Supreme Commander team

Technical Design Document

Project: RTS
Version: 1.0.0
Last update: 25/03/2024
Author: DEVINE Vincent, LISE Omaya

Sommaire

The game

- Type : RTS (Real-Time Strategy)
- One player
- Focus on **artificial intelligence**
- Free to use and mix previously seen AI architecture

Victory condition:
The opposite party's influence on the map reaches a low threshold.
To reach this goal the AI will need to **create, manage and upgrade its units and buildings**.

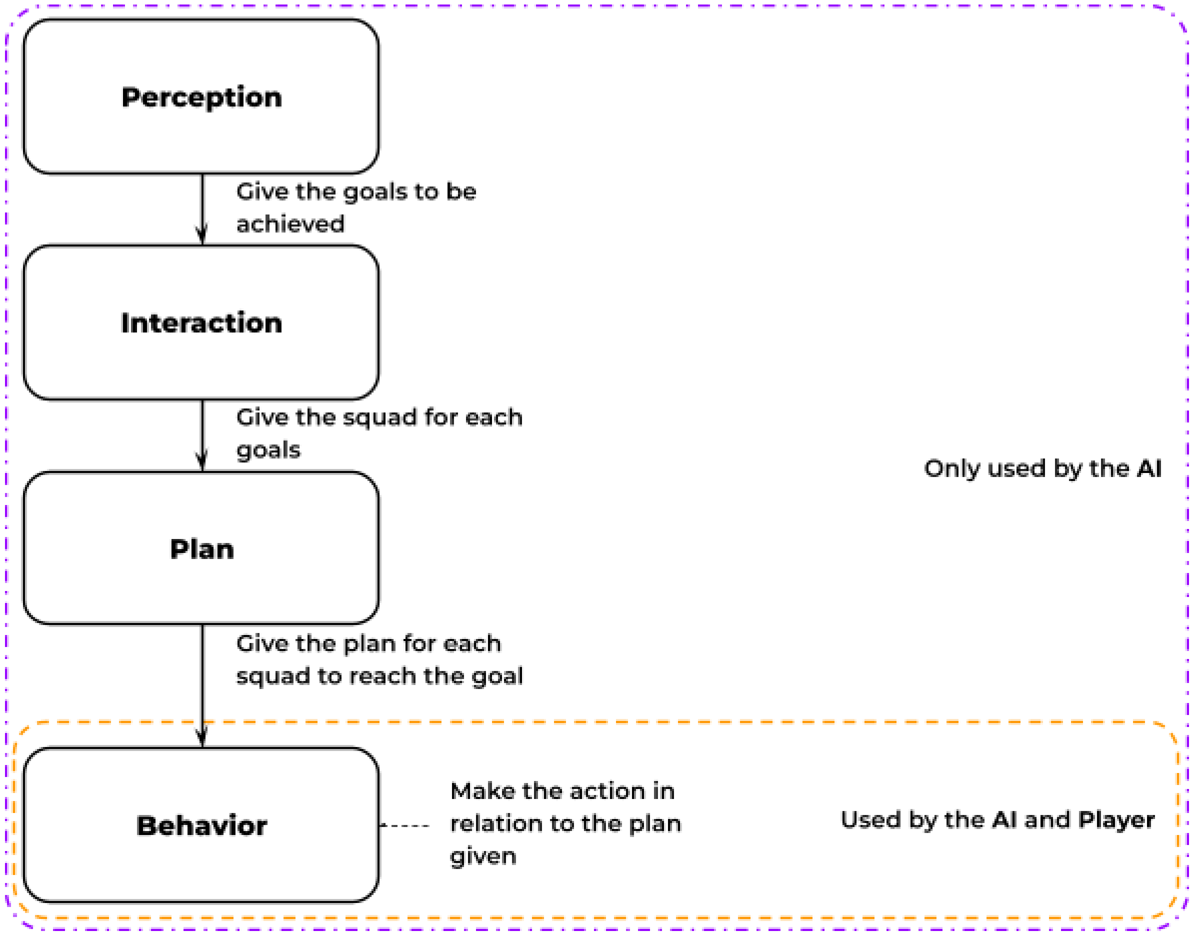
How to upgrade:
The number of resource types is limited to one.
The player/AI needs to **take control** of the **ores**. Each ore continuously **adds iron** to the **player's control**.
With iron, the player/AI can **create units or factories**.
Unit is used to **increase the influence** of the player.

Technological context

This project arrives with some restrictions:
- Engine: Unity 2022.3.9f1
- Versioning: Git (GitLab)
- Text Editor: Visual Studio 2022
- Documentation: Google suite (Google doc/sheet/slide)
- Communication: Teams, Discord
- Management: Trello

Technical choices

Technical choices for the AI in our RTS project


Perception
Give the goals to be achieved
Interaction
Give the squad for each goals
Plan
Give the plan for each squad to reach the goal
Only used by the AI
Behavior
Make the action in relation to the plan given
Used by the AI and Player

The architecture is linear. Some data can be updated or overridden by lower level systems (such as the map of influence) but the information will mostly be **flowing down**.
No utility system is formally used since no weight is calculated but there is a **priority system** which is close enough.

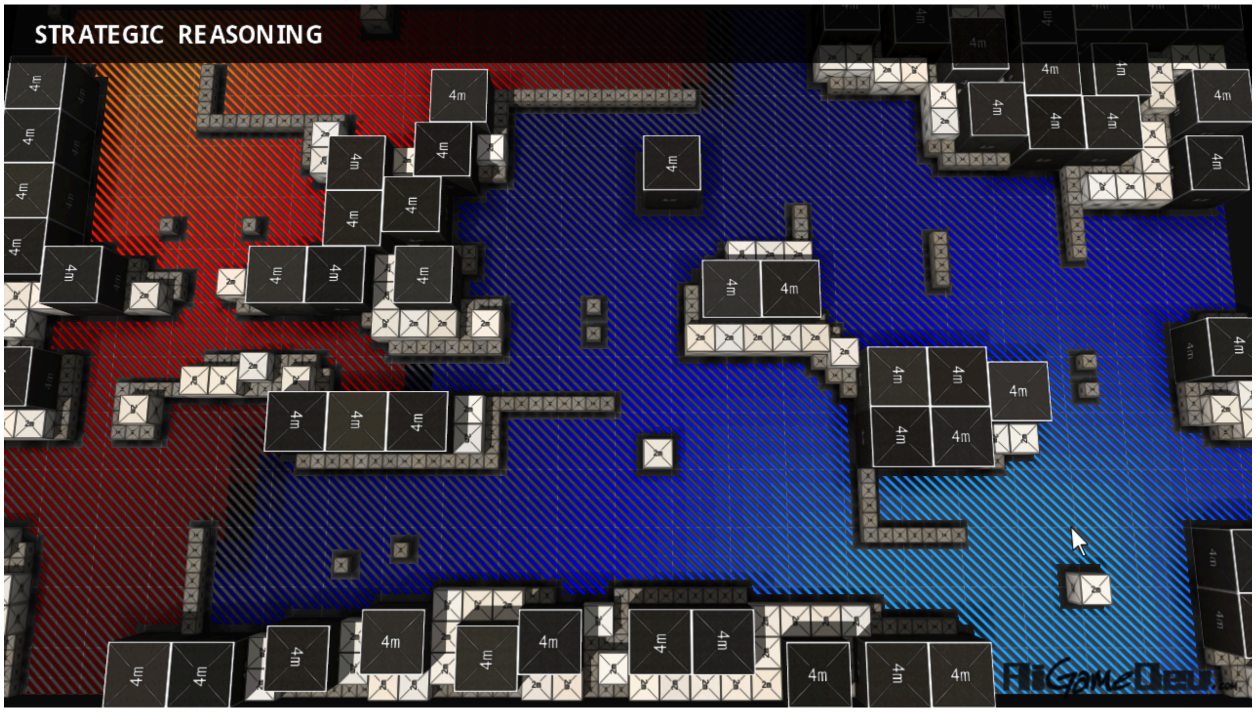
Architecture choices

- Preventing conflicts between systems (since the order always comes from the top)
- Easier to debug and to iterate from
- Less workload and less complicated AI
- Sending useless data for some systems if it is necessary for the next

Context used by AI

Influence map

The map will have a tile distribution. Each tile has a numeric value either negative or positive. This score is calculated from the power of each unit in the tile. The enemy's influence is negative and the ally's one is positive.



Resource map

The map also has a **tile distribution**, similar to the map of influence. Its score is calculated by adding the power of each ore in the tile.

Allies

The number of allies present on the map:

- Available / Not available / In formation
- Type
- Power

Ressources

The number of iron the player has.

Goals

Attack
This goal can be chosen if the AI has enough power of allies compared to the player.
The unit will be in *aggressive* stance

Defense
This goal can be chosen if the AI used ressources but didn't protect them *(ex: if the ressource power is 2, the defense needs to be 2)*. It will also call when a protected resource was attacked by power up to the protection.
The unit will be in a *defensive* stance.

Exploration
This goal can be chosen if the AI needs to find the player in the map or more ore to upgrade. If the squad finds ore, it will capture the ore.
The unit will be in a *neutral* stance.

Upgrade
This goal can be chosen if the AI needs to grow this unit power. It will consist of creating more units or factories.

Architecture choices
- System is goal-oriented.
- Only 4 goals defined to allow more particular behavior with stance from unit to unit.
- Goals are limited to two to prevent AI from being too strong.
- Split from the interactive system to be cleaner and easier to iterate from.

Perception

Objective

Give one or two goals maximum to be achieved

Input

- *Context*

Functionality

Compare player and Ai situation (*influence map*, *allies*)

- (1) If the player is as strong or stronger, the AI want to *upgrade*
- (2) If the player is weaker, the AI wants to *attack*
- (3) If the player attacks, all the units go to *defend* the base.
- (4) If the data about the player is too-old or inexistent, the AI wants to *explore* to update data

Taking account of the **world** (*resource map*)

- (5) If the AI can take control of a resource, the AI wants to *explore*
- (6) If the AI has control of a resource without protection, the AI wants to *defend* it.

Taking account of the **AI** (*resources*)

- (7) If the AI can make a factory, the AI want to *upgrade*

Prioritizing the choice of goals

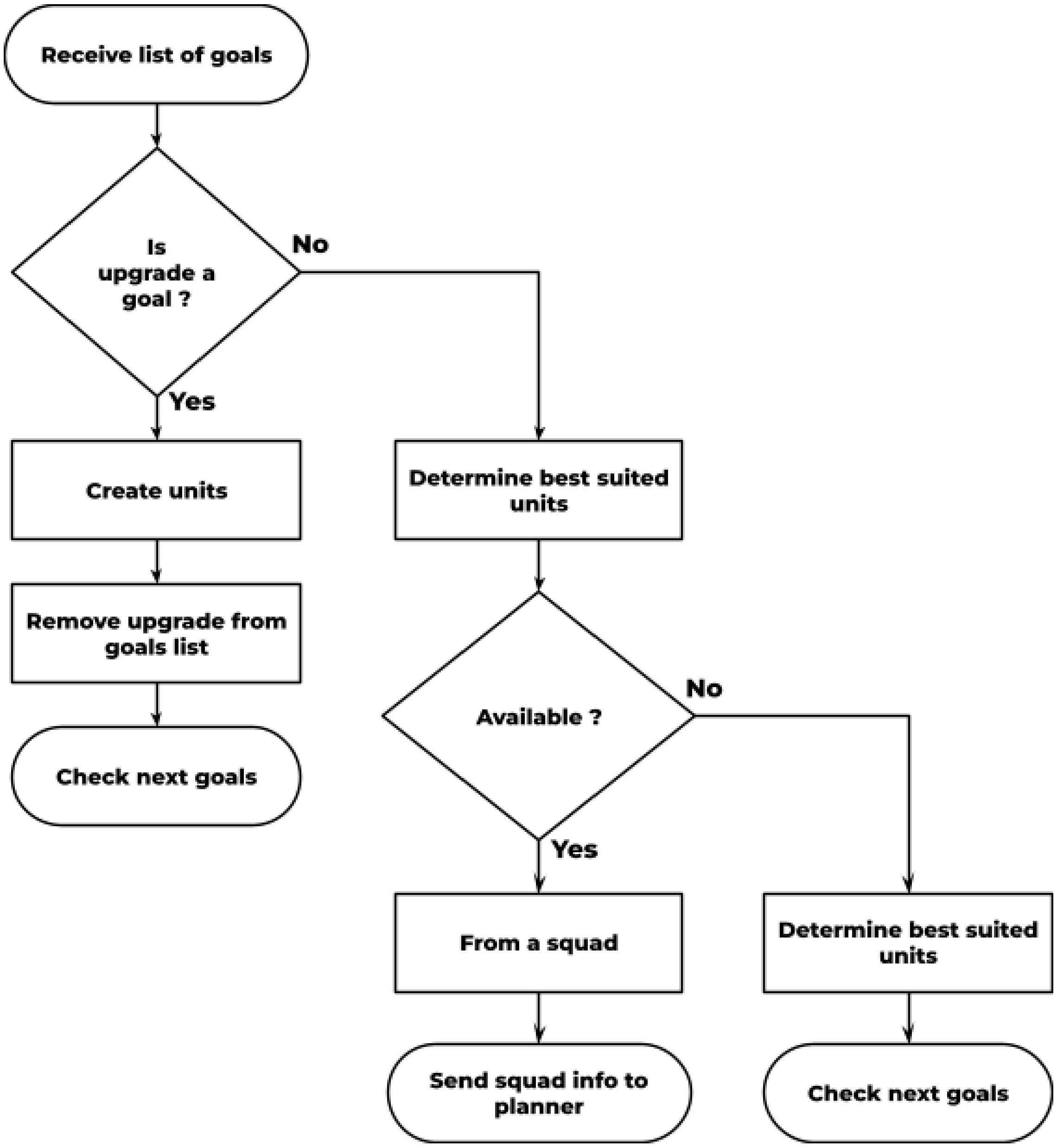
There can only be 2 goals simultaneously. The only exception is the priority 0.

Priority	ID
0 - all unit will stop what they doing to make this	3
1 - more important	1
2	2
3	5
4	6
5	4
6 - less important	7

Architecture choices

- The perception **analyze all the data of the map** (influence and resources map) then deduce which goals should be selected
- Used on a global scale but close to the **Unreal AI perception system** (received stimuli before sending signals to other part of our AI brain)
- Split from the perception system to be cleaner and easier to iterate from.
- As it rounds up all of the map information it can be a powerful debug tool.

Interaction



Objective
Create squads able to do the series of actions needed to reach the predetermined goals.

Input

- *Context*
- *Goals*

Functionality

Choose the best type of unit available for each goal. Fix the size of the squad. Get available units / create units and form a squad.

List of unit speciality :

- *Attack*: heavy unit
- *Defense*: all unit/heavy unit
- *Exploration*: light unit
- *Upgrade*: no unit

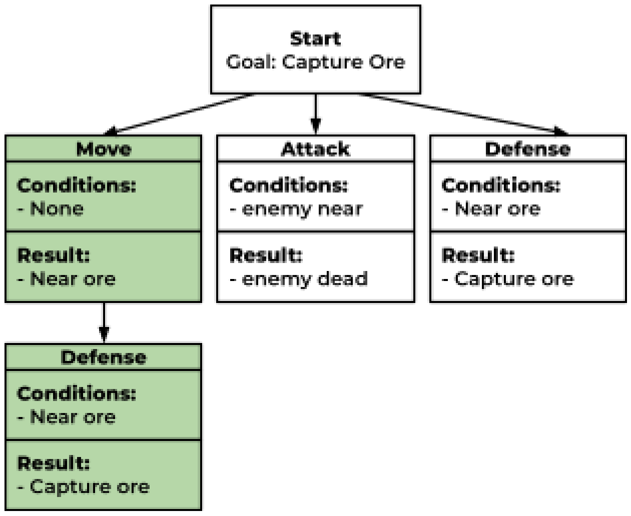
Architecture choices

- The interaction system checks **available units or creates units to form squads** able to reach the goals.
- It contains a list of the most optimized units for each goal.
- It also defines how big a squad should be.
- This architecture allows the planner to quickly find the most optimized solution to the problem.
- It won't get stuck if there is no solution to the goal but rather create units and/or buildings.

Plan

Objective
Chose the sequence of actions to be taken to achieve the objective

Design
GOAP Planner



Input
- *Context*
- *Goals*
- Squad

Predetermined actions for each goals
- *Attack*: *Move*, *Shoot*
- *Defense*: *Move*, *Protect*
- *Exploration*: *Move*, *Explore*, *Capture Ore*

Functionality
- Each goal has a predetermined list of actions.
- With the current state of the game, it checks if it can make this list.
- If the pre-create list is not feasible, it browse all the actions, check their conditions, their results and start again.

Architecture choices
- Although these sequences right now are short, It is easy to complexify the AI.
- Good to plan long term plans.
- Complex enough for the project’s objectives.
- Enumerating all the actions.
- Making reasonable predictions about the outcome of actions.

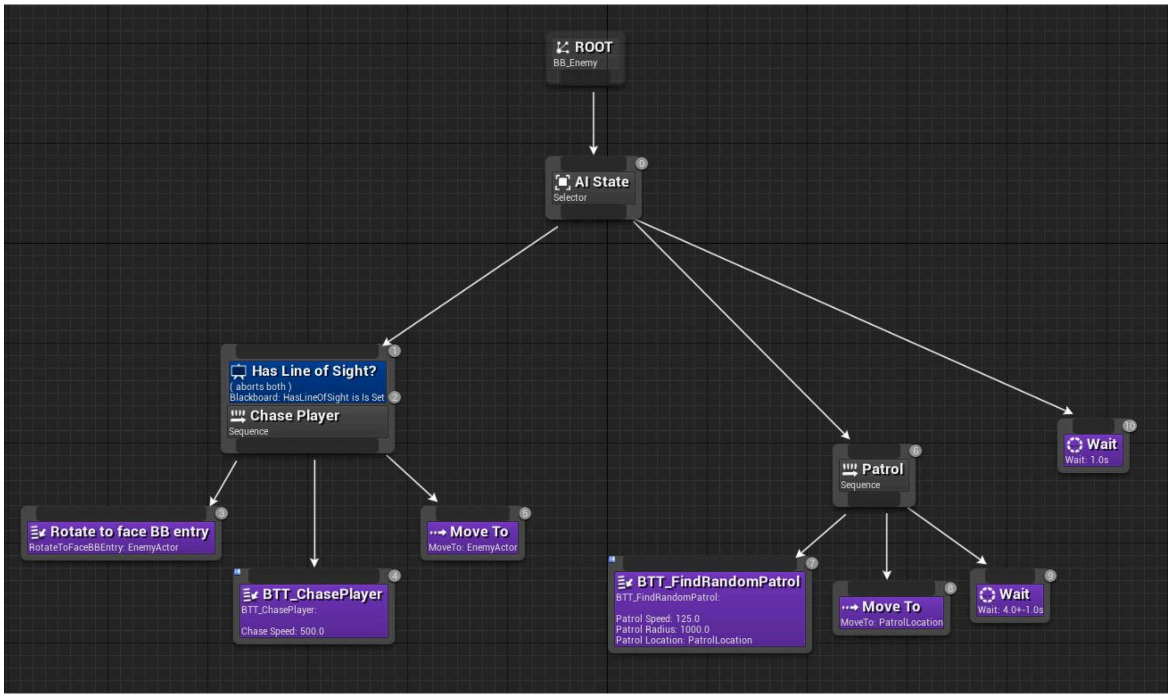
Behavior

Objective

Control the unit to realize the action. The realization is influenced by the unit stance.

Design

A Behavior Tree architecture used by both the AI and the player to control.



Architecture choices

- The behavior tree directly controls each unit (influenced by the squad behavior and the stance).
- Prevent the “spaghetti” code of a FSM (Finite-State Machine).
- More interesting from a learning perspective.
- Easy to iterate from once it is implemented.
- Harder than a FSM.

Stance

Stances have an impact on the interpretation of the order received from the planner.

Defensive

The unit attacks the enemy's unit if it is close to its squad. It chases it unless it goes too far. If so it returns to the original position.

Aggressive

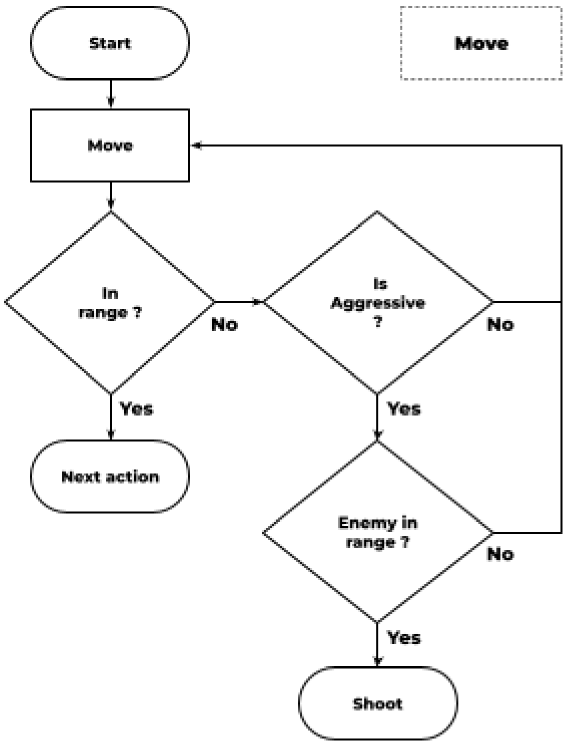
The unit attacks the enemy's unit if it is close to its squad and chases it until either die.

Neutral

It only does the action ordered without taking into account other units, allies or enemies.

Actions

Move



Move freely to a designed position.
If the unit is in a squad, it moves in formation defined by a fathom unit.
Utilization of the Unity NavMesh.

Input :

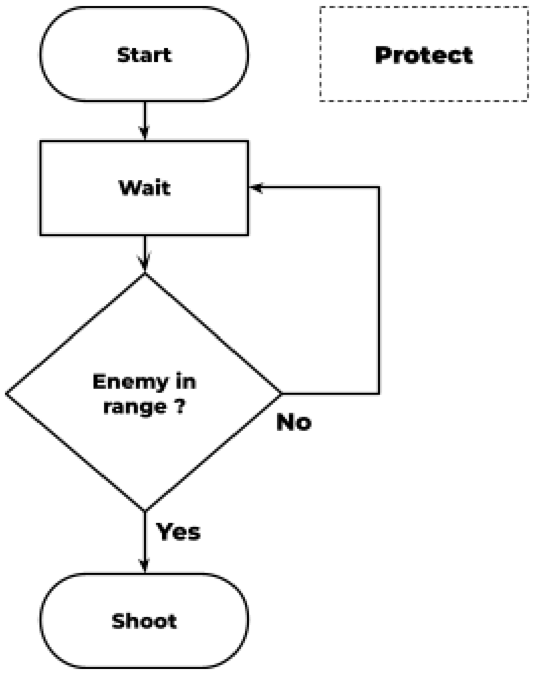
- Target (Vector3)
- Squad (List<Unit>)
- *Stance*

Protect

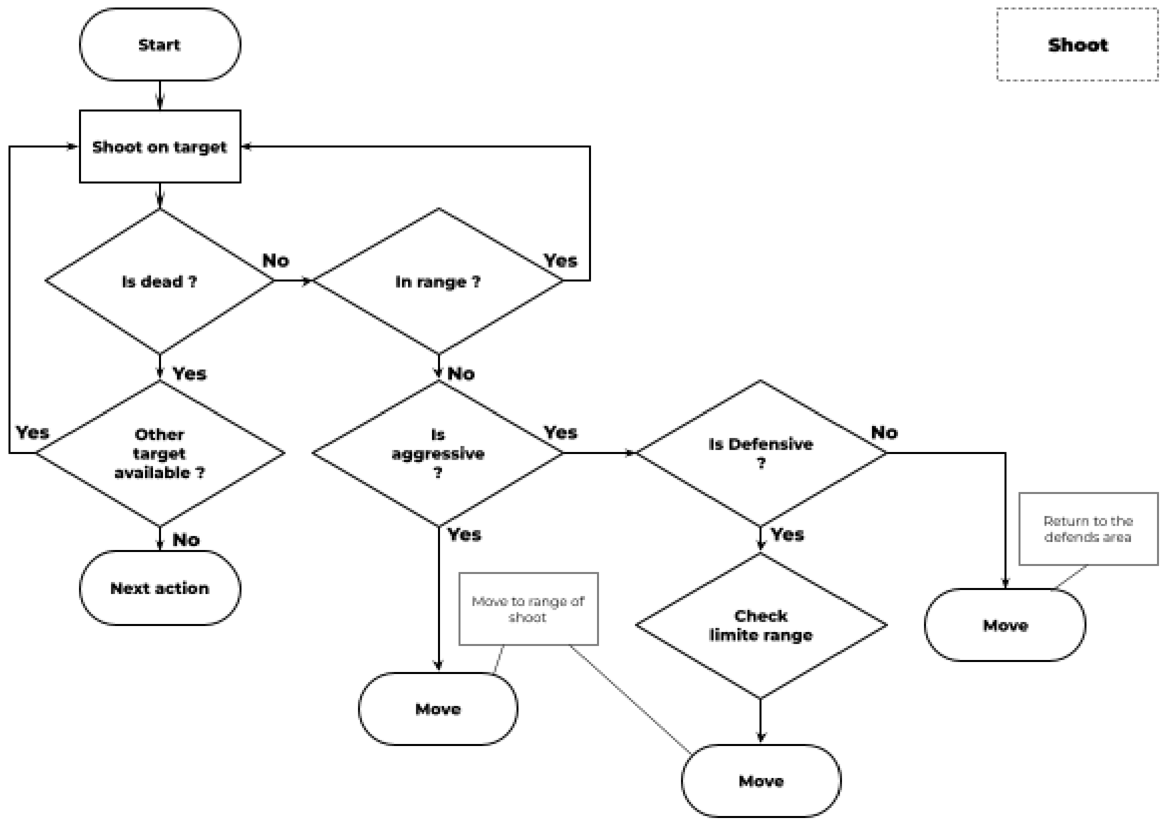
The unit waited in the defense area selectionned in a *defensive* stance.

Input:

- Defense area (Vector3)
- Squad (List<Unit>)



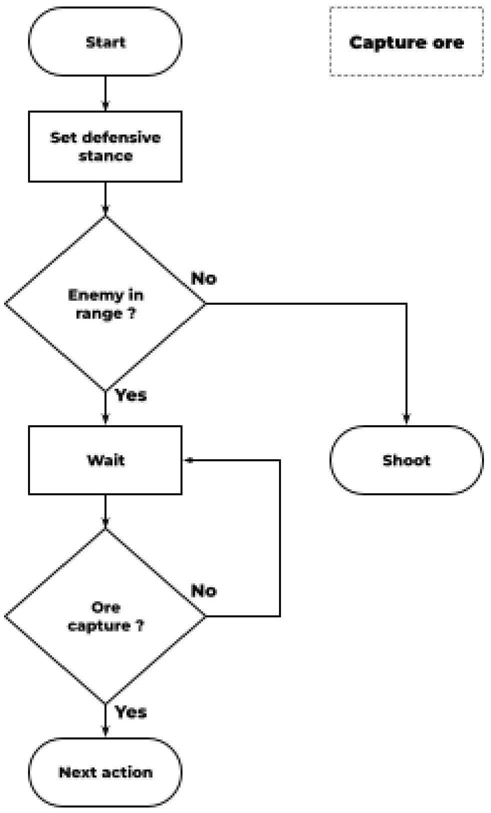
Shoot



If there is a target, The unit shoots the target. Or, the unit shoots the nearest opponent unit. And if there are no opponent units, the nearest opponent factory.

Input:

- Attack zone (Vector3)
- Target (Unit), can be null



Capture ore

If the opponent unit was in the ore area, the squad would attack this in a *defensive* stance. When the ore area is clean, the squad will wait until the ore is captured.

Input:

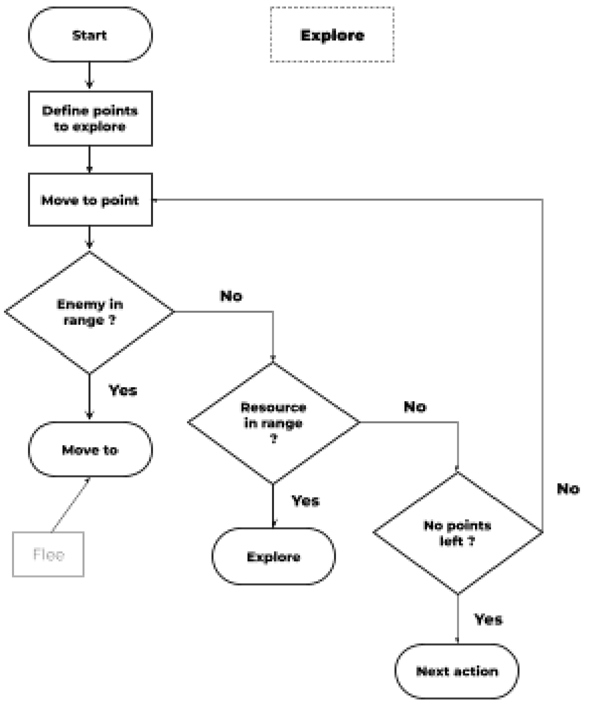
- Capture zone (Vector3)
- Squad (List<Unit>)

<u>Explore</u>

Move through a selection of points in a *neutral* stance. Flee if an enemy is found and capture if a resource is located.

Input:

- Explore zone (Vector3)
- Squad (List<Unit>)



Code convention

Function / Method and variables

Function is capitalized, not variable name.

```
float deltaTime;
private void MethodsName() { }
```

Enums

Enums components in all-cap.

```
public enum NODESTATE
{
	RUNNING,
	SUCCESS,
	FAILED
}
```

Const

In all-cap for a variable, not when given as parameter

```
const int ATTRIBUTE_IN_CONST;
void ClassName::Function(const int p_param) { }
```

Braces

Braces even if there is only one line of code.

```
if(condition)
{
   return;
}
```

Parameters

p_ in front of parameters in function definition.

```
private void Function(const unsigned int p_param) { }
```

Split up large functions into logical sub-functions (15 - 20 lines max approximately).

Backlog

Constraint

Number of days allocated to the project: 7.5 (= 15 half-days)

Number of people on the project: 2

ID	Task name	Duration (day)	Priority	Complexity
0	Researches	1	2	1
1	Conception of the AI global architecture	1	2	1
2	Making TDD	2	1.5	1
3	Map of influence	0.5	2	1.5
4	Map of resources	0.5	2	1.5
5	AI Perception	1	3	1.5
6	AI Interaction	1	3	1.5
7	GOAP Planner	2	2	4
8	Behavior Tree	4	3	**5**
9	Systems fusion	0.5	4	2
10	Test	0.5	2	1
11	Debugging	2	2	3

Bibliography

Perception / Interaction:

- Unreal AI perception example
- Definition of a Utility System

Plan:

- Definition and example of GOAP
- Definition of HTN planning

Behavior :

- Definition of behavior tree
- Tutorial for custom behavior tree on Unity
- Definition of FSM

Credit

3rd year ISART DIGITAL Paris project
Project made by

- DEVINE Vincent
- LISE Omaya

Special Thanks

- ROMETZ Baptiste
- WOLF Florian

Project Start: 03/18/2024
Project End: 04/17/2024