A Generic Model Framework for MOUT Simulations

Suiping Zhou

Nanyang Technological University
Singapore
asspzhou@ntu.edu.sg

Shang-Ping Ting

Defence Science & Technology Agency
Singapore
tshangpi@dsta.gov.sg

Abstract

Considerable efforts are needed to construct a virtual environment for Military Operations on Urbanized Terrain (MOUT). The models of various objects used in one application are often specifically designed for that application, thus are not generally suitable for other applications. We often have to rebuild many object models for a new simulation setup, which is very tedious and time consuming. To reduce the cost and time in building up different simulation environments, it is therefore highly desirable to have a repository of object models that are interoperable and extensible thus can be reused in different applications. To this end, we propose a generic model framework which supports model interoperability and reusability for MOUT simulations. Our experiences with Twilight City, a virtual training environment for MOUT, show that the proposed framework is successful in creating a repository of models for various simulation setups.

1. Introduction

According to a United Nation's report [1], the world population in urban areas will grow from 2.86 billion in 2000 to 4.98 billion by 2030. In view of the rapid urbanization, disasters in an urban area may have great impact on people's lives. Thus, Military Operations on Urbanized Terrain (MOUT) are getting increasingly important. In fact, MOUT are becoming a major priority for national security due to the threat of terrorist attacks in urban areas. However, live MOUT trainings are costly, time consuming, and require huge coordination effort in terms of preparation and administrative work. Thus, building virtual training environments for MOUT seems to be a logical and powerful alternative.

One of the major considerations in building a virtual MOUT environment is the time and resources needed for the design and development of various object models involved. We have noticed that many MOUT simulation projects are much task-focused and thus the possibility of collaboration of the efforts across different projects is often overlooked. As there are many different objects in a MOUT simulation, such as non-player characters, movable objects, static objects, terrains, etc., it is very time-consuming to construct the simulation from scratch. If simulation models across different projects are interoperable and extensible, then they can be reused in constructing new simulation projects, which will greatly save the cost and time in development. It will free the researchers and developers from the hassle of building up the various object models, and as a result, allows them to focus on the more important aspects of the project such as improving the conceptual model.

However, in most existing MOUT simulations, the models are not generic and are tightly coupled with the specific simulation objectives and setups. In addition, different MOUT simulation projects may adopt different game engines as development platforms. Currently there are no standards to ensure the interoperability of the models built on top of different game engines, thus the models in different projects may not be shared across different game platforms and are not interoperable with each other. Even for models developed for the same game engine, it is still hard to reuse the existing models due to the lack of standards to ensure model extensibility and composability.

Recently, the emphasis on model interoperability and reusability across different MOUT simulations is growing. The Simulation Interoperability Standards Organization (SISO) holds various conferences and workshops every year to promote model interoperability, reusability and composability [2]. In [3], Ray J. Paul discussed the impact of model reuse in the quality of simulations. Mark Ryan, Doug Hill, and Dennis McGrath modified the Gamebots interface to be interoperable with existing simulations to allow external simulations to control the state of internal game objects [15]. Bob Lutz studied various concepts of breaking down existing

models into reusable components. Paul Gustavson worked on the piece-part concepts of Bob Lutz and created the Base Object Models (BOM) [4], [8], [9].

In this paper, we propose a generic model framework to support model operability and extensibility so that various models in different MOUT simulations can be reused in construction of a new simulation project. With the generic model framework, we will be able to rapidly construct a new MOUT simulation and enjoy greater adaptability in face of the changing simulation requirements by leveraging on a rich repository of existing models. As the existing models have already undergone various tests in previous simulation projects, reusing them will also improve the quality and confidence of the simulation.

The remainder of this paper is organized as follows. In Section 2, we give an overview of the *Twilight City*, a virtual training environment for MOUT. The generic model framework is discussed in Section 3. Some examples of using the proposed framework in *Twilight City* are discussed in Section 4. Conclusions are drawn in Section 5.

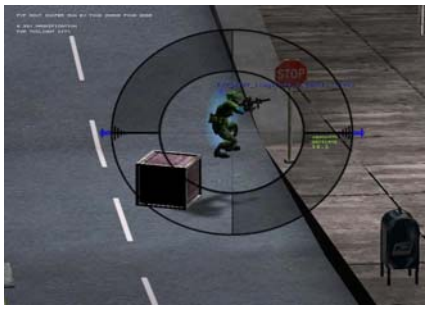
2. Overview of Twilight City

The objective of building the *Twilight City* [6] is to create a high fidelity training and simulation system for MOUT. *Twilight City* aims to act as a general testbed for various MOUT operations such as hostage rescue or raid operations. A typical scenario in *Twilight City* is the special squad operation to save hostage held in a building by some terrorists. The squad may consist of several human-controlled characters and some AI (artificial intelligence)-driven Non-player characters (also known as *bots*). The terrorists may also be some human-controlled characters and bots. Various tactics can be implemented in *Twilight City*, and the users (trainees) can get familiar with the real operation environments, assess the effect of different tactics, and choose the best tactics to be used in the real operations.

Twilight City is built using a commercial game engine, the *Unreal Tournament* (UT) engine. The motivation of using a game engine lies in the rapid growth of commercial game engines that are affordable and also provide excellent support for high fidelity 3D modeling. With the support of game engines, developers may focus on human behavior models and various tactics rather than on the 3D graphics.

Twilight City is designed to work on top of the UT engine with various modifications. These modifications adapt various UT game features to cater for MOUT simulations. With these modifications, *Twilight City* can be used as a MOUT simulation platform for various scenarios. *Twilight City* uses client-server architecture and allows for multi-player setup. In particular, an AI framework is implemented to generate realistic tactical behaviors for AI bots, a speech recognition module is introduced for human voice recognition, customized animations and special effects are used to enhance the fidelity of environment and enrich a human player's experiences in *Twilight City*. Various tactical scenarios are used to evaluate the performance of *Twilight City*. Case studies show that *Twilight City* has high fidelity on visual effects and the AI bots also demonstrate some human-like tactical behavior [6], [7]. Figure 1 shows some screen shots of *Twilight City*.

(a) Squad operation



(b) Sniper's view

Figure 1. Twilight City

Our experience in constructing *Twilight City* shows that it is very time consuming to build the simulation environment from scratch due the large number object models involved. In the meantime, we also noticed that though the behaviors of various objects seem different, they share a common structure in terms of how to respond to various events in the simulation, thus can be modeled using a

generic framework. In particular, to increase realism of *Twilight City*, there is a need to increase the number of movable objects yet keeping the computational costs low. Thus, we used qualitative physics for modelling movable objects in *Twilight City*. In this paper, we will show how the proposed generic model framework helps to model the movable objects using qualitative physics.

3. Generic Model Framework

In this section we will describe the software architecture of the generic model framework and the ways to compose reusable models for MOUT simulations using the framework.

3.1 Generic Objects

We adopt an object-oriented modelling approach to creating reusable models in MOUT simulations. With this approach, complex simulation models are broken down into some basic building blocks, called *generic objects*. Figure 2 shows the structure of a generic object.

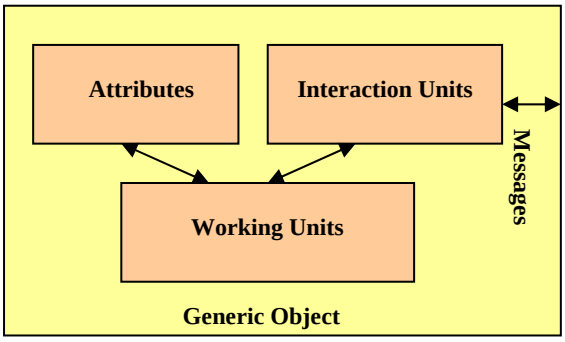


Figure 2. Structure of a generic object

A generic object is composed of three elements: *attributes*, *interaction units*, and *working units*. The attributes are used to hold the state of an object. The working units describe the functions that the object can perform. The interaction units are used to handle the interactions between different objects. All messages between different objects are through the interaction units.

The proposed structure is designed to capture the common features of all the models in MOUT simulations. The working units use the attributes values and the information from the interaction units to perform various functions. The results from the working units are used to update the attributes, and related information is sent to the simulation environment (and other objects) as messages via the interaction units. Figure 3 shows the class definition of a generic object.

Using the concept of encapsulation to package the generic object, their attributes are hidden from other generic objects. This is to ensure modularity and information hiding. Complex models can be built by extending the basic object class as show in Figure 3.

```
Class Generic Object{
// Interaction units
void EventHandlers();
//Working Units
void Initialize();
void Operate();
void Tick();
//Attributes
int ObjectID;
int Time;
int StateID:
}
```

Figure 3. Generic object class

3.2 Interoperable Components

Twilight City is built on the Unreal game engine. After a careful study of the Unreal game engine and various models used in different MOUT simulations, we identified two basic types of components: *entities* and *events*. Entities refer to the simulated characters, weapons and various movable objects like vehicle, boxes and bottles, etc. The state of an entity is characterized by its attributes. Entities can also interact with each other and with the environment via the event management mechanism. Events refer to various actions in the simulation. Examples of events include commands from the user, collisions, explosions, etc. When an event occurs, usually a message needs to be sent to other entities.

We further define a *model* component to describe the behavior of an entity. The model component resides within the entity component and controls the behavior of the entity. Figure 4 shows the relation between these components. The model component resides within the corresponding entity and is hidden from other entities and the external simulation world. The same entity may exhibit different behaviors if different model components are attached to it.

Through the event components, entities can communicate with each other and also update attributes dynamically.

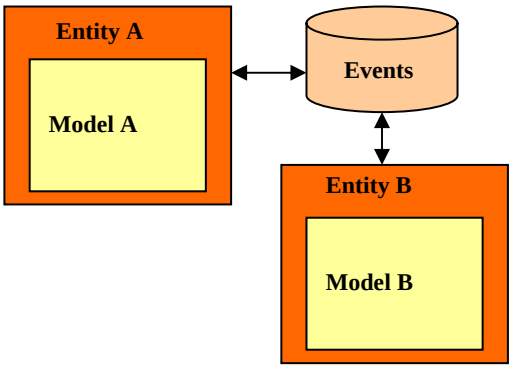


Figure 4. Interoperable Components

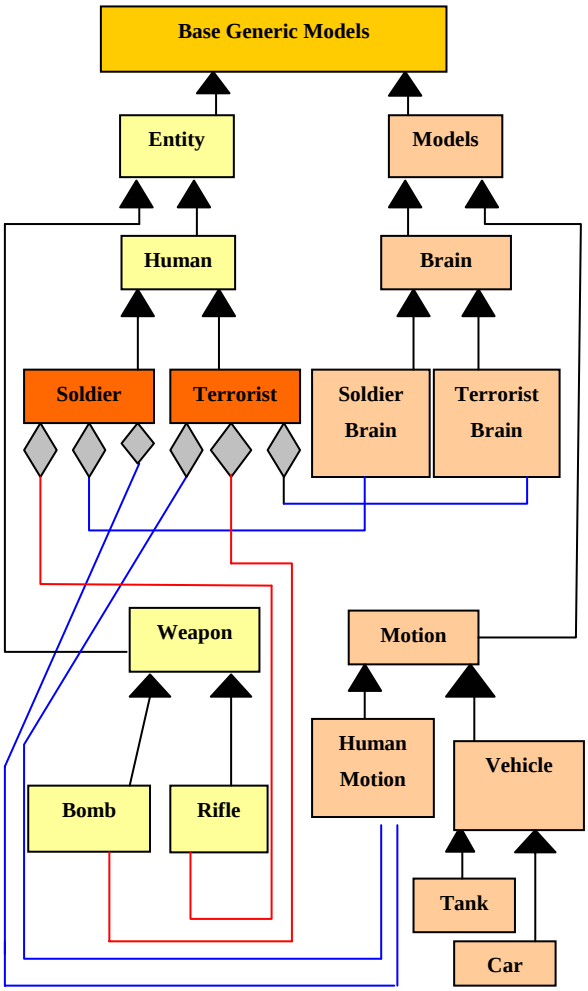


Figure 5. Class diagram of various object models

The class diagram in Figure 5 shows the aggregation of the basic components and the construction of various objects models. For example, by inheritance, weapons and human entities are extended from the base entity class. Model components such as *motion* and *brain* are also extended from the base model classes. Different characters such as terrorists and soldiers can be constructed by further extending the human entity class. Similarly, different types of weapons such as bombs for the terrorists and rifles for the soldiers can be built by extending the weapon class.

After the required components are described, a complete object model can be built by aggregating these components together. For example, the complete model of a soldier is an aggregation of the *rifle*, *human motion* and *soldier brain* components. Likewise, the complete model of a terrorist is an aggregation of the *bomb*, *human motion* and *terrorist brain* components.

3.3 Object Oriented Model Repository

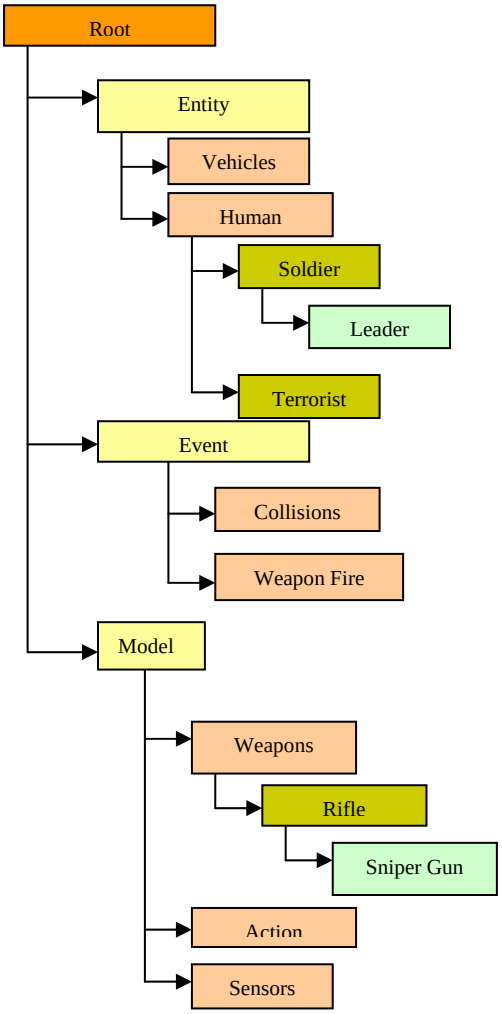


Figure 6. Generic models hub

To facilitate the construction of different MOUT simulations, we analyzed the modeling requirements of different setups. A set of generic simulation models is identified to be common across different simulations. We have built a large number of models used in MOUT simulations and store these models in a *generic model hub* with a tree-like architecture as shown in Figure 6.

The models are grouped into some main categories and further divided into some sub-categories as the repository expands. The different categories are divided according to the differences in their functions and tasks that they perform. New models can be built by aggregating and extending the existing models.

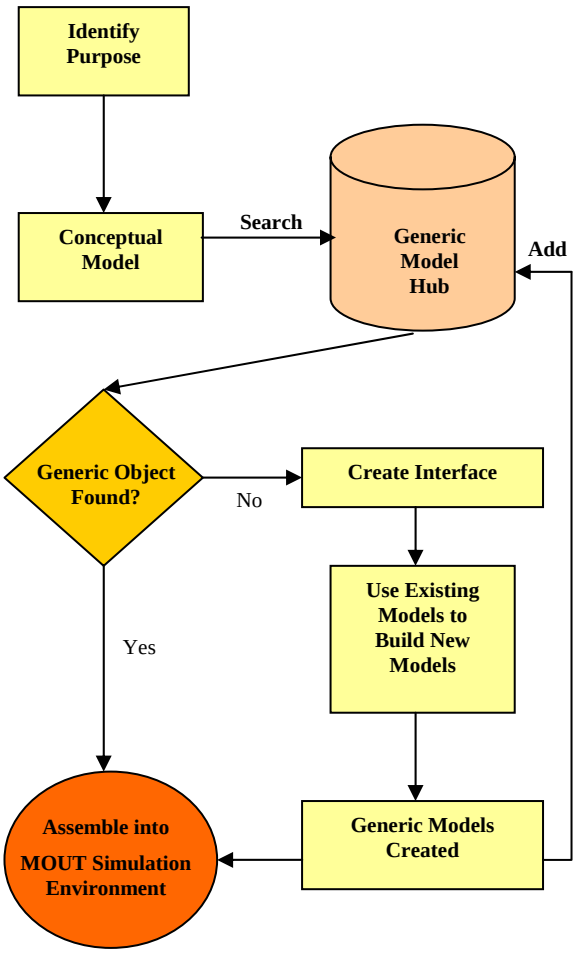


Figure 7. Model development flow

With the generic model framework, the model development flow for MOUT simulations is shown in Figure 7. After identifying the purpose of a new model, the conceptual model can be defined. Then the generic model hub is scanned. If a model corresponding to the conceptual model is found, then the model is assembled into the simulation environment. Otherwise, a new model will be built using existing components. The new model will then be added into the model hub for future use.

4. Example Results

In this section, we describe how the proposed framework helps to extend our previous work on *Twilight City*. As in most First Person Shooter (FPS) games, most objects in *Twilight City* are static objects, i.e., they cannot move. The reason for this is that modeling and implementing movable objects is a challenging task in terms of computational costs and mathematical complexities. The responsiveness of the system may be greatly affected by simulating the behaviors of the movable objects. However, as movable objects such as bottles, boxes and chairs, etc. are quite common in real-life, it is important to incorporate these objects in the virtual environments to enrich a human player's experiences.

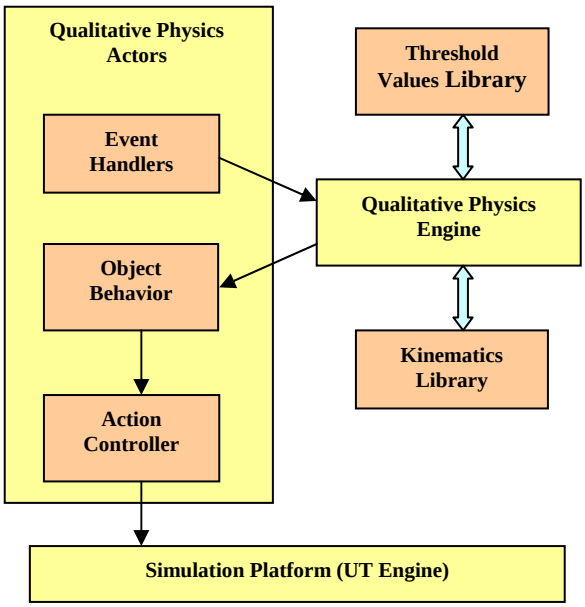


Figure 8. Qualitative physics system

In *Twilight City*, we adopt an approach of using Qualitative Physics (QP) to model the behaviours of movable objects in MOUT simulations. This approach will help to reduce the computational cost of simulating the behaviours of movable objects and increase the fidelity of the system. We have implemented a QP engine to simulate the behaviour of various movable objects. Figure 8 shows various

modules for simulating the behaviours of movable objects. This part of work has been presented in the 39th Annual Simulation Symposium [5].

The *Event Handlers* in Figure 8 are built using the *event* components within our generic model framework. These event components are constructed based on the generic object template to ensure composability and interoperability. Similarly, the *Object Behavior* module is built up by composing the *model* components in the generic model framework. *Qualitative Physics Actors* are extended from the existing *entity* components within the generic models hub. By leveraging the generic models framework and the existing generic models hub, the development time for creating various movable objects in *Twilight City* was shortened from a projected 8 months to the actual 2 months. Figure 9 shows some examples of movable objects that we implemented in *Twilight City*.

(a) Falling boxes

(b) Falling barrels

Figure 9. Movable objects in *Twilight City*

There are many other benefits of using the generic model framework, e.g., the rapid generation of terrorist behaviors in *Twilight City*. The terrorist behaviors are extended from the generic human entity and are combined with existing models such as the shooting actions of the soldier models. When there is a need to change the behavioral patterns of terrorists, we can conveniently adapt to this change by extending the existing models in the generic models hub. At the current stage of our project, we have implemented more than 50 generic model classes. We expect that the current model repository will greatly shorten the development time of various MOUT simulation setups.

5. Conclusions and Future Work

In this paper, we propose a generic model framework to promote interoperability, reusability and composability among various models across different MOUT simulations. A generic models hub is developed which defines various commonly used object classes for MOUT simulations. With this generic models hub, new models can be easily built by reusing or extending the existing models which helps to reduce the cost and time to build a new simulation environment. This also increases the confidence level of the developers, as the existing models have already undergone various testing. The developer can therefore focus on the conceptual models of the system rather than wasting time developing the computer models.

We will continue to work on the proposed framework, and enrich the object classes in the generic models hub. In particular, we shall focus on models for representing human behaviors and movable objects.

6. References

[1] Source: Findings of the 2004/05 UN-HABITAT report '*State of the world's cities*'

[2] Simulation Interoperability Standards Organization, www.sisostds.org

[3] Ray J. Paul and Simon J.E. Taylor, "What Use Is Model Reuse: Is There A Crook at the End of the Rainbow?", in Proceedings of the Winter Simulation Conference, December 8-11, 2002, San Diego, CA

[4] Paul Gustavson, "Building and Using Base Object Models (BOMs) for Modeling and Simulation (M&S) focused Joint Training", in Proceedings of the Interservice/Industry Training, Simulation, and Education

Conference (I/ITSEC), November 28 - December 1, 2005, Orlando, FL

[5] Shang Ping Ting and S. Zhou, "Qualitative Physics for Movable Objects in MOUT", in Proceedings of the 39th Annual Simulation Symposium, April 2-6, 2006, Huntsville, AL

[6] S. Zhou, S.P. Ting, Z. Shen, and L. Luo, "Twilight City – A Virtual Environment for MOUT", submitted to International Journal of Computers and applications

[7] Z. Shen., S. Zhou, C. Y. Chin, and L. Luo, "Design of an AI Framework for MOUTbots", in Proceedings of the 14th Conference on Behavior Representation In Modeling and Simulation (BRIMS 2005), May, 2005, Universal City, CA

[8] Paul Gustavson and Tram Chase, "Using XML and BOMS to Rapidly Compose Simulations and Simulation Environments", in Proceedings of the 2004 Winter Simulation Conference, December 5-8**,** 2004, Washington, D.C.

[9] Paul Gustavson, Phil Zimmerman, and Chris Turrell, 2003, "Capturing Intent-of-Use Meta Data for the Conceptual Model - A Key to Component Reuse", in Proceedings of the 2003 Fall Simulation Interoperability Workshop, September, 2003, Orlando, FL

[10] R. Adobbati and et al., "GameBots: A 3D Virtual World Test Bed for Multiagent Research", in Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS 2001, May 28 – June 1, 2001, Montreal, Canada

[11] Lewis M. and Jacobson J., "Game Engines in Scientific Research", Communications of the ACM, 45(1):27–31, January 2002

[12] IEEE 1516.3, IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), March 2003

[13] SISO, "Base Object Model (BOM) Template Specification", available at http://boms.info

[14] SISO, "Guide for Base Object Model (BOM) Use and Implementation", available at http://boms.info

[15] Dennis McGrath, Mark Ryan and Doug Hill, "Simulation Interoperability with a Commercial Game Engine", European Simulation Interoperability Workshop, June 27-30, 2005, Toulouse, France