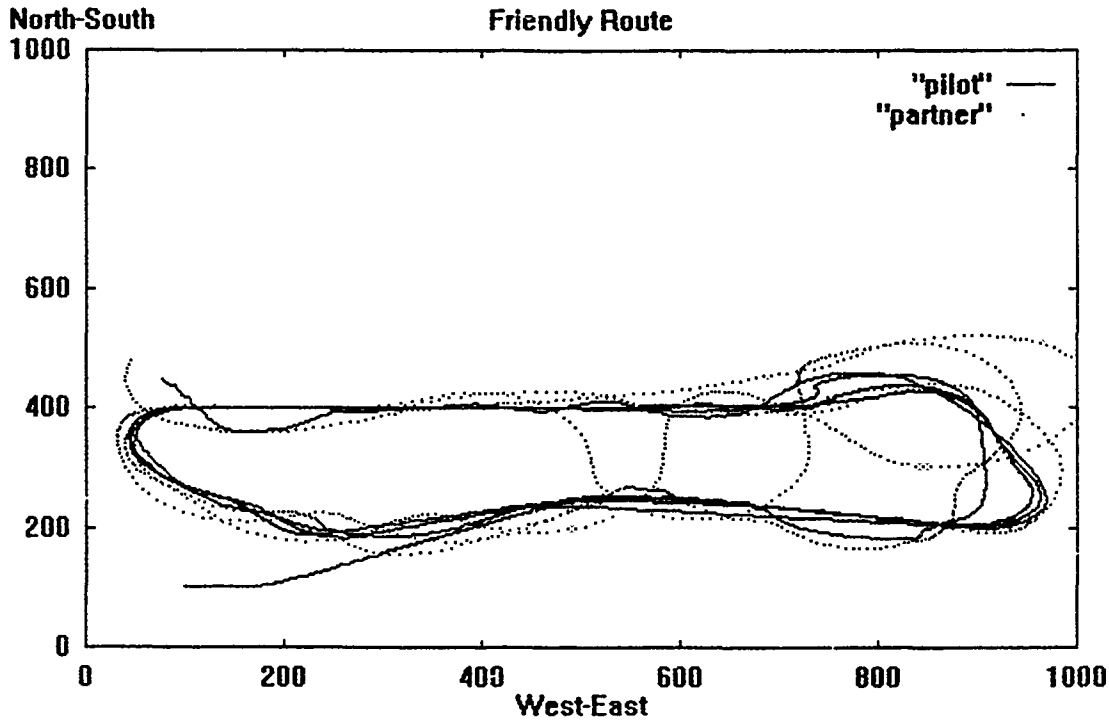


Figure 5.2. Planned Path of "Friendly" Forces Testing Mission.

5.1.3 Defensive Mission Scenario. In the defensive mission scenario, aircraft defended a portion of the grid. For the purpose of testing, I arbitrarily defined a border bisecting the playing grid. The southern half was defended by friendly forces while the northern half was assigned to enemy forces. Aircraft patrolled their assigned half of the grid searching for opposing forces. If an opposing aircraft entered defended airspace, it was attacked until it left the airspace. When all threats within defended airspace were

eliminated, defensive players resumed their assigned patrol plan. Two forms of patrol plan defined defensive routes: CAP-station and patrol.

5.1.3.1 Combat Air Patrol - Station. A CAP station was defined for each side in the simulation approximately halfway between a base and refueling point. Aircraft on patrol initially moved to and then circled the CAP station. While at the CAP station, players searched for opposing forces and responded depending on the threat to their air space. Cooperative players positioned themselves on opposite sides of the CAP station which expanded their effective search range beyond that of a single player. Independent players followed nearly identical paths, situating themselves in almost identical locations on each step. Consequently, independent players had a less effective search capability.

5.1.3.2 Combat Air Patrol - Patrol. After a preset time period, players patrolled their assigned airspace by changing to a different route plan. Using a patrol plan, aircraft moved between widely separated waypoints, again looking for opposing forces. Cooperative players flew in formation while solo players moved independently. At the transition from station to patrol assignments, rules detected changes in the leader's assignment and, in turn, changed the follower's assignment. The follower resumed a flight formation on the leader. After another preset time period, players returned to the CAP station. The planned routes for patrols followed the testing mission scenario.

5.1.4 Superiority Mission Scenario. Under superiority mission rules, players initially followed the defense mission plan, but once an opposing player was detected superiority aircraft attacked in either half of the grid. Defensive players chased opposing forces until they left defended airspace. Superiority players pressed the attack, even if the opposing player retreated.

5.1.5 Offense Mission Scenario. Under an offensive mission scenario, players initially behaved as drone aircraft. Unlike a drone, an offensive player will defend itself if attacked. Full implementation of an offensive player was left for future development.

5.2 Combat Scenarios

To test cooperative rules I used three types of combat scenario: one versus one, two versus one, and two versus two. A player's use of cooperative rules depended on the assigned role: leader, follower, or solo. For these tests, "friendly" forces were given cooperative abilities while enemy forces assumed the role of either drone or solo aircraft.

5.2.1 One Versus One Scenario. The one on one scenario pitted solo players from opposing sides against each other. This scenario provided a baseline of performance to compare against multiple aircraft scenarios. Target players were operated as drone and then defensive aircraft to isolate target acquisition from combat maneuvering capabilities. Detailed analysis of one versus one scenarios are treated by Hluck (Hluck, 1993).

5.2.2 Two Versus One Scenario. In the two versus one scenario, two players were pitted against a single opposing aircraft. The object was to attack the opposing aircraft and eliminate any threat. Team players were operated independently and then cooperatively to gauge the effect of cooperative rules. Solo players were operated as drone and then defensive aircraft as done in one versus one scenarios.

5.2.3 Two Versus Two Scenario. In the two versus two scenario, two players were pitted against two opposing aircraft. These tests offered a richer choice of combinations. Two independent or cooperative players maneuvered against an opposing team of two players. Target aircraft operated as drone players to clearly distinguish cooperative behaviors in attacking aircraft.

5.2.4 Neutral Players. The PDPC simulator has the capability of recognizing "neutral" players. Rules in place to defend against a hostile player also account for non-hostile players. Players whose "side" slot indicate they belong to a neutral third side are labeled benign threats and ignored. Neutral players were intended to build an escort mission capability. Future tests should assess this capability.

5.3 Performance Versus Drone Aircraft

5.3.1 Solo Player. Two players on opposing sides launch from opposite home bases and meet on the playing grid. The "friendly" player uses a defense mission while the enemy player follows the testing mission. Figure 5.3 depicts the trajectories followed by each player. (1) The pilot departs from the friendly base in the lower left corner and assumes position at the CAP station. (2) The drone, bogey1, launches from the "enemy" base at the top right hand corner of the grid and begins a patrol route. (3) As the drone approaches the friendly CAP station, the pilot continues patrolling. (4) At a distance of 500 units, the drone comes into radar range and is identified by the pilot as a threat. (5) The pilot tightly circles the drone to achieve a rear quarter approach. (6) The pilot chases the drone, achieves a rear quarter position and (7) launches a missile. (8) The missile destroys the bogey and the pilot returns to the CAP station.

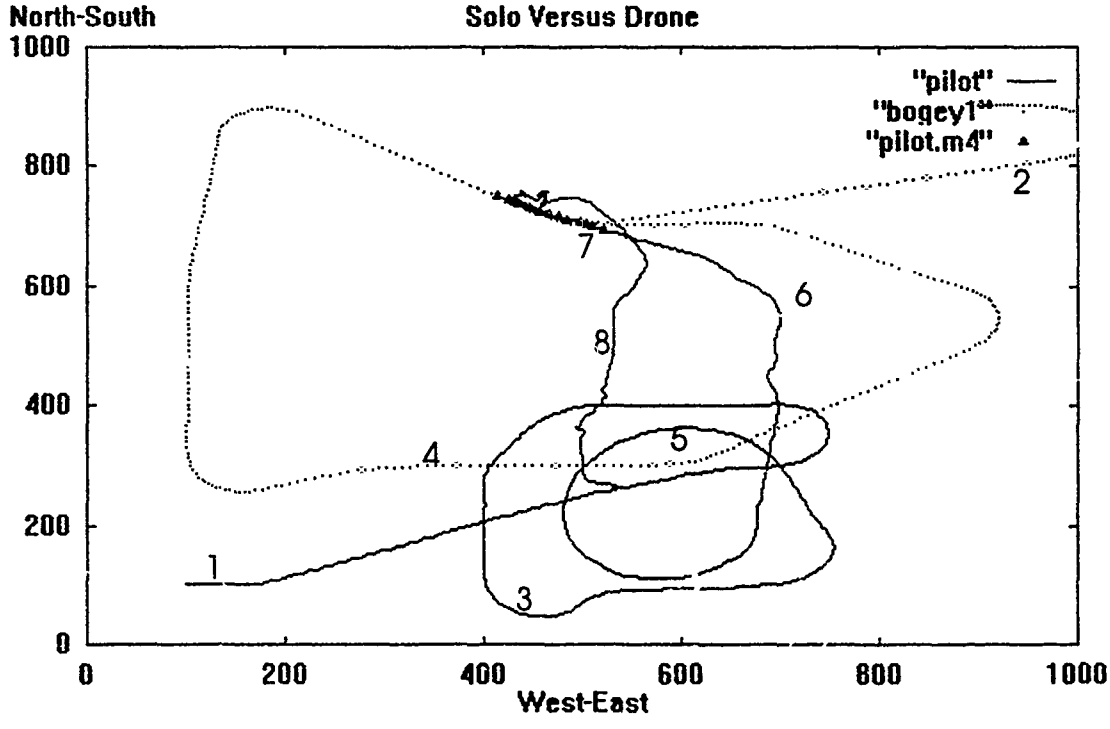


Figure 5.3. One Solo Player Versus a Drone.

5.3.2 Two Independent Players. Two solo players execute defense missions while an "enemy" drone enters "friendly" airspace. Team players act independently if their role slots have a "solo" designation and leader and follower slots are "none."

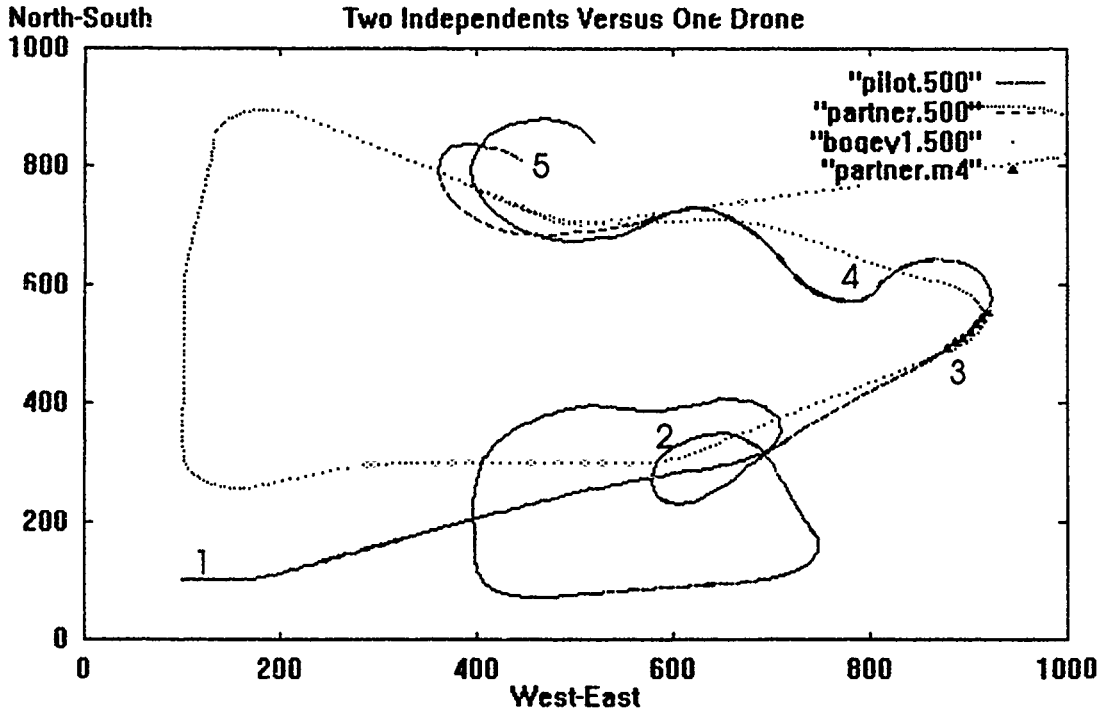


Figure 5.4. Two Independent Players Versus a Drone.

As in the solo player scenario, (1) aircraft depart from their home base and proceed to the CAP station. (2) Both "friendly" players detect the drone and turn tightly to achieve a rear quarter position. (3) Both players catch the drone on the eastern end of the grid, but only one fires a missile, the partner. The missile misses, and (4) both players continue to chase the drone without success.

Although only two traces appear in Figure 5.4, "friendly" force players followed the same path and fired the same rules to chase the bogey. Consequently, the pilot's trace and partner's trace overlapped identically. Late in the chase (point 5 in Figure 5.4) aircraft maneuvering parameters diverged sufficiently to produce separate traces for each player.

Typically, team players without an explicit leader-follower relationship copied each other's movements revealing the deterministic mechanisms of PDPC algorithms.

5.3.3 Two Cooperative Players. Two cooperative players execute the defense mission, but in this case "friendly" forces use cooperative rules. Figure 5.5 illustrates the paths in this two-on-one scenario where the partner's trace has been highlighted to distinguish "friendly" aircraft. Using cooperative rules, pilot and partner occupy opposite sides of the CAP-station (1a and 1b). (2) As the drone aircraft approaches from the west, "friendly" aircraft execute the CAP station plan. (3) The partner first detects the bogey at the point of turning south from the top of the CAP station loop. (4) The bogey continues through the CAP station with the partner chasing. (5) The pilot responds when the bogey nears the end of the CAP station. "Friendly" forces launch a total of three missiles during the ensuing chase (6, 7, and 8). Keys to this experiment are that "friendly" forces increased their effective kill range and tripled the number of missiles fired at the target.

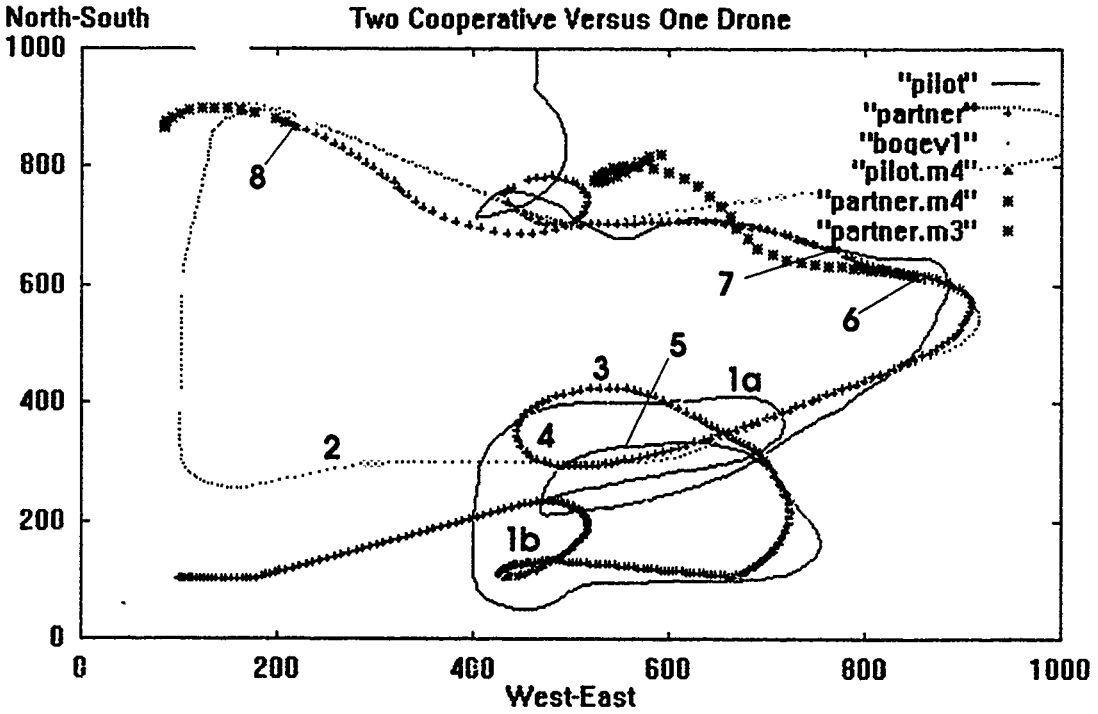


Figure 5.5. Two Cooperating Players Versus a Drone.

Cooperative players gained an advantage in effective search range by following rules for CAP station patrols. A standard procedure for CAP station patrols is for team members to occupy opposite sides of a CAP station loop (Shaw, 1985:327). The intent is for at least one team member to face the expected direction of a threat. The designated follower chose a different point of the CAP-station than the leader to simulate CAP-station patrol behavior. In this simulation, players occupying a CAP station function as solo aircraft. If the leader leaves to address a threat, followers return to a "follower" role. Complementary rules redirecting a leader when a follower addresses a threat were not fully functional which explains why the pilot did not respond to the partner's behavior.

The partner, however, does respond to the leader's behavior. Figure 5.6 depicts a modified two-versus-one scenario in which the leader first detects enemy aircraft. In this scenario, two drone bogeys follow a direct path to the CAP station (1). The leader detects both bogeys soon after reaching the CAP station (2) and begins a pursuit. (3) When the pilot leaves the CAP station, the partner attempts to rejoin the flight. (4) The pilot turns tightly into the rear quarter of both drones after a forward quarter pass. (5) Once in a firing position, the pilot launches a missile (which destroys one of the drones) and executes a "separation" maneuver taking him north of the second bogey. The partner is still trying to rejoin the pilot.

At point (6), the pilot is out of position for a shot attempt on the second drone as a result of the separation maneuver, but the partner is in position. (7) Before the partner can acquire the second drone, the pilot regains a rear quarter position and resumes the attack. (8) The partner defers to the pilot and again tries to approach the pilot from the rear quarter to assume a formation position. (9) The pilot pursues the second bogey while the partner chases the pilot. (10) A second missile launch misses the second drone, but (11) a third missile destroys the target. (12) Both "friendly" players return to the CAP station to resume patrol.

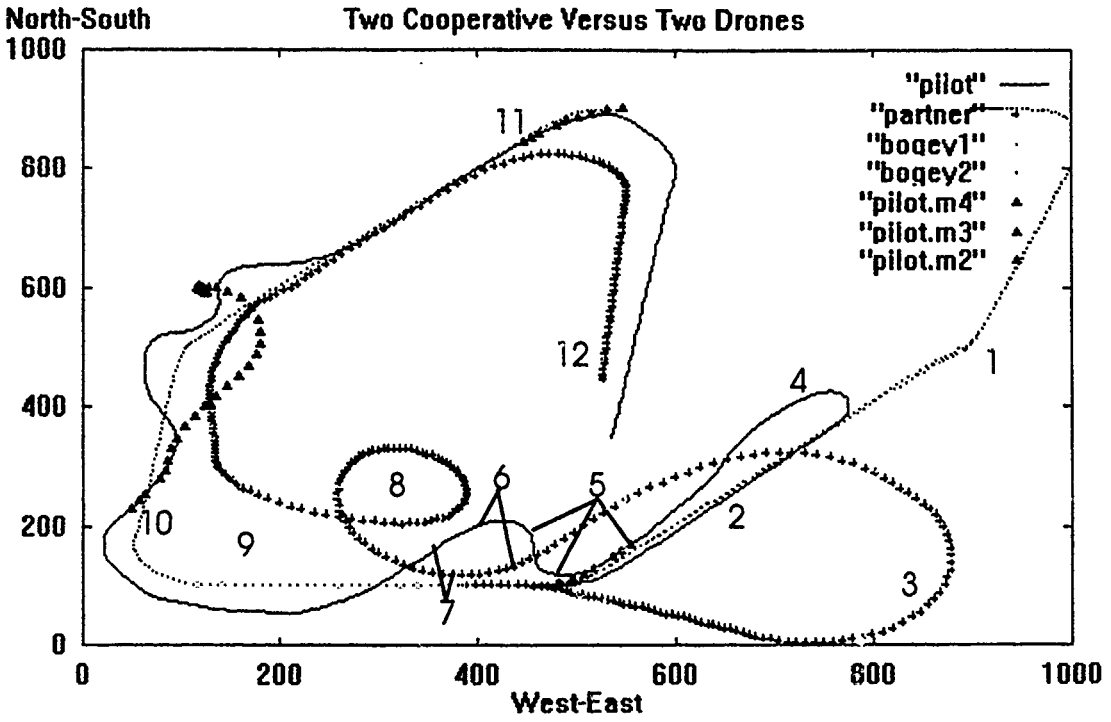


Figure 5.6. Two Cooperating Players Versus Two Drones.

This test illustrated how cooperating players use strategy when addressing threats. In the two-versus-one demonstration, both pilot and partner fired missiles at the bogey indicating they behaved as solo fighters. In the two-versus-two demonstration, the partner did not fire on the bogey, instead deferring to the pilot's leadership role. Pilot and partner enacted the fighting-wing strategy which stipulates that a wingman should cover his leader rather than engage the bogey (Shaw, 1985:198).

5.4 Performance Versus Defensive Solo Aircraft

5.4.1 Solo Player. In this test, two players have a defense mission and use the full battery of maneuver, decision, and engagement rules. Figure 5.7 depicts the engagement as players meet near the border. (1) The pilot enters from the south and turns west as the bogey moves east, so the combatants approach each other head-on with lateral separation.

(2) The pilot turns sharply to the north in a lag pursuit maneuver. (3) Bogey1 continues east to turns toward the south. (4) The pilot successfully continues lag pursuit through the bogey's turn. (5) Each player uses vertical maneuvers attempting to achieve a rear quarter position. (6) Players continue vertical maneuvers and drift toward the south. Neither player gains sufficient advantage to fire a missile.

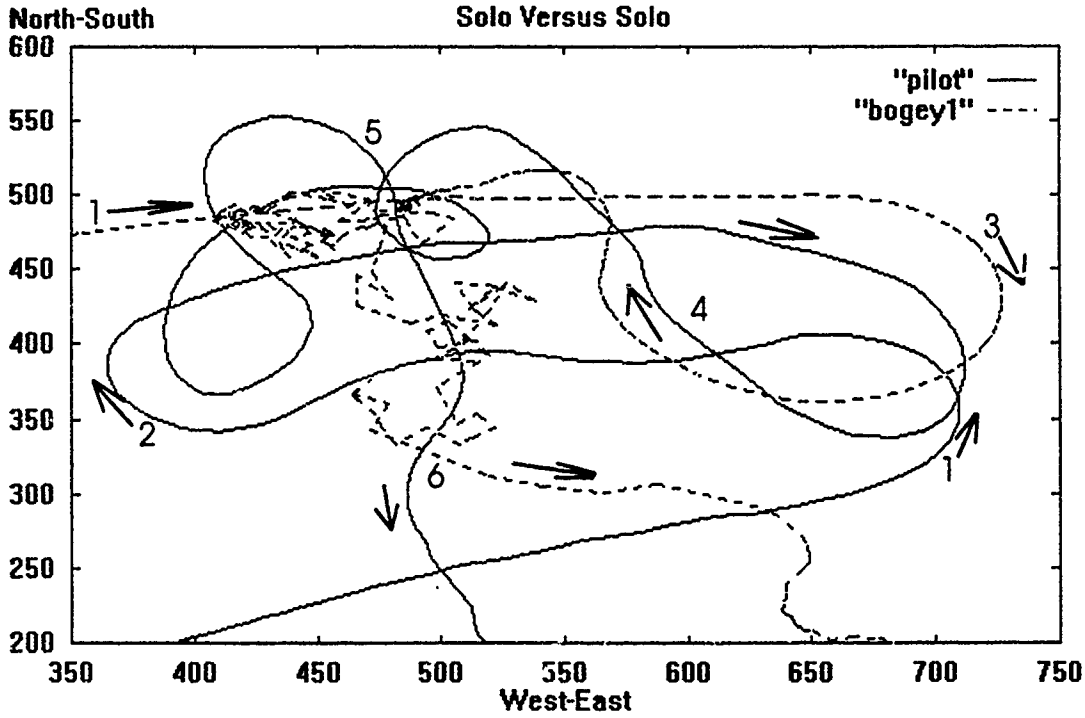


Figure 5.7. One Versus One - Fully Functional Players.

The one-versus-one engagement using fully functional adversaries illustrates the complexity of engagements when drones are not used. Hluck provides detailed analysis of solo player capabilities and one versus one engagements (Hluck, 1993).

The scenario in Figure 5.7 also highlights deficiencies in the PDPC flight model. The bogey's erratic behavior occured during vertical maneuvers because roll, pitch, and yaw are inter-related within flight model calculations. Heading is calculated as the angle between the velocity vector and the positive x-axis. Roll angle is calculated as the angle between the wing vector and the xy-plane. As an aircraft passes through a climb angle of

90 degrees, heading shifts by 180 degrees and roll angle passes through zero degrees even if the aircraft was trying to maintain constant heading and roll angles. The result is erratic behavior displayed by any player operating near climb angles of 90 or -90 degrees.

5.4.2 Two Independent Players. Figure 5.8 depicts a scenario of two independent players versus a single opponent. In this test, "friendly" forces use a pair of fully functional players that have solo, or independent, roles. The "enemy" force is composed of a single fully functional player in a defensive role. (1) Bogey1 launches from the north-east corner while pilot and partner launch from the south-west. As in the demonstration of two independent players versus one drone, the pilot's track masks the partner's since both follow identical rules. (2) The engagement begins much like the one-on-one scenario with opposing sides facing each other near the center of the grid. Bogey1 abandons maneuvering, however, in an attempt to evade superior numbers. (3) The ensuing chase

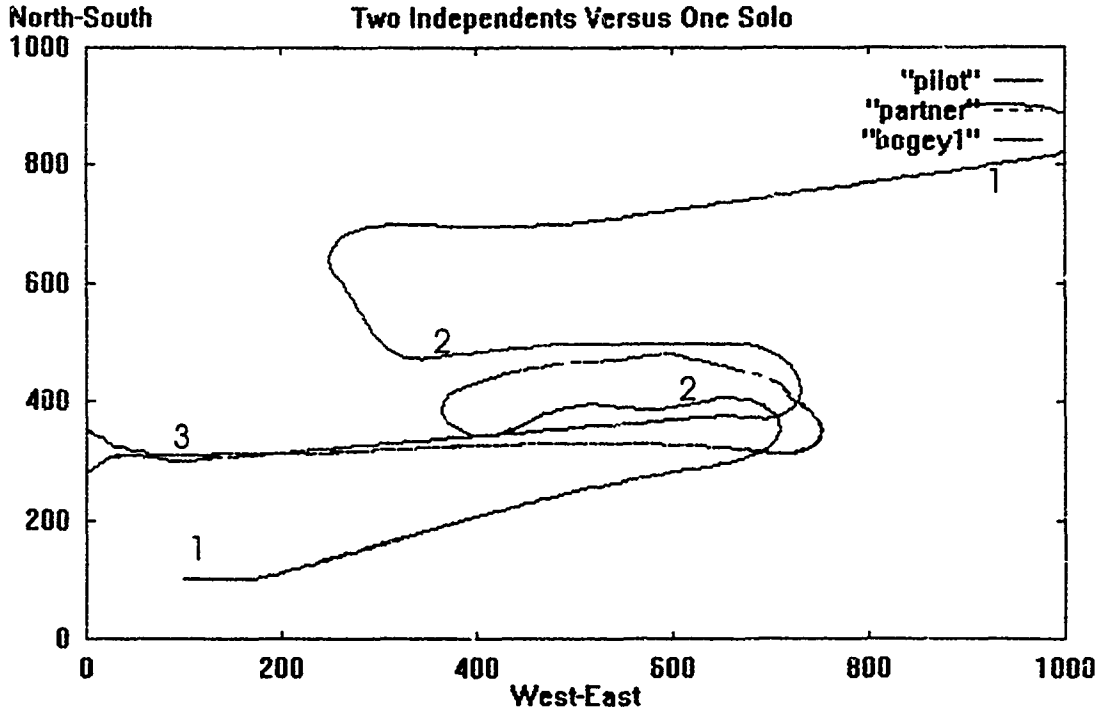


Figure 5.8. Two Versus One - Fully Functional Players.

proceeds west without conclusion. The two-versus-one with fully functional players restates the problem of independent players: identical actions and reactions.

5.4.3 Two Cooperative Players. In this test, "friendly" forces use cooperative rules against a single "enemy" defender. Figures 5.9 through 5.11 depict the encounter through time steps 250, 500 and 750, respectively. At first, all three players assume a defensive posture at a CAP station. The pilot and partner go to opposite sides of the "friendly" CAP station in the south (a2 and b2) while bogey1 proceeds to the "enemy" CAP station in the north (c2). Just before time step $t = 250$ the pilot detects bogey1 and signals the partner to rejoin in a formation. The partner acknowledges and moves toward the pilot. The positions of pilot, partner and bogey1 at time $t=250$ are shown in Figures 5.9 and 5.10 as a3, b3, and c3, respectively.

The behaviors of "friendly" players were different than in either the one-on-one or two independent player scenario. Pilot and partner traces did not overlap because the

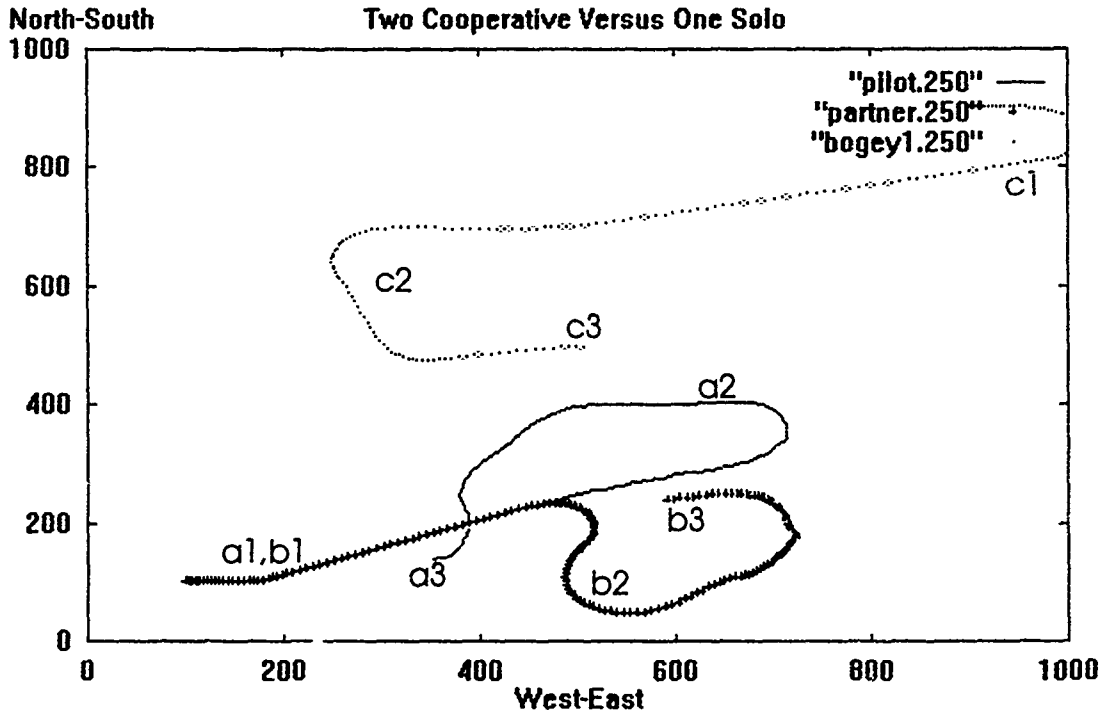


Figure 5.9. Two Cooperative Players Versus One Defensive Player (t=250).

partner employed rules explicitly designed for follower actions. When the pilot decides to engage the bogey, the partner returns to formation flight rather than attack the bogey directly. The pilot and bogey did not immediately engage, but this was due to code refinements that did not affect leader-follower relationships.

Figure 5.10 illustrates mid-game maneuvering. The bogey has not responded yet because the pilot is not considered a threat (c4). The pilot, however, considers the bogey to be a threat and moves north to intercept (a4). As the pilot reaches the border, the bogey threat level is reassessed and reduced since the bogey is retreating north (c5). This leads the pilot to a return to CAP station duties (a5). Pilot and partner (b5) both return to CAP station duties until the pilot again comes within range of bogey1 (a6 and c6). At time *t=500* the pilot and bogey engage (a7,c7) in a nose-to-nose turn. As bogey1 turns west, the partner crosses his path (b8 and c8) attempting to rejoin the pilot (a8).

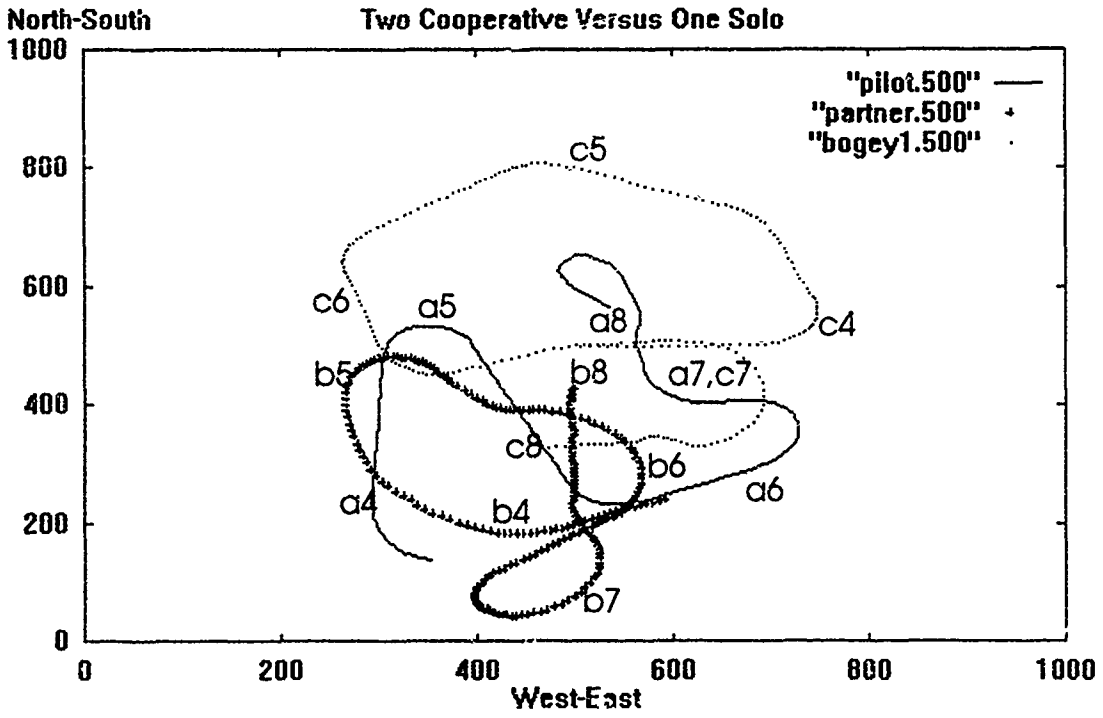


Figure 5.10. Two Cooperative Players Versus One Defensive Player (t=500).

The mid-game illustrates how cooperative rules indirectly affect players. Shortly after the pilot and bogey executed a nose-to-nose turn, the bogey detected the partner crossing his path. Instead of continuing a turn into the pilot, bogey1 turned into the rear quarter of the partner interrupting his primary engagement with the pilot. The bogeys behavior changed because the partner out of position. If the pilot and partner were defined as independent players, the partner would have followed the pilot's track identically and the bogey would not have been interrupted.

Figure 5.11 depicts the end-game. Bogey1 turns south (c8) and approaches the pilot, who is proceeding south toward the bogey. Pilot and bogey pass (a9 and c9) and the pilot pursues (a10). The partner turns in the east (b9) attempting to rejoin the pilot. Partner and pilot pass each other (a10 and b10) as the pilot proceeds north to engage the bogey. The bogey turned south (c10) to meet the pilot. The pilot and bogey engage using vertical maneuvers (a11, c11) and the simulation ends with no player gaining an advantage.

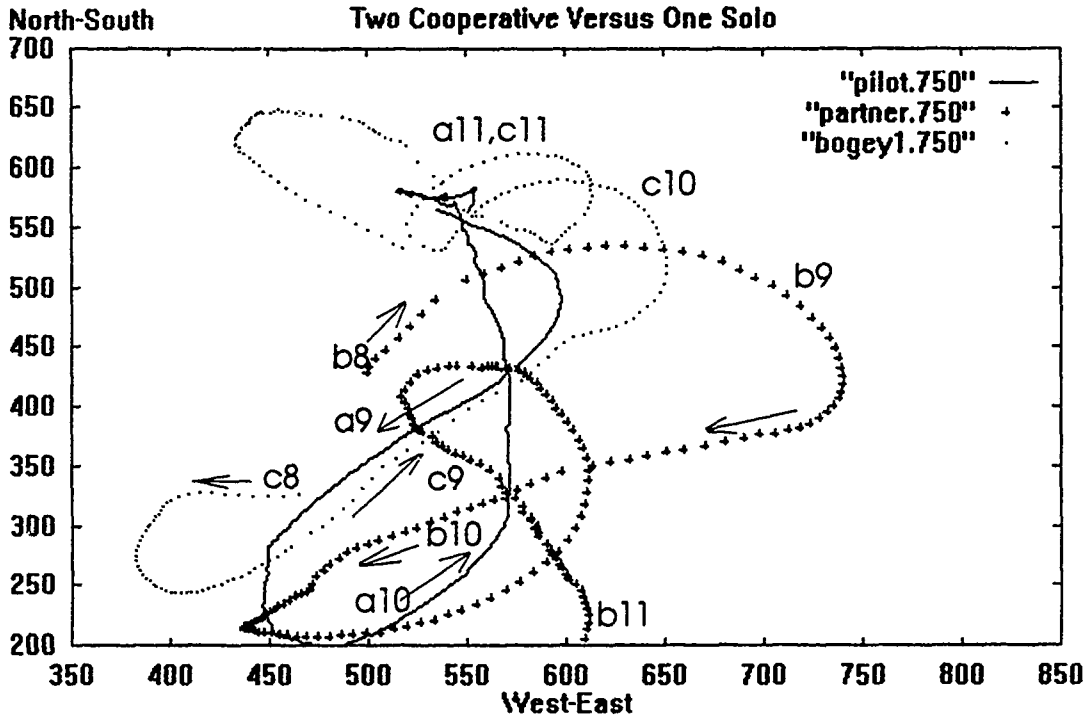


Figure 5.11. Two Cooperative Players Versus One Defensive Player (t=500).

5.5 Cooperative Maneuvering

Cooperative maneuvers require that two players have knowledge of each other and some mechanism exists to coordinate actions. Figure 5.12 depicts one cooperative maneuver within PDPC: an offensive split turn. Shaw describes an "offensive split", or "bracket", as an attempt by two team players to surround a target (Shaw, 1986:204). I called this maneuver a "split-turn".

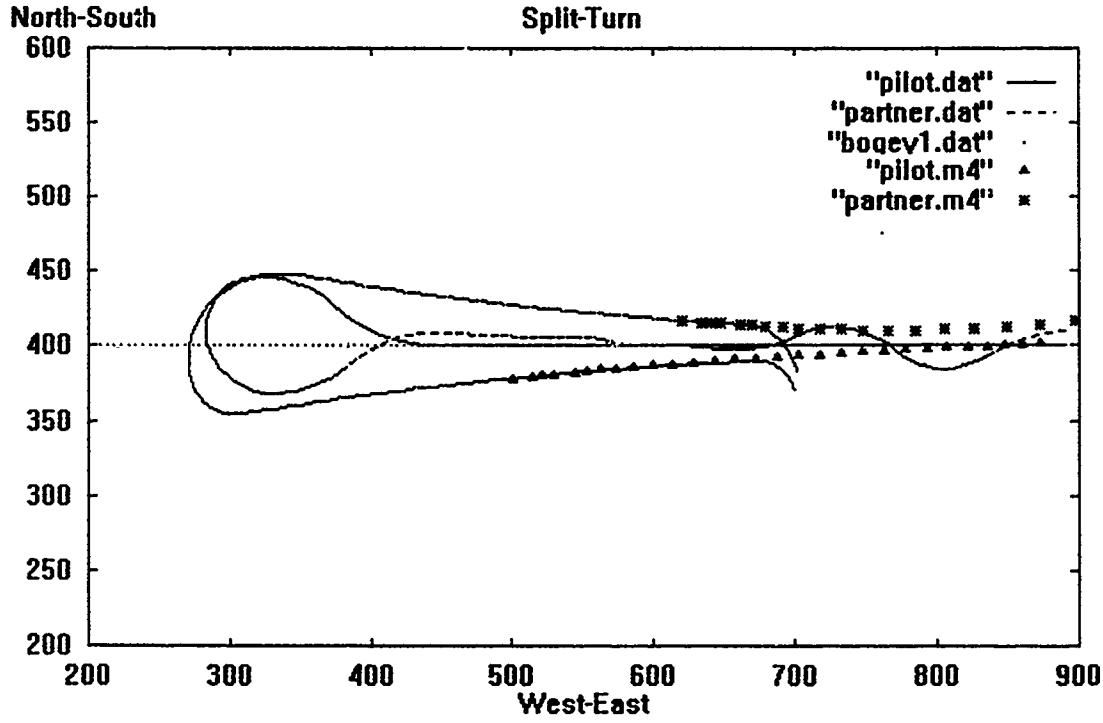


Figure 5.12. Offensive Split-Turn Maneuver.

5.5.1 Offensive Split Turn Maneuver. In Figure 5.12, two "friendly" players enter from the east as a single "enemy" drone approaches from the west. The partner moves in formation with the pilot until the team reaches a point ten closing-velocity units in front of bogey1 (the drone) where the pilot calls for a "split-turn" maneuver. The pilot

executes a "separation-pursuit" maneuver generating a northward turn (Hluck, 1993). The partner executes a "split-turn" maneuver which uses identical code, but negated parameters to generate a way point on the opposite side of the target . The partner follows a southward turn. As bogey1 passes, both "friendly" players execute lead, lag, and pure pursuit maneuvers to achieve a rear quarter position and acquire the target. The pilot acquires bogey1 first and launches a missile that destroys the target. The partner also launches a missile because bogey1 still exists when the partner fires. When the target is destroyed, it is deleted from the simulation. The partner's missile loses track of the target and flies to the east.

5.5.2 Coordinated Team Players. This serves to illustrate the problems in coordinating team players. Separate rules must be in place for leaders and followers. Depending on strategy selection, followers typically remain in Chase phase while the leader engages a target. To execute a cooperative maneuver, followers must first be prompted by the leader. Rules communicating this prompt can use either object slot data or factlist messages. Once prompted, followers need rules that are distinct from the leader or identical behaviors will result.

5.6 System Performance

Overall system performance depended on the number and phase of players in a simulation scenario. The greater the number of players and the more complex a phase module, the longer PDPC took to process. Preliminary timing analyses show promise that PDPC could be converted to a "real-time" base. The following two sections analyze preliminary timing data collected during simulation runs. Simulations that generated the timing data were run on a Sparcstation 2 which was part of an AFIT computer network.

5.6.1 Two Versus One Timing Analysis. Timing data for a two versus one scenario is shown in Figure 5.13. The "real-time" increment for each simulation time step varied as simulations progressed. During the simulation depicted in Figure 5.13, a single drone remains in Cruise or Search phase throughout the test. Two players, the pilot and partner, progress through different phases in an effort to destroy the drone.

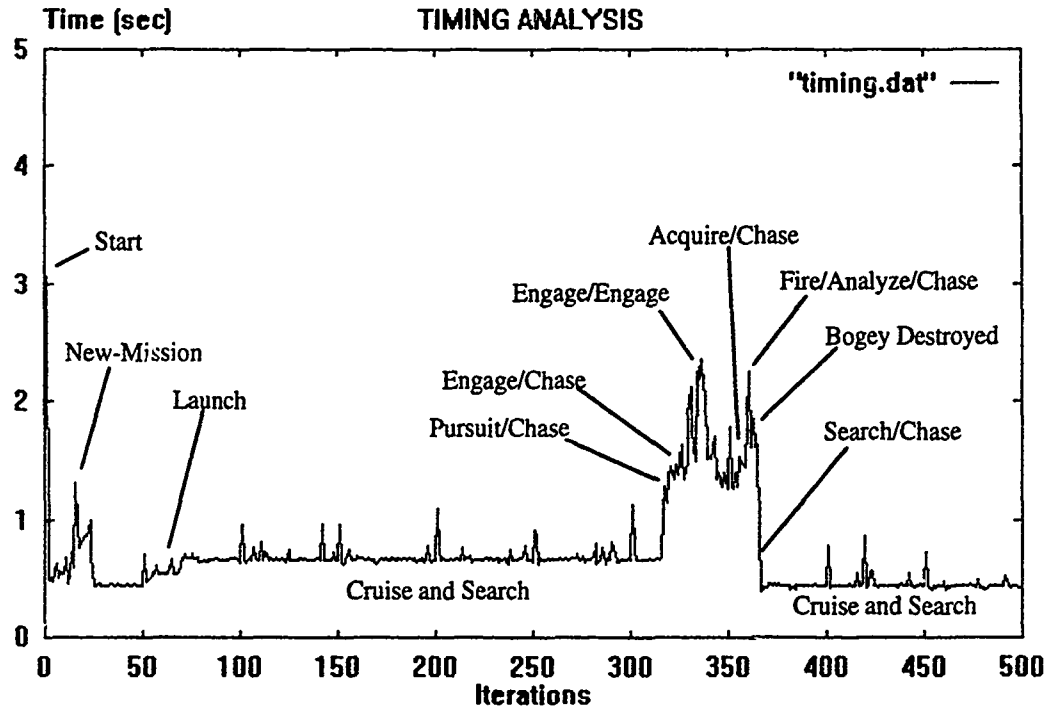


Figure 5.13. Two Versus One Timing Analysis.

When the drone comes within range, the pilot and partner increase processing time demands on the system. The point of highest demand occurs when both the pilot and partner enter the Engage phase to execute a "split-turn." Each execution of a "split-turn" increment required approximately 2.1 seconds. A second peak occurs when the pilot launches a missile. This action introduces a fourth player into the simulation (the missile) causing processing time demand to increase. Demand reduces significantly when the missile hits the target because two players are deleted from the simulation and the pilot

and partner return to the Cruise phase. With only two players in the simulation operating in the same phase module, time demand remains less than one second per iteration.

5.6.2 Two Versus Two Timing Analysis. An analysis of a two versus two scenario is depicted in Figure 5.14. In this scenario two teams of cooperating players engage each other. Preliminary analysis indicates all phases have a higher average time per iteration, but the amount required in any phase remained less than five seconds.

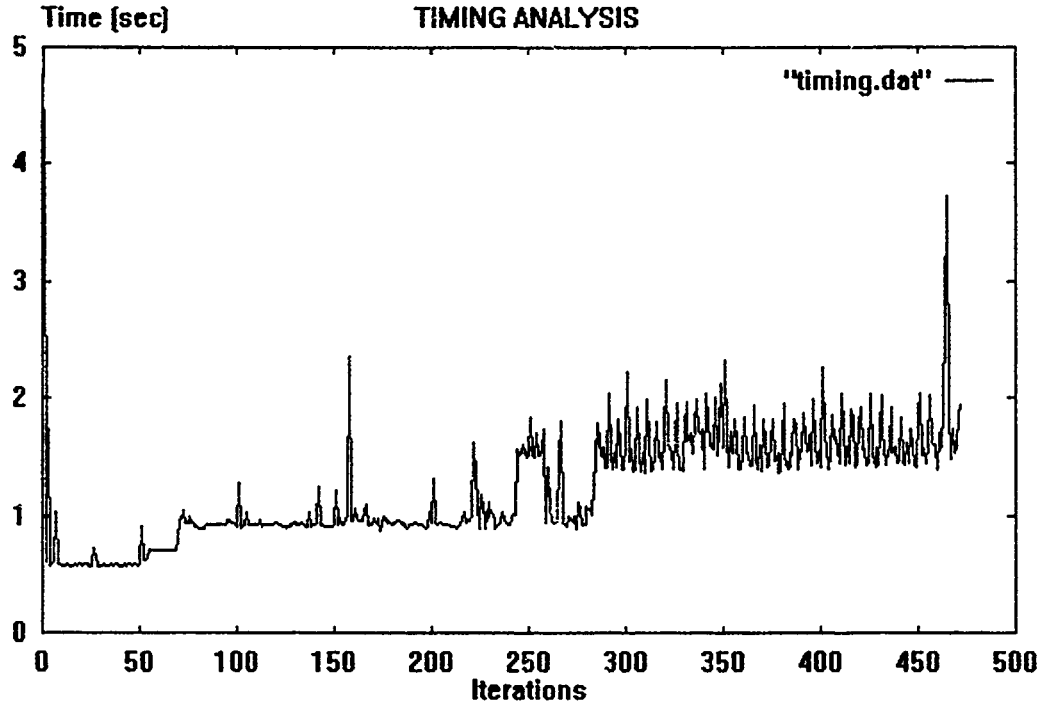


Figure 5.14. Two Versus Two Timing Analysis.

5.7 Summary

The PDPC simulated various air combat scenarios including one-versus-one, two-versus-one, and two-versus-two. Players used planned routes to move through simulated airspace unless they encountered opposing players. Engaged players used the rules in

phase and decision modules to decide on a course of action. The same rules were used for both independent and cooperative team players. Cooperative players gained advantages over independent players in search and maneuvering capabilities. Overall system performance depended on the number of players employed and modules accessed.

6. Conclusions and Recommendations

Cooperative behaviors in autonomous agents of an air combat simulation can be implemented using universal reactive plans in a modularized knowledge-base system. The intent of this thesis was to investigate, develop, and implement cooperative behaviors in autonomous air combat agents for operation within the ARPA Distributed Interactive Simulation program. The PDPC implements pilot decision-making processes and behavior patterns, including cooperative behaviors, in an air combat simulation.

6.1 Conclusions

6.1.1 Cooperative Behaviors PDPC agents demonstrated cooperative behaviors in terms of leader/follower role switching, cooperative maneuvering, and coordinated action.

6.1.1.1 Role Switching. The PDPC implemented rules that resulted in role switching between leader and follower. Depending on strategy, followers chased the leader or assumed the lead role during engagements. If the current strategy was "fighting-wing", followers retained a follower role and deferred to the leader during engagements. Followers assumed a lead role if the current strategy was not "fighting-wing" and the follower had a greater position advantage on a target. On a role switch, leaders assumed a follower role until the position advantage reversed. Role switching improved team performance by increasing the number of shot opportunities presented to players.

6.1.1.2 Cooperative Maneuvering. Cooperative players maneuvered in concert during engagements. The "split-turn" maneuver required coordination between players attempting to surround a target. Pre-conditions on executing the "split-turn"

required that the leader ensure the follower was in position before beginning the maneuver. Once initiated, leader and follower used complementary rules to achieve a rear-quarter position advantage. Cooperative maneuvering also resulted in improved team performance since both leader and follower engage a target.

6.1.1.3 Coordinated Actions. Coordination between players required either knowledge of teammate object data or use of explicit messages. Followers used the status of a leader's phase to key some actions. For example, when a leader destroyed a target and returned to Cruise phase, the follower detected the phase change and responded by also returning to Cruise phase. Leaders prompted followers to key other actions. A leader on CAP station assignment detected an incoming threat and used an explicit message to instruct the follower to reform for an attack.

In phase control modules, special rules coordinated leaders and followers. The primary task of a follower was to follow the leader. Followers executed rules designed to achieve a flight formation on the leader rather than fly to the leader's goal. Scenarios involving two solo players on the same team resulted in overlapping flight paths because players used the same goals over identical rule firing sequences. Teams using a leader/follower relationship produced separate tracks because players neither had the same goals nor fired the same rules.

In the decision modules, rule design allowed activation by either a leader or follower. For example, followers who detected new threats performed threat analysis and shared data with their leaders. Followers also committed to threats, planned intercepts, and reacted to missile warnings regardless of team role. Coordination rules allowed leaders and followers to act independently while maintaining a leader/follower relationship.

6.1.2 Air Combat Simulation The PDPC enacted air combat simulations sufficient for research purposes. Agents demonstrated enhanced realism in maneuvering, strategy and tactics, and pilot decision-making.

6.1.2.1 Realism in Air Combat Maneuvering. Simulation results depicted air combat maneuvering sufficient for research of computer generated forces. Adversary players recognized and confronted each other when they came within range and were assessed as a threat. Team players cooperated on a leader/follower basis and demonstrated improved performance both over single player threats and independent threats. Firing distinct rule sequences for leaders and followers added realism to combat simulations.

PDPC players executed a variety of maneuvers, including both and cooperative maneuvers. A predecessor of PDPC, MAXIM, used universal reactive plans to implement intelligent adversaries in an air combat simulation. MAXIM agents employed maximum turn-rate maneuvering tactics which resulted in abrupt course changes rather than smooth curves (Dyer and Gunsch, 1993). PDPC maneuvers operated at variable turn-rates resulting smooth curve trajectories rather than abrupt turns.

6.1.2.2 Air Combat Tactics and Strategy. The PDPC implements strategies and tactics discussed by Shaw in his book *Fighter Combat, Tactics and Maneuvering* (Shaw, 1985). Individual players adopted an "energy" or "angles" fighter strategy depending on relative energies between aircraft. Team players determined a strategic approach and executed two player team strategies such as "fighting wing" and "loose deuce." MAXIM agents followed a "shoot first" philosophy and so did not allow a choice of strategies or tactics beyond search, attack, and evade (Dyer and Gunsch, 1993). PDPC implements a comprehensive range of strategies, tactics, and maneuvers to enhance realism in air combat simulations.

6.1.2.3 Combat Pilot Decision-Making. Rules were based on parameters and metrics outlined in the Titan report (Titan, 1986). Each of 40 parameters were implemented as either an object slot or function call. Function-based parameters derived a given parameter from data contained in object slots. Rules used both slot data and

function-derived information to determine appropriate actions. The result is a set of six rule modules that provide a second level of reasoning for PDPC agents.

6.1.3 Rule Modules as Reactive Plans. Rule modules act as reactive plans for each agent in PDPC. Since the basic CLIPS rule follows a predicate-consequent construction, individual rules enact a stimulus-response behavior. Rules could act singly, such as with a phase change, or in concert, such as with maneuver selection. Since rules fire once on each iteration, PDPC implements the basic approach of a reactive system (Rich and Knight, 1991:331). Rule modules provide a reactive plan of action for each phase of a mission.

6.1.4 Modularized Knowledge-Based System The PDPC captures the knowledge and experience of air combat pilots in a modular knowledge-based architecture. Knowledge is contained in rules which execute actions dependent on specific pre-conditions. The PDPC architecture compartmentalized rules into modules based on the stages of an air combat mission and the levels of a pilot decision-making model.

6.2 Recommendations

6.2.1 PDPC as a Research Tool. The PDPC simulator should be used to further investigate cooperative behaviors in autonomous agents. PDPC required extensive knowledge engineering, design, and implementation. The result is an air combat simulation environment offering several advantages. First, PDPC offers an extensive knowledge base defining maneuvers, tactics, and strategies for agents in an air combat simulation. Second, agents function autonomously, so experiments could be designed to depict complex air combat scenarios. Third, all code was written in CLIPS to take advantage of rapid prototyping, portability, and development environment features. PDPC could prove valuable to research in the field of air combat simulation.

6.2.2 Input and Output. Considering the most reliable output to date is Gnuplot files, output systems are inadequate for development of an interactive "real-time" system. PDPC generates location, timing, log, and "Red Flag" data files. Location data depicts the x, y, z, coordinates of each agent during a simulation run. Timing data depicts the time per iteration of simulation runs. Location and timing data files were used to generate figures for this thesis. Log files collect output from format statements embedded in PDPC rules. A log file contains a chronological history of rule firings during simulation runs. "Red-Flag" data files contain data suitable for broadcast over the AFIT BattleSim simulator to a "Red Flag Terrain Map" simulator. PDPC output formats are suitable for research purposes and could support development of an interactive interface.

6.2.3 CLIPS Pitfalls. One coding pitfall was in how CLIPS handled global variables on the left-hand side of rules. All global variables were initially defined in the Main module to ensure each rule used common values. For example, maximum radar range was defined globally to ensure all rules that checked target distance only considered objects within maximum radar range. When phase or decision modules imported global variables, they were not visible to CLIPS when used on the left-hand side of a rule. The solution was to redefine global variables within each module when they were needed on the left-hand side of a rule.

6.2.4 Flight Model Deficiencies. The algorithms used to generate agent motion did not provide sufficient control for formation flight. Movement control was based on reorienting a player's nose vector before each move rather than on reorienting a lift vector. Redefining motion in terms of lift vectors could produce more realistic flight trajectories.

6.2.5 Strategy Limitations. The PDPC limits players to acquiring a target and launching a missile from a rear quarter approach. Forward and beam quarter attacks could be coded, but heuristics measuring the value of each approach would need to be developed. This aspect of PDPC should be developed in future efforts.

6.3 Future Work

6.3.1 Distributed Interactive Simulation Interface. Before connecting PDPC to DIS, an interactive network interface must be developed and installed. The PDPC described in this thesis can generate location data files, but can neither generate nor receive DIS protocol packets. An interface module could be developed to process network functions. The module would need packet receivers and translators to position DIS participants within the CLIPS environment. It would also need packet builders and transmitters to put player information onto the network.

6.3.2 Prioritized Parameters. A scheme to prioritize data parameters with respect to each other should be implemented. The Titan report rank ordered data parameters according to results of a survey of combat fighter pilots (Titan, 1986:79). For example, on average, Air Force pilots felt that the type and identification (friend or foe) of aircraft were the most important parameters to know about a target. Rate of climb was the least important. In PDPC, all of the parameters are available as object slot data, but none are weighted with respect to each other. A scheme to prioritize data parameters was left for future research.

6.3.3 Maneuver Modules. Phase control modules contained duplicate, or near duplicate code to generate maneuvers. As a consequence of rapid prototyping, many rules contained identical predicate and consequent code sequences. Rule modules contained sets of nearly identical rules. A potential method of improving system performance would identify common characteristics of rules and rule modules and extract those characteristics to a separate rule module. For example, lead, lag, and pure pursuit rules were built into six phase control modules. These rules could be removed from phase modules and replaced by a separate "maneuver" module containing three rules: one for lead, lag, and pure pursuit. A rule to focus onto the maneuver module would replace the lead, lag, and

pure pursuit rules in each phase module. The overall rule count would reduce by nine rules. Similar efficiency questions are contained in each rule module. A thorough study could result in significant system performance improvements.

6.3.4 Generative Planning. An alternative form of planning could improve agent effectiveness within simulation runs. The PDPC currently contains a rudimentary intercept trajectory planner, but does not provide means to integrate overall mission goals or assign target value. A high level generative planner could replace the "target-data" sub-module (see Figure 4.32) and provide a more effective means of updating threat assessments and target values.

6.3.5 Offensive Missions. To enhance simulation realism, the offensive mission should be fully developed. The offensive mission would use a route plan similar to a patrol route plan, but pilots should consider targets of opportunity, terrain masking, and the value of retreating versus attacking.

6.3.6 Networked Simulations. A step toward developing a fully interactive DIS interface would be to network PDPC simulations running on separate machines. A network interface would need to be developed. As an initial effort, a simple interface would use the CLIPS "save-instance" and "restore-instance" functions to write object data to a common data structure. An advanced network interface would need to manage many sources.

6.3.7 "Fogging" Perfect Data. Another improvement to realism would prevent agents in the simulation from using perfect data. Players in PDPC do not change phase, select a maneuver, or fire a missile unless the predicate portion of rules are satisfied *exactly.* Data used to test each predicate comes from object slot data; perfect data. For example, a PDPC agent measures target distance by taking the difference between two x, y, z coordinate points. The result is a measurement precise to several decimal places. "Fogging" the data would reduce the precision, but enhance the realism.

6.4 Summary

The PDPC simulator demonstrates a means of eliciting cooperative behaviors from autonomous agents in an air combat simulation. The resulting system offers an environment to explore agent interaction across complex air combat scenarios. Potential performance improvements could result from code "tuning" and rule extraction. Realism could be enhanced with the addition of offensive mission modules and a means of "fogging" the data. As a research tool, PDPC could serve as a basis for future work on interactive, networked air combat simulations.

Appendix A. Vector Mathematics for Air Combat Maneuvering

A.1. Introduction.

This appendix reviews basic mathematical concepts for analyzing vectors and then defines vector spaces used in air combat maneuvering. My source is *Calculus and Analytic Geometry* by Mizrahi and Sullivan (Wadsworth, Inc. 1982). My intent is to introduce some key terms and definitions we used to develop our simulator.

A.2. Vector Math.

To implement computer generated forces in a multi-dimensional simulation, we need a firm grounding in basic vector mathematics.

A.2.1. Vectors and Vector Space. Vector space is a rectangular coordinate system based on any three vectors: **x**, **y**, and **z**. Vectors are defined on these axes and any vector will have the form:

$$\mathbf{V} = a\mathbf{x} + b\mathbf{y} + c\mathbf{z} \tag{A.1}$$

where **V** is the three space vector, a, b, and c are constants, and **x**, **y** and **z** are unit vectors along each of three coordinate axes. Since our simulator contains several vector spaces, we adopted a shorter notation for vectors: $\mathbf{V} = (a, b, c)$. Unit vectors are implied and depend on the definition of the vector space.

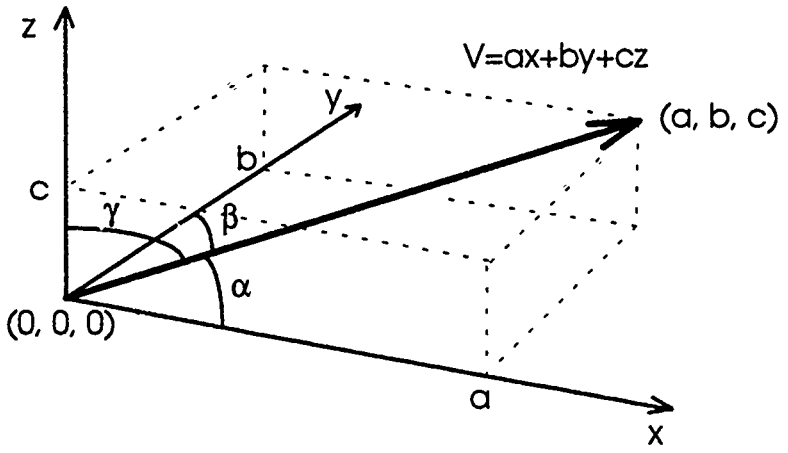


Figure A.1. Vectors in Vector Space.

A.2.2. Vector Magnitude. Vector magnitude is defined as:

$$\|\mathbf{V}\| = \sqrt{a^2 + b^2 + c^2} \tag{A.2}$$

where **V**, a, b, and c are as defined above.

A.2.3. Direction Cosines. Direction cosines indicate the oblique angles between a vector and each of the three coordinate axes.

$$\cos\alpha = \frac{a}{\|\mathbf{V}\|} \quad \cos\beta = \frac{b}{\|\mathbf{V}\|} \quad \cos\gamma = \frac{c}{\|\mathbf{V}\|} \tag{A.3}$$

where **V**, a, b, and c are as defined above and α , β , and γ are angles off the **x**, **y** and **z** axes, respectively.

A.2.4. Dot Product. The dot or scalar product is:

$$\mathbf{V} \bullet \mathbf{W} = a_1 a_2 + b_1 b_2 + c_1 c_2 \tag{A.4}$$

where $\mathbf{V} = (a_1, b_1, c_1)$ and $\mathbf{W} = (a_2, b_2, c_2)$.

A.2.5. Angle Between Vectors. The angle between two vectors is:

$$\cos\theta = \frac{\mathbf{V} \bullet \mathbf{W}}{\|\mathbf{V}\| \cdot \|\mathbf{W}\|} \tag{A.5}$$

where **V** and **W** are as defined above and θ is the angle between them.

A.2.6. Vector Projection. The projection of one vector along a second is then:

$$\|\mathbf{V}\|\cos\theta = \frac{\mathbf{V}\bullet\mathbf{W}}{\|\mathbf{W}\|} \tag{A.6}$$

where all symbols are as previously defined.

A.2.7. Cross Product. The cross product between two vectors is:

$$\mathbf{V}\times\mathbf{W} = (b_1c_2 - b_2c_1)\mathbf{x} + (a_2c_1 - a_1c_2)\mathbf{y} + (a_1b_2 - a_2b_1)\mathbf{z} \tag{A.7}$$

All symbols are as previously defined. Some useful properties of the cross product are:

$$\begin{aligned} &\mathbf{V}\times\mathbf{V} = \mathbf{0} && \mathbf{U}\times(\mathbf{V}+\mathbf{W}) = (\mathbf{U}\times\mathbf{V}) + (\mathbf{U}\times\mathbf{W}) \\ &n(\mathbf{V}\times\mathbf{W}) = n\mathbf{V}\times\mathbf{W} && \|\mathbf{V}\times\mathbf{W}\|^2 = \|\mathbf{V}\|^2\|\mathbf{W}\|^2 - (\mathbf{V}\bullet\mathbf{W})^2 \\ &\mathbf{V}\times\mathbf{W} = -\mathbf{W}\times\mathbf{V} && \|\mathbf{V}\times\mathbf{W}\| = \|\mathbf{V}\|\cdot\|\mathbf{W}\|\sin\theta \end{aligned} \tag{A.8}$$

The rule of "thumb" applied to the cross product is the *Right Hand Rule*. Extend the fingers of your right hand along the vector **V**. Curl your fingers the shortest angular distance to the vector **W**. Your thumb will extend in the direction of the cross product.

A.2.8. Triple Product. Two vector triple product rules are:

$$\mathbf{U}\bullet(\mathbf{V}\times\mathbf{W}) = (\mathbf{U}\times\mathbf{V})\bullet\mathbf{W} \tag{A.9}$$

and

$$\mathbf{U}\times(\mathbf{V}\times\mathbf{W}) = (\mathbf{U}\bullet\mathbf{W})\mathbf{V} - (\mathbf{U}\bullet\mathbf{V})\mathbf{W} \tag{A.10}$$

Equation A.9 returns the magnitude of one vector projected onto the cross product of two other vectors. The triple cross product (A.10) we found useful in calculating parameters relative to two moving objects.

A.2.9. Normal of a Plane. The normal of a flat plane is a vector of unit length which is orthogonal to a plane. For example, the **z** axis in Figure A.1 is orthogonal to the **xy**-plane. If **z** was of unit length, it would be "normal" to the plane. The normal, **N**, of

any plane in which two vectors, **V** and **W**, lie is given by the cross product divided by the magnitude of the cross product:

$$\mathbf{N} = \frac{(\mathbf{V} \times \mathbf{W})}{\|\mathbf{V} \times \mathbf{W}\|} \tag{A.11}$$

where **N**, **V**, and **W** are all vectors.

A.2.10. Equation of a Plane. The equation of a flat plane defined by two vectors:

$$(\mathbf{V} - \mathbf{W}) \bullet \mathbf{N} = 0 \tag{A.12}$$

where **N** is the normal as defined by Equation A.11.

A.2.11. Angle Between Planes. The angle between two flat planes is defined by their normals:

$$\cos\theta = \frac{|\mathbf{N}_1 - \mathbf{N}_2|}{\|\mathbf{N}_1\| \cdot \|\mathbf{N}_2\|} \qquad 0 \le \theta \le \pi/2 \tag{A.13}$$

where θ is the angle between normals.

A.2.12. Distance Between a Point and a Plane. The shortest distance between a flat plane defined by $ax + by + cz = d$ and a point not lying in that plane defined by $P_1=(x_1, y_1, z_1)$ is:

$$dist = \frac{|ax_1 + by_1 + cz_1 - d|}{\sqrt{a^2 + b^2 + c^2}} \tag{A.14}$$

where *dist* is the *distance* from the point to the plane.

A.3. Air Flight Vector Spaces.

Several sets of axes interact with each other during air combat maneuvering. Airspace is defined on the points of a compass. Airframes, or fuselages, have a nose, tail, and wings. The attitude of an aircraft, roll, pitch, and yaw, also have three-space sets of axes. The

following paragraphs clarify define relationships between coordinates and coordinate frames used in our simulation.

A.3.1. Airspace Axes. The airspace in which simulations occur can be defined in several ways. First, there is latitude, longitude, and altitude in units of degrees (north or south), degrees (east or west), and either feet or meters, respectively. This form most directly represents the earth coordinate system. Second, there is simply north, east, and up, with the distinctions of south and west arbitrarily dismissed. This second form also has units of degrees, degrees, and feet. The directions of south and west are defined as negative north and negative east, respectively. Lastly, there is a rectangular coordinate frame arbitrarily defined as a right-handed coordinate frame with all axes in a common unit of measure. This coordinate frame defines the location of a platform in three-space without the need to calculate distance from degree measurements.

We used the rectangular coordinate representation since units are common to all axes and switching between units of measure only requires a simple conversion utility. To relate back to a world coordinate system, we arbitrarily defined east as the x-axis, north as the y-axis, and altitude as the z-axis. The set of coordinate axes defining a coordinate grid system is sometimes called an earth grid coordinate (EGC) frame.

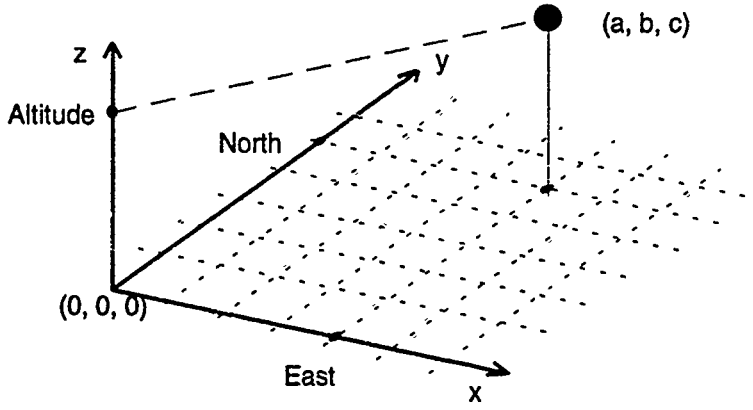


Figure A.2. Earth Grid Coordinate Frame.

A.3.2. Aircraft Axes. Each airframe can be defined using three orthogonal axes: nose, wing, and apex. The nose axis, also known as the "gun bore line" extends coaxially with the aircraft fuselage. The wing axis extends outward along the left wing, perpendicular to the fuselage. The apex axis extends upward, along the rudder and perpendicular to both nose and wing axes. These axes remain fixed in relation to the airframe regardless of airframe attitude. The set of coordinate axes defining an airframe is called the aircraft body coordinate frame. I used the abbreviation ABC frame to refer to this coordinate system.

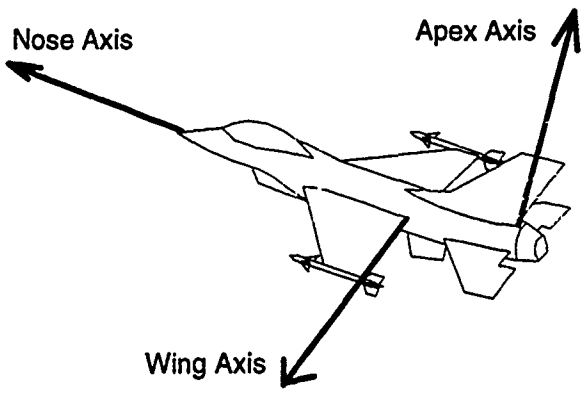


Figure A.3. Aircraft Body Coordinate Axes.

A.3.3. Attitude Axes. The pilot controls roll, pitch, and yaw of the aircraft. In level flight, roll is the angle between apex and z vectors. In oblique flight, roll is the radial angle between the nose-apex plane and nose-z plane. Pitch is the angle between the nose vector and the horizontal x-y plane. Yaw, also called side-slip, is the angle between the nose vector and direction of travel when in level flight. In oblique turns, yaw is the angle between the nose-apex plane and velocity-apex plane. Another attitude parameter is called angle of attack, or AOA. The AOA is the angle between the nose-wing plane and the velocity-wing plane.

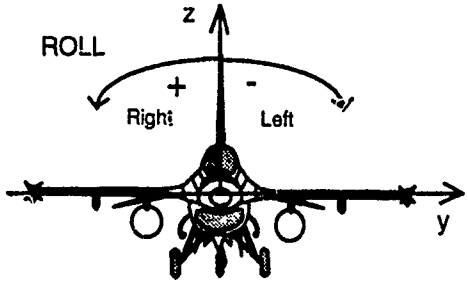


Figure A.4. Roll Attitude Parameter.

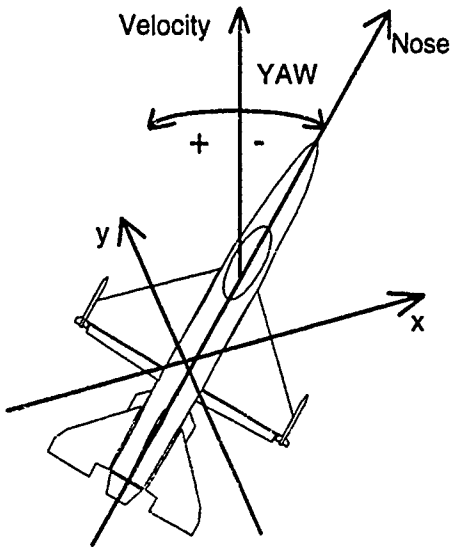


Figure A.6. Yaw Attitude Parameter.

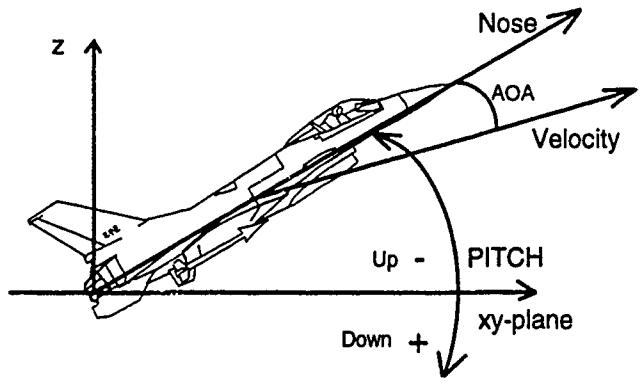


Figure A.5. Pitch and Angle of Attack Attitude Parameters.

A.3.4. Orientation Space. We distinguish aircraft orientation and aircraft attitude using different reference frames. Aircraft attitude is with reference to the pilot in the cockpit. Aircraft orientation is with reference to an observer on the ground. The distinction is necessary since combinations of roll, pitch and yaw produce different effects. For example, increasing pitch on an aircraft in level flight will increase altitude, but increasing pitch on an aircraft in inverted flight will decrease altitude. As a more complex example, consider how changes in yaw affect an aircraft in 90 degrees of roll. To a pilot, yaw still has the effect of moving the aircraft left or right. To an earth-bound observer, the aircraft would climb or dive. As a more succinct example, consider 180 degrees of yaw. Is the aircraft flying backwards, or westwards?

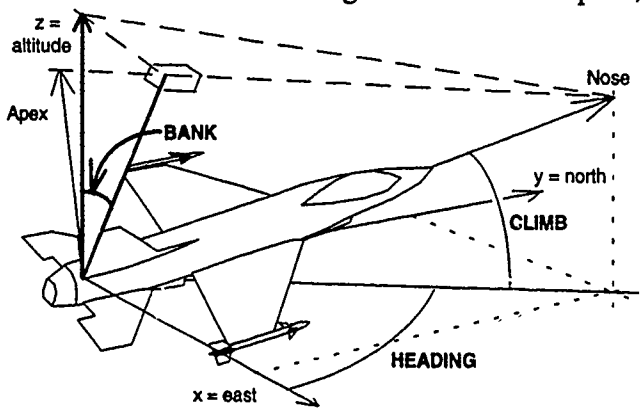


Figure A.7. Orientation Parameters.

To separate pilot and ground reference, we defined an orientation vector composed of a bank angle, climb angle, and heading angle. The three angles are analogous to roll, pitch, and yaw, but serve to distinguish the pilot reference from the ground reference.

A.3.5. Velocity Space. Important parameters in velocity space include the velocity vector, speed, direction and heading. The velocity vector is the magnitude and direction the aircraft is moving. Speed and direction are derived from velocity. Heading is the projection of the velocity vector into the horizontal x-y plane. Relative velocity parameters are discussed in the section entitled Air Combat Vector Spaces. To simplify some calculations, we defined the nose and velocity vectors to be identical.

A.3.6. Acceleration Space. Acceleration parameters affect changes in the velocity vector. Important parameters include thrust, lift, and radial-g. Thrust is the force produced by the engine to propel the aircraft forward along a heading. Forward thrust changes the speed component of the velocity vector. Lateral thrust causes drift. Vertical thrust changes the altitude. Lift depends on wing surface area and engine thrust. To maintain level flight, lift must at least counteract the effects of gravity. In a loop, the lift vector rotates through 360 degrees, so lift and gravity do not always oppose each other.

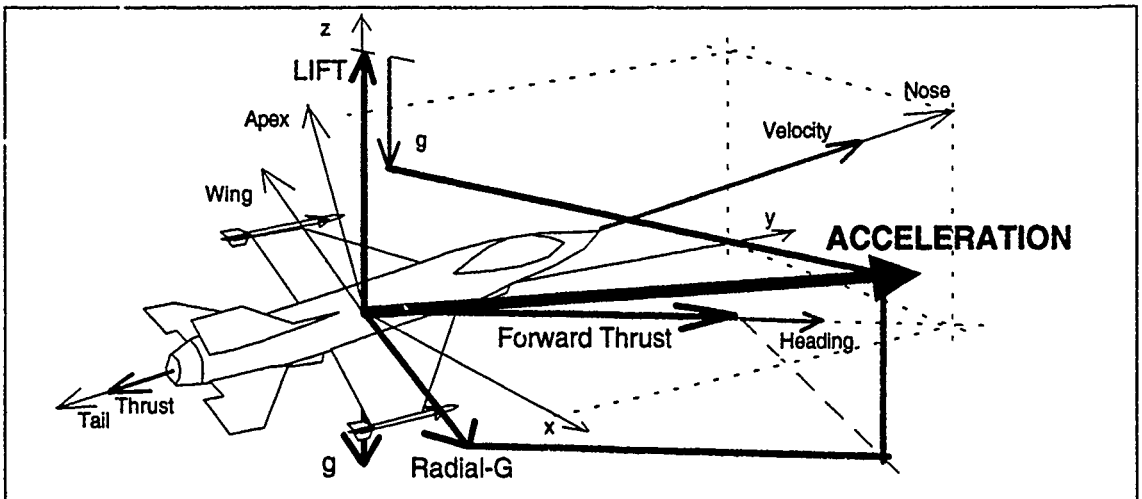


Figure A.8. Velocity and Acceleration Parameters.

Radial-G is the amount of force used to turn the aircraft left or right in the horizontal x-y plane. Acceleration is the vector sum of thrust, lift, gravity and radial-g.

A.4. Air Combat Vector Spaces.

Air combat is conducted between aircraft. Aircraft have a relation to the surrounding airspace in terms of hemispheres and quarter spheres. Opponents must orient airframes to claim the best advantage. Failing that, pilots must not concede the worst disadvantage.

A.4.1. Approach Hemispheres. Combat pilots refer to aircraft in slang terms. Aircraft not only have noses and tails, but left and right beams, and hot and cold sides. The nose is the front of the aircraft while the tail is the back. The beam is off to one side, either left or right. The hot side is the top while the cold side is the bottom of the aircraft.

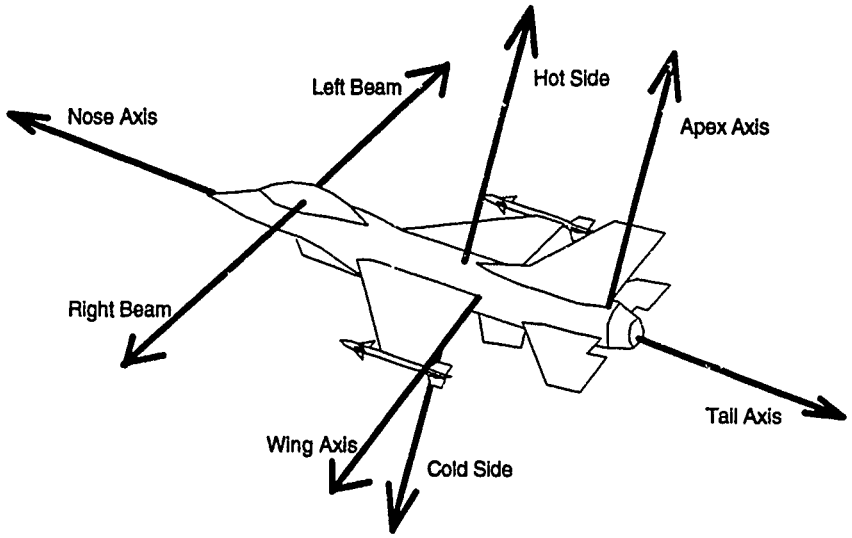


Figure A.9. Approach Hemispheres.

Six hemispheres define the surrounding airspace. The forward hemisphere is centered on the nose vector and extends spherically for 90 degrees off the nose vector. The rear hemisphere is similar, but is centered on the tail vector. Four other hemispheres,

left-side, right-side, hot-side, and cold-side, are centered on the left-beam, right-beam, apex, and negative apex, respectively.

A.4.2. Approach Quarter-Spheres. Six quarter-spheres also define the surrounding airspace, but provide more accuracy for targeting. The forward quarter is actually a cone projected along the nose vector. The nose vector runs down the center of the cone. The sides of the cone make an angle of 45 degrees with the nose vector. Similarly, the rear quarter is centered on the tail vector and beam quarters are centered on beam vectors. For completeness, I've defined an overhead quarter centered on the apex axis and an underside quarter centered on the negative apex axis.

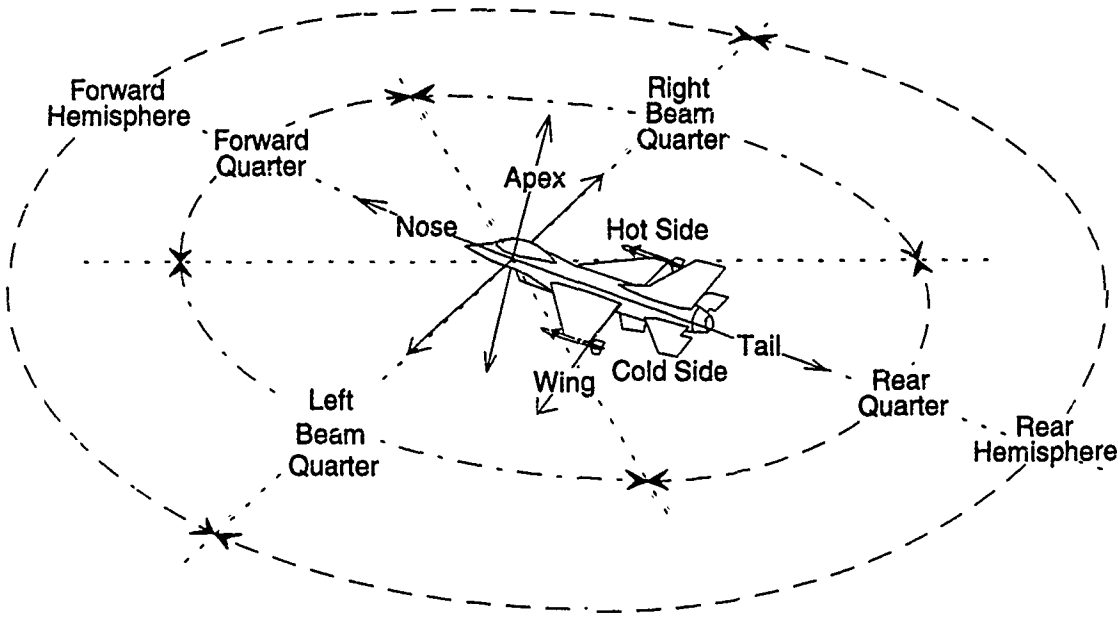


Figure A.10. Approach Quarter-Spheres.

The airspace can also be defined in terms of quadrants. Eight quadrants correspond to the quadrants in rectangular coordinate frames. Quadrants are defined in Appendix B and finer cones are defined under the section entitled Targeting Parameters later in this appendix.

A.4.3. Maneuvering Parameters. Important parameters in maneuvering space include distance, angular and rate measurements. Perhaps the most critical measure is the line of sight (LOS) between two aircraft. The LOS is the vector difference between two points in three-space. Most maneuvering parameters refer to the LOS either directly or indirectly.

Three distance measurements help calculate maneuvering parameters. Range is the two dimensional distance between aircraft locations as measured in the horizontal xy-plane. The magnitude of the LOS indicates three dimensional distance. Separation is the lateral range between aircraft.

Angular measures include lead (or lag) angle, angle off the nose (AON), angle off the tail (AOT), target aspect angle (TAA), and track crossing angle (TCA). Lead (or lag) angle is the angular difference between the velocity vector and LOS. The AON is the angle between the LOS and the target's nose vector. The AOT is the angle between the LOS and the target's tail vector. The TAA is the smaller of AON and AOT, so target aspect angle has a range of 0 to 90 degrees. The TCA is the angle between velocity vectors.

Rate parameters include closing velocity, crossing velocity, and LOS rate. Closing velocity indicates whether the distance between aircraft is increasing or decreasing. It is calculated as the vector difference between velocity vectors. Crossing velocity indicates how fast a target moves across an aircraft's path. It is calculated as the closing velocity times the sine of the lead angle. The LOS rate is the rate of change in LOS and is estimated by the crossing velocity.

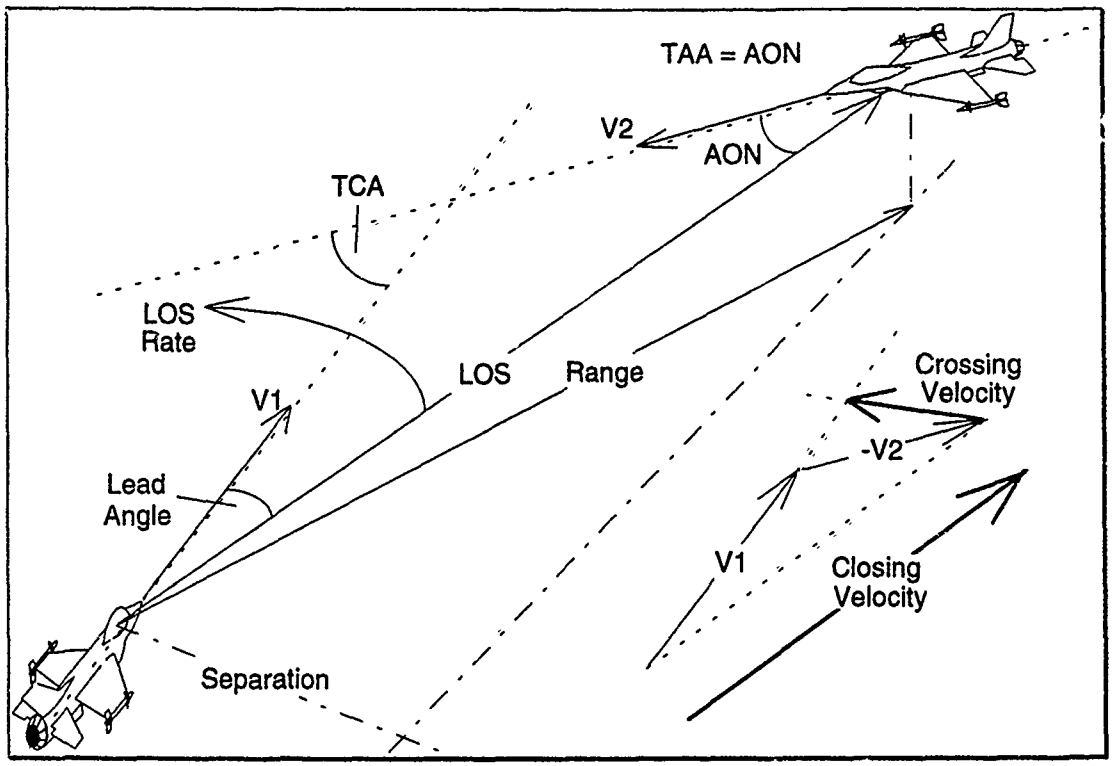


Figure A.11. Maneuvering Parameters.

A.4.4. Targeting Parameters. Two firing envelopes describe targeting: guns and missile envelopes. The guns envelope contains two effective firing areas. The tracking area allows the attacker to hold the target within its gunsights for an extended period of time. The snapshot area gives the attac' ' brief glimpse of the target within its gunsight. Minimum and maximum ranges depend relative velocity and maneuvering.

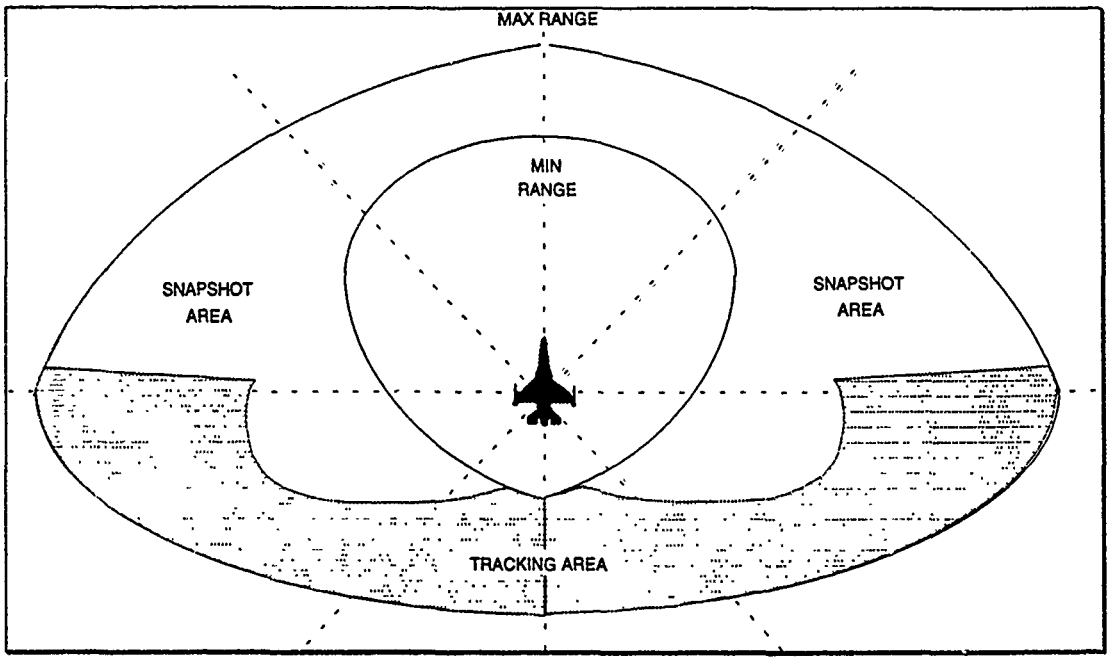


Figure A.12. Targeting Parameters. (Reconstructed from Shaw1985:24)

The missile envelope contains areas of high and low kill probability as well as areas where only "look-up" shots are effective. A high kill probability means the missile is likely to find the target. A low kill probability means a missile is likely to pass the target and detonate on the far side. A look-up shot reduces ground clutter so missiles do not lose track of the target.

Under maneuvering, missile envelopes change dramatically. If closing speed is high, then both maximum and minimum firing ranges increase. This means a missile can be fired early. If closing speed is low (or negative), then maximum and minimum ranges decrease and missile firing should wait. If crossing speed is high, then the missile field of view shifts to compensate. Under a lead angle with high crossing speed, missile field of view shifts toward the target. But with a lag angle, missile field of view shifts away from the target.

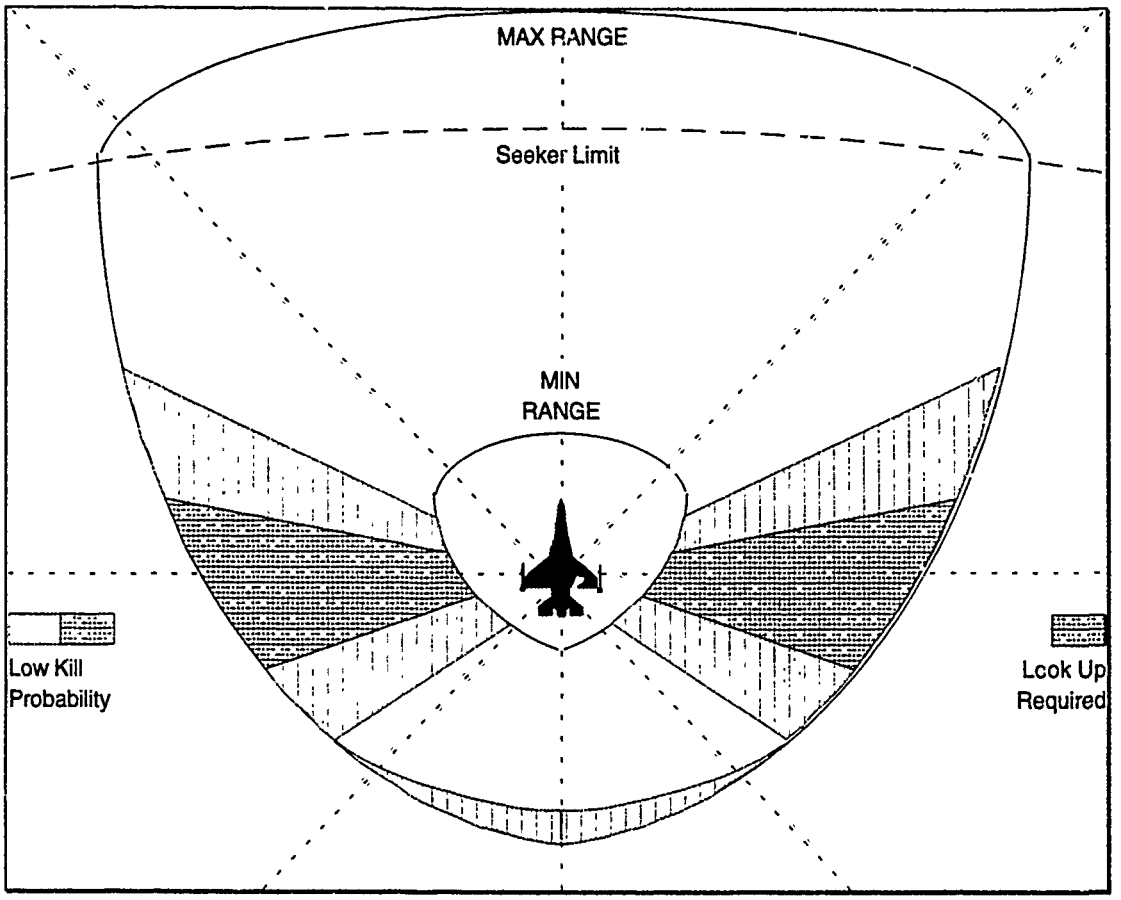


Figure A.13. Targeting Parameters. (Reconstructed from Shaw, 1985:47)

A.5. Summary.

Basic mathematics concepts are presented here in an attempt to understand aerial combat maneuvering and introduce key terms. These terms define basic parameters and provide the beginning of a data dictionary for our simulator. Apperdix B contains a detailed parameter set and data dictionary.

Appendix B. Key Parameters and Data Dictionary

B.1. Introduction

This appendix is a dictionary of key parameters measured, monitored, or otherwise used during air combat maneuvering. Most of the parameters listed here are taken from the book, *Fighter Combat, Tactics and Maneuvering* (Shaw, 1985). Parameters were implemented in the PDPC as indicated in the "defined as" column in each table.

B.2. Background.

In air combat maneuvering there are several different kinds of parameters. Low-level detail parameters include aircraft specifications, such as maximum speed and altitude. These low-level parameters can be static or dynamic. By static I mean the value of the parameter should not change during the life of the platform. By dynamic, I mean the pilot controls a parameter's value within a specific range. Some parameters can be associated with others. These associations are pointed out where necessary. This appendix distinguishes parameters and their types.

B.3. Organization.

This appendix lists and describes parameters that were used in implementing the PDPC air combat simulator. Parameters are not all inclusive since the intent was not to research air combat maneuvering, but to reveal methods for cooperative decision-making

in autonomous agents. These parameters provide insight into decision-making processes undertaken by pilots during air combat.

Key parameters that were implemented are explained in the "defined as" column. Parameters may be defined as one of several types. A global constant indicates the parameter is visible from any phase, module, or function. A local constant is only visible within a specific module. A class slot means the parameter is held in a slot of an object definition. A fact slot means the parameter appears as a field in a fact or template. This also typically means the information is hard coded in one or more rules. The phrase "not defined" means the parameter was not implemented in the PDPC simulation, but parameters that were not implemented are listed for completeness.

B.4. Key Parameter Tables.

B.4.1. Static Parameters. Static parameters are very-low-level details that either do not change or do not change significantly to affect decision making. These parameters indicate maximums or minimums for given platforms. For example, the type of aircraft remains the same regardless of its condition. Other parameters, such as maximum turning radius, were assumed constant for the purpose of decision-making.

Aircraft performance parameters describe limitations of aircraft performance. For the PDPC simulator, these parameters are constant. Approach parameters itemize terms defined in Appendix A.

B.4.2. Decreasing Parameters. Decreasing parameters are low-level details that decrease over the time period of a mission. For example, aircraft decrease in weight as a mission progresses and fuel is used. Weight contributes to key dynamic parameters.

B.4.3. Dynamic Parameters. Dynamic parameters are low-level details controlled by a pilot within some range of operation. Each parameter indicates an aspect

of the current state of an aircraft. Some parameters are defined as part of (p/o) a class slot. In these cases several parameters compose a multifield slot.

Table B.1. Aircraft Performance Parameters.

PARAMETER	DESCRIPTION	DEFINED AS
Aircraft Type	Type of aircraft e.g. F-15	class slot "type-of"
Max-Roll	Highest degree of roll	continuous range
Max-Roll-Rate	Defines maximum change in roll	phase constant
Max-Pitch	Highest degree of pitch	continuous range
Max-Pitch-Rate	Defines maximum change in pitch	phase constant
Max-Yaw	Highest degree of yaw	continuous range
Max-Yaw-Rate	Defines tightest "flat" turn	phase constant
Min-Turn-Radius	Defines tightest turn	not defined
Max-Angle-of-Attack	Max nose-up from direction vector	not defined
Max-Speed	Fastest in level flight	global constant
Max-Altitude	Highest in level flight	global constant
Cornering-Speed	Critical-Velocity @ Max-Aircraft-G	not defined
Max-Aircraft-G	Structural limitation.	not defined
Max-Pilot-G	Limitation of the pilot	not defined
Gravity-G	Force of gravity, 9.8 m/s^2	global constant

Table B.2. Approach Parameters.

PARAMETER	DESCRIPTION	DEFINED AS
Forward-Hemisphere	Centered on the nose axis	+/- (π/2)
Rear-Hemisphere	Centered on, but opposite the nose axis	+/- (π/2)
Hot-Side	Centered on the apex axis	+/- (π/2)
Cold-Side	Centered on, but opposite the apex axis	+/- (π/2)
Left-Side	Centered on the wing axis	+/- (π/2)
Right-Side	Centered on, but opposite the wing axis	+/- (π/2)
Forward-Quarter	Centered on the nose vector	+/- (π/4)
Rear-Quarter	Centered on the negative nose vector	+/- (π/4)
Overhead-Quarter	Centered on the apex axis	+/- (π/4)
Underside-Quarter	Centered on the negative apex axis	+/- (π/4)
Left-Beam-Quarter	Centered on the wing vector	+/- (π/4)
Right-Beam-Quarter	Centered on the negative wing vector	+/- (π/4)
Upper-Left-Quadrant	Bounded by +nose, +wing, +apex axes	not defined
Upper-Right-Quadrant	Bounded by +nose, -wing, +apex axes	not defined
Lower-Left-Quadrant	Bounded by +nose, +wing, -apex axes	not defined
Lower-Right-Quadrant	Bounded by +nose, -wing, -apex axes	not defined
Right-Shoulder-Quadrant	Bounded by -nose, +wing, +apex axes	not defined
Left-Shoulder-Quadrant	Bounded by -nose, -wing, +apex axes	not defined
Left-Tail-Quadrant	Bounded by -nose, +wing, -apex axes	not defined
Right-Tail-Quadrant	Bounded by -nose, -wing, -apex axes	not defined

Table B.3. Decreasing Parameters.

PARAMETER	DESCRIPTION	DEFINED AS
Fuel	Amount remaining	class slot "fuel"
Weight	Aircraft weight	class slot "mass"
Endurance	Defines limitations of the pilot	not defined
IR-Missiles	Infra-red guided missiles	not defined
Radar-Missiles	Radar guided missiles	class slot "missile-load"
Laser-Missiles	Laser guided missiles	not defined
Rounds	Bullets	not defined
Gravity-Bombs	Drop bombs	not defined
Laser-Guided-Bombs	Smart bombs	not defined
Cluster-Bombs	Scatter bombs	not defined

Table B.4. Dynamic Parameters.

PARAMETER	DESCRIPTION	DEFINED AS
Altitude	Height or flight level	p/o class slot "location"
Speed	Scalar (e.g. mach, mph, KTAS, KIAS)	function "speed"
Direction	Vector indicating direction of travel (bank, climb, heading angles)	class slot "orientation"
Velocity	Vector indicating speed and direction (dx, dy, dz)	class slot "velocity"
Acceleration	Vector indicating change in velocity	class slot "abc-acceleration"
Location	Position on a grid representing sim-space (x,y,z)	class slot "location"
Position	World coordinates (latitude, longitude, altitude)	not defined
Attitude		
Roll	Angle of apex vector from nose-z plane	p/o class slot "abc-attitude"
Pitch	Angle of nose vector from wing-nose plane	p/o class slot "abc-attitude"
Yaw	Angle of nose vector from nose-apex plane	p/o class slot "abc-attitude"
Angle-of-Attack	Angle of nose vector from velocity-wing-plane	not defined
Forces		
Lift	Acceleration along apex vector	not defined
Gravity-G	Acceleration along negative z vector	global constant
Thrust	Acceleration forward along nose vector	class slot "abc-thrust"
Drag	Acceleration backward along negative nose vector	not defined
Radial-G	Acceleration toward a focal point	not defined
Turn-Rate	Rate of change in velocity vector	not defined
Turn-Radius	Distance from a focal point	not defined
Specific-Energy	See formula below	rule "specific-energy"
Excess-Power	See formula below	rule "excess-power"

B.4.3.1. Specific Energy. Shaw identifies two critical parameters in air combat maneuvering: specific energy and excess power (Shaw, 1985:394). The specific energy indicates maneuver capability and is key to determining if an aircraft should adopt the "energy" or "angles" fighter strategy. Specific energy, E_S, depends on altitude, speed, and gravity as defined in the following equation:

$$E_s \approx Altitude + \frac{Speed^2}{2 \cdot Gravity - G} \tag{B.1}$$

where G is the radial acceleration induced on the aircraft due to maneuvering.

B.4.3.1. Excess Power. The second of Shaw's key parameters, excess power indicates climb performance. Although a fighter may not have an explicit energy advantage, excess power indicates the ability to gain an advantage. Excess power, P_S, depends on velocity, thrust, drag, and aircraft weight:

$$Ps \approx Velocity \frac{Thrust - Drag}{Weight} \tag{B.2}$$

In the PDPC simulation, *drag* is ignored.

B.4.4. Relative Parameters. The relation between two aircraft dictates tactics and maneuvers and play the critical role in firing decisions. Relative parameters define the relationship between two maneuvering opponents.

Three types of envelopes define air combat maneuvering. The visual envelope describes the physical position of one aircraft with respect to another. The maneuvering envelope describes limiting factors so that a pilot can choose between different maneuvers. The firing envelope imposes further limitations, but the choice is whether to fire or not. Parameters were weighed and used differently within different PDPC phase modules.

Table B.5. Visual Envelope.

PARAMETER	DESCRIPTION	DEFINED AS
Line-of-Sight (LOS)	Vector from aircraft to target locations	function "line-of-sight"
LOS-Rate	Angular change in line-of-sight vector	function "LOS-rate"
Angle-Off-Tail	Difference between LOS and target tail vector	function "angle-off-the-tail"
Angle-Off-Nose	Difference between LOS and target nose vector	function "angle-off-the-nose"
Target-Aspect-Angle	Acute angle between LOS and target velocity vector	function "target-aspect-angle"
Target-Velocity-Vector	Estimate of target speed and direction	class slot "velocity"
Target-Attitude	Estimate of target roll, pitch and yaw	class slot "orientation"

Table B.6. Maneuvering Envelope.

PARAMETER	DESCRIPTION	DEFINED AS
Target-Specific-Energy	Estimate of capabilities	rule
Target-Excess-Power	Estimate of potential	rule
Predicted-Location	Estimate of target path	functions; class HISTORY
Maneuver-to-Execute	Result of deliberation	rules
Target-Maneuver	Estimate of target intentions	rules
Closure-Rate	Difference between velocity vectors	function
Track-Crossing-Angle	Angle between velocity vectors	function
Location	Coordinates in x, y, z space	class slot "location"
Separation	Radial distance from nose vector	function

Table B.7. Firing Envelope.

PARAMETER	DESCRIPTION	DEFINED AS
Range	Distance to target in xy-plane	function
Distance	Distance to target along line-of-sight	function
Lookup	Positive nose vector angle with Earth plane	not defined
Lookdown	Negative nose vector angle with Earth plane	not defined
Counterflow	Angle between velocity vectors >= 90 degrees	not defined
Inflow	Angle between velocity vectors < 90 degrees	not defined
Time-of-Flight	Estimate of time for weapon to reach target	not defined

B.4.5. Basic Maneuvers. Basic maneuvers constitute the minimum elements of controlled flight. Simple maneuvering is embedded within rules in each phase module.

Table B.8. Basic Maneuvers.

PARAMETER	DESCRIPTION	DEFINED AS
Level-Upright	All orientation angles equal zero.	rules
Level-Inverted	Level flight at a bank angle (roll) of precisely π or $-\pi$. Other orientation angles equal zero.	rules
Level-Rolled	Level flight at a bank angle between π and $-\pi$, but not zero.	rules
Climb	Positive dz value	rules
Dive	Negative dz value	rules
Roll-Left	Rotate aircraft clockwise about nose axis	rules
Roll-Right	Rotate aircraft counterclockwise about nose axis	rules
Pitch-Up	Rotate aircraft clockwise about wing axis	rules
Pitch-Down	Rotate aircraft counterclockwise about wing axis	rules
Yaw-Left	Rotate aircraft counterclockwise about apex axis	rules
Yaw-Right	Rotate aircraft clockwise about apex axis	rules
Turn-Left	Roll-Left, Pitch-Up, Yaw-Right	rules
Turn-Right	Roll-Right, Pitch-Up, Yaw-Left	rules
Break-Left	Roll-Left-Max, Pitch-Up-Max, Yaw-Left	rules
Break-Right	Roll-Right-Max, Pitch-Up-Max, Yaw-Right	rules
Accelerate	Increase throttle	rules
Decelerate	Decrease throttle	rules
Accelerate-Max	Increase throttle to 125% of maximum positive	rules

B.4.6. Composite Maneuvers. These maneuvers are composites of basic maneuvers performed in a sequence. Composite maneuvers fall into one of five categories: pursuit curves, attack rolls, attack turns, crossing maneuvers, vertical maneuvers, and combination maneuvers. Composite maneuvers are defined in rules or sets of rules. Not all of the maneuvers defined here were implemented, but they are shown here for completeness.

Pursuit curves ascribe a simple curved flight path through three-space in the plane of attack. Typically, the curve is directed toward a target or some point offset from a target. The primary purpose is to close on a target in a way that brings the attacker into the rear quarter of the target. Three types are lead-pursuit, pure-pursuit, and lag-pursuit.

Table B.5. Pursuit Curves.

TYPE	PRE-CONDITIONS	DESCRIPTION	GOALS	DEFENSE
Lead-Pursuit	Fighter is inside target's turn.	Aim nose-vector ahead of target. Turn with decreasing radius (increasing rate). Avoid "blind" turns by using parallel turn plane. Use proportional navigation course.	Increase AOT & closure	Energy Fighter: Extend. Angles Fighter: Turn at Min-Turn-Radius.
Pure-Pursuit	Equal fighter energy.	Aim nose-vector at target.	Maintain AOT & closure	
Lag-Pursuit	Fighter is outside target's turn.	Aim nose-vector behind target Turn with increasing radius (decreasing rate).	Decrease AOT & closure	Reverse with "impunity".

Attack rolls are conducted out of the plane of attack and either maintain an advantage or reverse a disadvantage. For example, lag rolls are used as an alternative to slowing down. Barrel rolls and "yo-yo" maneuvers change a position disadvantage into a position advantage. Five types of attack rolls are lag-roll, displacement-roll, barrel-roll, high-yo-yo, and low-yo-yo.

Attack turns are conducted in the plane of attack, but use the target as a focus to orbit rather than a point to fly toward. Attack turns depend on the response of the target.

Table B.9. Attack Rolls.

TYPE	PRE-CONDITIONS	DESCRIPTION	GOALS	DEFENSE
Lag-Roll	Imminent overshoot, high closure, excess lead pursuit, medium AOT (30~60 degrees)	Increase lag angle. Reduce speed by increasing altitude. Roll to point apex at target. Decrease lag angle.	Lag-Pursuit	Extend dive with same Roll-Angle
Displace-ment-Roll	Close range, low AOT (0~30 degrees), low closure.	Increase lag angle. Reduce speed by slowing down. Move to opposite side of target flight path. Decrease lag angle.	Increase range Reduce AOT	Extend dive with same Roll-Angle
Barrel-Roll	Forward-Hemisphere, lead-pursuit, long range.	Wings level pull-up. Roll to point apex toward target. When passing overhead, dive onto target using lead pursuit.	Rear-Hemisphere, Lag-Pursuit	Extend dive with same Roll-Angle
High-Yo-Yo	Imminent overshoot, low closure (increasing), Pure-Pursuit, medium AOT (30~60 degrees)	Roll to Level-Upright. Climb. Roll to place lift-vector ahead of target for lead-pursuit (GUNS), onto target for pure-pursuit, or behind target for lag-pursuit (MISSILES).	Lead: gun snapshot. Lag: missile min-Range.	Attack-Turn
Low-Yo-Yo	Long range, Hot-Side, Lag-Pursuit, high AOT (60~90 degrees)	Nose-down toward inside of turn. Roll to match turn rates. Climb back to target's altitude.	Decrease AOT, increase closure	Attack-Turn

For example, a nose-to-nose turn means both attacker and target turned in the same direction and will pass again after 180 degrees of turn. A nose-to-tail turn means attacker and target turned in opposite directions with one achieving a rear quarter advantage. Five types of attack turns are lead-turn, nose-to-nose, nose-to-tail, turn-toward, and turn-away.

Table B.10. Attack Turns.

TYPE	PRE-CONDITIONS	DESCRIPTION	GOAL	DEFENSE
Lead-Turn	Forward-Hemisphere with separation of one min-turn-radius. Not Energy-Fighter.	Level-upright until passing (beam quarter) then turn toward target.	Rear-Hemisphere	Attack-Roll
Nose-to-Nose	Angles Fighter. Target Lead-Turn.	Turn away from target until target overshoots. Then turn toward target.	Rear-Quarter	Attack-Roll
Nose-to-Tail	Energy Fighter. Target Lead-Turn.	Turn toward target and prevent target overshoot.	Inside turn, Flat Scissors.	Attack-Roll
Turn-Toward	None	Point nose vector at target.	Nose = LOS	Turn-Away
Turn-Away	None	Point nose vector away from target.	Nose = -LOS	Turn-Toward

Crossing maneuvers describe dynamic and dangerous situations. For example, a flat-scissors maneuver is a repeated sequence of nose-to-nose turns with direction reversals on each pass. The goal is to achieve a rear-quarter posture by staying slow and maneuverable. The danger is that so is the opponent. Whoever can fly slowest without stalling wins. Another crossing maneuver is a downward spiral. Opponents barrel-roll around each other, but the general direction of the maneuver is toward the ground. Whoever pulls out first loses the advantage, but not necessarily the engagement.

Table B.11. Crossing Maneuvers.

TYPE	PRE-CONDITIONS	DESCRIPTION	GOAL	DEFENSE
Counter-flow	Nose-To-Nose	Fighter turn-rate @ equal but opposite target turn rate.	Turn min-turn-radius	Attack-Roll
In-flow	Nose-To-Tail	Fighter turn-rate @ equal target turn rate.	Turn min-turn-radius	Attack-Roll
Flat-Scissors	Angles Fighter, Nose-To-Nose, same maneuver plane.	Nose-to-nose ==> Reversal ==> Lead-Turn ==> Repeat	Rear-Quarter	vertical maneuver or extend run with turn toward.
Rolling-Scissors	Energy Fighter, Nose-To-Nose, high track-crossing-angle.	Barrel-Roll ==> Repeat	Rear-Quarter, but hold energy advantage	
Spiral-Down	Angles Fighter, Nose-To-Tail	Vertical Rolling-Scissors	Rear-Quarter	Extend-Dive
Spiral-Up	Energy Fighter, Nose-To-Tail	Vertical Rolling-Scissors	Rear-Quarter	Extend-climb

Vertical maneuvers describe pursuit curves and attack turns in the z-plane. The primary goal of vertical maneuvers is to gain either energy or speed. Climb maneuvers increase specific energy, but decrease excess power. Dive maneuvers increase both specific energy and excess power, but can only be conducted for short time periods. Both types require periods of straight flight and may cause a position disadvantage.

Combination maneuvers depend heavily on the position of an opponent and typically implies the opponent has an advantage. For example, a reversal indicates the opponent is executing a maneuver that the attacker cannot follow. A jinking maneuver is executed

when the attacker crosses in front of an opponent. All combination maneuvers involve movement away from an opponent rather than toward.

Table B.12. Vertical Maneuvers.

TYPE	PRE-CONDITIONS	DESCRIPTION	GOAL	DEFENSE
Oblique-Climb	Level-Upright, below max-altitude	Increase pitch to optimum climb angle, roll left/right to induce lateral motion, accelerate	Gain energy w/o losing position.	
Zoom-Climb	Oblique-Climb, below max-altitude	Increase pitch to optimum climb angle for altitude, Max-Accelerate	Gain energy.	
Steep-Climb	Zoom-Climb, below max-altitude	Increase pitch to maximum climb angle for altitude, Max-Accelerate	Gain altitude.	
Oblique-Dive	Level-Upright, above Ground-Level	Decrease pitch to optimum dive angle, roll left/right to induce lateral motion, Max-Accelerate	Gain speed w/o losing position	
Zoom-Dive	Oblique-Dive, above Ground-Level	Decrease pitch to optimum dive angle for altitude, Max-Accelerate	Gain speed.	
Steep-Dive	Level-Upright, above Ground-Level	Decrease pitch to maximum dive angle for altitude, Max-Accelerate	Max speed gain.	

Table B.13. Combination Maneuvers.

TYPE	PRE-CONDITIONS	DESCRIPTION	GOAL	DEFENSE
Reversal	Give-Rear-Quarter, tight turn, high bank angle (45~90 degree)	Roll to negative bank angle and go the opposite direction.	Increase Give-AOT and distance	
Jink	Give-Snapshot is imminent (nose overshoot)	Pitch-down or pitch-up, then return after overshoot.	Avoid snapshot	
Oblique-Turn	Level-Upright	Turn-Toward or Turn-Away with altitude change (vertical & lateral combined)	Lag-Pursuit, Rear-Quarter	
Extend-Run	Target not in rear quarter.	Turn-Toward, Max-Accelerate, increase Lag-Angle	Increase distance.	

B.5.7. Targeting Goals. Targeting goals indicate incremental gains over an adversary. I established this sequence as a way of gauging advantage or disadvantage. Parameters are not explicitly defined, but appear as parts of rules and rule sets.

Table B.14. Advantage Targeting Goals (In Precedence Order).

GOAL	DESCRIPTION	DEFINED AS
Missile-Lock	Target acquired	p/o rule
Tracking-Shot	Target crossing speed near zero	p/o rule
Snap-Shot	Target crossing speed near maximum	p/o rule
Lethal-Range	Min-Range < Range < Max-Range	p/o rule
Overshoot-Tail	Fighter overshoots target tail vector	p/o rule
Rear-Quarter	Fighter is within 45 degrees of target tail vector	p/o rule
Rear-Hemisphere	Fighter is within 90 degrees of target tail vector	p/o rule
Underside-Quarter	Fighter is within 45 degrees of target -apex axis	p/o rule
Cold-Side	Fighter is within 90 degrees of target -apex axis	p/o rule
Beam-Quarter	Fighter is within 45 degrees of target wing axis	p/o rule
Beam-Hemisphere	Fighter is within 90 degrees of target wing axis	p/o rule
Hot-Side	Fighter is within 90 degrees of target apex axis	p/o rule
Overhead-Quarter	Fighter is within 45 degrees of target apex axis	p/o rule
Forward-Hemisphere	Fighter is within 90 degrees of target nose axis	p/o rule
Forward-Quarter	Fighter is within 45 degrees of target nose axis	p/o rule

Table B.15. Disadvantage Targeting Goals (In Precedence Order).

GOAL	DESCRIPTION	DEFINED AS
Give-Forward-Quarter	Target is within 45 degrees of Fighter nose axis	p/o rule
Give-Forward-Hemisphere	Target is within 90 degrees of Fighter nose axis	p/o rule
Give-Overhead-Quarter	Target is within 45 degrees of Fighter apex axis	p/o rule
Give-Hot-Side	Target is within 90 degrees of Fighter apex axis	p/o rule
Give-Beam-Hemisphere	Target is within 90 degrees of Fighter wing axis	p/o rule
Give-Beam-Quarter	Target is within 45 degrees of Fighter wing axis	p/o rule
Give-Cold-Side	Target is within 90 degrees of Fighter -apex axis	p/o rule
Give-Underside-Quarter	Target is within 45 degrees of Fighter -apex axis	p/o rule
Give-Rear-Hemisphere	Target is within 90 degrees of Fighter tail vector	p/o rule
Give-Rear-Quarter	Target is within 45 degrees of Fighter tail vector	p/o rule
Give-Overshoot-Tail	Target overshoots Fighter tail vector	p/o rule
Give-Lethal-Range	Target-Min-Range < Range < Target-Max-Range	p/o rule
Give-Snap-Shot	Fighter crossing speed near maximum	p/o rule
Give-Tracking-Shot	Fighter crossing speed near zero	p/o rule
Give-Missile-Lock	Fighter acquired	p/o rule

B.5.8. Tactical Decisions. These parameters form the basis of the PDPC phase control architecture. Tactical decisions are based on decision and maneuver rules. As an architecture, entering one set of rules implies information and events occurred to require a decision. For example, firing on a target implies the target was hostile, entered protected

airspace, constituted a threat, was pursued, engaged in combat, and came into the firing envelope of the attacker. All of the information leading to a firing decision is moot once the missile is launched. So, each tactical decision requirement was implemented as a phase module with transitions between modules implying a decision was made.

Table B.16. Tactical Decisions.

PARAMETER	DESCRIPTION	DEFINED AS
New-Mission	Mission, leader, and followers assigned	Phase module
Launch	Take off from airfield	Phase module
Cruise	Move from point to point	Phase module
Search	Locate other aircraft	Phase module
Identify	Determine target relationship	Phase module
Chase	Pursue to border	Phase module
Pursuit	Pursue beyond border	Phase module
Engage	Seek position advantage through maneuvering	Phase module
Acquire	Lock on target	Phase module
Fire	Release weapon	Phase module
Analyze	Measure result of weapon use	Phase module
Breakoff	Break away due to counter-maneuver	Phase module
Avoid	Break away due to counter-attack	Phase module
Disengage	Break away to address new target	Phase module
Evade	Avoid confrontation because of mission	Phase module
Retreat	Break away to address new mission	Phase module
Refuel	Break away to address low fuel alarm	Phase module
Recall	Return to base	Phase module
Landing	Execute a landing at a base	Phase module

B.5.9. Strategic Decisions. Tables B.17 through B.26 define strategy options for PDPC agents. Mission assignment, planning and tactical coordination identify specific strategy options. Tables B.20 through B26 are derived from the Titan report (Titan, 1986) and provide base definitions for decisions modules.

Table B.17. Mission Assignment.

PARAMETER	DESCRIPTION	DEFINED AS
CAP-station	Fly oval pattern along "threat" axis.	class slot value
Patrol	Fly linear pattern along defined boundaries.	class slot value
Assault	Fly direct to target. Egress out the back side.	not defined
Sweep	Feign assault on collateral target, then attack primary from flank.	not defined

Table B.18. Planning.

PARAMETER	DESCRIPTION	DEFINED AS
Phase-Plan	Defines default sequence through phase modules.	class
Route-Plan	Defines waypoint sequence for pre-planned routes	class
Intercept-Plan	Defines waypoint sequence for un-planned routes.	class
Bingo-Plan	Defines waypoint sequence to refueling stations.	class
Ingress-Plan	Scenario file for assault	not defined
Egress-Plan	Scenario file for assault	not defined

Table B.19. Tactical Coordination.

PARAMETER	DESCRIPTION	DEFINED AS
Solo	Single fighter attacks target	class slot value
Fighting-Wing	Lead fighter attacks target. Partner follows.	class slot value
Dual-Attack	Lead fighter attacks target. Partner stays out.	class slot value
Loose-Deuce	First fighter attacks, then second. Alternate.	class slot value

Table B.20. Pre-Engagement Strategy Decisions.

PARAMETER	DESCRIPTION	DEFINED AS
Radar-Check	Select radar mode	rules
Commitment	Declares when to engage	rules
Correlate-Target	Compare target data with partners	rules & slot values
Guess-Intent	Decide who a target will attack	rules & slot values
Estimate-Threat	Attach a value to a target	rules & slot values
Call-For-Support	Get help before engaging	rules
Assign-Priority	Attach numeric value to a target	rules & slot values

Table B.21. Engagement Strategy Decisions.

PARAMETER	DESCRIPTION	DEFINED AS
Determine-Approach	Select a direction to approach a target	rules & slot value
Supporting-Role	Select tactical coordination method	rules & slot value
Formation	Select an attack formation	rules & slot value
Energy-Advantage	Advantage due to speed and/or altitude.	not defined
Energy-Fighter	Declares energy advantage.	rule
Angles-Fighter	Declares energy disadvantage	rule
One-on-One	Use one versus one tactics	not defined
One-on-Two	Use one versus two tactics	not defined
Two-on-One	Use two versus one tactics	not defined
Two-on-Two	Use two versus two tactics	not defined

Table B.22. Intercept Geometry.

PARAMETER	DESCRIPTION	DEFINED AS
Entry-Point	Define the first waypoint in a plan	plan slot value
Intermediate-Point	Define waypoints between entry-point and intercept-point	plan slot value
Intercept-Point	Define expected rendezvous	plan slot value
Free-Fire-Area	Define points to avoid	object instance
Ingress-Altitude	Define the altitude to use during a plan	rules & slot value
Ingress-Airspeed	Define the altitude to use during a plan	rules & slot value
Egress-Altitude	Define the altitude to use during a plan	rules & slot value
Egress-Airspeed	Define the altitude to use during a plan	rules & slot value
Create-Plan	Make an instance of a plan	rule
Update-Plan	Update the information in a plan	method

Table B.23. Evasion Tactics.

PARAMETER	DESCRIPTION	DEFINED AS
Evade-Ground-Site	Use path around or under opposing capability, e.g. radar-site, SAM-site, etc.	Intercept-Trajectory rules
Evade-Aircraft	Conduct mission without detection	Evade phase
Evade-Missile	Hard maneuvering to avoid being shot down	Avoid phase

Table B.24. Weapons Employment.

PARAMETER	DESCRIPTION	DEFINED AS
Discriminate-Targets	Distinguish individual targets in a group	rule
Assign-Target	Assign a target to attack	rule
Assign-Goal	If not a target, then determine what partner should do	rules
Modify-Orbit	Keep partner aircraft near an engagement	rule
Engage-Criteria	Decide whether to engage a target	rules
Abort-Criteria	Decide whether not to engage a target	rules
Activate-Weapon	Arm a missile	rule
Deactivate-Weapon	Disarm a missile	rule
Reactivate-Weapon	Rearm a disarmed missile	rule
Lock-Radar	Decide to commit radar to fire control	rules
Launch-Envelope	Determine the bearing and range to launch a missile	rule

Table B.25. Counter-Action Strategy.

PARAMETER	DESCRIPTION	DEFINED AS
Counter-Jamming	Determine how to counter communications jamming	rule
Counter-Maneuver	Determine what maneuver to use against an opponent's maneuver	not defined
Missile-Defense	Determine what maneuver to use against a missile attack	Avoid phase
Throttle-Setting	Determine a throttle setting before intercept point	not defined

Table B.26. Post Engagement Strategy.

PARAMETER	DESCRIPTION	DEFINED AS
Next-Missile	Decide whether another missile should be armed	not-defined
Disengage-Criteria	Decide whether to re-attack or disengage	engage criteria
Post-Missile-Launch	Decide maneuver to execute after launching a missile	not-defined
Post-Missile-Miss	Decide maneuver to execute if the missile misses the target	not-defined
Disengage	Determine how to disengage	Disengage phase
Retreat	Determine direction to disengage	Retreat phase

B.5.10. Information Gathering. These parameters indicate raw data collected during each mission. Specific parameters may. or may not, be used in an engagement. The major categories of information gathering include intelligence, communications, counter-measures, and environment data.

Intelligence information characterizes threats and expected threats. In real combat, data could be pre-loaded. In PDPC simulations, only some intelligence data is pre-loaded. Known fixed threat locations, such as opponent bases and surface-to-air missile sites, are established as instances of the STATION class. Unknown threats are defined when they come within radar range.

Communications systems include radio and radar capabilities. Counter-measure systems include chaff and flare systems and rules to counter radio jamming signals. Environment information describes weather and terrain conditions.

Table B.27. Intelligence Information.

PARAMETER	DESCRIPTION	DEFINED AS
Threat-Environment	Defines specific expected threat, e.g. SAMs	class slot "type-of"
Threat-Type	Guns, rear quarter missile, or all aspect missile	class slot "type-of"
Threat-Location	Define opposing target in x, y, z space	not defined
Threat-Source	Launch platform	not defined
Target-Thrust-to-Weight	Estimate of capability beyond own capability	not defined
Target-Name	Identify opposing targets	class slot "target-name"
Target-Status	Detected, assigned, destroyed, etc.	class slot "target-status"
Target-Victim	Assume where the target intends to go	class slot "target-victim"
Victim-Location	Define the location in x, y, z space	class slot "victim-location"
Target-Threat	Estimate of potential damage threat poses	class slot "target-threat"
Target-Priority	Importance of target. Highest priority first.	class slot "target-priority"

Table B.28. Communications.

PARAMETER	DESCRIPTION	DEFINED AS
Radio	Pass message traffic between aircraft	class slots "radio-status", "radio-mode", and "radio-channel" and template "message"
Radar	Provide target location and velocity data	class slots "radar-status", and "radar-mode"
RWR-Warning	Incoming missile warning signal	rule

Table B.29. Counter Measures.

PARAMETER	DESCRIPTION	DEFINED AS
Chaff	Missile defense system	not defined
Flare	Missile defense system	not defined
Jamming	Electronic counter measure	rule and class slot

Table B.30. Environment Information.

PARAMETER	DESCRIPTION	DEFINED AS
Weather	Specific conditions within 10 mile radius	not defined
Wind-Speed	Average speed	fact "wind-velocity"
Wind-Direction	Average direction	fact "wind-velocity"
Wind-Velocity	Average speed and direction vector	fact "wind-velocity"
Wind-Gusts	random velocity fluctuations	not defined
Rain	Describes precipitation variable	not defined
Cloud-Cover	Describes environment variable	not defined
Cloud-Ceiling	Describes environment variable	not defined
Cloud-Tops	Describes environment variable	not defined
Visibility	Distance that can be seen below cloud cloud-ceiling	not defined
Terrain	Specific conditions within 10 mile radius	not defined
Ground-Level	Describes minimum flight level	not defined
Ground-Variation	Describes average variation in ground level	not defined
Ground-Excursions	Describes mountains and valleys	not defined

B.5.11. Mission Purpose. A statement of mission purpose identifies primary operating modes. For example, Air Superiority implies aircraft search for opposing targets over the leading edge of the battlefield and all opposing aircraft are targeted and attacked. Defensive doctrine implies opponent aircraft are chased, but not necessarily destroyed. In a defensive role, distance from a defended area must be weighed against distance to a potential target.

B.5.12. Mission Planning Mission plans provide specifics in terms of primary mission, targets, units involved and the most current information on intelligence, communications and other aspects of the threat environment. The PDPC simulator does not provide a specific mission plan, but does provide mission planning information.

Table B.31. Mission Purpose.

PARAMETER	DESCRIPTION	DEFINED AS
Air-Superiority	Destroy airborne opposing aircraft over battlefield	slot value: "superiority"
Defensive-Counterair	Destroy airborne opposing aircraft over airfield	slot value: "defense"
Fighter-Sweep	Destroy all opposing aircraft	not defined
Offensive-Counterair	Destroy grounded opposing aircraft	slot value: "offense"
Area-Defense	Patrol an own-side area and search for targets	slot value: "defense"
Air-Interdiction	Attack strategic targets, air and non-air	not defined
Strike-Escort	Protect own-side airborne elements	not defined
Close-Air-Support	Protect own-side ground elements	not defined
Reconnaissance	Search for strategic ground targets	not defined
Support	Electronic warfare, AWACs, communications, etc.	not defined
Transport	Cargo airlift	not defined

Table B.32. Mission Planning Parameters.

PARAMETER	DESCRIPTION	DEFINED AS
Mission-Purpose	Defines primary mission.	class slot: "mission"
Target	Defines primary and secondary targets.	class slot: "target-name"
Participants	Defines friendly, opposing, and team-mates	class
Intelligence	Describes expected types of air and ground threats	not defined
Communications	Defines call signs of local and global elements	AWACs
Weapons-Load	Defines available firepower options	class slot: "missile-load"
Terrain	Helps in route planning	Route-Plan
Leading-Edge-of-Battlefield	Describes when to expect opposing threats	fact: "border"
Weather	Describes difficulty in route or combat maneuvers	not-defined
Route-Plan	Primary ingress and egress routes	class slot: "plan"
Alternate-Routes	In case of emergency	not defined

B.5.13. Operations Orders. These requirements provide general guidance in single scenarios. In the PDPC simulator, operations orders are not specifically defined except for a rule in New-Mission phase that makes assignments depending on mission purpose.

Table B.33. Operations Orders.

PARAMETER	DESCRIPTION	DEFINED AS
Assign-Mission	Determine a mission assignment depending on a given mission.	New-Mission phase rule
Execute-Mission	Defined by mission, assignment, plan, and circumstances.	Pre-Engagement module, "commitment" checks
Abort-Mission	Defined by mission, assignment, plan, and circumstances.	Weapons-Employment module, "engage-criteria" checks

B.5.14. Directives. These requirements provide very high level guidance valid in many several scenarios. Its purpose is to allow coordination of adjacent missions (at least at the top level) and implement a chain-of-command. In the PDPC simulator, chain-of-command is pre-loaded and rules to either execute or abort a plan depend on the current phase. An overall "recall" rule aborts all plans after a preset time limit

Table B.34. Directives.

PARAMETER	DESCRIPTION	DEFINED AS
Execute-Plan	Choose an action from a pre-set plan	Depends on current phase and current plan.
Abort-Plan	Change plans from one currently executing to a default plan	Depends on current phase and current plan.

B.5.15. Doctrine. AFM 1-1, *Basic Aerospace Doctrine,* lists eleven principles of war that should be achieved as a prelude to combat.

Table B.35. Doctrine.

PARAMETER	DESCRIPTION	DEFINED AS
Offensive	Act rather than react	Pursuit versus evade phase rules
Security	Protect own-side elements from surprise	Counter-Action module, "cooperative-rules" check
Surprise	Act at time and place unexpected	not defined
Objective	Defines intentions to act upon	Pre-Engagement module "target-data" checks
Timing-and-Tempo	Optimizes use of friendly forces	Engage phase rules
Mass-Vs-Economy	Balanced to address primary objectives	not defined
Maneuver	Movement in relation to opposing forces	Engagement-Strategy module, "strategy" checks
Unity-of-Command	Dictates coordinated effort	class slots: "leader-of" and "follower-of"
Cohesion	Coordinated rather than disjointed	not defined
Logistics	Defines adequate supplies	not defined

B.6. Summary.

This appendix is a dictionary of key terms used in the PDPC simulator. Air combat maneuvering parameters and concepts were encoded as constructs. A construct may have the form of a rule, set of rules, class, class slot, class slot value, or simply as a fact for general consumption. Not all parameters and concepts were actually coded, but all are listed in this appendix for completeness.

Appendix C. Pilot Situation Awareness

C.1. Introduction.

This appendix lists a series of questions measuring a pilot's awareness of combat situations. They are taken from an idea given to me by Amburn who conducted a study measuring pilot situation assessment (Amburn, 1993). His study attempted to identify the merits of using virtual reality goggles versus computer monitor displays. I used the idea of a questionnaire to elaborate the decision process used by pilots. The questionnaire below itemizes potential questions used by pilots entering a combat situation. Questions and their answers are derived from information and explanations contained in the Shaw text and Titan report. Each question is phrased to keep answers as simple as possible.

C.2. Questionnaire.

1. What am I doing here? (What mode am I in?)
Answers:

New-Mission	Intercept	Fire	Evade
Launch	Chase	Analyze	Retreat
Cruise	Pursue	Breakoff	Refuel
Search	Engage	Avoid	Recall
Identify	Acquire	Disengage	Landing

2. Where's the bogey?
Answers: Over there. - Causes a tactical decision sequence:
identify, intercept, pursue, engage, acquire, fire, analyze.
No where. - Causes continuation of search or cruise phase.

3. Is the target really a bogey?
Answer: Yes. I've identified the target as hostile enemy.
Maybe. I haven't identified the target, yet.
Maybe not. I've identified the target as neutral or civilian.
No. I've identified the target as friendly (same side).

4. Has he seen me?
Answers: Yes. The bogey is maneuvering hard right toward me.
Maybe. The bogey is maneuvering hard, but not directly toward me.
Maybe not. The bogey is maneuvering easy.
No. The bogey is still flying straight and level.

5. Is the bogey's radar on?
Answers: Yes. The bogey is actively searching.
No. The bogey is not searching.

6. Is the bogey's radar locked onto me?
Answers: Yes. I'm in trouble.
No. I'm still undetected.

7. Has he fired a weapon at me?
Answers: Yes. Take evasive action.
No. He's maneuvering into position.

8. How much fuel do I have left?
Answers: Plenty. Enough for prolonged engagement.
Enough. More than half remaining with less than half the mission done.
Not Enough. Less than half remaining with more than half the mission done.
Panic. Only enough to return to base.

9. What weapons do I have left?
Answers: All of my missiles and all of my guns.
Missiles (all, >half, half, <half, none), and/or Guns (all, >half, half, <half, none)
None of my missiles and none of my bullets for guns.

10. Am I injured to the point it affects my ability?
Answers: Yes. I cannot properly control the aircraft.
Maybe. I have partial control of the aircraft, but my is reduced.
Maybe not. I can control the aircraft, but my endurance has weakened.
No. I am fully functional.

11. Can I shoot him down now?
Answers: Yes. The bogey is within the range and firing envelope of my weapon.
Weapon choice is one of:
All-aspect missile {long, medium, or short range}
Rear quarter missile {medium or short range}
Guns {short range}
Maybe. I could simply "fire for effect".
No. I'll have to maneuver to get into firing position.
No. I don't have any weapons left.

12. Can I maneuver into a better firing position?
Answers: Yes. Execute a maneuver toward the bogey for best effect.
No. Execute a maneuver around the bogey until a better opportunity exists.
No. I don't have enough fuel left. Go into Withdraw or Escape mode

13. Is the bogey in my front or rear hemisphere?
Answers: Front hemisphere.
Rear hemisphere.

14. Is the bogey moving toward or away from me.?
Answers: Toward.
Away.
Neither. The bogey is moving across my field of view.

15. How many bogeys are there?
Answers: One.
Two.
Three.
Four.
Gaggle.

16. How many fighters are we?
Answers: One.
Two.
Three.
Four.
Gaggle.

17. Measure the following parameters.

	Us	Them	Advantage (+/-)
Altitude	feet	feet	
Speed	knots	knots	
Direction	(x,y,z)	(x,y,z)	
Thrust	lbs	lbs	
Weight	lbs	lbs	
Drag	lbs	lbs	

18. Who has the higher "specific energy"?
Answers: Us. Be the "energy" fighter.
Them. Be the "angles" fighter.

19. Who has the higher "excess power"?
Answers: Us. Be the "energy" fighter.
Them. Accelerate and try to be the "energy" fighter.

20. What is the best strategy to employ?
Answers: Energy fight.
Angles fight.

21. What is the best tactic to employ?
Answers: Opening.
Mid-game.
Transition game.
End game.

22. What is the best attitude to take?
Answers: Panic. I'm hit and going down.
Hard. I'm under missile attack and at a disadvantage.
Aggressive. I'm under attack, but not at a disadvantage.
Firm. I'm not under attack, but I must maintain an advantage.
Easy. I'm not under attack, but I must act on the first opportunity.

23. What is the best maneuver to make?
Answers depend on strategy, tactic, and attitude employed.

24. How are the bogeys employing toward me?
Answers: Fighting wing.
Double Attack.
Loose Deuce.

25. How can I best employ toward the bogeys?
Answers: Fighting wing.
Double Attack.
Loose Deuce.

26. What doctrine should we employ?
Answers: Fighting Wing.
Double Attack.
Loose Deuce.
Task Assignment.
Solo.

27. Where is the ground? (How high are we flying?)
Answers: We're at high altitude.
We're at medium altitude.
We're at low altitude.
We're at tree-top level.
We're at sea-level.
There's a mountain (or other obstacle) over there.

28. Where are there any SAM sites to avoid?
Answers: Over there.
No where.

29. Is the SAM site's radar on?
Answers: Yes. Look for launching missiles.
No. Look for SAM sites.

30. Is the SAM sites radar locked?
Answers: Yes. Prepare for evasive maneuvers.
No. Maneuver away from the site.

31. Has a SAM been fired at me?
Answers: Yes. Take evasive action.
No. Keep watching out for it.

32. How's the weather?
Answers: Visibility unlimited.
Low-level clouds.
Mid-level clouds (maybe I can hide there).
High-level clouds.
Thunder clouds (at all levels).

33. Where did everybody (friendly) go?
Answers: The planned rendevous.
Over there (an unplanned rendevous).
Retreating.
No where (they're all dead).

34. What's the best way out of here?
Answers: Combination of the following:
{Break Left, Break Right, Stay Straight}
{Dive, Climb, Stay Level}
{Extend, Stay at Speed}
{Stay with Plane, Eject}

C.3. Summary.

This appendix lists a series of questions measuring a pilot's awareness of combat situations. The intent was to elaborate pilot situation awareness in terms both the reader and a computer programmer could understand.

Appendix D. Class Slot Definitions

This appendix defines classes, subclasses and object slots for each class in the PDPC simulator.

Table D.1. Player Class.

SLOT	TYPE	DEFINITION
name-of	slot	Symbol uniquely identifying player
side	slot	One of "friendly", "enemy", or "neutral" to help players sort out who is who.

Table D.2. Platform Class.

SLOT	TYPE	DEFINITION
phase	slot	The module of rules to activate for the player
state	slot	One of "moveable", "move", or "moved" helps the simulator decide which player to consider next. STATE also ensures each player receives one move per cycle.
location	multislot	(x,y,z) coordinates of a player's location in EGC space.
velocity	multislot	(dx,dy,dz) player speed and direction in EGC space.
orientation	multislot	(b,c,h) player bank, climb and heading angles in EGC space
goal	slot	Symbol identifying where a player should fly.
goal-location	multislot	x, y, z coordinates of the goal.
desired-direction	slot	One of "toward" or "away" indicating whether a player wants to approach or escape another player.
abc-velocity	multislot	Indentifies aircraft velocity in the ABC frame. (U, V, W) represents velocity in the nose, wing, and apex directions.
abc-acceleration	multislot	(dU,dV,dW) the rate of change in velocity relative to the ABC frame.
abc-thrust	multislot	Change in acceleration in the ABC frame. (F, G, H) represents thrust in the forward (nose), lateral (wing), and vertical (apex) directions.
abc-attitude	multislot	(R,S,T) roll, pitch, and yaw in the ABC frame
attitude-rate	multislot	(dR,dS,dT) change in roll, pitch, and yaw in the ABC frame.
attitude-moment	multislot	(mR,mS,mT) describes torque applied to aircraft body.
throttle	slot	A number to indicate whether the simulator should increase or decrease a player's speed.
mass	slot	weight of the aircraft
on-the-ground	slot	boolean value indicating whether the aircraft is actually on the ground (TRUE) or in the air (FALSE). If on-the-ground is true, gravity effects are cancelled.
fuel	slot	Describes how much fuel is left

Table D.3. Aircraft Class.

SLOT	TYPE	DEFINITION
role	slot	One of "leader", "follower", "solo", or "support" describes how same side players interact.
leader-of	slot	For a leader, this slot contains the players on the team. For other roles, this slot is "none".
follower-of	slot	Indicates who is the leader.
mission	slot	One of "defense", "offense", "superiority", "escort", or "testing".
assignment	slot	A statement of "patrol", "CAP-station", or "flight-test" dictates the initial plan to be used.
plan	slot	A plan name containing the waypoints to fly to and the order in which to fly them.
condition	slot	Indicator of the pilot's status: "alive", injured", "dead".
missile-load	slot	The number of missiles a player owns
type-of	slot	One of "fighter", "bomber", or "tanker" for aircraft.
number-of	slot	The number of aircraft within a target grouping.
tactical-coordination	slot	The way a follower aircraft behaves with respect to a leader. One of: "patrol", "fighting-wing", "double-attack", or "loose-deuce".
formation	slot	Decsribes the formation a leader wants follwers to use. One of "spread", "echelon", "trail", or "none".
approach	slot	Describes how an aircraft will approach a target. One of "head-on", "side", "rear", "above", or "below".
bearing-to-defensive-target	slot	Bearing from a player to any ground site.
distance-to-defensive-target	slot	Distance between player and any ground site.
kill-radius-of-defensive-target	slot	Estimate of how far to go around the target.
maneuver	slot	Identifies the maneuver selected to execute.
target-name	multislot	A list of targets within radar range.
target-status	multislot	For each target, one of "detected", "correlated", "assigned", "destroyed", "retreating", or "escaped".
target-victim	multislot	An assumption who target aircraft intend to attack
victim-location	multislot	x, y, z coordinates of assumed victims
target-threat	multislot	Threat level of targets. One of "high", "medium", or "low" for each target.
target-priority	multislot	Numerical assessment of threat.
radar-status	slot	One of "on", "off", or "jammed"
radar-mode	slot	One of "search", "fire-control", or "locked"
IR-source-status	slot	One of "on", or "off"
IR-detector-status	slot	One of "on", "off", or "jammed"
radio-status	slot	One of "on", "off", or "jammed"
radio-channel	slot	The current channel number
radio-mode	slot	One of "receive" or "transmit"
radio-jammer-status	slot	One of "on", or "off"
radio-jammer-detector	slot	One of "on", or "off"
radar-jammer-status	slot	One of "on", or "off"
radar-jammer-detector	slot	One of "on", or "off"
jamming-source-name	slot	A symbol indicating the source of jamming.
jamming-source-location	slot	x, y, z coordinates of the jammer

Table D.4. Missile Class.

SLOT	TYPE	DEFINITION
missile-name	slot	A symbol identifying the missile
type-of	slot	Currently, only "sidewinder" available
number-of	slot	A number corresponding to the aircraft missile load
belongs-to	slot	The source of the missile
status	slot	Indicates the current state of a given missile. One of "inactive", "active", "armed", "ready", "fire", "tracking", "missed", "detonated", or "expired".
envelope	multislot	Describes missile capability to reach a target. Expressed as min-range, max-range, max-angle. Max-angle is the mximum AON for firing.
kill-probability	slot	Defines probability that a missile will hit the target.
reliability	slot	Defines the probability that the missile will ignite and travel toward the target.

Table D.5. Station Class.

SLOT	TYPE	DEFINITION
type-of	multislot	Describes the type of fixed station. One of "base", "tanker", or "SAM-site".
location	multislot	x,y,z coordinates of fixed station
orientation	slot	bank, climb, and heading angles used for aircraft approaches, such as for landing.

Table D.6. History Class.

SLOT	TYPE	DEFINITION
belongs-to	multislot	Describes the type of fixed station. One of "base", "tanker", or "SAM-site".
locations	multislot	Last five x,y,z coordinates of given station

Glossary of Acronyms

ABC	aircraft body coordinate (frame)
AFIT	Air Force Institute of Technology
AOA	angle of attack
AON	angle off the nose
AOT	angle off the tail
ARPA	Advanced Research Projects Agency
CAP	combat air patrol
CGF	Computer Generated Forces
CLIPS	C Language Integrated Production System
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling and Simulation Office
EGC	earth grid coordinate (frame)
IST	Institute for Simulation and Training
LOS	line of sight
PA	Pilot's Associate
PDPC	Pilot Decision Phases in CLIPS
SAF	Semi-Automated Forces
TAA	target aspect angle
TCA	track crossing angle
TSS	Task Support Subsystem
USAF	United States Air Force

Bibliography

(Amburn, 1993) Amburn, Philip. Instructor of Computer Science, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH. Personal communication. July 1993.

(Banks and Lizza, undated) Banks, Sheila B., and Carl S. Lizza. *Pilot's Associate: A Cutting Edge Knowledge-Based System Application.* Wright Research and Development Center, Wright-Patterson AFB, OH. Undated.

(Barr, Cohen, and Feigenbaum, 1981) Ba[illegible], Avron, Paul R Cohen, and Edward A. Feigenbaum. *The Handbook of Artificial Intelligence, Volume 4.* Addison-Wesley Publishing Company, Inc., 1989.

(CLIPS, 1993) *CLIPS Reference Manual, Volume I, Basic Programming Guide, CLIPS Version 6.0.* Software Technology Branch, Lyndon B. Johnson Space Center, Houston. June 1993.

(DMSO, 1993) Defense Modeling and Simulation Office. *1993 DMSO Survey of Semi-Automated Forces.* Defense Modeling and Simulation Office, July 30, 1993.

(Downes-Martin, 1992) Downes-Martin, Stephen. *Open and Extensible Architecture for Computer Generated Forces,* part of Volume II of the *Strawman Distributed Interactive Simulation Architecture Description Document* (contract No. ADST /WDL/TR-92-003010). Orlando FL: Loral Systems Company, February 1992.

(Dyer and Gunsch, 1993) Dyer, Douglas E., and Gregg H. Gunsch. "Enlarging the Universal Plan for Air Combat Adversaries," in *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation.* 255-261. Orlando FL: Institute for Simulation and Training, University of Central Florida, March 1993.

(Funk and Lind, 1992) Funk, Kenneth H. and Judith H. Lind. "Agent Based Pilot-Vehicle Interfaces: Concept and Prototype," in *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 6*: 1309-1321. November/December 1992.

(Giarratanno and Riley, 1989) Giarratanno, Joseph, and Gary Riley. "Expert Systems, Principles and Programming." PWS-KENT Publishing Company, 1989.

(Hluck, 1993) Hluck, George S. *Developing Realistic Behaviors in Adversarial Agents for Air Combat Simulation.* MS Thesis AFIT/GCE/ENG/93D-06. Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1993.

(IST, 1992a) Institute for Simulation and Training. *Distributed Interactive Simulation, Operational Concept (Draft).* Orlando FL: Institute for Simulation and Training, University of Central Florida. February 1992.

(IST, 1992b) Institute for Simulation and Training. *Distributed Interactive Simulation, Standards Development Guidance Document (Draft).* Orlando FL: Institute for Simulation and Training, University of Central Florida. February 1992.

(Kilpatrick, 1992) Kilpatrick, Freeman A., Jr. *An Investigation of Discovery-Based Learning in the Route Planning Domain.* MS Thesis AFIT/GCE/ENG/92D-07. School of Engineering, Air Force Institute of echnology (AU), Wright-Patterson AFB OH, December 1992.

(Levi , 1992) Levi, Keith R. and others. "An Explanation-Based-Learning Approach to Knowledge Compilation, A Pilot's Associate Application," *IEEE Expert, Vol. 7, No. 3, 44-51* (June 1992).

(Lun and MacLeod, 1992) Lun, Vernon, and Ian M. Macleod. "Strategies for Real-Time Dialogue and Interaction in Multiagent Systems," in *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 4,* 671-679 July/August 1992.

(McMahon, 1990) McMahon, Daniel C. "A Neural Network Trained to Select Aircraft Maneuvers During Air Combat: A Comparison of Network and Rule Based Performance," *IEEE INNS International Joint Conference on Neural Networks, Vol. 1.* I107-I112. 1990.

(Rich and Knight, 1991) Rich, Elaine and Kevin Knight. *Artificial Intelligence, Second Edition.* McGraw-Hill, Inc. 1991.

(Rumbaugh, 1991) Rumbaugh, James, and others. *Object-Oriented Modeling and Design.* Prentice-Hall, Inc. 1991.

(Sarma and Raju, 1991) Sarma, V. V. S. and Savithri Raju. "Multisensor Data Fusion and Decision Support for Airborne Target Identification," in *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 5*: 1224-1230. September/October 1991.

(Shaw, 1985) Shaw, Robert L. *Fighter Combat, Tactics and Maneuvering.* Naval Institute Press; 1985.

(Titan, 1986) *Pilot's Decision Definition and Analysis, Volume I - Information Requirements Assessment.* Technical Report AFWAL-TR-86-0002, Vol. I. Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base OH. August 1986.

(Whelan, 1992) Whelan, Michael Anthony. *An Intelligent Real-Time System Architecture Implemented in Ada.* MS Thesis AFIT/GCE/ENG/92D-12. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis

4. TITLE AND SUBTITLE

DEVELOPING REALISTIC COOPERATIVE BEHAVIORS FOR AUTONOMOUS AGENTS IN AIR COMBAT SIMULATION

5. FUNDING NUMBERS

6. AUTHOR(S)

Dean P. Hipwell, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCE/ENG/93D-05

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This thesis investigated, developed and implemented cooperative decision-making behaviors in an air combat simulation by using a knowledge-based system. Knowledge-based systems were well suited for this task because of built-in features such as inference engines and rule-based constructs. This thesis addresses the specific problem of generating autonomous forces for inclusion in the Advanced Research Projects Agency Distributed Interactive Simulation program. Existing autonomous forces implementations lacked flexibility, realistic behaviors, real-time planning and other features. The simulation system in this thesis addresses the problem of realistic behavior by modeling pilot decision processes rather than aircraft platforms. The system is based on phased control of a blackboard architecture. Modular knowledge bases partition rules to process decision data. Cooperative behaviors are based on a leader-follower relationship. Agents share the workload in assessing threats. Leaders make the initial decision, but followers react independently if necessary. The simulator described in this thesis provides an architecture and design for modeling combat pilot decision processes. The system was developed using the C Language Integrated Production System Object Oriented Language.

14. SUBJECT TERMS

ARTIFICIAL INTELLIGENCE; AIR COMBAT; SIMULATION; KNOWLEDGE-BASED; CLIPS; COOPERATIVE BEHAVIORS

15. NUMBER OF PAGES

171

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL