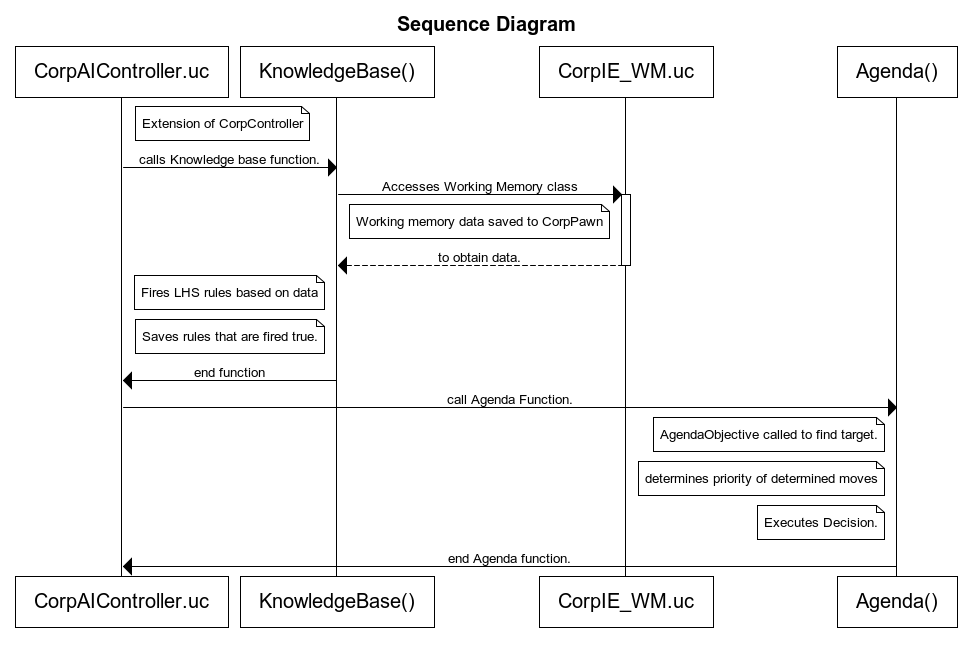
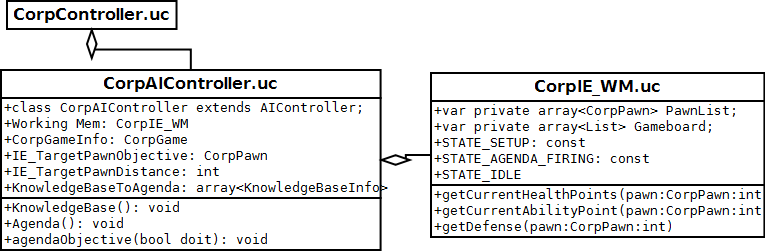
**Inference Engine - Integration Team**

**Final Documentation**

**1. Diagram**





**2. Explanation**

The game runs using a state machine. When the program takes control of a pawn, the controller class is being used. When the NPC pawn is being controlled, the CorpAIController is being used.

The Inference Engine is integrated using 2 classes. The CorpIE\_WM class which is the entire working memory portion of the Inference engine. The Working memory will take a snapshot of the game field and put the relevant data in variables for the rest of the Inference Engine to access. When it is needed, the Inference engine could call the “Pawn” and use a call such as “thispawn.health” to determine the health of the current pawn.

The CorpAIController is the second class. Everything that is needed for the AI to make a decision and execute is already integrated in this class, so it made sense to integrate our decision making engine into this class. It’s here that we put the Knowledge Base and Agenda.

The Knowledge base then calls the Pawn data structure that is held in the working memory and calls variables to access the specific data needed, such as health or ability points.

At that point, the knowledge base sets a number of IF-THEN rules based on that data and stores the resulting decisions in an array.

The Agenda accesses the array saved by the knowledge base. There is a function, agendaObjective() which find the target in which the currently possessed pawn will use an ability on, attack, or walk/run towards or away from. Once that is determined the agenda will use an algorithm to determine the priority of the rules that was determined to be true. Once a rules has been determined to be the best decision to make, the execution of that decision is done.

**3. IF-THEN rules**

RULE # ACTION1 ACTION2 DESCRIPTION

1 Genu - If unit health is less than 10% of its max AND unit has enough action points...

2 SelfMot - If unit has enough action points...

3 Move Genu If unit is melee AND is adjacent to target AND has less than 10% health AND has enough action points...

4 Genu - If unit is melee AND is adjacent to target AND has less than 10% health AND has enough action points...

5 Rush - If unit is melee AND is adjacent to target AND has less than 30% health AND has enough action points AND target has less than 25% health...

6 Move - If unit is melee AND target is within 6 squares AND unit has less than 30% health AND has enough action points AND used SelfMot last...

7 Move Bash If unit is melee AND target is within 6 squares AND unit has less than 30% health AND has enough action points AND used Genu last...

8 Move - If unit is melee AND target further than 6 squares...

9 Move Attack If unit is melee AND target is within 6 squares AND unit is NOT adjacent to target...

10 Attack - If unit is melee AND is adjacent to target...

\*11 Attack - If unit is melee AND health is greater than

(target1 potential damage) \* (unit max health) AND unit health is greater than (target2 potential damage) \* (unit max health) AND unit is adjacent to target...

12 Move Bash If unit is melee AND target is within 7 squares [AND unit is NOT adjacent to target] AND unit has enough action points...

13 Bash - If unit is melee AND is adjacent to target AND has enough action points...

14 Move SelfMot If unit is melee AND unit is NOT adjacent to target AND unit has enough action points...

15 SelfMot - If unit is melee AND is adjacent to target AND has enough action points... (different amount of points than rule 13)

16 Move - If unit is ranged AND target is further than 10 squares away AND unit has more than 30% health...

17 Move Attack If unit is ranged AND target is furter than 5 squares away AND target is less than 10 squares away AND unit has more than 30% health...

18 Attack Run If unit is ranged AND target is less than 3 squares away AND unit has more than 30% health...

19 Attack - If unit is ranged AND target is further than 3 squares away AND target is within 5 squares AND unit has more than 30% health...

20 MoveStayOutOfRange - If unit is ranged AND target is further than 5 squares AND unit has less than 30% health...

21 Attack Run If unit is ranged AND target is within 3 squares AND unit has less than 30% health...

22 Attack Run If unit is ranged AND target is fuerther than 3 squares away AND target is within 5 squares AND unit has less than 30% health...

23 MoveStayOutOfRange Genu If unit is ranged AND target is further than 5 squares away AND unit has less than 30% health AND has enough action points...

24 Genu Run If unit is ranged AND target is within 3 squares AND unit has less than 30% health AND has enough action points...

25 Genu Run If unit is ranged AND target is further than 3 squares away AND target is within 5 squares AND unit has less than 30% health AND has

enough action points...

26 Move SelfMot If unit is ranged AND target is further than 10 squares away AND unit has more than 30% health AND has enough action points...

27 Move SelfMot If unit is ranged AND target is further than 5 squares away AND target is within 10 squares AND unit has more than 30% health AND has enough action points...

28 SelfMot Run If unit is ranged AND target is within 3 squares AND unit has more than 30% health AND has enough action points...

29 SelfMot - If unit is ranged AND target is further than 3 squares away AND target is within 5 squares AND unit has more than 30% health AND has enough action points...

30 MoveStayOutOfRangeSelfMot If unit is ranged AND target is further than 5 squares away AND unit has less than 30% health AND has enough action points...

31 SelfMot Run If unit is ranged AND target is within 3 squares AND unit has less than 30% health AND has enough action points...

32 SelfMot Run If unit is ranged AND target is further than 3 squares away AND target is within 5 squares AND unit has less than 30% health AND has

enough action points...

33 Move FocusShot If unit is ranged AND target is further than 5 squares AND target is within 10 squares AND unit has more than 30% health AND has enough action points...

34 FocusShot Run If unit is ranged AND target is within 3 squares AND unit has more than 30% health AND has enough action points...

35 FocusShot - If unit is ranged AND target is further than 3 squares away AND target is within 5 squares AND unit has more than 30% health AND has enough action points...

36 FocusShot Run If unit is ranged AND target is within 3 squares AND unit has less than 30% health AND has enough action points...

37 FocusShot Run If unit is ranged AND target is further than 3 squares away AND target is within 5 squares AND unit has less than 30% health AND has enough action points...

**4. Data structures.**

**Working Memory**

//The Working Memory created their own class.

// Their class uses the functions previously written in the game to store

// variables in a pawn data structure.

.

var private CorpPawn currentEnemy; // A Target

var private byte IE\_state; // Flag used for determining the state

var private array<CorpPawn> PawnList; // Array of Pawns on the board.

var private array<List> Gameboard; // snap shot of the game board.

/\*

The CorpPawn data structure possesses all the key stats such as health, location, job,

ability points, range, attack modifier, defence modifier.

\*/

**Knowledge Base**

Class scope variables.

var string action1; // Used for holding the first action in the decision

var string action2; // Used for holding the second action in the decision

var CorpPawn target; // The target of the action we decide.

Local scope variables.

/\*

// these are the variables gathered from the working memory in use as

// test variables for the LHS of the IF-THEN rules.

\*/

local KnowledgeBaseInfo knowledge;

local KnowledgeBaseinfo ktest;

local CorpPawn TempPawn;

local array<int> EnemyPawnDist;

local string unitID;

local float unitHealth,unitActionPoints,enemyUnit1Health,enemyUnit2Health;

local float unitPotentialDamage,enemyUnitPotentialDamage1,

enemyUnitPotentialDamage2;

local string unitLastMove; // set after agenda

WorkingMem = new () class'CorpIE\_WM';

unitHealth = WorkingMem.getCurrentHealthPoints(ThisPawn);

unitActionPoints = WorkingMem.getCurrentAbilityPoints(ThisPawn);

unitID = ThisPawn.CharacterJob;

**Agenda**

Class scope variables.

var bool FriendTarget;

var bool EnemyTarget;

var bool KnowledgeBaseTarget;

var string RHS; // The decision

var bool Pri1, Pri2, Pri3, Pri4, Pri5, Pri6; // priority flags.

var string RHSattack;

//used as flag for each of the RHS rules that are determined true.

var bool GenuAfter;

var bool MotivateAfter;

var bool RushAfter;

var bool BashorFocusedAfter;

var bool Attack;

var bool Move;

var bool Runaway;

var bool GenuRun;

var bool GenuMove;

var bool Genu;

var bool NormalRun;

var bool AttackMove;

var bool SelfMot;

var bool SelfMotMove;

var bool SelfMotRun;

var bool BashFocus;

var bool BashFocusRun;

var bool BashFocusMove;

var bool Rush;

var bool RushMove;

var bool RushRun;

var bool MoveBash;

var bool MoveGenu;

var bool MoveAttack;

var bool MoveSelfMot;

var bool AttackRun;\*/

// agenda also has a function agendaObjective which finds the target

// which the AI pawn will use once it knows what action to take.

local CorpPawn PossiblePawn, TempPawn;

local int PossiblePawnDist, TempPawnDist;

local CorpPawn\_BlueCollar\_Ranged RangedPawn;

local int CorpOwner;

local int friend;

local array<CorpPawn> PossiblePawns, PossibleTempPawnsOne, PossibleTempPawnsTwo;

local int friendsInArea;

local array<CorpCell> PossibleCells, EnemyPositions, TempCells;

local CorpCell TempCell;

local int i;

local int NumberOfCells;

local int RandomCell;

// Agenda Fuction uses:

local KnowledgeBaseInfo choice, choiceTemp;

local int num\_choices;

local int chooser;

local array<string> ActionHolder;

local string actions;

local int i;

local array<KnowledgeBaseInfo> P1, P2, P3, P4, P5, P6;

**5. How to compile and run Inference Engine.**

The Inference Engine is integrated within the Office Tactics game. There is no special instructions to have the inference engine compile and run. Office Tactics can be accessed from the repository.