# Game AI R&D Project

## Design Document

**Synopsis:** This document outlines the Immersive AI engine concepts, objectives and provides a high-level and detailed design of the components of the system.

Reference: GameAI.Design v1.0

Date: 3 May 2006

Author: Gavin Bunney & Tom Romano

Status: Definitive

iAI

immersive ai engine

# Document Control

### Version History

Every change to this document is logged in the table below.

| Ver. | Date | Author | Description |
|---|---|---|---|
| v0.1 | 2006-04-15 | Gavin Bunney | Initial document creation |
| v0.2 | 2006-04-17 | Gavin Bunney | Added System architecture notes & diagrams |
| v0.3 | 2006-04-17 | Gavin Bunney | Added System / TGE / Package sections |
| v0.4 | 2006-04-17 | Tom Romano | Added States & Goals to HLD |
| v0.5 | 2006-04-20 | Gavin Bunney | Added iAIManager / State machine & classes |
| v0.6 | 2006-04-24 | Tom Romano | Added iAISeek detailed design / class diagram |
| v0.7 | 2006-04-29 | Gavin Bunney | Added iAICombat / StrikePrediction design |
| v0.8 | 2006-04-30 | Tom Romano | Added iAIInteract detailed design |
| v0.9 | 2006-05-03 | Gavin Bunney<br>Tom Romano | Changed made from internal review feedback |
| v1.0 | 2006-05-03 | Gavin Bunney<br>Tom Romano | Definitive Issue |

## Project Abstract

Implemented game AI, particularly in RPG/MMORPG games, is very script based. The NPC's walk in a set path, speak with set scripts; the mobs have scripted actions. The concept behind the project is to create a more realistic AI, to both provide unpredictability in NPC behaviours, and to immerse a player in the game world.

The project is to research AI techniques, design and implement a goal-based AI system. Goal-based AI is a technique used to create NPC's which act as real players; in that they are given an objective to achieve - e.g. Given a goal of "rake leaves" it is up to the NPC to work out how to achieve their goal, whether through buying a rake, stealing one, or killing another NPC for their rake.

The implemented AI system will be in the form of various classes, created in C++ and Torque Script for the Torque Game Engine, version 1.4. For more information on the Torque Game Engine, visit http://www.garagegames.com.


# Contents


**1 INTRODUCTION ..... 6**

1.1 PURPOSE ..... 6
1.2 SCOPE ..... 6

**2 IMMERSIVE GAME ENGINE ..... 7**

2.1 IMMERSIVE GAME WORLD ..... 7
2.2 AIMS & OBJECTIVES ..... 7
2.3 IMMERSIVE AI – IAI ..... 7
2.4 AI CONCEPTS IMPLEMENTED ..... 7

**3 HIGH-LEVEL DESIGN ..... 8**

3.1 TORQUE GAME ENGINE ..... 8
3.1.1 Game Architecture ..... 8
3.2 STATES AND GOALS ..... 9
3.2.1 Goal Oriented States ..... 9
3.3 SYSTEM COMPONENTS ..... 10
3.3.1 State & Goal Manager ..... 10
3.3.2 State Controls ..... 10
3.4 PACKAGE AND SUBSYSTEMS ..... 11

**4 DETAILED DESIGN ..... 12**

4.1 IAIMANAGER ..... 12
4.1.1 Purpose ..... 12
4.1.2 Goal Manager ..... 12
4.1.3 State Manager ..... 13
4.2 IAISEEK ..... 14
4.2.1 Purpose ..... 14
4.2.2 Core Functions ..... 14
4.2.3 Goal-State Lists ..... 15
4.3 IAICOMBAT ..... 19
4.3.1 Purpose ..... 19
4.3.2 Core Functions ..... 19
4.3.3 Goal-State Lists ..... 19
4.4 IAIINTERACT ..... 21
4.4.1 Purpose ..... 21
4.4.2 Core Functions ..... 21
4.4.3 Goal-State Lists ..... 21

**5 GLOSSARY ..... 24**

**6 REFERENCES ..... 25**

## List of Figures

Figure 3-1 Game Architecture ..... 8
Figure 3-2 System Components ..... 10
Figure 3-3 Package Components ..... 11
Figure 3-4 Subsystems ..... 11
Figure 4-1 iAIManager State Machine ..... 12
Figure 4-2 Goal Manager ..... 13
Figure 4-3 State Manager ..... 13
Figure 4-4 Path Manager ..... 14
Figure 4-5 Strike Prediction Class ..... 19

## List of Tables

Table 3-1 Goal List ..... 9
Table 4-1 Buy Food State-Transition Table ..... 15
Table 4-2 Steal Food State-Transition Table ..... 15
Table 4-3 Seek out Food State-Transition Table ..... 16
Table 4-4 Buy Health Pack State-Transition Table ..... 16
Table 4-5 Steal Health Pack State-Transition Table ..... 16
Table 4-6 Seek out Health Pack State-Transition Table ..... 17
Table 4-7 Do Explore State-Transition Table ..... 17
Table 4-8 Do Patrol State-Transition Table ..... 17
Table 4-9 Go Home State-Transition Table ..... 18
Table 4-10 Kill Opponent State-Transition Table ..... 19
Table 4-11 Wound Opponent State-Transition Table ..... 20
Table 4-12 Defend Self State-Transition Table ..... 20
Table 4-13 Wound Opponent State-Transition Table ..... 20
Table 4-14 Dance State-Transition Table ..... 21
Table 4-15 Sing State-Transition Table ..... 21
Table 4-16 Talk to Self State-Transition Table ..... 22
Table 4-17 Talk to Another State-Transition Table ..... 22
Table 4-18 Have Sleep State-Transition Table ..... 23
Table 4-19 Have Relax State-Transition Table ..... 23

# 1 INTRODUCTION

## 1.1 Purpose

The purpose of this document is to outline the Immersive AI engine concepts, objectives and provides a high-level and detailed description of the various components of the system. It aims to encapsulate the entire definition and design phases of the projects, to outline explicitly the functionality of the system.

## 1.2 Scope

The scope of this document is limited to the Immersive AI engine which will be completed by 2nd October 2006.

# 2 IMMERSIVE GAME ENGINE

## 2.1 Immersive Game World

An immersive game world is an environment which engages players wholly within the confines of the game, through creating an alternate reality for the gamer to explore.

Many key elements make up an immersive game world; graphically intense environments, complex story lines and plots, interactive objects and the reactions of the game world all play a part in the creation of an immersive game environment. With advances in hardware and software, visually immersive environments are now completely achievable. This has lead to further developments and research into game artificial intelligence systems, to help creating game worlds which absorbs the player, making them feel apart of the system.

## 2.2 Aims & Objectives

The aim of this project is to create a game artificial intelligence system, to assist in the creation of an immersive game world. The implemented system will allow computer controlled game agents to act, react and interact with the game world in a seemingly intelligent manner.

The key objectives of the implemented system are as follows:

1. To create an artificial intelligence system which controls AI agents that interact with the game as seemingly intelligent entities

2. To design a general system which can be implemented for various goal-states and game types

3. To implement a system which has minimal impact on game performance

## 2.3 Immersive AI – iAI

The name of the completed system, and referenced frequently as iAI, will be ‘immersive AI engine’; this demonstrates clearly the immersive nature of the implemented AI system.

## 2.4 AI Concepts Implemented

In order to meet the aims and objectives of the project, a number of AI concepts will be implemented. These include, but are not limited to, A* Path Finding and Bayesian Network reasoning. The A* Path Finding implementation will allow for the agents to traverse the game world, whilst the Bayesian Networks will form the base for decision making processes under uncertainty.

# 3 HIGH-LEVEL DESIGN

## 3.1 Torque Game Engine

Immersive AI will be implemented within the Torque Game Engine (TGE). Torque was chosen as it is a fully fledged game engine, not simply a rendering environment; it is also cross platform with Windows, Mac OSX and Linux operating systems being natively supported.

The engine source code for TGE is written in platform independent C++, with the in game controls written in a proprietary TorqueScript language. TorqueScript is based on C++ and allows for abstraction of in game rules from the backend engine code.

### 3.1.1 Game Architecture

The type of game implemented for Immersive AI will be an RPG-style game, consisting of client-server architecture. The AI agents are controlled by the server and ghosted to the client connection. Although it will be primarily run on a single machine, this design allows for two main advantages:

- Separation of server world processing code and the client interaction code
- Ease of expansion onto a multi-server or multiplayer game

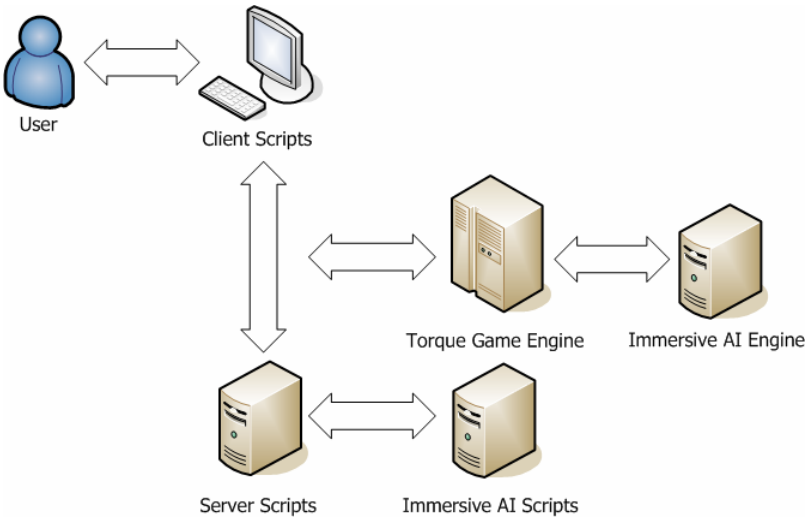


**Figure 3-1 Game Architecture**

## 3.2 States and Goals

Immersive AI, being a goal-based AI system means that AI agents are controlled by the need to fulfil various objectives. An agent being assigned a particular goal can be caused from various triggers, such as low health invoking a goal of finding a health pack.

The table below lists all the goals implemented within Immersive AI.

| Goal | Description |
|---|---|
| Get Food | The agent has to acquire some food |
| Get Health | The agent has to acquire medical assistance |
| Attack | The agent has to attack some other object or agent |
| Defend | The agent has to defend against another object or agent |
| Have Fun | The agent has to perform some fun action |
| Have Talk | The agent has to talk to another agent |
| Do Rest | The agent has to have a rest and relax and/or sleep |
| Do Explore | The agent has to venture around to explore surroundings |
| Do Patrol | The agent has to venture around and guard an area |
| Go Home | The agent has to venture back to their home or origin |

**Table 3-1 Goal List**

In order to satisfy a master goal, the AI agents must transition through various states; sub stages to achieve the set goal. The way in which the goal based logic for the iAI engine will function is based on a finite state machine. Mat Buckland in his book, "Programming Game AI by Example" offers this explanation of "*A finite state machine [as] a device, or a model of a device, which has a finite number of states it can be in at any given time and can operate on input to either make transitions from one state to another or to cause an output or action to take place. A finite state machine can only be in one state at any moment in time.*" (p44).

For each of the goals within the iAI system, a finite state machine will be created. This will enable for a step by step process which will be carried out in order to achieve the goal.

### 3.2.1 Goal Oriented States

Each of the goals has a set of states that it will process in order to achieve said goal; outlined here in a goal by goal fashion. Each of the goals have a state transition table which highlights a high level overview of the transitions that will occur from one state to the next and the conditions upon which these changes will be triggered.

## 3.3 System Components

Immersive AI is designed to accommodate a wide range of states and goals and thus will contain various generic functions to help the AI agents to act, react and interact with the game world.

The Immersive AI system is split into two main components – the state and goal manager, and the implemented state controls.

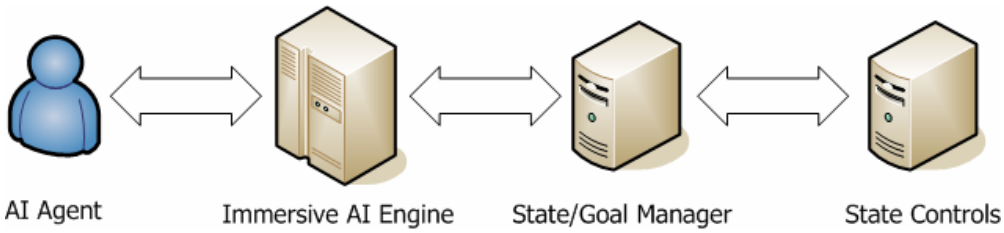


**Figure 3-2 System Components**

### 3.3.1 State & Goal Manager

The State and Goal Manager's primary responsibility is to assign and update the various AI agent goal lists. It is responsible for altering an agent's goal list and assigning various state transitions to meet its respective goal.

### 3.3.2 State Controls

The State Controls are a collection of core functions and implemented states; divided into modules of common functionality. The core functions are controls which are required by the various states to meet a specific aim. A* path finding is one such core function which allows the AI agents to traverse the game world.

### 3.4 Package and Subsystems

Immersive AI will consist of a single package, with four subsystems; these subsystems will control the various actions of the AI agent. The State & Goal manager will be implemented in iAIManager, with various state controls implemented as subsystems iAISeek, iAICombat and iAIInteract.

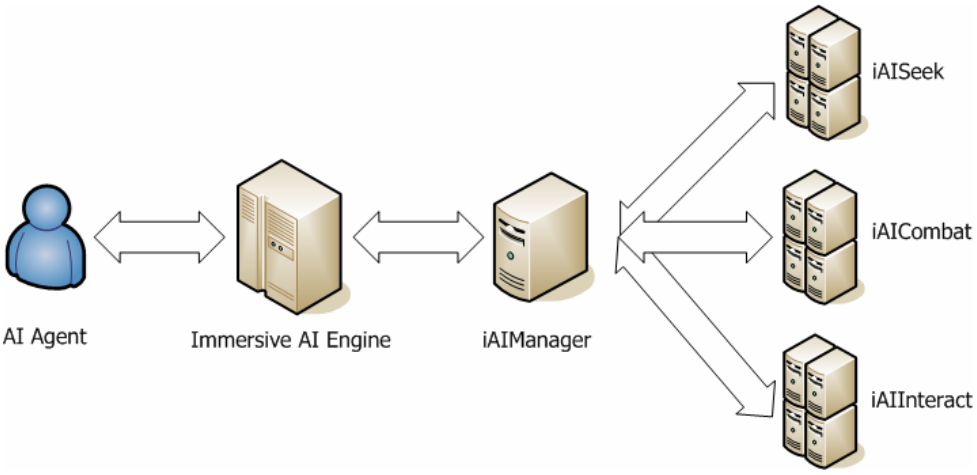


**Figure 3-3 Package Components**

The various subsystems, explained in more detail in Section 4, are designed to accommodate any function an AI agent could complete within the game environment.

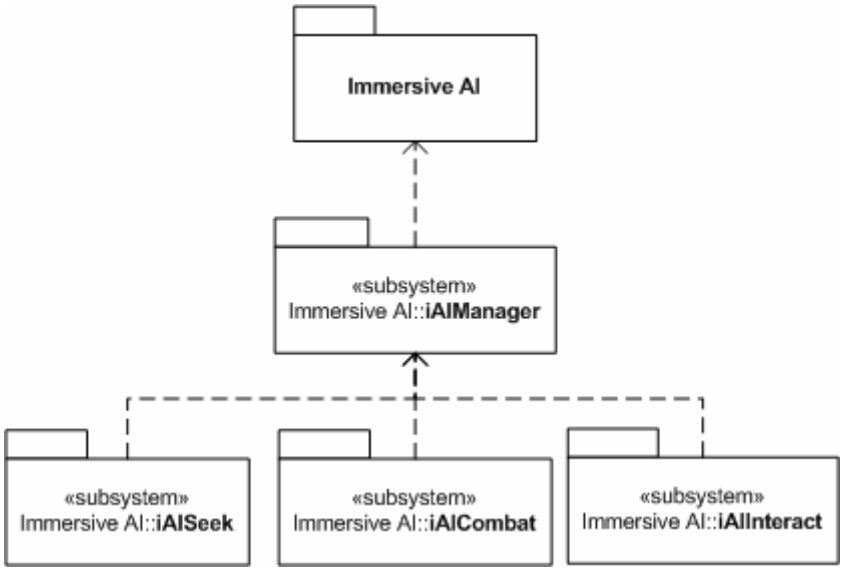


**Figure 3-4 Subsystems**

# 4 DETAILED DESIGN

## 4.1 iAIManager

### 4.1.1 Purpose

This module is designed to manage the state and goal lists for AI agents. It is responsible for recognising goal-change triggers and forming state machine processes for agents to complete.

The figure below illustrates the iAIManager internal state machine.

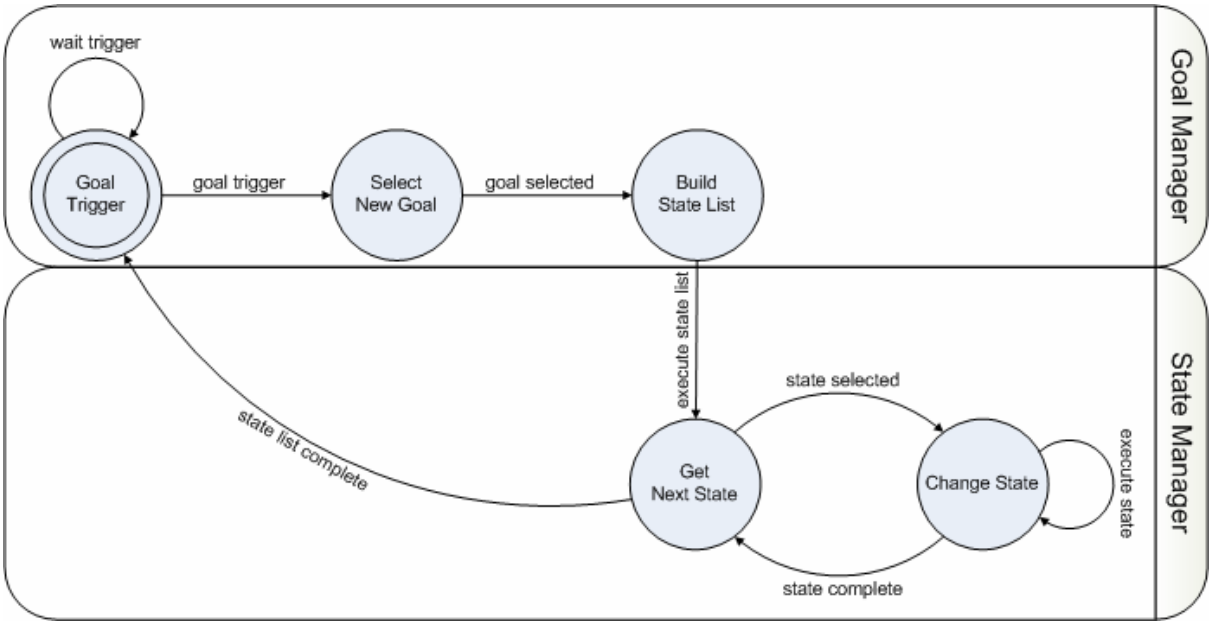


**Figure 4-1 iAIManager State Machine**

### 4.1.2 Goal Manager

The Goal Manager is responsible for detecting any goal change triggers within the game environment. These triggers range from an AI agent having low health, feeling hungry, being attacked and many more.

The goal triggers and state lists are sent programmatically to the Goal Manager during the game world creation. Upon detecting of a goal trigger, the Goal Manager will select an appropriate goal and build a state list to complete said goal. This state list is then executed by the State Manager, until each has been completed.

It is important to note that the goal triggers can be set to act as interrupts, overriding any currently set goals: such as if the agent is current executing the goal “Do Explore” and gets attacked, the primary goal of the agent would change to “Defend”.

#### 4.1.2.1 Class Diagram

The Goal Manager will consist of a single class, containing a primary goal manager and interface structure to allow adding of goals and triggers.

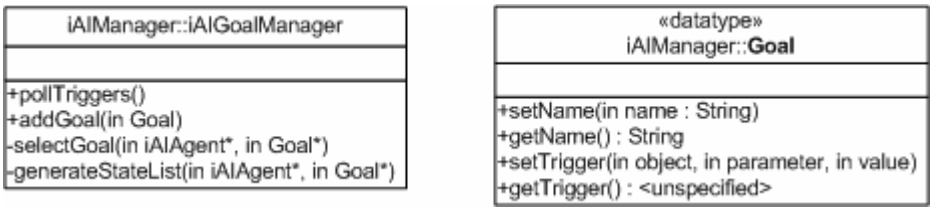


**Figure 4-2 Goal Manager**

### 4.1.3 State Manager

The State Manager is responsible for managing each AI agent's current state.

#### 4.1.3.1 State Manager Classes

The State Manager will consist of 2 classes; a primary state manager class and an interface class to implement the various states.

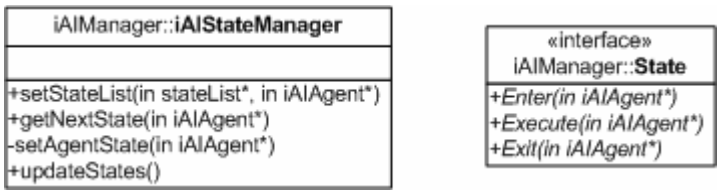


**Figure 4-3 State Manager**

During normal operation, the individual AI agents, iAIAgent, will call getNextState from the iAIStateManager class, the function then sets the agents next state. If a goal interrupt has occurred, such as being attacked whilst exploring, the iAIStateManager will issue an updateStates function call to update an iAIAgent's state, ensuring that the agents will change states immediately, rather than waiting till it has finished with the current state execution.

Each state implemented in iAI will implement the State interface. This ensures that a common entry, execution and exit function exists for all State classes.

## 4.2 iAISeek

### 4.2.1 Purpose

This module is designed to accommodate for all seek, search and path finding based actions and algorithms for an AI agent.

### 4.2.2 Core Functions

This module will have as its core functionality an implementation of an A* Path Finding Manager in order to carry out its required tasks. Using this A* algorithm it will be able to effectively and efficiently complete the tasks required within each of its states.

#### 4.2.2.1 A* Path Finding

In many AI systems, AI agents do not have good path finding implementations that can sometimes lead to a player recognizing the AI system. A good AI implementation should be so well represented that it acts almost in an invisible fashion to the player. The iAISeek module is the controller class for all these critical actions which require A* path finding algorithms and techniques in order to achieve their goals. More information on A* path finding techniques can be found in the “iAI Research” document prepared by Gavin Bunney and Tom Romano.

Below is the class diagram for the path finding manager class.

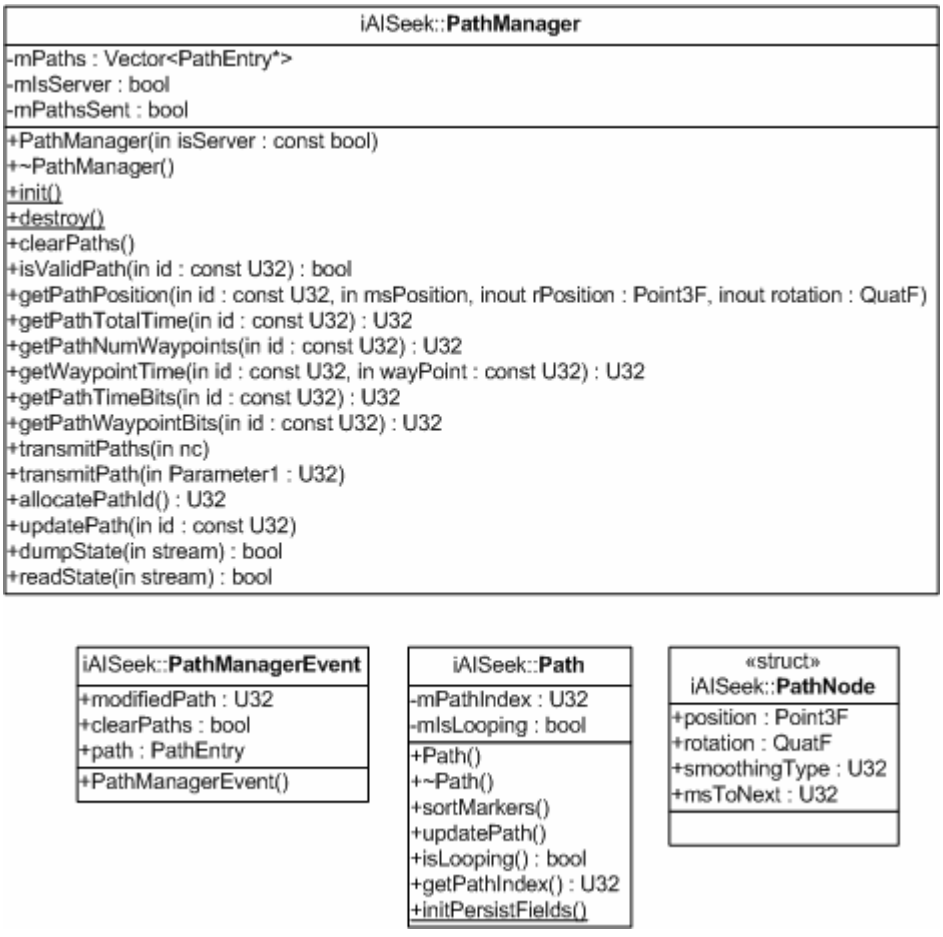


**Figure 4-4 Path Manager**

#### 4.2.3 Goal-State Lists

##### 4.2.3.1 Goal: Get Food

For the goal of "Get Food" to be achieved there are three major possibilities: buy the food, steal the food or seek out (search for) the food. Once the state **eatFood** is reached the goal is said to have been achieved.

###### 4.2.3.1.1 State-List: Buy Food

These are the possible states if an Agent wishes to achieve the "Get Food" goal by purchasing the food.

| Current State | Condition | Next State |
|---|---|---|
| buyFood | Need to find somewhere to buy food from | findVendor |
| findVendor | Found place to buy food | buyFromVendor |
| buyFromVendor | Sufficient Money | completeSale |
| buyFromVendor | Insufficient Money and Not Hostile | findVendor |
| buyFromVendor | Insufficient Money and Hostile | stealFood |
| completeSale | Sale Completed Successfully | takeFood |
| takeFood | Agent now has food | eatFood |

**Table 4-1 Buy Food State-Transition Table**

###### 4.2.3.1.2 State-List: Steal Food

These are the possible states if an Agent wishes to achieve the "Get Food" goal by stealing the food.

| Current State | Condition | Next State |
|---|---|---|
| stealFood | Need to find somewhere to steal food from | findVendor |
| stealFood | Previous State buyFromVendor | killVendor |
| findVendor | Found place to steal food | killVendor |
| killVendor | Vendor is killed and has food | takeFood |
| takeFood | Agent now has food | eatFood |

**Table 4-2 Steal Food State-Transition Table**

###### 4.2.3.1.3 State-List: Seek out Food

These are the possible states if an Agent wishes to achieve the "Get Food" goal by searching for the food.

| Current State | Condition | Next State |
|---|---|---|
| seekFood | Need to find some food objects | findFood |
| findFood | Location that contains food not found | findFood |
| findFood | Find location that contains food | takeFood |

| Current State | Condition | Next State |
|---|---|---|
| takeFood | Agent now has food | eatFood |

**Table 4-3 Seek out Food State-Transition Table**

4.2.3.2 Goal: Get Health

For the goal of "Get Health" to be achieved there are three major possibilities: find a health pack, buy a health pack or steal a health pack. Once the state **repairHealth** is reached the goal is said to have been achieved.

4.2.3.2.1 State-List: Buy Health Pack

These are the possible states if an Agent wishes to achieve the "Get Health" goal by purchasing the health pack.

| Current State | Condition | Next State |
|---|---|---|
| buyHealth | Need to find somewhere to buy health pack from | findVendor |
| findVendor | Found place to buy health pack | buyFromVendor |
| buyFromVendor | Sufficient Money | completeSale |
| buyFromVendor | Insufficient Money and Not Hostile | findVendor |
| buyFromVendor | Insufficient Money and Hostile | stealHealth |
| completeSale | Sale Completed Successfully | takeHealth |
| takeHealth | Agent now has health pack | repairHealth |

**Table 4-4 Buy Health Pack State-Transition Table**

4.2.3.2.2 State-List: Steal Health Pack

These are the possible states if an Agent wishes to achieve the "Get Health" goal by stealing the health pack.

| Current State | Condition | Next State |
|---|---|---|
| stealHealth | Need to find somewhere to steal health pack from | findVendor |
| stealHealth | Previous State buyFromVendor | killVendor |
| findVendor | Found place to steal health pack | killVendor |
| killVendor | Vendor is killed and has health pack | takeHealth |
| takeHealth | Agent now has food | repairHealth |

**Table 4-5 Steal Health Pack State-Transition Table**

4.2.3.2.3 State-List: Seek out Health Pack

These are the possible states if an Agent wishes to achieve the "Get Health" goal by searching for the health pack.

| Current State | Condition | Next State |
|---|---|---|
| seekHealth | Need to find some health pack objects | findHealth |
| findHealth | Location that contains health pack not found | findHealth |
| findHealth | Find location that contains health pack | takeHealth |
| takeHealth | Agent now has health pack | repairHealth |

**Table 4-6 Seek out Health Pack State-Transition Table**

#### 4.2.3.3 Goal: Do Explore

For the goal of "Do Explore" to be achieved there is really only one major possibility: to explore the surrounding area. Once the state **finishExplore** is reached the goal is said to have been achieved.

##### 4.2.3.3.1 State-List: Explore Surrounding Area

These are the possible states if an Agent wishes to achieve the "Do Explore" goal by adventuring around the surrounding area. Note that the defend state is part of the Combat module.

| Current State | Condition | Next State |
|---|---|---|
| startExplore | Want to explore the surrounding area | exploreArea |
| exploreArea | Timer not expired and not being attacked | exploreArea |
| exploreArea | Timer expired | finishExplore |
| exploreArea | Timer not expired and being attacked | defend |

**Table 4-7 Do Explore State-Transition Table**

#### 4.2.3.4 Goal: Do Patrol

For the goal of "Do Patrol" to be achieved there is really only one major possibility: to patrol a certain area. Once the state **finishPatrol** is reached the goal is said to have been achieved.

##### 4.2.3.4.1 State-List: Patrol Certain Area

These are the possible states if an Agent wishes to achieve the "Do Patrol" goal by patrolling around the surrounding area. Note that the attack state is part of the Combat module.

| Current State | Condition | Next State |
|---|---|---|
| startPatrol | Want to patrol a certain area | patrolArea |
| patrolArea | Timer not expired and not being attacked | patrolArea |
| patrolArea | Timer not expired and being attacked | attack |
| patrolArea | Timer expired | finishPatrol |

**Table 4-8 Do Patrol State-Transition Table**

#### 4.2.3.5 Goal: Go Home

For the goal of “Go Home” to be achieved, there is really only one major possibility: to find and return to place of origin. Once the state **arriveHome** is reached the goal is said to have been achieved.

##### 4.2.3.5.1 State-List: Search for Home

These are the possible states if an Agent wishes to achieve the “Go Home” goal by searching for the point of origin (or home or creation point) of an agent.

| Current State | Condition | Next State |
|---|---|---|
| searchHome | Want to find home or origin | findHome |
| findHome | Home not found | findHome |
| findHome | Home found | arriveHome |

**Table 4-9 Go Home State-Transition Table**

## 4.3 iAICombat

### 4.3.1 Purpose

This module is designed to accommodate all combat based actions for an AI agent.

### 4.3.2 Core Functions

The core function implemented in this module is a Bayesian Network based strike predication algorithm.

#### 4.3.2.1 Strike Prediction

In many AI systems, AI agents do not seem to avoid attacks; this leads to a perception of "dumb" agents. The Strike Prediction class will consist of a Bayesian Network whereby an opponents next move can be predicted, allowing for evasive action to be taken.

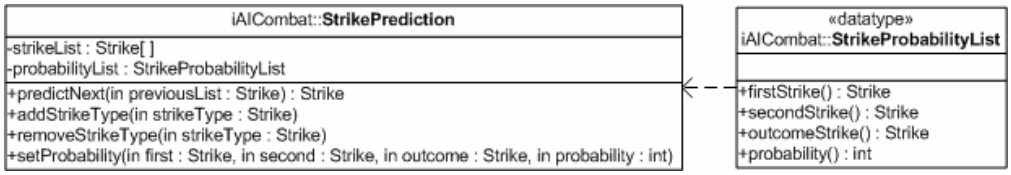


**Figure 4-5 Strike Prediction Class**

### 4.3.3 Goal-State Lists

#### 4.3.3.1 Goal: Attack

For the goal of "Attack" to be achieved there are two major possibilities: kill the opponent or wound the opponent. Once the state **attackComplete** is reached the goal is said to have been achieved.

##### 4.3.3.1.1 State-List: Kill Opponent

These are the possible states if an Agent wishes to achieve the "Attack" goal by killing the opponent.

| Current State | Condition | Next State |
|---|---|---|
| killOpponent | Need to be in range of opponent to attack | locateOpponent |
| locateOpponent | In range of opponent to attack | attackOpponent |
| attackOpponent | Opponent is not dead | attackOpponent |
| attackOpponent | Opponent is dead | attackComplete |

**Table 4-10 Kill Opponent State-Transition Table**

##### 4.3.3.1.2 State-List: Wound Opponent

These are the possible states if an Agent wishes to achieve the "Attack" goal by wounding the opponent.

| Current State | Condition | Next State |
|---|---|---|
| woundOpponent | Need to be in range of opponent to attack | locateOpponent |
| locateOpponent | In range of opponent to attack | attackOpponent |
| attackOpponent | Opponent is wounded to sufficient level | checkHostile |
| checkHostile | Opponent is still hostile | attackOpponent |
| checkHostile | Opponent is not hostile | attackComplete |

**Table 4-11 Wound Opponent State-Transition Table**

#### 4.3.3.2 Goal: Defend

For the goal of "Defend" to be achieved there are two major possibilities: defending self or fleeing. Once the state **agentSafe** is reached the goal is said to have been achieved.

##### 4.3.3.2.1 State-List: Defend Self

These are the possible states if an Agent wishes to achieve the "Defend" goal by defending itself from an opponent.

| Current State | Condition | Next State |
|---|---|---|
| beingAttacked | Opponent is hostile | attackOpponent |
| attackOpponent | Opponent is wounded to sufficient level | checkHostile |
| checkHostile | Opponent is still hostile | attackOpponent |
| checkHostile | Opponent is still hostile & high threat | fleeOpponent |
| checkHostile | Opponent is not hostile | agentSafe |

**Table 4-12 Defend Self State-Transition Table**

##### 4.3.3.2.2 State-List: Flee

These are the possible states if an Agent wishes to achieve the "Defend" goal by fleeing.

| Current State | Condition | Next State |
|---|---|---|
| fleeOpponent | Opponent is hostile | leaveArea |
| beingAttacked | Opponent is hostile | leaveArea |
| leaveArea | Run in opposite direction of opponent | leaveArea |
| leaveArea | Opponent out of range | agentSafe |

**Table 4-13 Wound Opponent State-Transition Table**

## 4.4 iAIInteract

### 4.4.1 Purpose

This module is designed to allow an AI agent to interact with the game world as a normal player would.

### 4.4.2 Core Functions

This modules core functionality is based on functions common between both the AI agents and a real player character; all the functions implemented use the player namespace and thus do not require any extra base functionality.

### 4.4.3 Goal-State Lists

#### 4.4.3.1 Goal: Have Fun

For the goal of "Have Fun" to be achieved, there are really only two major possibilities: to either dance or to sing. Once the state **completeFun** is reached the goal is said to have been achieved.

##### 4.4.3.1.1 State-List: Dance

These are the possible states if an Agent wishes to achieve the "Have Fun" goal by dancing.

| Current State | Condition | Next State |
|---|---|---|
| beginDance | Bored | doDancing |
| doDancing | Tired or Bored or being attacked or Timer expired | finishDance |
| doDancing | Having fun and Not Bored or Tired | doDancing |
| finishDance | Finishing up dancing | completeFun |

**Table 4-14 Dance State-Transition Table**

##### 4.4.3.1.2 State-List: Sing

These are the possible states if an Agent wishes to achieve the "Have Fun" goal by singing.

| Current State | Condition | Next State |
|---|---|---|
| beginSing | Bored | doSinging |
| doSinging | Tired or Bored or Being attacked or Timer expired | finishSing |
| doSinging | Having fun and Not bored or tired | doSinging |
| finishSing | Finishing up singing | completeFun |

**Table 4-15 Sing State-Transition Table**

#### 4.4.3.2 Goal: Have Talk

For the goal of "Have Talk" to be achieved, there are really only two major possibilities: to either talk to self or to talk to another agent. Once the state **completeTalk** is reached the goal is said to have been achieved.

##### 4.4.3.2.1 State-List: Talk to Self

These are the possible states if an Agent wishes to achieve the "Have Talk" goal by talking to themselves.

| Current State | Condition | Next State |
|---|---|---|
| beginChat | Bored | chatToSelf |
| chatToSelf | Tired or Bored or Being attacked or Timer expired | finishChat |
| chatToSelf | Having fun and Not bored or tired | chatToSelf |
| finishChat | Finishing up talking to self | completeTalk |

**Table 4-16 Talk to Self State-Transition Table**

##### 4.4.3.2.2 State-List: Talk to Another

These are the possible states if an Agent wishes to achieve the "Have Talk" goal by talking to another agent.

| Current State | Condition | Next State |
|---|---|---|
| beginChat | Bored and another agent nearby | chatToAnother |
| chatToAnother | Tired or Bored or Being attacked or Timer expired or Other agent left | finishChat |
| chatToAnother | Having fun and Not bored or tired | chatToAnother |
| finishChat | Finishing up talking to another agent | completeTalk |

**Table 4-17 Talk to Another State-Transition Table**

#### 4.4.3.3 Goal: Do Rest

For the goal of "Do Rest" to be achieved, there are really only two major possibilities: to either have a sleep or to sit down and wait. Once the state **completeRest** is reached the goal is said to have been achieved.

##### 4.4.3.3.1 State-List: Have Sleep

These are the possible states if an Agent wishes to achieve the "Do Rest" goal by having a sleep.

| Current State | Condition | Next State |
|---|---|---|
| getSleep | Tired | doSleep |
| doSleep | Wait until timer elapsed or being attacked or not tired | doAwake |

| Current State | Condition | Next State |
|---|---|---|
| doSleep | Timer not expired and still tired | doSleep |
| doAwake | Awaken from sleep, no longer tired | completeRest |

**Table 4-18 Have Sleep State-Transition Table**

4.4.3.3.2 State-List: Have Relax

These are the possible states if an Agent wishes to achieve the “Do Rest” goal by having a sit down and relax.

| Current State | Condition | Next State |
|---|---|---|
| getRest | Tired | doSitDown |
| doSitDown | Wait until timer elapsed or being attacked or not tired | doStandUp |
| doSitDown | Timer not expired and still tired | doRest |
| doStandUp | Stand up from resting (sitting), no longer tired | completeRest |

**Table 4-19 Have Relax State-Transition Table**

# 5 GLOSSARY

| Item | Description |
|---|---|
| AI | Artificial Intelligence - The ability of a computer or other machine to perform those activities that are normally thought to require intelligence |
| Agent | A computer controlled entity within a game environment |
| Ghosted | The process of simultaneous updating of server-side objects to any connected clients; used for client-server architectures where all objects are created on server side and displayed on clients |
| Goal | The purpose toward which an endeavor is directed, an objective; describes the AI agent's current overall objective. |
| RPG | Role-Playing Game - A game in which players assume the roles of characters and act out fantastical adventures, the outcomes of which are partially determined by chance |
| State | A condition of being in a stage or form; describes the AI agent's current action algorithmic state |

# 6 REFERENCES


- Buckland, Mat. “Programming Game AI by Example” 2005 Wordware Publishing, USA.
- Bunney, Gavin & Romano, Tom. “iAI Research” 2006 irombu.com, Brisbane.