Project Proposal For BBM486 Group 10
State Design Pattern

Ahmet Ensar Sönmez, Berkay Kısa, Ali Arda Özdemir, Berke Bayraktar, Muhammet Ali Çaki

1 Introduction

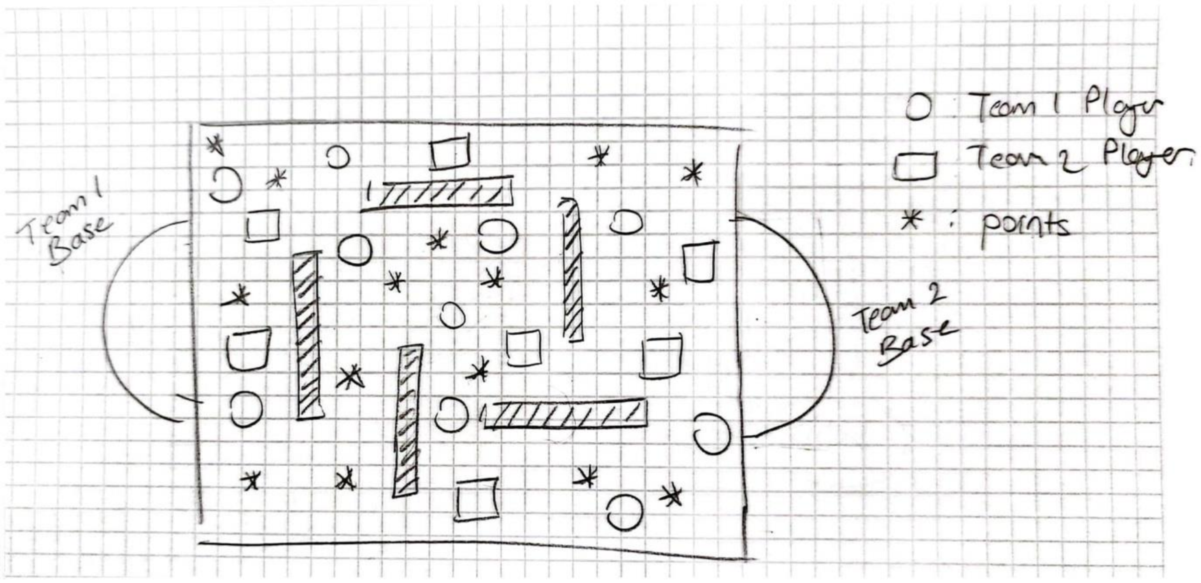
State pattern is a behavioral design pattern that shows close resemblence to the concept of finite state machines (FSMs). Finite state machines have been used in the area of game programming for a very long time, mainly for programming NPC[1] behaviour. However today's modern and complex games use more sophisticated methods such as Behaviour Trees for this task given the fact that FSMs become harder to maintain as the number of states increase, which is the case for modern game NPCs. Nonetheless, they still work great for simpler NPCs with fewer states.

2 Project

In our project, we will design and implement a simple multi-agent simulation game which makes use of NPC behaviours where these behaviours are defined by a state machine.

2.1 Simulation Setup

Before moving onto the explanation for the state machine itself we will briefly talk about the game setup.

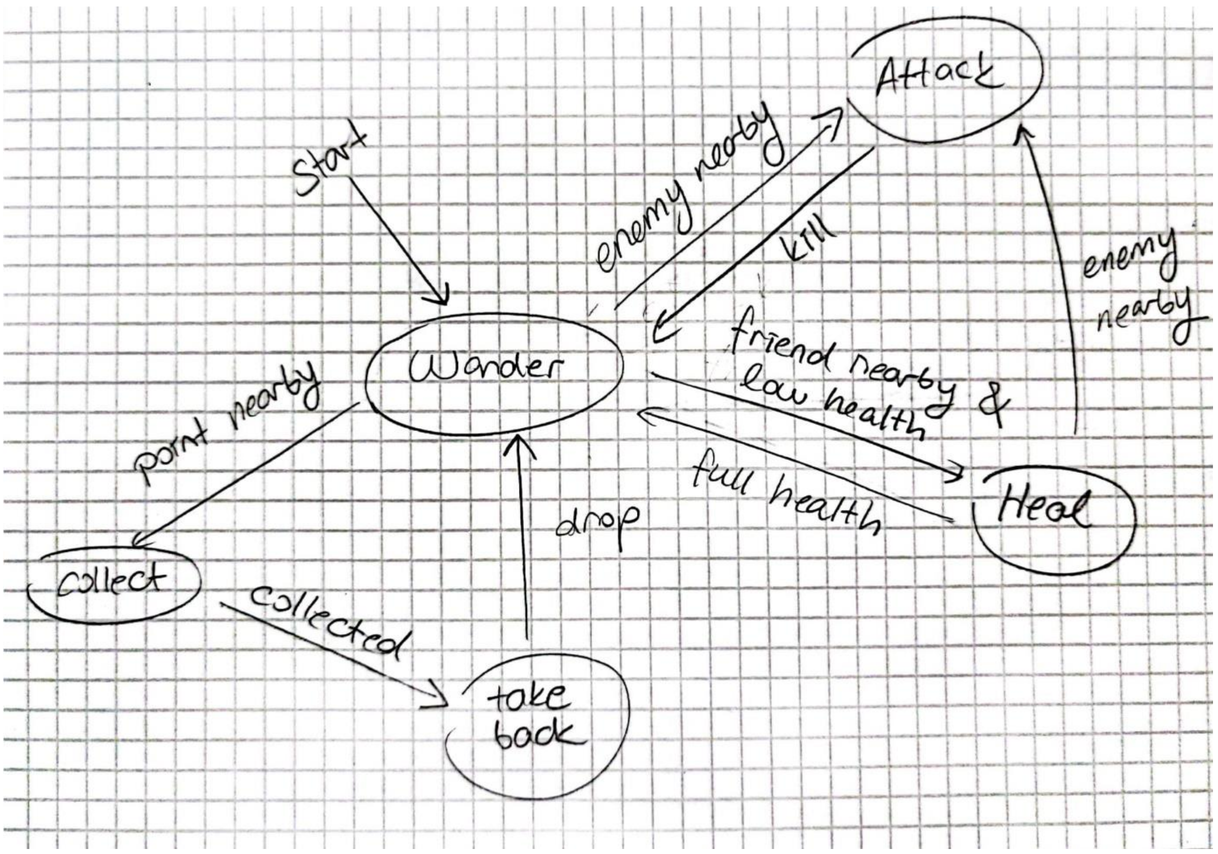


There are 2 teams, represented as circles and squares, with each team having its own "base" and "points" represented with the star (*) symbol. Goal of each team is to carry as much points as possible to their own base. At the end of the game the team with higher points is declared as winner. In this simulation, agents exhibit both inter-team and intra-team interactions such as "attacking" enemy players or "healing" ally players.

[1] A non-player character (NPC) is any character in a game that is not controlled by a player.

2.2 The State Machine

Here, we define a state machine consisting of 5 states. This choice was made so that everyone gets to implement a state. The agents' transition between states based on their proximity to other game objects such as enemy, ally or point; or based on their current objective such as carrying a point to its base.



2.3 Notes on Game Setup & States

We would like to mention that previously discussed game setup and state machine can and probably will change as we progress further into the project. For example, considering the state machine, there are many more ways to define transitions and on what conditions these transitions should occur. Similarly the game logic might be extended with additional features or actions on agents that make the simulation run more smoother. Having said these we aim not to deviate from the main idea behind the project, that is, representing an agent simulation game using state machine. Additionally, we will make sure that there are *at least 5 states* on the state machine so that everyone get to implement *at least a single state*.

3 Responsibility of Each Team Member

There are 3 major parts of this project: design and implementation of the state pattern, main body of the code which acts as the client for the state pattern and also manages the game environment and lastly design of the platform where agents reside.

For the first part, as stated earlier each team member will implement at least a single state. The remaining 2 parts can not be further broken down into smaller parts without actually proceeding with the project. So a fair division of the tasks in part 2 and 3 is not possible for the time being. However we will make sure that everyone contributes roughly equally to the project and this can be verified later on since we will use GitHub for project management where contribution of each individual can be examined.