Decision-making System for Cooperative NPC in a Shooter Game

Rémi Giner[a], Noé Mouchel[a], Benjamin Marin[a], Mael Ahmad Addoum[a], and Florian Wolf[a]

[a]Isart Digital, Departement of Game AI, Paris, France

ABSTRACT

Typical shooter games are usually focused on the player characters and their actions against AI-controlled opponents. In this project, we attempted to explore how a shooter game could be technically influenced if now the player played the role of an assault leader, placed within a squad filled with supporting NPC (Non-Player Character) soldiers.

In our project, the NPCs should be able to perform three different main actions:

1. Protect their leader
2. Heal the other members of the squad
3. Fire back at opponents

In addition, a barrage fire action was added and can be activated by the player via a specific input. When triggered, the entire squad fires at will at a specific point on the map. Also, in order to prevent NPCs from following the player randomly, we gave them movement rules based on squad formations.

To help the player distinguish between the different NPC's capabilities, a supporting role (see Fig. 1) was given for each NPC that can be easily identified visually. Protection NPCs hold an assault shield, healing NPCs carry a backpack, while combat NPCs simply hold a gun.

Moreover, all of the NPC behaviours and transitions between actions had to be seamless. For example, if a squad member is killed during a fight, the squad should be able to rearrange its positions so that the NPCs can still perform the chosen action. The decision system is mainly based on customized utility system techniques. This allowed us to obtain reactive behavioral changes depending on the current world situation. Each desired behaviour is represented by what we called an action, and the priority of each action is evaluated by a mathematical Consideration Curve at runtime. In our implementation, an action is not only a simple call to a single method of an agent component. It is also possible to assign multiple methods from different components for the action and hence create advanced custom behaviours, such as simultaneous defense and firing (see Fig. 2).

Note that the performance depends on the number of agents populating the map and the number of actions available for each agent. In this work, we added the possibility to set the evaluation frequency rate to decrease the CPU load. This frequency can also be modified at runtime and adapted to the number of awake agents.

The few playtests have shown that the gameplay and NPC's reactions are lively, enjoyable and adapt with player's skills. Even though the primary objective was to create interesting cooperative behaviour with the player, the same decision system was successfully used to design the behaviour of the enemy squads. Furthermore, the proposed architecture is ergonomic and can be easily customized by adding, modifying, or deleting actions/behaviours using a user-friendly edition.

Keywords: Real Time Shooting Game, Squad Behaviours, Utility Systems, Decision Making

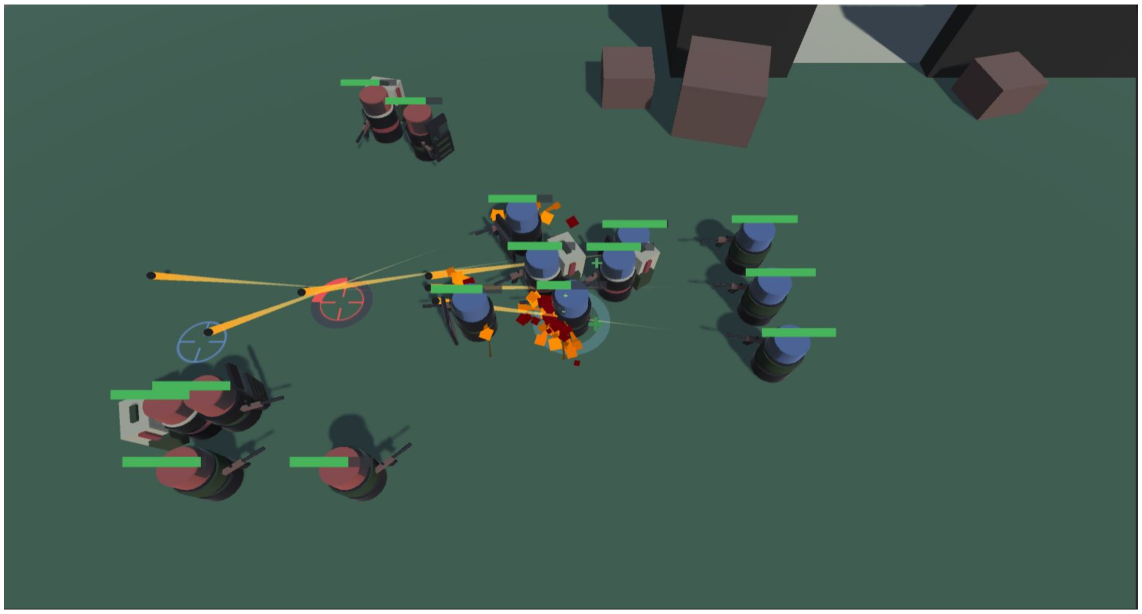
Figure 1. NPC squad members assisting the player during a fast-paced action sequence against the enemy guards.

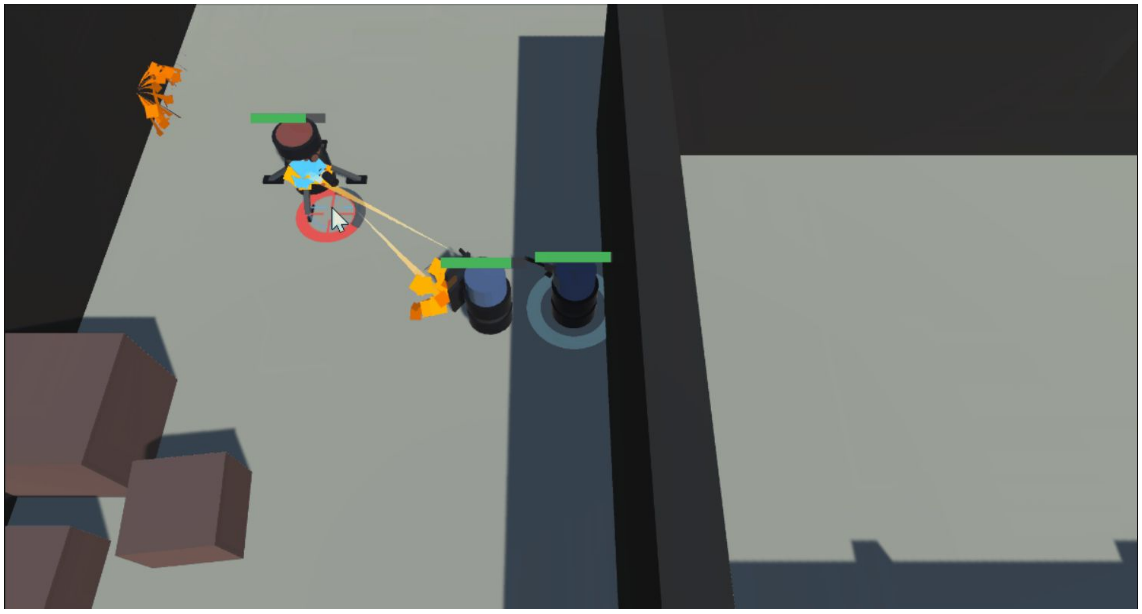
Figure 2. A NPC protecting the player and firing at an enemy at the same time.