Chapter 5

Design of the Multi Agent System for 3D Game Environments Design

5.1 Introduction

The previous chapter described the proposed approach to solve the identified issues by using multi agent technology. This chapter describes the design of the Multi Agent System to Assist 3D Game Environments Design.

5.2 High Level Design of the System

The proposed system has been decomposed in to 8 modules and the functionality of each module is described with interconnection between modules.

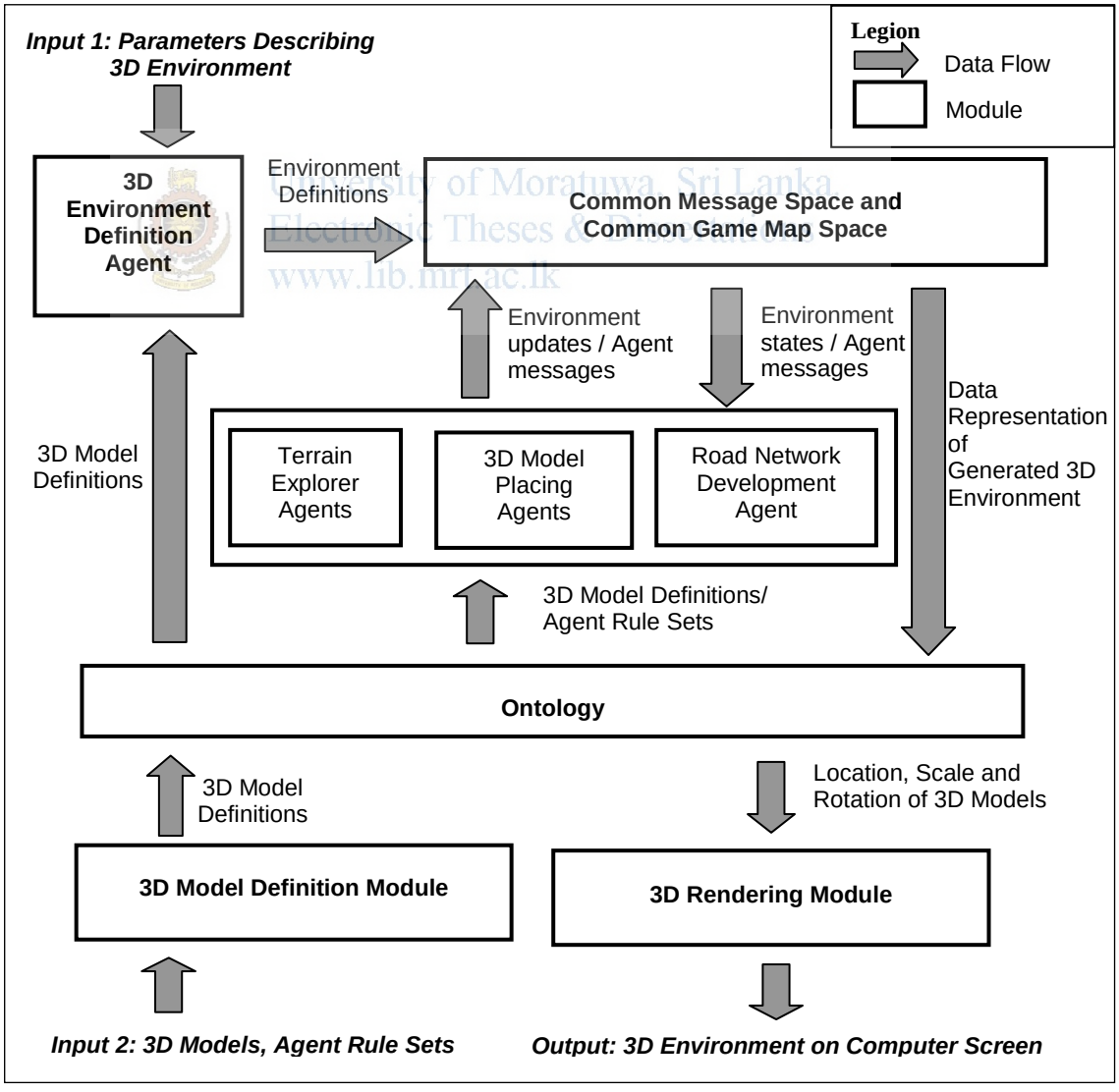

Figure 5.1 : Top Level Architecture of the Multi Agent System

The architecture of Multi Agent System for 3D Game Environments Design is shown in Figure 5.1. There are 2 main inputs and 1 output shown in the diagram. The block arrows show the interconnection between modules, inputs and outputs. Following sections describes main modules in this system.

5.3 3D Environment Definition Agent

This agent is used to capture a brief user description of the 3D environment. The input of the system would be a parameterized description of an imaginary 3D environment and a height map. A height map is a greyscale raster image used to represent surface elevation data, for display in 3D computer graphics. In a height map lighter areas shows terrain with height elevation and darker areas shows terrain with lower elevation. Figure 5.2 shows a sample height map which can be used as an input for the system.

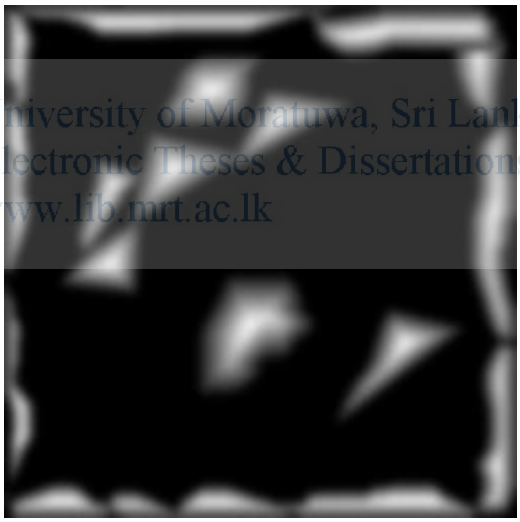

Figure 5.2 : A Sample Height Map

In addition, the user will specify following details as input.

- Size of the terrain
- List of 3D models selected from ontology with number of instances required
- Size and number of towns/ villages/ industries
- Water level

After capturing above inputs the *3D Environment Definition Agent* sends a request to *Common Message Space* to start the environment generation.

5.4 3D Model Definition Module

In order to use this system to design 3D environment, there should be sufficient amount of 3D model definitions available in the ontology. After defining a 3D model, that definition can be reused in any number of 3D environments created. And it is not necessary to define 3D models at each time generating a 3D game environment, if required 3D models are already defined in the system.

3D Model Definition Module is used to introduce new 3D models (such as trees, rocks and houses) to the system and associate agent classes with newly introduced 3D models (for example tree agent class associated with a 3D model of a tree).

When defining a 3D model, following parameters should be specified.

- 3D model name
- Location of 3D model
- Category of 3D model (tree, rock etc)
- Associated agent class (selected from ontology)
- Default scale

After defining above parameters, this 3D model definition will be stored in ontology can be reused in any 3D environment.

5.5 Ontology

The ontology will act as the main medium to store the knowledge of agents. The ontology contains following.

- 3D model binaries
- 3D model definitions
- Agent type definitions
- Data representations of generated 3D environments

5.5.1 3D Model Binaries

3D model binaries in which contains the actual 3D model (E.g.: 3D model of a house or tree) are also considered as the part of ontology.

5.5.2 3D Model Definitions

3D model definitions are defined using *3D Model Definition Module (Section 5.4).* These definitions are saved in the ontology and retrieved when a 3D game environment is generated.

5.5.3 Agent Rule Set Definitions

Users can define agent rule sets of a given 3D model type. The behaviours are simple rules. Also users can specify the hierarchies of agent rules in order to inherit rules of other agents. In 3D model definitions, these agent rules are associated with 3D models. As a result, a set of agent rules get assigned to each 3D model. These simple rules are used to arrange 3D models in game environment with surprisingly complex outcome.

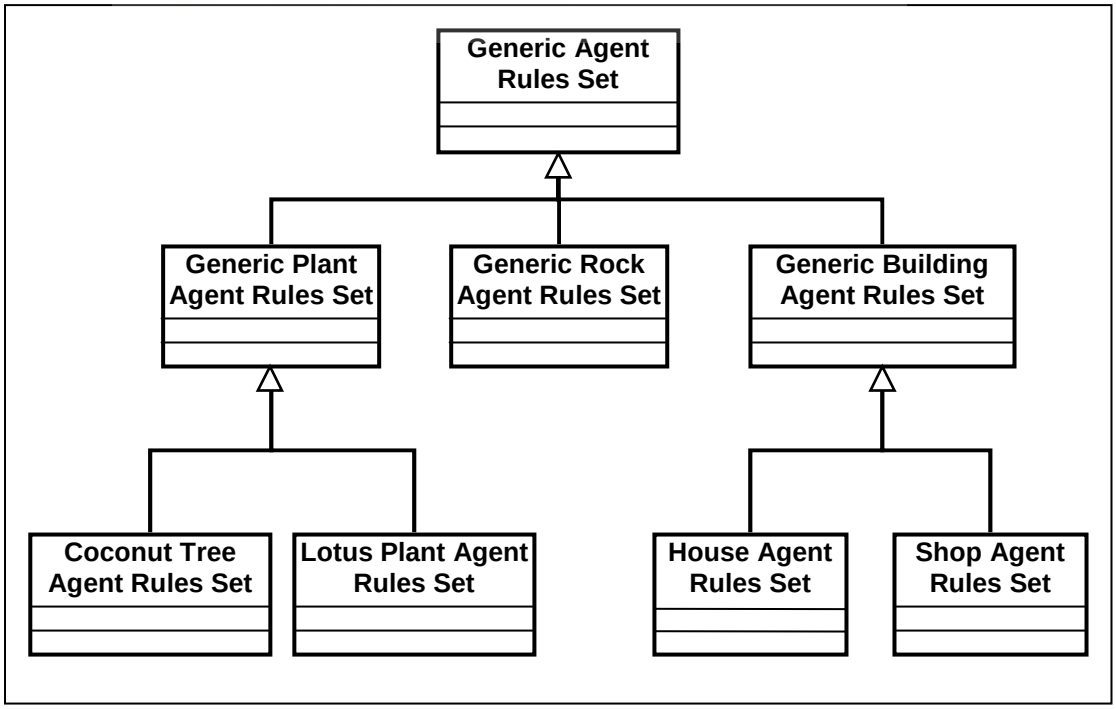

Figure 5.3 : Example of a Hierarchy of Agent Rule Sets

Figure 5.3 shows an example of a hierarchy of agent rule sets. Having a hierarchy of agent rule sets helps users of the system to reuse common rules of agents in the top of the hierarchy. These rules are used by *3D Model Placing Agents* to place 3D models in appropriate places. In the given example, Generic Plant Agent Rules Set inherits the rules of Generic Entity Agent Rules Set. In a similar manner Coconut Tree 3D Model and Lotus Plant 3D Model inherit the rules of Generic Plant Rules Set. However at that level Coconut Tree and Lotus Plant, there are specific rules in addition to rules generic to every plant. These specific rules help 3D models of lotus plants to place on water surfaces and 3D models of coconut trees to place on ground.

5.6 Common Message Space and Common Game Map Space

The Common Message Space acts as a bulletin board or a common white board for agents. Agents can publish common messages and requests on common message space. These messages or requests are visible to all agents and relevant agents respond to them. This is an important module of multi agent systems where message passing is asynchronous.

The *Common Game Map Space* can be also considered as a common message space. The *Common Game Map Space* represents the current state of game environment. The state of *Common Game Map Space* changes as a result of the actions of agents. These changes are visible to all agents and their future actions are always based on the current state of *Common Game Map Space*. Operating on an environment based on the current state of that environment is a very common aspect of multi agent systems as the same concept is used in this system also.

5.7 Terrain Explorer Agents

Terrain Explorer Agents are a special type of agents who search for a given type of terrain. Upon the request from *3D Environment Definition Agent* to generate the 3D environment, the *Terrain Explorer Agents* explore different areas of the terrain to search for suitable lands for the required 3D game environment. The locations to explore are selected randomly.

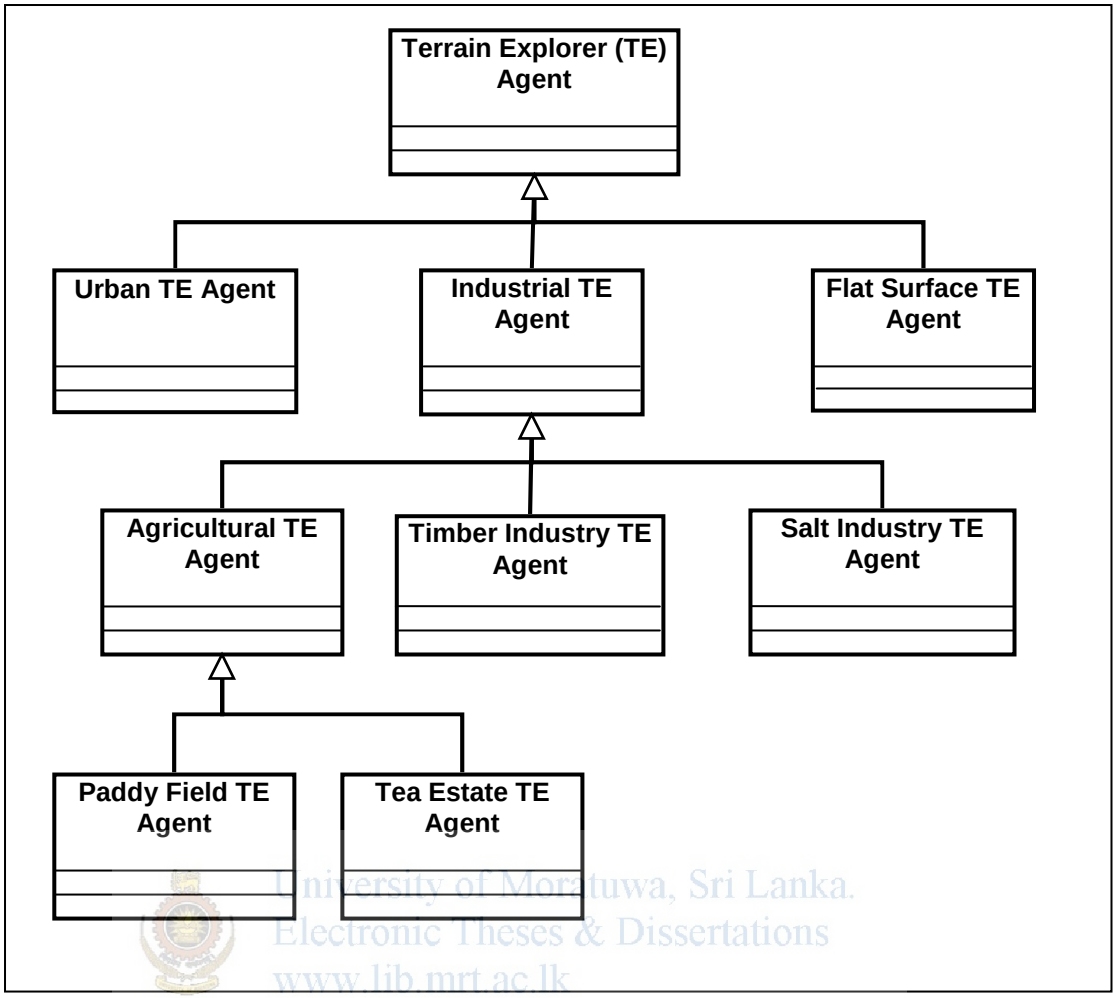

Figure 5.4 : Example of a Hierarchy of Terrain Explorer Agents

Figure 5.4 shows an example of a possible hierarchy of different specializations of *Terrain Explorer Agents*. However number of *terrain explorer agents* will be not limited to agents shown in Figure 5.4 and A*gent Definition Module* of the system provides facilities to define any number of specializations for any given agent type.

These *Terrain Explorer Agents* seek for suitable locations based on the type of *Terrain Explorer Agent*. For example an *Urban TE Agent* will seek for a land suitable to build a city and *Industrial TE Agent* will seek for a land suitable to build a given industry. When evaluating the suitability of land, A T*errain Explorer Agent* will consider following attributes.

- Type of surface (flat / hills / water / sand / rock)
- Size of the surface
- Distribution of barriers on surface such as trees and rocks

- Cost of cleaning up barriers
- Available areas that can be cleaned up within a predefined cost

When a *terrain explorer agent* finds required number of suitable places, it marks the required area *Common Game Map Space* as “Reserved Land” and broadcast about that in *Common Message Space*. For example, if the five areas are reserved for a city then the message will be “5 Areas Reserved for City”. After that the agent will be disposed. Other *Terrain Explorer Agents* will avoid this reserved land and they will continue searching in other areas.

5.8 3D Model Placing Agents

3D Model Placing Agents are responsible for placing 3D models in 3D game environment, Upon the request of desired game environment from *3D Environment Definition Agent*, the relevant 3D model definitions will be loaded with associated agent types (Such as tree agent type, building agent type. These associations are defined in *Ontology* using *3D Model Definition Module*). After the Terrain Explorer Agents have marked the areas suitable for various purposes, the 3D Model Placing *Agents* start marking places on *Common Game Map Space* to place 3D models in appropriate locations.

The proposed system can be extended by introducing various types of *3D Model Placing Agents* based on different types of 3D models. Also it is possible to define hierarchies of agents to inherit attributes and behaviours of *3D Model Placing Agents* as shown in Figure 5.5.

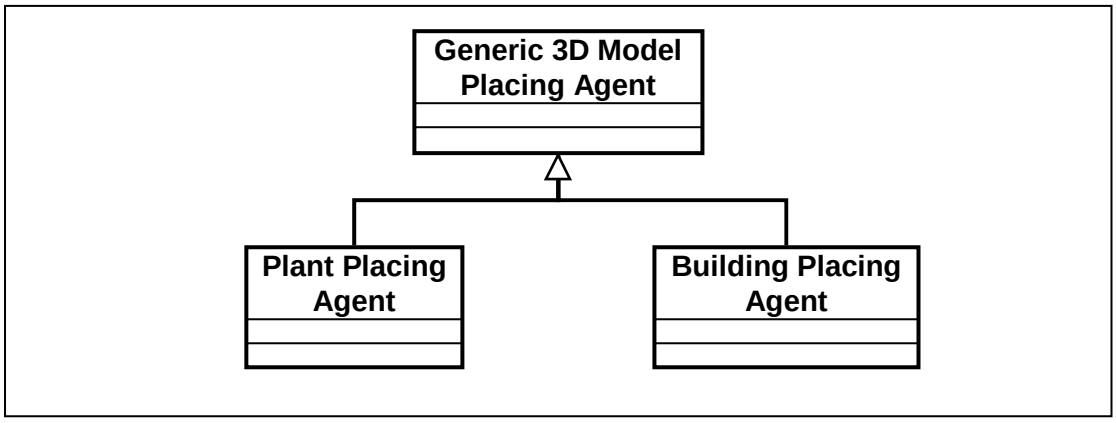

Figure 5.5 : Example of a Hierarchy of 3D Model Placing Agents

In order to mark a place for a 3D model, a *3D Model Placing Agent* reads the 3D model definition of the given 3D model from ontology. From that definition the agent type associated with the 3D model is identified. The simple rules defined in the agent type are used by *3D Model Placing Agent* to place the 3D model in a suitable place. Depending on the rules, 3D *Model Placing Agent* may consider all or some of the following things before marking a place for a 3D model.

- The type of the terrain
- Types of neighbouring 3D models
- Sizes of neighbouring 3D models
- Distance to roads
- Distance to cities
- Locations of neighbouring 3D models
- Orientation of neighbouring 3D models and
- Messages/ requests received from other agents

5.9 Road Network Development Agent

Road Network Development Agent is responsible for the development of road networks within the game environment. There are 2 types of roads namely primary roads and secondary roads. Primary roads are used to connect cities and secondary roads are used to connect buildings to primary roads, if those buildings are not connected to any primary road.

To generate primary roads, this agent searches for the city areas which are not connected to any primary road. Then this agent searches for the nearest city or primary road to the given city. After that it creates a new primary road from the given city to nearest city or road. To generate secondary roads, this agent searches for building areas which are not connected to any primary or secondary road. Then this agent searches for the nearest road or city. Then it creates a new secondary road from given building area to nearest road or city.

When creating roads, this agent avoids barriers such as mountains, rocks etc. At the same it always tried to create the road on flat surfaces or surfaces with minimum slopes. After creating the road the *Common Game Map Space* is updated by marking

the road on the map to make it visible to other agents. Also a message is placed on *Common Message Space* to inform other agents about the completion of road creation.

5.10 3D Rendering Module

A free and open source game engine is used to render 3D output of the system. When starting the game engine, it reads information of generated 3D game environment from ontology. The game engine first creates the terrain based on the height map provided. Then it loads the relevant 3D models and places them in appropriate places in 3D space with required scale and orientation. In the game engine the height is represented by Y axis. When placing a 3D model in a [x, z] location, the y value of the location is calculated based on the height of the terrain. Also the game engine reads the road network information from ontology and draws roads on the train as a texture. After creating the 3D game environment, the system provides facilities to flythrough the 3D environment using keyboard and mouse.

5.11 Interaction between Modules

Following section describes the interactions between above modules using a 3D environment generation scenario. In Figure 5.6 the interaction between modules are shown using arrows. The number on the arrow shows the sequence of interaction. Following are the description of interactions labelled by numbers.

(1) When it is required to introduce new 3D models, the *3D Model Definition Module* is used define and store 3D model definition in *Ontology*

(2) To define a required 3D environment to be generated the *3D Environment Definition Agent* is used and this agent loads 3D models details from *Ontology*

(3) *3D Environment Definition Agent* request to generate a 3D environment by passing a message to *Common Message Space*

(4) *Terrain Explorer Agents* observe the request of *3D Environment Definition Agent*

(5) *Terrain Explorer Agents* read the *Ontology* to access relevant agent definition/ rule sets etc.

(6) *Terrain Explorer Agents* explore the required terrain and mark it on *Common Game Map Space* and sends a work completion message to *Common Message Space.*

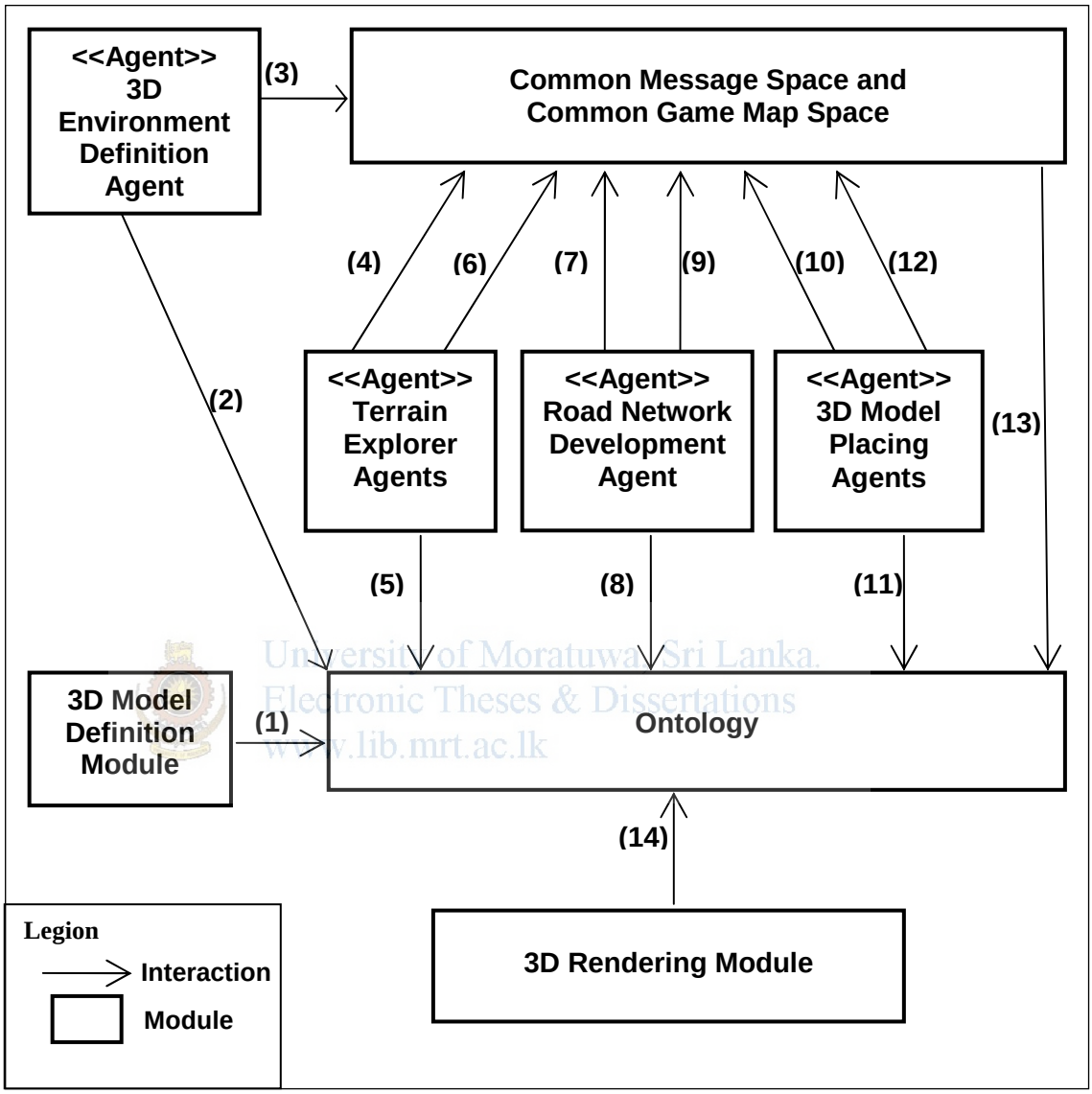

Figure 5.6 : AUML Collaboration Diagram to Show Interactions in Multi Agents System

(7) *Road Network Development Agent observes* the work completion message sent by *Terrain Explorer Agents*

(8) *Road Network Development Agent reads* the *Ontology* to access relevant agent definition/ rule sets etc.

(9) *Road Network Development Agent* mark roads on *Common Game Map Space and sends a work completion message to Common Message Space*

(10) *3D Model Placing Agents observe the work completion message sent* by *Road Network Development Agent*

(11) *3D Model Placing Agents reads* the *Ontology* to access relevant agent definition/ rule sets etc.

(12) *3D Model Placing Agents mark locations to place 3D models on Common Game Map Space and sends a work completion message to Common Message Space*

(13) The generated 3D environment representation in *Common Game Map Space* is stores in *Ontology*

(14) The *3D Rendering Module* reads the generated 3D environment representation from Ontology, generates the 3D environment using game engine and shows the output in 3D and this is the end of 3D game environment generation process

5.12 Summary

This chapter describes the design of the Multi Agent System to Assist 3D Game Environments Design. The proposed system has been decomposed in to 8 modules and the functionality of each module is described. The next chapter discusses about the implementation of each and every module identified in this chapter.