Engineering Emergent behavior in Computer Generated Forces (CGF) using Multi-Agent System

Sumant Mukherjee[1] and Saroj Kaushik[2]

[1]Institute of Systems Studies and Analyses, DRDO, Delhi
Email: sumantiitd@yahoo.com
and
[2]Dept. of Computer Science & Engineering, IIT, Delhi
Email: saroj@cse.iitd.ernet.in

Abstract

Over the previous decade, there has been a significant amount of work done on the development of intelligent, Computer Generated Forces (CGF) capable of combat behavior within synthetic battlefields. One of the pioneering efforts is credited to Ilachinski who developed ISAAC (Irreducible Semi-Autonomous Adaptive Combat) in 1997. The central thesis of ISAAC is that land combat can be thought of as a complex adaptive system - combat forces are composed of large numbers of nonlinearly interacting parts and are organized in a command and control hierarchy. ISAAC is intended to be used as a conceptual playground in which to explore high-level emergent behaviors arising from various low-level interaction rules. However in such a system it is very hard to engineer the emergent behavior. This paper attempts to develop an approach that tries to significantly reduce laborious trial and error experimentation usually required to engineer the emergent behavior. A prototype of synthetic battlefield environment and a generic Multi-Agent based CGF has been built to demonstrate the approach. A toolkit named JACK has been used for implementation.

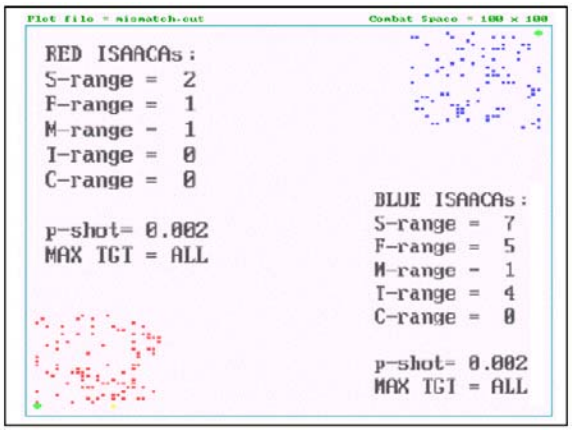
Key Words: Computer Generated Forces, Multi-Agent System, Emergent behavior, BDI Model.

1. Introduction

Combat simulation systems are used by Armed forces around the world as an important tool to train its personnel and to devise new doctrines and strategies. In a realistic training exercise the cost of involving human players in the battle space is very expensive. Computer Generated Forces (CGF) comes to the rescue here. CGF have been used in training as well as tactics development. CGF can potentially replace humans in Combat simulation systems to reduce cost of training exercises.

Irreducible Semi-Autonomous Adaptive Combat (ISAAC) was an attempt to model land combat using agent-based simulation techniques. Completed in 1997, the central thesis of Ilachinski's model is that land combat can be thought of as a complex adaptive system - combat forces are composed of large numbers of nonlinearly interacting parts and are organized in a command and control hierarchy [1].

ISAAC represents combat units as abstract Blue or Red entities (ISAAC Agent) fighting on a two dimensional battlefield [1] (see Fig 1). Each agent evaluates the space around it and based on its internal goals reacts to the changes in the environment. The agents are limited in what they can detect and thus can only respond to changes in the local environment.



```
Plot file = mismatch.out                    Combat Space = 100 x 100

RED ISAACAs:
S-range =  2
F-range =  1
M-range -  1
I-range =  0
C-range =  0

p-shot= 0.002
MAX TGT = ALL
                                  BLUE ISAACAs:
                                  S-range =  7
                                  F-range =  5
                                  M-range -  1
                                  I-range =  4
                                  C-range =  0

                                  p-shot= 0.002
                                  MAX TGT = ALL
```

Figure 1: Snapshot of ISAAC Run

ISAAC is based on mobile cellular automata model. ISAACs intended use is as a conceptual playground in which to explore high-level emergent behaviors arising from various low-level interaction rules. Emergent behavior suggests that relationship between individual behaviors, environment and overall behavior is not completely understood. This makes it hard to engineer agents to fulfill specific tasks. One has to laboriously go thru trial and error to engineer an agent.

In this paper we have attempted to develop an approach that tries to significantly reduce trial and error experimentation usually followed to engineer the emergent behavior. The approach taken is to develop a standalone prototype of Combat simulation system (Combat server) in which human players and/or CGF client software can play against each other. The proposed approach is then demonstrated by engineering a generic Multi-Agent based CGF (CGF-AGENT) that runs on top of Combat server.

2. Combat Server

Combat server enables Human players and/or CGF software to control combat force, which play against each other within the simulation environment [2]. Such an environment should support generation of combat forces (Blue and Red) with units having configurable number of personal & ammunition, ranges for sensing and firing weapon. It should also support virtual terrain and generate casualty due to combat. At interface level it should give graphical user interface for human players to interact and allow CGF to plug in so as to play the combat exercise.
Combat server developed has four major components (Fig. 2)

- Graphical Interface
- Game Controller
- Combat Process
- Database

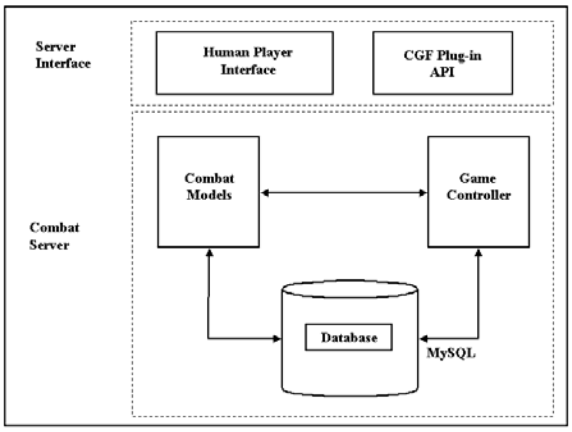


Figure 2: Combat Server Architecture

2.1 Graphical Interface

Graphical Interface displays the map of the virtual battlefield and provides a set of Dialog boxes for Human players to interact with the Combat server.
Map (Fig. 3) depicts the current state of virtual battlefield for human players to observe and control the proceedings of the battle. Map area is a two-dimensional grid in which each square of the grid may be occupied by Blue or Red units. An influence map is overlaid on top of this map to depict areas where each player wields control [3].

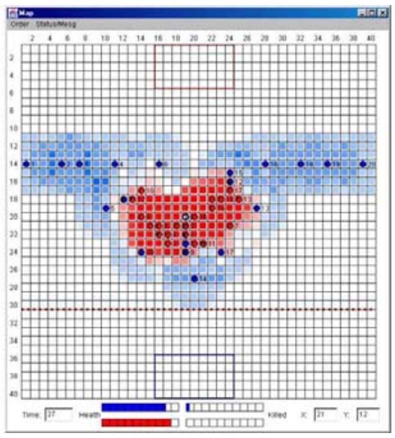


Figure 3: Snapshot of Map Interface of Combat Server

Combat server provides a set of GUIs to human players to issue Move and Fire order to units, monitor status of their combat units and keep track of messages given by Combat server. For e.g. using Move / Fire GUI (Fig. 4) it is possible for a unit to move in any of eight neighboring square. Similarly in single cycle, units can fire at most one detected enemy targets.

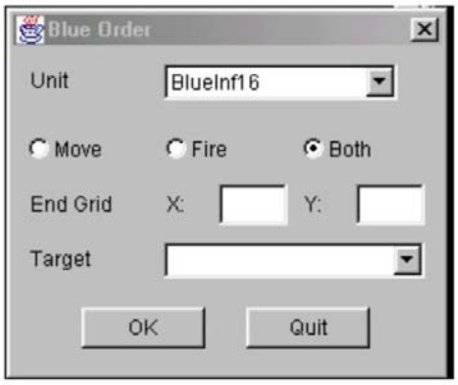


Figure 4: Move and Fire Order GUI

2.2 Game controller

Game controller controls the internal execution of Combat server and synchronizes the clients with the game clock. Thus it carries out two important functions viz.,

- Game Time Management
- Process Synchronization

Execution loop of Game Controller is as follows -
1. Increment Game time.

2. Gives “Go Ahead” to Combat process for current cycle and wait till Combat process done.
3. Update Map Display.
4. Gives “Go Ahead” to Blue client to input their orders for current cycle.
5. Gives “Go Ahead” to Red client to input their orders for current cycle.
6. Go to Step 1

2.3 Combat process and Database

Combat process is the heart of Combat server and implements the rules of combat. It performs movement, firing, casualty generation and enemy unit detection.

Database provides persistence storage to Combat **server**. It also provides a channel thru which Game Controller, Combat process and Graphics interface communicate with each other. Database is hosted on MySQL and primarily consists of tables holding information regarding current game for both sides. Before Game starts, initialization of database needs to be done by generating -

- Terrain
- Units and Resources

3. CGF-AGENT

CGF-AGENT is a generic Multi-Agent based CGF that has been developed to plug-in to Combat server. Overall goal of developing the system was to create an experimental framework in which CGF model can be flexibly configured.

3.1 Specifications

CGF-AGENT should be able to display the capability of interacting with highly dynamic combat environment and taking autonomous decision. They should have limited perceptions restricting their information about the environment.

CGF-AGENT should be able to choose current goal by judging all possible goals at each game step without being instructed. To demonstrate truly realistic behavior in dynamic battlefield the tactics must not be hard coded [4].

In developing **CGF-AGENT** the domain knowledge of Combat tactics and behavior has been adopted from ISAAC although there are important differences (Refer Section 5)

3.2 Design

A Combat force in usually organized in hierarchy of Command and Control with each level of hierarchy having a specific set of tasks to fulfill. CGF-AGENT is also organized in three levels of hierarchy (Fig. 5).

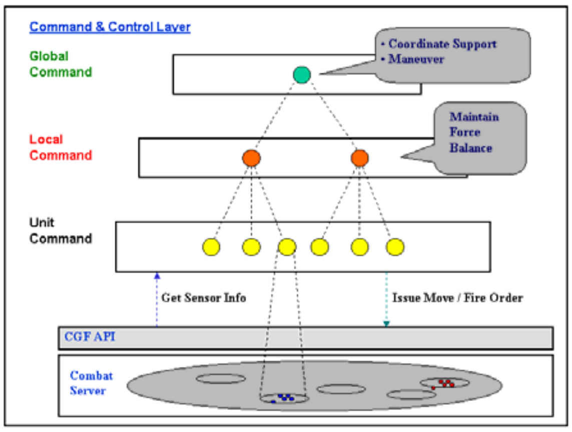


Figure 5: CGF-AGENT Design

Unit Commander (UC) :-
The lowest hierarchy level has UC. UC controls the physical unit existing within the Combat server and gives out orders to the server. UC can also get (limited to sensor range) information from Combat server about number of friendly and enemy units around it and their health.

Local commanders (LC) :-
LC is middle level Command and Control layer that coordinates lower level UCs under its command. It compiles information communicated from UCs and uses it to maintain force balance in its command area and massing of units by adjusting their movement vectors

Global commanders (GC) :-
GC is top level Command and Control layer that coordinates actions of the LCs. GC uses the information communicated to them from all LCs to issue force movement vectors to LCs and coordinates mutual help between LCs.

3.3 Agents Internal Design

CGF-AGENT uses one of the most popular and successful agent models - Belief, Desire and Intention (BDI agents) framework by Rao and Georgeff [5]. BDI Model, based on folk psychology, is highly suited for complex dynamic environments like combat (Fig. 6)

Beliefs represent the agent’s current information about the state of the environment inferred from its sensor. Desires represent a state which agent is trying to achieve. Intentions are the means to achieve the agent’s desires, and are generally implemented as plans (procedures which come with pre-conditions to determine when a plan is applicable). An agent can have a number of plans applicable at given time but only one chosen plan is active at any one time [6].

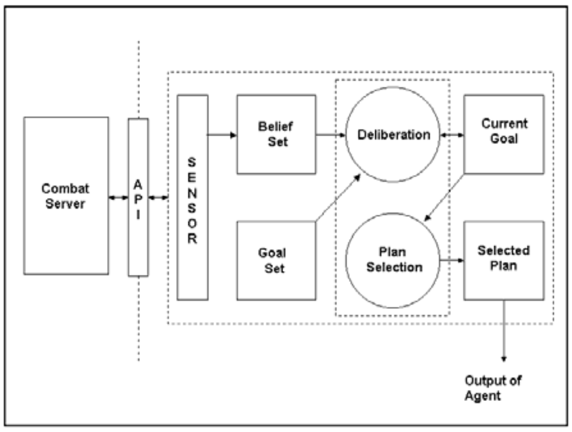


Figure 6: BDI Architecture

Classical planning systems (such as STRIPS) have similar methodology. However in case of situated agents in dynamic environment, a balance must be struck between reactive behavior (responding to changes in the environment) and proactive behavior (persisting with the current plan in order to achieve the desired outcomes). In such systems it is possible that a plan can become inapplicable or fail during its execution because of changes in the environment. The balance is achieved by having a library of precompiled plans and periodically checking for applicability of the plan currently executing. It should be possible to switch between concurrently executing plans or abandon current plan in favor of another.

The BDI framework technically realizes goal-directed behavior, which is the key difference from mainstream computing paradigms. Goals play a central role in rational systems described by BDI theories [7]. Goals are state of the world, which the agent has decided to attempt to achieve. Goals represent a certain level of commitment and need to be consistent.

3.4 CGF-AGENT Personality

Like ISAAC, how the agent uses the information it senses to select movement and firing strategy is decided by the personality of agent in CGF-AGENT.

Basic Personality : Basic personality traits an agent has in CGF-AGENT are

- Tendency to Cluster with healthy friendly units
- Tendency to Support injured friendly units
- Tendency to Preserve itself by avoiding healthy enemy units
- Tendency to be Aggressive to injured enemy units
- Tendency to Fire at Nearest or Weakest enemy targets.

Under special circumstances, agents in CGF-AGENT can modify their personality to deal with the circumstance and assume back its original personality later. This meta-personality uses the concept of fractional force difference to specify various thresholds at which personality mutates. Fractional Force difference (FFD) value gives the relative advantage or disadvantage a combat force has vis-à-vis its enemy force in sensed area for given unit. It is computed as -

Own Force Strength (OFS) = $(F_{healthy}+F_{injured}/2)$
Enemy Force Strength (EFS) = $(E_{healthy}+E_{injured}/2)$
FFD = (EFS – OFS)/OFS

Where

$F_{healthy}$ = Number of healthy friendly units
$F_{injured}$ = Number of injured friendly units
$E_{healthy}$ = Number of healthy enemy units
$E_{injured}$ = Number of injured enemy units

For e.g. if FFD = 0 both force are in balance. If FFD = 0.5 then enemy force is 50% stronger than own force and if FFD = -0.5 then own force is 50% stronger than enemy force.

Meta Personality :CGF-AGENT caters for four configurable rules of meta-personality. They are defined by setting thresholds at which the rules fire. These thresholds are based on FFD value at which Personality mutates as explained below -

Attack :Agent mutates to attacking personality and starts moving towards enemy units when Attack threshold is crossed

Group / Help : Agent adopts supporting personality and starts moving towards friendly units when Group / Help threshold is crossed

Survival: Agent assumes completely defensive personality when under extreme threat and starts moving away from enemy units when Survival threshold is crossed

4. Implementation

Overall Software architecture of CGF-AGENT has a Configuration interface to define the force settings for GC, LC and UC Agents. It also has a plan library, which encodes the procedural knowledge of agents (see Fig 7). The Tactical agent (UC) takes C^2 orders from Strategic agents. The Strategic agents (GC and LC) get reports from UC and use plan library to carry out coordination at their level. It is UC that finally issues Move/Fire order to Combat server.

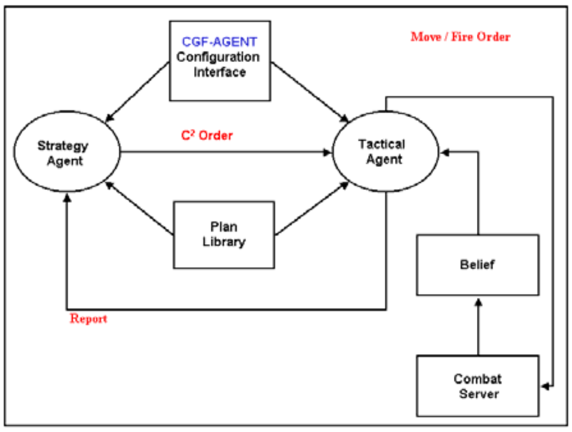


Figure 7: CGF-AGENT Architecture

4.1 Configuration Interface

CGF-AGENT configuration includes

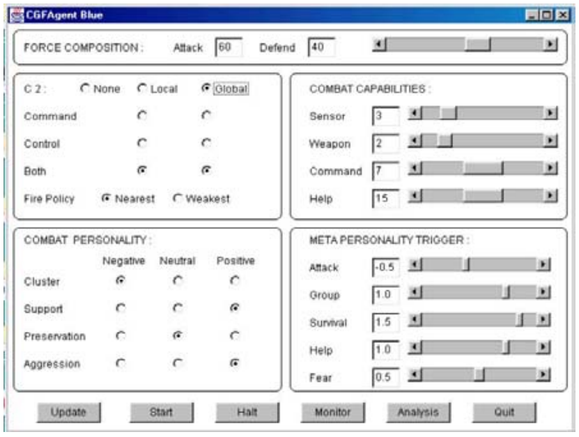
- Force Composition
- Command and Control structure
- Combat Capabilities
- Combat Personality

Force Composition :-
Usually Combat forces have two major tasks. One is to capture enemy base and second is to defend own base from enemy attack. Thus it should be possible to decompose CGF-AGENT into attack and defense task forces.

Command and Control structure :-
Various levels of Command and Control structure can exist in a given combat force. Thus combat force may not have any C^2 and is essentially decentralized. Individual units take independent decision without coordinating or cooperating with other units. At another extreme a combat force may have rich C^2 hierarchy with each higher level coordinating action between entities of subordinate level.

Combat capabilities :-
Combat forces vary in their capabilities to sense, fire and coordinate. CGF-AGENT encodes capabilities thru ranges -
Sensors range. The maximum range up to which unit can sense other units in its neighborhood.
Weapon range. The maximum range up to which unit can fire.
Command and Control range. The maximum range up to which Commanders can coordinate actions of subordinate units.

Combat Personality :-
CGF-AGENT Combat personality is a combination of Basic and Meta personality. Basic personality traits are defined by tendencies and Meta personality is defined by specifying the enabling condition or rule.


CGFAgent Blue

FORCE COMPOSITION : Attack 60 Defend 40

C 2 : None Local [✓ Global]

	Local	Global
Command		
Control		
Both	✓	✓

Fire Policy [✓ Nearest] Weakest

COMBAT CAPABILITIES :

Sensor	3
Weapon	2
Command	7
Help	15

COMBAT PERSONALITY :

	Negative	Neutral	Positive
Cluster	✓		
Support			✓
Preservation		✓	
Aggression			✓

META PERSONALITY TRIGGER :

Attack	-0.5
Group	1.0
Survival	1.5
Help	1.0
Fear	0.5

Figure 8: CGF-AGENT Configuration Interface

CGF-AGENT is configured using GUI (Fig. 8) to set up Force Composition where user specifies the decomposition of attack and defense task force in terms of percentages. Units playing attack role move towards enemy base while defensive units stay near own base to defend it from attacking enemy units. Configuration of Command and Control in CGF-AGENT simulate number of C^2 situations ranging from no C^2 to complete C^2 hierarchy. Between these two extremes various LC and GC Command and/or Control capabilities can be configured. CGF-AGENT combat capability configuration includes defining Sensor, Weapon and C^2 ranges. Combat personality in CGF-AGENT can be configured in various combinations of Basic and Meta personality. Each of basic personality traits can have three possible values – Negative, Neutral and Positive. Similarly rules of Meta-Personality are defined by setting the threshold value of FFD for that rule.

4.2 Execution within CGF-AGENT

CGF-AGENT implementation uses an Agent toolkit called JACK Intelligent Agents™ [8][9][10]. The JACK agents model reasoning according to Belief Desire Intention (BDI) model. To describe how they should go about achieving goals, these agents are programmed with a set of plans. Each plan describes how to achieve a goal under varying circumstances. Set to work, the agent pursues its given goals (Desires) by receiving an event, adopting the appropriate plans (Intentions) according to its current set of data (Beliefs) about the state of the world. This combination of Desires and Beliefs initiating context sensitive intended behavior is part of what characterizes a BDI agent [5].

Events :-
Agents in CGF-AGENT execute logic by interacting with each other using events. Inter agent interaction in CGF-AGENT takes place between Global and

Local command layer and Local and Unit Command Layer (Figure 9).

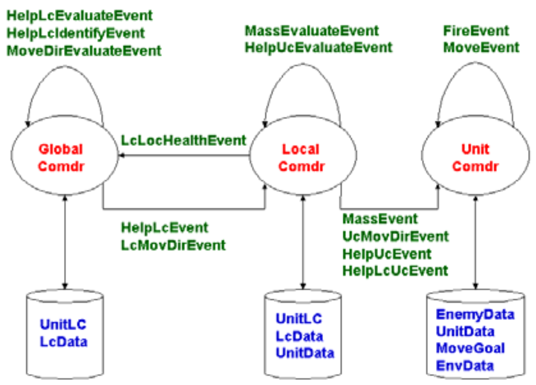


Figure 9: CGF-AGENT Interaction

Global – Local Command Interaction
Events that are exchanged between global and local command layers are:
1. All LC send their health and location report to GC.
2. GC identifies LCs who is required to give support to LC in trouble and instruct them to move towards that LC.
3. GC also identifies individual LCs movement sector.

Local – Unit Command Interaction
Events that are exchanged between local and unit commander layers are:
1. LC periodically checks if its sub-ordinate UCs has drifted too far and if so instruct UC to move closer.
2. LC also instructs UCs to move to sub-area within its command where it is at disadvantage compared to enemy force.
3. LC is also responsible for executing orders it receives from GC with the help of UCs under its command.

Plans
When Agent receives an event it selects and executes a plan. Plan describes a sequence of actions that an agent takes and represents agent's skills and procedural knowledge. CGF-AGENT has over 25 plans in its plan library encoding basic combat and C^2 skills [11][12].

Global Commander Plans
1. Concurrently search for LCs, which are within helping distance of LC in need of help
2. Update it beliefs about LC when it receives report form LC.
3. Get sector wise information of area surrounding LC and choose sector with least FFD. Sends computed move direction to LC
4. Periodically iterate thru all LC information currently held looking for LC in requirement of help (with health below threshold value)

Local Commander Plans
1. Clears beliefs currently held by LC.
2. Concurrently check all UC location in its command and issues massing order if UC has drifted away.
3. Concurrently sends messages to all UCs to converge to sub-area where force balance is not in favor.
4. Used for sending messages to all UCs about GC order for helping LC in need.
5. Used for sending messages to all UCs about GC order for moving towards given sector.
6. Concurrently sends messages to all UCs about GC order for LC help.

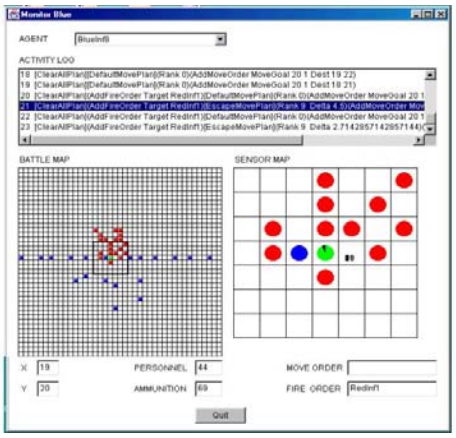
Unit Commander Plans
1. Clears beliefs currently held by UC.
2. Computes Utility of firing at each detected target within weapon range based on distance from it
3. Computes Utility of firing at each detected target within weapon range based on health of target unit.
4. Recognizes opportunity to move towards and attack a single enemy unit within its sensor range without any support from its friendly unit
5. Move towards Injured friendly unit if FFD is below grouping threshold set by user.
6. Move towards Healthy friendly unit if FFD is below grouping threshold set by user.
7. Move away from enemy units if FFD is lower than escape threshold set by user.
8. Move towards Injured enemy unit to attack it if FFD is less than aggression threshold set by user
9. Move towards Healthy enemy unit to attack it if FFD is less than aggression threshold set by user.
10. When environment informs UC that its move goal has been satisfied (i.e. it has reached its destination) UC deletes the move goal from its beliefset and henceforth does not move.
11. Incorporates massing goal issued by LC into its move goal
12. Incorporates Force balance goal issued by LC into its move goal
13. Incorporates Helping LC in need goal issued by LC into its move goal
14. Incorporates Move direction goal issued by LC into its move goal
15. In absence of any special circumstances UC move towards its assigned move goal.

Beliefs
An agent's beliefset is represented in a first order, tuple-based relational model. Jack automatically maintains the logical consistency of the beliefs contained in the beliefset. The beliefset being used by different agents within CGF-AGENT are:
1. Mapping between UC and its LC.

2. LC data like locations, current move goals and health.
3. Data about enemy units in contact
4. 4. Data of location of sub-area with in LC command area where force balance not in favor.
5. Data of UC location and status
6. Maintain current Move goal of UC Data of sensed information about friendly and enemy units and also store FFD within sensed area.

4.3 Monitoring / Analyzing Agent Actions

CGF-AGENT provides an interface to monitor decision-making processes of all agents. Using Monitor GUI (Fig. 10) user can select the agent whose activity he or she wants to track. The Activity log gives details of events sent to that agent and plans used by it handle the event for simulation cycle. It also depicts overall state of the battlefield and sensed environment of the agent at that instant of time. The personal and ammunition held by the unit in Combat server and order issued to by the CGF-AGENT are also shown.



Monitor Blue

AGENT: BlueInf8

ACTIVITY LOG

```
18  [ClearAllPlan][DefaultMovePlan](Rank 0)(AddMoveOrder MoveGoal 20 1 Dest 19 22)
19  [ClearAllPlan][DefaultMovePlan](Rank 0)(AddMoveOrder MoveGoal 20 1 Dest 18 21)
20  [ClearAllPlan](AddFireOrder Target RedInf1)(DefaultMovePlan](Rank 0)(AddMoveOrder MoveGoal 20 1
21  [ClearAllPlan](AddFireOrder Target RedInf1)(EscapeMovePlan](Rank 9  Delta 4.5)(AddMoveOrder Mov
22  [ClearAllPlan](AddFireOrder Target RedInf1)(DefaultMovePlan](Rank 0)(AddMoveOrder MoveGoal 20 1
23  [ClearAllPlan](AddFireOrder Target RedInf1)(EscapeMovePlan](Rank 9  Delta 2.7142857142857144)(
```

X: 19
Y: 20
PERSONNEL: 44
AMMUNITION: 69
MOVE ORDER:
FIRE ORDER: RedInf1

Figure 10: CGF-AGENT Monitoring Interface

Additionally a facility to analyze a run of CGF-AGENT has been provided which gives out comparison graphs (Fig. 11) of Blue and Red side with respect to criteria like

- Total number of Move Order executed
- Total number of Fire Order executed
- Overall health of side
- Units out of action
- Total ammunition consumed
- Advance made by a force

5. Results and Comparison

Sample simulation runs done to generate results aim to explore relevance of developed infrastructure (Combat server and CGF-AGENT) in engineering emergent behavior in CGF.

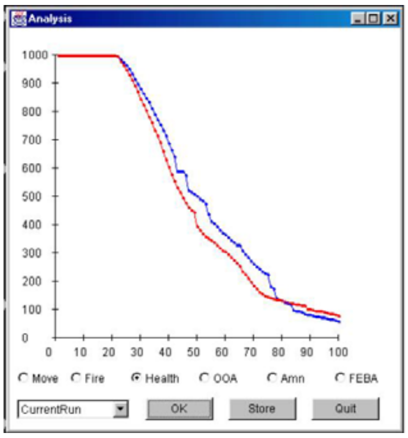


Figure 11: Sample CGF-AGENT Analyzing Interface

For each run required emergent behavior to be engineered along with the personality set for both force are documented. A general description of simulation is given with help of map snapshots (see Fig. 12). Analysis of run is done with respect to measures like amount of move and fire executed by both the forces, their health and casualty suffered over time and consumption of resources like ammunition (see Fig. 13). A special Playback Program was developed to record and replay simulation for this purpose.

ISAAC / CGF-AGENT Comparison

ISAAC heavily influences current work. However there are important differences between ISAAC and Combat Server / CGF-AGENT.

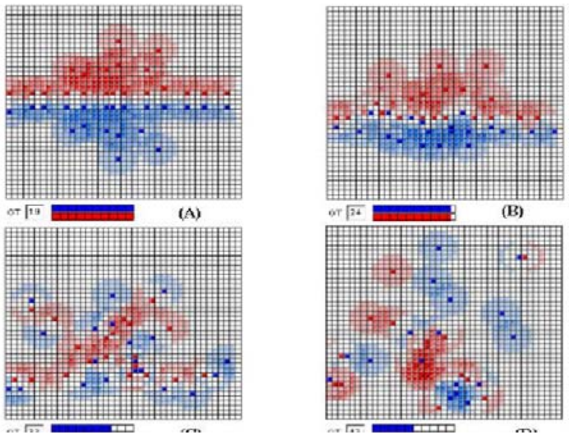


Figure 12: Sample CGF-AGENT Run

1. Combat Server / CGF-AGENT is not a pure agent based simulation but runs the agent based CGF on standalone Combat simulation server. This allows human-in-loop Combat game scenario. In contrast ISSAC is a pure agent based closed loop simulation.
2. Unlike ISAAC, meta-personality in CGF-AGENT is all threshold based and not constraint-

based rule. All rules are based on Fractional force difference

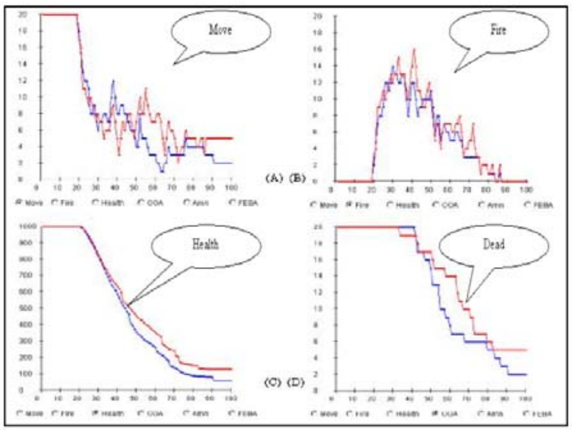


Figure 13: Analysis of CGF-AGENT Run

3. ISAAC Agent is given an opportunity to fire at all detected enemy. In contrast agents in CGF-AGENT select a single enemy target to fire based on user specified policy
4. In ISAAC Global commanders (GCs) issue orders to local commanders using battlefield-wide information. GC effectively knows everything about the overall state of the battle. In CGF-AGENT GCs knows only what LC consolidated reports inform.

6. Conclusion

Land combat has been thought of as a complex adaptive system having large numbers of nonlinearly interacting components with high-level emergent behaviors arising from low-level interaction rules. However it is hard to engineer agents with specific behavior and extensive experimentation is the only way out. The focus of this paper was on developing an approach to reduce trial and error experimentation required to engineer the required emergent behavior.

The approach taken in this paper was to develop an architecture, which allows for human-in-loop Combat game scenario with a generic configurable Multi-Agent based CGF-AGENT, pitted against it. Combat server developed acts as standalone, distributed and interactive combat simulation platform providing easy user interface for humans to interact with the system while giving CGF well defined interface to plug-in.

The Multi-Agent based CGF, CGF-AGENT, can be flexibly configured and can be given wide range of personalities. All relevant functionalities at different layers of Global, Local and Unit Command are included. Decision-making follows BDI model and uses Belief and Plan library. Monitoring Agent activity and analyzing a run provides transparency to the internal working within CGF-AGENT and can be used for validation.

Test runs of CGF-AGENT has proved usefulness of combining an open human-in-loop combat simulation infrastructure with configurable CGF-AGENT having scalable BDI reasoning model in significantly reducing effort required to engineer emergent behavior.

References

1. Ilachinski, Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Warfare, Center for Naval Analyses Research Memorandum CRM 97-61, June1997.
2. Atkin, Marc S., Westbrook David L. and Cohen Paul R., "Capture the Flag: Military Simulation Meets Computer Games". AAAI Spring Symposium on AI and Computer Games, 1999.
3. Tozour, P. (2001) Influence Mapping. In M. Deloura (Ed.), Game Programming Gems 2. Hingham, MA: Charles River Media, Inc.
4. Headquarters, Department of the Army, Washington, DC, 2001, Tactics (Field Manual 3 - 90) http://www.globalsecurity.org/military/library/policy/army/fm/index.html
5. Kinny D, Georgeff M and Rao A. A Methodology and Modelling Technique for Systems of BDI Agents. In W van der Velde and J Perram (eds), Agents Breaking Away, Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), Springer-Verlag, 1996.
6. Shoham Y. Agent-oriented Programming, Artificial Intelligence, Vol. 60, No. 1, pp. 51-92, 1993.
7. Anand S.Rao, Michael P.Georgeff BDI Agents: From Theory To Practice Proceedings of the First International Conference on Multi-Agent Systems 1995.
8. Paolo Busetta, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas, "JACK Intelligent Agents - Components for Intelligent Agents in Java", Technical Report 1, Agent Oriented Software Pty. Ltd www.agent-software.com/shared/resources/reports
9. Nick Howden, Ralph Rönnquist, Andrew Hodgson, Andrew, "JACK Intelligent Agents - Summary of an Agent Infrastructure", 5th International Conference on Automated Agent, Montreal, Canada, 2001.
10. JACK Manual Agent Oriented Software Pty. Ltd.
11. Sumant Mukherjee, Saroj Kaushik, "Agent based Implementation of Automated Command and Control Process", Proceedings of the Sixth International Conference on Cognitive Systems, 2004.
12. Aparna Malhotra, Sanjay Bisht, S.B. Taneja, "Using Intelligent Agent to Simulate Battle Tank Tactics", Proceedings of the Sixth International Conference on Cognitive Systems, 2004.