

University of Essex School of Computer Science and Electronic Engineering

Tactical Level Game AI with natural Language Input

Initial Report

Piers Williams
10/19/2012

Table of Contents

Introduction.....	3
Goals of the project.....	3
To demonstrate that it is possible to merge NLP with RTS Game AI.....	3
To demonstrate that Game AI doesn't have to be as poorly made as many AAA games	3
An abstract guide to the process	4
The dimensions of play.....	4
The environment	4
Initial Run	5
Units.....	5
Technologies	5
Construction.....	5
End game weapons	5
The AI.....	5
General Overview	5
The order system.....	6
Individual Systems.....	6
AI Types.....	7
Work Ethic	7
Organisation of sprints	7
Gantt Chart.....	8
Appendix.....	9
Gantt Chart.....	9
Strategic Planning Language	9
References	12

Introduction

The general project is to design and develop a Real Time Strategy (RTS) game with an Artificial Intelligence (AI) that can be guided, influenced and negotiated with by Natural Language (NL) input. The main reason for doing this is to demonstrate that AI systems can co-operate with human players in a realistic and easy way, paving the way for improved RTS games in the future.

The RTS component of the game will be basic in the areas of graphics and physics, focusing more on game play and AI. This is due to this being developed by myself on my own and with significantly less experience than the team of programmers that a standard game has at its disposal. This game compared to a standard RTS in the industry:

	Typical	Project
Graphics	3D	2D
Physics	Realistic	None simulated
Networking	Capable	Not Capable for multiplayer, May offload computation
Artificial Intelligence	Predictable, Uncontrollable. Best are adaptable a little	Intelligent, Guidable if on same team. Adaptive
Unit diversity	Great, many factions	Sufficient, single faction or two factions

Goals of the project

To demonstrate that it is possible to merge NLP with RTS Game AI

- Create a RTS AI that can co-operate and collaborate with a human player smoothly
 - Assist player in attacks and defence
 - Trade with player to achieve goals or assist
 - Request help from player when necessary
- Create a RTS AI that can understand and process orders in NL
 - Understand and generate an in-game representation of a player's request
 - Assess the players request and calculate costs and chance of success
 - Accept or reject or amend the order based on assessments and the trust level between the AI and the player
- Create a 2-Dimensional (2D) game world that:
 - Has minimal complexity
 - Is large scale
 - Has Reasonable variety for game units

To demonstrate that Game AI doesn't have to be as poorly made as many AAA games

- Create a convincingly good game AI that can:
 - React well to new information
 - Be smart with its attacks and defences
 - Interact with the player as a single unified entity

An abstract guide to the process

At the heart of the project will be the process of turning NL into a defined and rigid AI Instruction Language. This language has been termed the Strategic Planning Language (SPL). The SPL will map nicely to function calls with arguments.

The first part of the process is for the NL to be pre-processed to convert references to game references or co-ordinates. For example, the text “attack the base south of GV-34” would be replaced with “attack GV-56”, assuming that GV-56 is the most reasonable meaning of “south of GV-34”. This will work mostly through some keywords such as “base” that cause the surrounding sentence to be processed for meaning. These can be implemented crudely by maintaining a set of regular expressions.

The second part of the process is for the NL to be Part Of Speech (POS) tagged. This will assign each word with a tag that defines its place and purpose in the sentence. Nouns for example are very likely to wind up as arguments to the SPL functions whilst Verbs will likely be used to determine which functions are selected. For example, splitting up "attack GV-34 with those units" would yield a noun phrase of "those _{DTS} units _{NN}" and a verb phrase with "attack _{VB} GV-34_{NN}". "those _{DTS}" would require anaphora to resolve, based on previous units, likely an order that referenced an existing group or prescribed a construction order.

The third part of the process will be to analyse the instructions given and evaluate them. If accepted, then the instructions will be rendered on screen so that the user can approve the interpretation. If rejected, then the AI will simply output some text into the console. If amended, the AI will render the new version and point out that it is an amendment.

The dimensions of play

All units are split into 3 tech levels. The players start at the 1st tech level, where the units are basic and cheap and weak. Moving up the tech levels increases these things significantly, although the cost-strength ratio will improve for the player providing an advantage. Some types of unit or building will only be available in certain tech levels.

The environment

The world will be a square tile based rectangle. All tiles will be of the same size virtually, although may appear larger or smaller on different platforms/zoom levels.

The game will be set in a space, where it is relatively easy to imagine that there would be no complex obstacles. At worse nebulae and asteroid fields will be around to add variety and introduce some tactical dimensions to the environment. Larger asteroids will contain one or more mass points, with energy generating power stations being available to build anywhere in open space. Asteroid fields will cause damage to larger units, prevent large construction and slow down all movement. Nebulae will give a stealth effect but at the cost of units losing their shielding. Nebulae and Asteroid fields may be postponed in development till late into the project as they are fairly low priority.

Setting things in space allows an easier way to make things look realistic without any extra work. Asteroids are simple to draw and collision detection is also quite easy with them compared to oceans, mountain ranges and craters etc. I also already have some Java code for space based movement and generating star backgrounds. This will nicely improve the speed at which I can code the environment and display it on screen.

Initial Run

For the initial run, units will simply have health and a single weapon each that varies in power and accuracy and type. There will also be very limited unit choice (starting at a single unit) and the BCV's and defence turrets in the initial run. All other additions will be made if time is available; however I have documented many of them here.

Units

Units will have a health meter that they take damage on, and may have their own shield meter that will take damage first.

If the unit is in a shielded area, the shield damage will transfer to the shield building until it is depleted and then damage can be transferred either to another in range shield building or the unit.

Technologies

Units by default output an energy signature when they are active. Units can't generally be deactivated to hide them from radar. Radar displays all units in its radius that has an energy signature. Buildings also emit a signature, though some buildings may be able to shutdown to save power and hide from radar.

Stealth field generators can be built to hide active units from radar. The cost of generating the field will be proportional to the strength of the signatures within the field. Stealth fields will be assigned a budget of Energy which will include a buffer to account for sudden influxes of requirements.

Construction

Bases are built around a Base Construction Vehicle (BCV) or a Stealth BCV (SBCV). The advantage of the SBCV is that the enemy won't notice it travel to the new bases location, making it ideal for sneaky tactics. BCV's and SBCV's provide a small income to enable the player to begin construction.

End game weapons

Artillery and missiles will play a large part of the game, though will be expensive to make and easy to lose. Artillery devices are in-accurate but cheap per shot and do more wide-spread damage. Great for bringing down shields or disrupting incoming land forces they are a staple part of any land base. Missile silos are for providing high damage accurately, though missiles will cost mass and energy to construct and can be shot down by specialised defences. Keep your missile silos secret and the enemy will overlook strong defences making it easier to use them successfully. Like all solid matter items, missiles pass through shields though splash damage from the explosion will also be exempt if it's inside the shield radius.

The AI

General Overview

The artificial intelligence will function as a Multi-Agent System (MAS). This design will reduce the amount of work that a single system will have to handle, and thus by minimising and defining the interactions between them they can be made relatively robust. Additionally the individual components will be largely simplified, making their construction and testing quicker and easier. This tactic is described well in [1].

The order system

The system of orders is split into two main halves. These orders are the machine understandable version of your NL input and are hidden from the user.

The first half of this is the Query section. The query sections allow you to query an AI to discover information about its forces or its perception of the environment.

The second half of this is the Order section. The order section allows you to submit commands to an AI for it to review. It will then review the action, and propose an alteration or accept it.

The most recent version of the work in progress Strategic Planning Language is in the appendix section.

Individual Systems

The lowest form of agent will be the AI that handles an individual unit. The AI for this will be loaded with a unit's data, and then it will decide and output the next move/action for the unit. Furthermore, many of these AI's can work in parallel to improve efficiency without creating an AI for every unit in the game (Potentially thousands). Some buildings will also be given an AI in this manner. Units may well possess a static FSM with an object level State stored. FSM's as unit AI is covered in [3].

The next layer up will be the squad AI systems. These will be responsible for the operations of groups of units or buildings (such as turrets) to ensure that they behave in co-ordination and don't display erratic behaviour.

Further up the scale will be the AI that manages bases. Individual bases will be given an AI that will use its individual budget to best upgrade itself based on the goals it has been assigned by the upper AI systems.

Next up in the chain is an AI that will be a basic resource allocator unit, that analyses the needs and wants of the other AI's and assigns a portion of the input stream/storage streams of resources.

Finally there is an AI that decides what the long term goals are going to be by adjusting a priority matrix which the other AI's base their decisions on. So if a sudden artillery threat is discovered, then the shield building priority can be increased. Base Construction AI's will notice this, assess the threat to their individual base and act accordingly. Requests for more resources for these tasks will be allocated with more resource than previously assigned due to the higher priority.

AI Types

There are two main types of AI in the game. There is the active and the informative AI. The active AI's are charged with acting on the world while the informative AI's are charged with analysing the world. There may be multiple instances of the same type of Active AI with different goals. The active AI will base its actions on the analysis of the informative AI, and thus in doing so can provide new or updated information with which the informative AI can base its analysis. There will always be one perceived instance of each Informative AI, although the workload may be parallel. Tactical analysis is covered in depth in [2] and the major part that I would use is influence maps. These are easy to construct and allow the AI to calculate many useful metrics. The AI's are split as follows:

Informative AI	Active AI
Threat Perception	Scouting (Single)
Opportunity Perception	Attacking (Multiple)
Strength Perception	Defending (Multiple)
Weakness Perception	Base Construction (Multiple)
Terrain Perception	Resource Expansion
Enemy Resource Analysis	
Path Finding	

The Informative AI's are not given a budget at all of in-game resources and simply continuously tick over, whilst Active AI's are assigned a budget based on their requirements for the tasks they have been set, and the priority of that task.

Work Ethic

For this project I will be using Agile Development as a method of keeping myself and the workload organised. Each sprint will be split into smaller tasks that will each be tackled in order. The Gantt chart will document these sprints and their corresponding tasks in detail. The earlier sprints will be more detailed at the moment, with the NL sprints being less detailed due to myself not knowing as much about that topic. Once a sprint has been achieved in the simplest way and level of functionality possible, it will be closed and I will move on. This means that

- Graphics will be simple and quick
- Code will start off simple and less intelligent
- Things will only be added if necessary to complete the project












Once the base goals of the project have been achieved, the simple code can be replaced with more complicated code to flesh out the project. At this point however, the project can always be back-tracked to a working state.

Organisation of sprints

From this, the project will be split into two main halves, the first half will be as quick and basic as possible and is the development phase. It will be split into a number of sprints. The second half is arbitrarily long and will consist of a number of sprints designed to improve the feature set of the project. These can involve a number of tasks such as improving unit AI, optimisation of a particular function(s) or addition of a new feature or capability.

Gantt Chart

Here below, you can see an extract from my Gantt chart. I've omitted the graph section because it is superfluous to the point and takes up too much space. Here you can see the rough plan of how the entire project is arranged, with a task denoting the main sprints (Sprint A, Sprint B here) and smaller tasks that have Start-to-Start dependencies leading from those sprints. Those tasks make up the elements of the sprint so to say. Project wouldn't allow me to add a dependency for the sprint that the last task be finished before it finishes due to some rule about cyclical dependencies however the idea is valid. The sprints don't particularly exist; they just collate together tasks to achieve a higher single goal. So far there are sprints that go up to the letter L in the complete Gantt chart (Included with this submission). I have also included the major milestones set to us by the university, such as all the hand in dates. This helps me to see when things are due in by so I don't forget and work too hard on my project itself, but also to show perhaps why I work a little slower in these times. The duration estimates are how many days I think it will take me to complete the task, and does not represent 8 hours a day as Microsoft Project feels it will. Expressing things in hours is very pie in the sky for this sort of thing and would likely introduce large amounts of error for me.

	 Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Initial Report	15 days	Mon 01/10/12	Fri 19/10/12	
2		Interim Report	31 days	Sat 20/10/12	Fri 30/11/12	
7		Sprint A: Basic Infrastructure	5 days	Mon 22/10/12	Fri 26/10/12	1
19		Get World Running	2 days	Mon 22/10/12	Tue 23/10/12	7SS
20		Get Maps Loading	2 days	Wed 24/10/12	Thu 25/10/12	19
8		Sprint B: User Interface	5 days	Mon 29/10/12	Fri 02/11/12	7
21		Add Resource Overlay	1 day	Mon 29/10/12	Mon 29/10/12	8SS
22		Add Build Menu Overlay	1 day	Tue 30/10/12	Tue 30/10/12	21
23		Add MiniMap	2 days	Wed 31/10/12	Thu 01/11/12	22
40		Add Console Window for NLP Entry	1 day	Fri 02/11/12	Fri 02/11/12	23

Appendix

Gantt Chart

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Initial Report	15 days	Mon 01/10/12	Fri 19/10/12	
2		Interim Report	31 days	Sat 20/10/12	Fri 30/11/12	
7		Sprint A: Basic Infrastructure	5 days	Mon 22/10/12	Fri 26/10/12	1
19		Get World Running	2 days	Mon 22/10/12	Tue 23/10/12	7SS
20		Get Maps Loading	2 days	Wed 24/10/12	Thu 25/10/12	19
8		Sprint B: User Interface	5 days	Mon 29/10/12	Fri 02/11/12	7
21		Add Resource Overlay	1 day	Mon 29/10/12	Mon 29/10/12	8SS
22		Add Build Menu Overlay	1 day	Tue 30/10/12	Tue 30/10/12	21
23		Add MiniMap	2 days	Wed 31/10/12	Thu 01/11/12	22
40		Add Console Window for NLP Entry	1 day	Fri 02/11/12	Fri 02/11/12	23
9		Sprint C: Buildings	5 days	Mon 05/11/12	Fri 09/11/12	8
24		Add BCV	1 day	Mon 05/11/12	Mon 05/11/12	9SS
25		Add Small Buildings	1 day	Tue 06/11/12	Tue 06/11/12	24
26		Add Medium Buildings	4 hrs	Wed 07/11/12	Wed 07/11/12	25
27		Add Large Buildings	4 hrs	Wed 07/11/12	Wed 07/11/12	26
10		Sprint D: Resource System	5 days	Mon 12/11/12	Fri 16/11/12	9
28		Make PowerPlants generate Power	2 days	Mon 12/11/12	Tue 13/11/12	10SS
29		Make Mass Points generate Mass	2 days	Wed 14/11/12	Thu 15/11/12	28
11		Sprint E: Tech Tree Implementation	5 days	Mon 19/11/12	Fri 23/11/12	10
30		Restrict Buildings based on Tech Level	1 day	Mon 19/11/12	Mon 19/11/12	11SS
31		Allow Tech Level To Increase	3 days	Tue 20/11/12	Thu 22/11/12	30
12		Sprint F: Units and Unit AI	5 days	Mon 26/11/12	Fri 30/11/12	11
18		Basic Unit Implementation	2 days	Mon 26/11/12	Tue 27/11/12	12SS
17		Basic Pathfinding	2 days	Wed 28/11/12	Thu 29/11/12	18
3		Open Day Poster	69 days	Fri 30/11/12	Wed 06/03/13	
4		Open Day Abstract	69 days	Fri 30/11/12	Wed 06/03/13	
5		Project Presentation Availability Form	69 days	Fri 30/11/12	Wed 06/03/13	
6		Final Report	104 days	Fri 30/11/12	Wed 24/04/13	
13		Sprint G: Squad AI	5 days	Mon 03/12/12	Fri 07/12/12	12
32		Implement Ghost Unit for Squads	1 day	Mon 03/12/12	Mon 03/12/12	13SS
33		Cause Units in Squad to Follow Ghost (Flocking)	3 days	Tue 04/12/12	Thu 06/12/12	32
14		Sprint H: Base AI	5 days	Mon 10/12/12	Fri 14/12/12	13
34		Basic Build Plan To Follow	2 days	Mon 10/12/12	Tue 11/12/12	14SS
35		Choose different plans based on base type	2 days	Wed 12/12/12	Thu 13/12/12	34
15		Sprint I: Resource Allocator AI	10 days	Mon 17/12/12	Fri 28/12/12	14
36		Priority System	1 day	Mon 17/12/12	Mon 17/12/12	15SS
37		Allocator Based On Priorities	4 days	Tue 18/12/12	Fri 21/12/12	36
16		Sprint J: Objective Allocator AI	10 days	Mon 31/12/12	Fri 11/01/13	15
38		Goal Allocator	2 days	Mon 31/12/12	Tue 01/01/13	16SS
39		Priority Calculator	5 days	Wed 02/01/13	Tue 08/01/13	38
41		Sprint K: Natural Language Parsing	14 days	Mon 14/01/13	Thu 31/01/13	16
43		Base Location Inference System	3 days?	Mon 14/01/13	Wed 16/01/13	41SS
44		Group Location Inference System	2 days?	Thu 17/01/13	Fri 18/01/13	43
45		POS Tagger	3 days?	Mon 21/01/13	Wed 23/01/13	44
46		Sprint L: Order Comprehension and Analysis	14 days?	Fri 01/02/13	Wed 20/02/13	41
47		Order analysis	5 days	Fri 01/02/13	Thu 07/02/13	46SS
48		Order Alteration	3 days	Fri 08/02/13	Tue 12/02/13	47
49		Order Following	5 days?	Wed 13/02/13	Tue 19/02/13	48
42		Project Demonstration	8 days	Thu 07/03/13	Mon 18/03/13	3

Strategic Planning Language

 *** Strategic Planning Language Specification ***

Two Parts to the language.

Query and Order

 *** Query ***

A Query is a request of information from the AI

Query

QueryObject

HowMany - How many of these do you have?

1 Argument(ObjectType)

ObjectType - Object type to count

HowManyGreedy - How many of these and related do you have?

1 Argument(ObjectType)

ObjectType - Object type to count, as well as sub and parent types

HowManyAvailable - How many of these to spare?

1 Argument(ObjectType)

ObjectType - Object type to count

HowManyGreedyAvailable - How many of these and related to spare?

1 Argument(ObjectType)

ObjectType - Object type to count, as well as sub and parent types

QueryTime

HowLong - How long to complete an order?

1 Argument(Order)

Order - A previously agreed Order

HowLongStart - How long till start an order?

1 Argument(Order)

Order - Either previously agreed Order or OrderWIP

QueryAssessment

AttackStrength - Evaluate attack strengths / weaknesses

1 Argument(Player/Team)

Player/Team - Evaluate the player or team given with all available information to AI's team

DefenseStrength - Evaluate defense strengths / weaknesses

1 Argument(Player/Team)

Player/Team - Evaluate the player or team given with all available information to AI's team

TotalStrength

AttackStrength(Player/Team) && DefenseStrength(Player/Team)

AssessAll - Assess TotalStrength of all players and return matching strengths to weaknesses

*** Order ***

An Order is a request to the AI to carry out a task

Order ::= Order | Order && Order

4 Main Types of order

Attack, Defend, Trade and Explore

Attack - Attack orders are given to direct the AI to attack a location or structure or base.

Assassinate - Orders the AI to take out a single structure or unit without regard to others. Should withdraw after completion

2 Arguments (Object, Severity)

Object - The Target

Severity - How important it is to succeed

Destroy - Orders the AI to completely destroy a base

2 Arguments (Base, Severity)

Base - The Base to destroy

Severity - How important it is to succeed

Neutralize - Orders the AI to keep the enemy out of a region

3 Arguments (Location, Radius, Severity)

Location - Location to use as centre point

Radius - How wide around the centre point to go

Severity - How important it is to succeed

Defend - Defend orders the AI to perform defensive tasks

Grow - Orders the AI to improve its own base defences

3 Arguments (Base, Type, Severity)

Base - The Base to grow

Type - The thing to defend against

Severity - How much resources to put into this. Out of 5.

Occupy - Orders the AI to construct a base in an area
 3 Arguments (Location, Type, Severity)
 Location - The location to use as centre point
 Type - The purpose of the base
 Severity - How large/Strong will the base be
 OccupyForMe - Orders the AI to construct a base for me in an area
 (Occupy followed by GiveBase)
 3 Arguments (Location, Type, Severity)
 Location - The location to use as centre point
 Type - The purpose of the base
 Severity - How large/Strong will the base be

Explore - Explores the terrain
 Explore - Explore a geographical location
 3 Arguments (Location, Radius, Severity)
 Location - The Location to use as centre point
 Radius - How much to explore around the point
 Severity - How soon should we explore

Search - Search for a particular item
 4 Arguments (Resource, Location, Radius, Severity)
 Resource - The Resource we are looking for
 Location - The Location to use as centre point
 Radius - How much to explore around the point
 Severity - How soon should we explore

Trade - Negotiates trade of units/resource/buildings
 Give - Orders the AI to hand over control
 GiveObject
 1 Argument (Object)
 Object - The Object to be transferred
 GiveGroup
 1 Argument (Group)
 Group - The Group to be transferred
 GiveBase
 1 Argument (Base)
 Base - The Base to be transferred
 GiveAll
 0 Arguments

Receive
 ReceiveObject
 1 Argument (Object)
 Object - The Object to be transferred
 ReceiveGroup
 1 Argument (Group)
 Group - The Group to be transferred
 ReceiveBase
 1 Argument (Base)
 Base - The Base to be transferred
 ReceiveAll
 0 Arguments

Swap
 SwapObject
 2 Arguments (ObjectMine, ObjectYours)
 GiveObject(ObjectMine) && ReceiveObject(ObjectYours);
 SwapGroup
 2 Arguments (GroupMine, GroupYours)
 GiveGroup(GroupMine) && ReceiveGroup(GroupYours);
 SwapBase
 2 Arguments (BaseMine, BaseYours)
 GiveBase(BaseMine) && ReceiveBase(BaseYours);

References

- [1] I Millington and J Funge, "Chapter 6.4.1" in "*Artificial Intelligence for Games 2nd Edition*" on "page 559", Morgan Kaufmann
- [2] I Millington and J Funge, "Chapter 6.2" in "*Artificial Intelligence for Games 2nd Edition*" on "page 518", Morgan Kaufmann
- [3] I Millington and J Funge, "Chapter 5.3" in "*Artificial Intelligence for Games 2nd Edition*" on "page 309", Morgan Kaufmann