Twilight City - A Virtual Environment for MOUT

Suiping Zhou, *Member*, *IEEE*, Shang-Ping Ting, Zhuoqian Shen, Linbo Luo
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: asspzhou@ntu.edu.sg

Abstract

We describe our work on *Twilight City*, a virtual training environment for Military Operations on Urbanized Terrain (MOUT). MOUT are often characterized by building-to-building, room-to-room, and person-to-person close combat, therefore the virtual training environment for MOUT requires high fidelity in 3D graphics and realistic behaviors of various objects. *Twilight City* is constructed on a commercial multi-player first-person-shooter (FPS) game engine, the *Unreal Tournament* (UT), which provides very good support for 3D modeling. Various modifications to the UT game engine have been made to incorporate various aspects in real MOUT scenarios. For example, a layered Artificial Intelligence (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 the virtual environment and enrich a human player's experiences in *Twilight City*. Case studies show that the AI bots are able to demonstrate some human-like tactical behaviour. Human factor tests are conducted to evaluate the performance of *Twilight City*. The results demonstrate the effectiveness of *Twilight City* as an urban warfare simulation environment. The lessons learned from these experiments are also discussed which provide directions for future improvement.

***Index Terms*—virtual environments, simulation, training**

I. INTRODUCTION

"The worst policy is to attack cities. Attack cities only when there is no alternative." As pointed out by the ancient Chinese general Sun Tzu [1], Military Operations on Urbanized Terrain (MOUT) are notoriously tough

especially for the attackers. However, it is getting more and more important nowadays for modern armies to adapt to urban warfare due to the terrorist attacks in cities, for example, the Russian hostage crisis in 2004 [2]. As training in urban environments is costly and often restricted to the physical mock-ups of buildings and other urbanized terrain features, building virtual training environments for MOUT seems to be a logical alternative.

MOUT are often characterized by building-to-building, room-to-room, and person-to-person fighting. Therefore, the virtual training environments for MOUT require high-fidelity and high-performance models. Several such systems have been built, and in general, they adopted two different design approaches. The first approach is to design and implement the whole system completely. For example, the Institute for Creative Technology (ICT) at the University of Southern California developed the *Full Spectrum Command* [3] from scratch to produce immersive training simulations for the US army. While this approach is very flexible, it needs a lot of time and programming effort. The other approach adopted by other developers is to implement a separate AI engine while using an off-the-shelf game engine to implement other features of the game. For example, the MOVES Institute at the Naval Postgraduate School developed the *America's Army* game [4] for military training using the Unreal Tournament (UT) game engine [5]. The motivation behind the second approach lies in the rapid development of commercial game engines which are affordable and also provide excellent support for high-fidelity 3D modeling. With the help of game engines, developers may focus on various AI techniques to generate more realistic human behaviors.

In recently years, game engines (especially the UT engine) are adopted by more and more researchers and developers for constructing high-fidelity 3D environments. Michael Lewis and Jeffrey Jacobson advocated the adoption of game engines such as the UT engine and the Quake engine for research [6]. David Arendash emphasized the strength of the UT engine as a 3D visualization tool [7]. Marc Cavazza and his team used the UT engine to create virtual environment for digital arts [8]. Their work was focused on applying qualitative physics to model various processes. Phongsak Prasithsangaree and et al. used the UT engine as a visualization tool for large-scale wargame simulations [9]. Various modifications to the UT engine have been made in recent years to enhance its support for different applications. In particular, Kaminka and et al. developed the *Gamebots* as an infrastructure for multi-agent research [10]. *Sentry Studios* developed a popular UT modification, *Infiltration*, which replaced the futuristic players/bots in UT with more realistic soldier and weapon models [11]. John Laird's group at the University of Michigan is also shifting to the UT engine from the Quake engine for creating intelligent bots [12]. They developed a game called *Haunt 2*, which is an adventure game rather than a warfare game.

This paper describes our work on *Twilight City*, a multi-player game for MOUT. The remainder of this paper is organized as follows. The major considerations in the design of *Twilight City* are discussed in section II. Section III describes the design details and some example results of *Twilight City*. Section IV describes the human factor tests that we have conducted to evaluate the performance of *Twilight City*. Finally, conclusions and future work are given in section V.

II. MAJOR DESIGN CONSIDERATIONS OF TWILIGHT CITY

The goal of our *Twilight City* project is to create a high-fidelity training and analyzing system for MOUT. A typical application scenario is the special squad operation for saving hostages held in a building by a group of terrorists. Various operation tactics need to be investigated before the real squad operation. The constructed virtual environment needs to resemble the real operation environment not only in terms of human's visual and audio perceptions but also in terms of the behavior of the non-player characters (i.e., AI bots). Creating such a high-fidelity virtual environment is a challenging task. Although there is some work on the application of FPS engines to military training, the design requirements may be different for different training tasks and scenarios. As pointed out by E. Lewis and M. Barlow in [13], "*The games must have sufficient fidelity to provide valid results. This fidelity does not have to be complete in all aspects, just enough in the variables of interest*." In the following sections, we identify the major considerations on the design of *Twilight City*.

A. Urbanized Environment

To simulate military operations around a city block, various urbanized terrain features need to be incorporated into *Twilight City*, for example, high-rise buildings, shops, public activity areas, lifts and staircases inside a building, streets and roads, etc. In particular, various sewers also need to be implemented in *Twilight City* to simulate the terrorist attack on a city's sewage system. To enhance the realism of the virtual environment, some special effects need to be implemented, for example, fire, smoke, water, rain, night vision, etc.

B. AI Framework for Intelligent Bots

Another important consideration on *Twilight City* is the AI framework for intelligent bots. To generate human-like behaviors, an intelligent bot should have a certain level of situation awareness, be able to perform tactical terrain reasoning and move/act coordinately and cooperatively with other teammates. In military training, predictable behaviors of the opponents may lead to negative training effects. Thus, it is also important for the bots to have the ability to learn or adapt from experiences so that they are able to cope with unpredictable situations and to demonstrate unpredictable behaviors.

Though many different kinds of AI framework have been proposed by the AI community [14] [15], few can be applied directly to MOUT simulations either due to over-simplicity or inefficiency. In general, the design of intelligent bots involves several different issues such as action selection, learning, adaptation, multi-agent coordination and cooperation. Based on different action selection mechanisms, the AI systems for intelligent bots can be classified into *reactive systems*, *deliberative systems*, and *hybrid systems*. Reactive systems intend to produce prompt and robust actions in response to the dynamic environment. However, the major problem with a reactive system is its poor scalability as well as the difficulty in understanding and predicting its behaviors. On the contrary, a deliberative system usually has a symbolic representation of the world and selects actions through reasoning and planning, which is the trend of the mainstream AI. Such systems are generally good at producing goal-directed behaviors. However, high computational cost of these systems usually leads to difficulty in producing real-time responses. Therefore, a natural idea is to design a hybrid system as a combination of a reactive system and a deliberative system. Usually, such an idea leads to layered architectures [16].

We believe that the development of an AI framework for MOUT bots is an incremental process. A full-fledged "brain" may consist of a number of modules in a complex architecture. Therefore, we shall start from a relatively small scale and gradually expand the framework by adding new modules. All the modules should ultimately work as an integral system. The whole system should also run as efficiently as possible. Therefore, the functionality of each module needs to be carefully determined and the interfaces between different modules need to be clearly defined.

Based on the above observations, the AI framework should meet the following requirements:

- *Flexible*: the bots should be able to take different actions according to different situations; they can respond promptly to the emergent situations and carry out a task intelligently to achieve a goal either individually or in collaboration with other teammates;
- *Extensible*: various new features can be added to the framework without much difficulty;
- *Integrable*: different behaviors and different levels of reasoning process can be integrated into a consistent hierarchy.

C. Speech Synthesis and Recognition

In most existing MOUT simulations, it seems that much more efforts are put on the visual effects rather than on the audio interactions. However, we believe that speech synthesis and recognition are integral parts of immersive MOUT simulations. With these functionalities, human players could communicate with the AI bots verbally, which may greatly enhance a human player's sense of immersion in the simulation. For example, instead of sending command messages to the AI bots in the squad through the keyboard, a human player may issue verbal commands to those bots. The bots can also respond to a human player “verbally” rather than by sending some text messages to the human player’s screen. As will be described later, the human factor tests with the *Twilight City* show that these modules indeed help to enhance the realism of the virtual environment.

D. Customized Animation and Models

The UT engine is originally developed to support futuristic FPS games. Its embedded character appearances and the associated motion models (e.g., holding guns, jumping and running) are not very realistic for MOUT simulations. Thus, there is a need to enhance the UT engine with some customized character models and animations. Moreover, to enhance the realism of the virtual environment, it is also important to introduce some movable objects to the virtual environment, as they are very common in our daily lives.

III. DESIGN DETAILS AND SOME EXAMPLE RESULTS

A. MOUT Test Bed

To simulate MOUT in a highly populated city, we have created a large and detailed map using *Unreal Editor* 3.0, which provides excellent support for 3D modeling. As shown in Fig. 1, the map consists of about 50 buildings including shops, offices and some high-rise buildings, which simulates a city block of 7 km by 6 km. Each building consists of a number of rooms, corridors and lifts. Some of these buildings are inter-connected through various links. In addition, we have also modeled the streets, traffic and parks with some civilians. As the threat of sabotage of civilian structures through the sewage system is recognized, we have modeled the underground sewage system which allows the bots to move through. Furthermore, we have implemented different scenes of the map, e.g., daytime scene, night scene and raining scene have been implemented to simulate different operations scenarios. Theses features make *Twilight City* a good test bed for MOUT simulations. For instance, the high-rise buildings provide good hiding places for snipers, and the *L-shape* and *T-shape* turns in the buildings can be used to test various tactical behaviors of the bots.

Fig. 1. Twilight City

B. AI Framework for Intelligent Bots

We propose a layered architecture to address the requirements on the AI framework as pointed out in section II.B. Figure 2 shows the conceptual model of the framework.

The layered architecture has the following major benefits:

- The architecture reflects the natural pattern of human decision-making process. In general, low-level decisions, such as coordinating movements of the bots in the same group, correspond to reactive behaviors which are responsible for immediate handling of urgent situations. High-level decisions, such as scheduling different tasks in a mission, correspond to deliberative behaviors which are responsible for achieving long term goals.
- The layered architecture is flexible and extensible. Different decision-making procedures and knowledge representations may be employed in different layers without affecting other layers, provided that the functions of each layer and its interfaces to neighbouring layers are clearly defined.

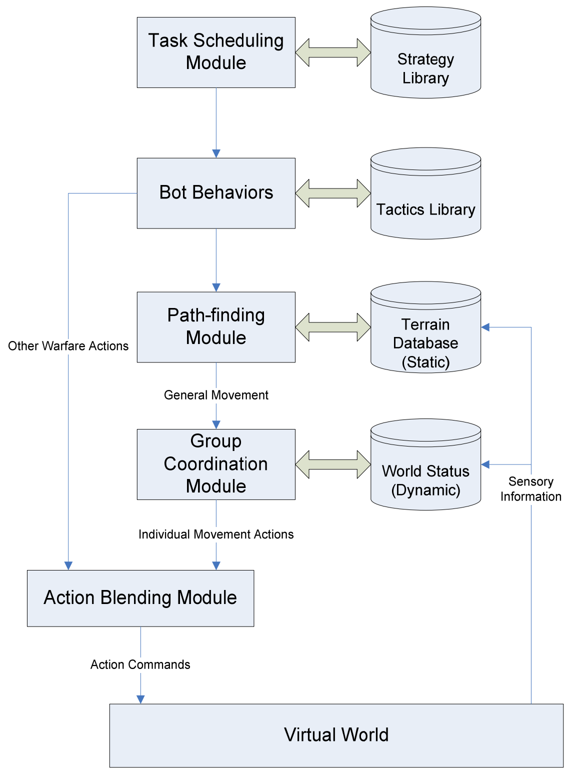


Fig. 2. The conceptual model of the AI framework

In the proposed architecture, the *Group Coordination Module* coordinates a bot's movement with other bots. For the members in a squad to move as a group, the movement of a bot should take into consideration the movements of other bots in the squad. The input of this module is a desired path for the group, and the output is the specific locations that a bot should move to. The calculation is based on the desired path, the group information, and the positions of the squad members. To this end, a bot should regularly update the information of its teammates as the *World Status*.

The *Path-finding Module* is used to find a path from a bot's current location to a destination location. The path calculation may also include some tactical information, like the positions of the enemies or threats, sniping spots, covering spots, etc. Therefore, the resulting path may not be the shortest one or the fastest one. In fact, it may be desirable for a bot to make a detour to avoid the enemies.

In the *Bot Behaviors* module, each behavior can be regarded as a sequence of basic actions including movement actions and other warfare actions, such as shooting. These behaviors are the building blocks of MOUT simulations. Each behavior achieves certain objective in a task reflecting certain warfare tactics. For example, when a squad plans to enter a room, some bots may need to secure the entrance of the room first. Here "*secure_room*" is a behavior. To make the behaviors reusable in different scenarios, a set of parameters are used to represent different situations. Thus, the same behavior may generate different actions according to different parameter settings.

Task Scheduling Module is responsible for scheduling bot behaviors appropriately, and it accomplishes the highest level reasoning or planning that reflects various warfare strategies. The input to this module is the description of a specific task, and the output of this module is the desirable behaviors to accomplish the task.

The *Action Blending Module* blends different actions into an executable sequence. In general, to blend different actions, this module needs to check for possible conflicts of different actions. For instance, a bot can shoot at enemies while moving to a desired location. However, a bot cannot pick up an item on the ground while shooting at enemies.

The implementation details of these modules in the AI framework are described in our previous work [17]. Here we only describe the *Bot Behaviors* module. This module can be regarded as a behavior reservoir. Each behavior is an independent program. Once its parameters are set, it generates action commands correspondingly. Different behaviors should be able to be added into or removed from the reservoir easily. In addition, as behaviors are the building blocks of the bot AI, the implementation should be as efficient as possible. We have implemented this module using a set of hierarchical *Finite State Machines* (FSMs). The FSM is widely used in both commercial games and the bot community due to its simplicity and efficiency. Figure 3 shows a sample FSM that we have implemented. Each rectangle represents a state in the FSM. The arrows connecting different states denote transitions. For example, in Fig. 3, when the squad forms a group (a state transition is triggered), each bot in the squad moves following their leader.

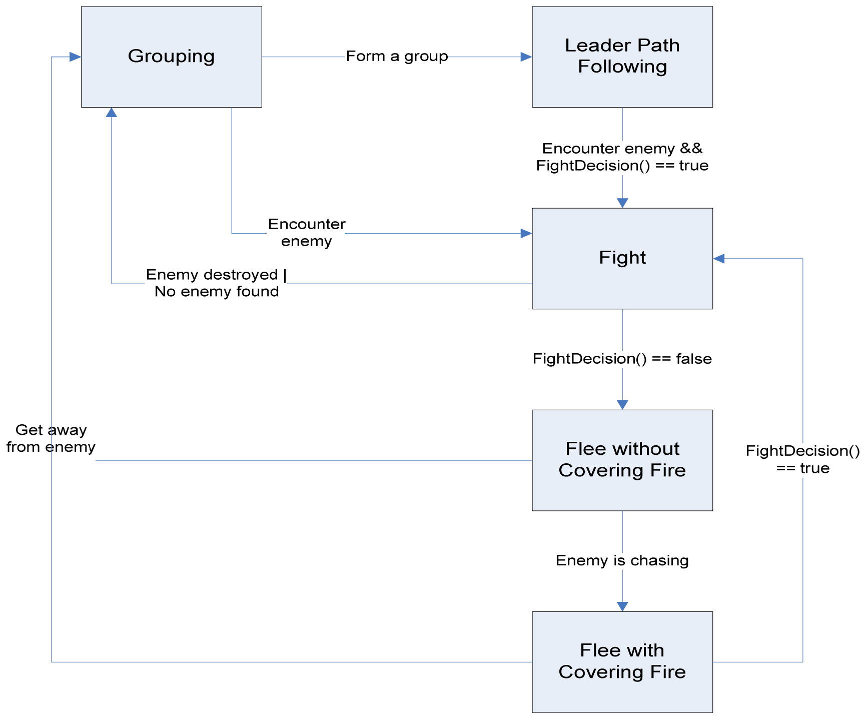


Fig. 3. A sample FSM

However, the problem with FSM is its poor scalability. On one hand, the transitions between the states will become very complicated if there are many states in a single FSM. On the other hand, once there is a change in the design, it will be difficult to modify the FSM structure. Adding or removing a state may lead to a significant change in all state transitions. To address this problem, we take the following two steps. First, the FSMs are organized into a hierarchy with no more than three levels, where a higher level state corresponds to a lower level FSM. Those states that can not be further decomposed into a lower level FSM will generate actions commands. Second, the behaviors implemented using FSMs are strictly limited to some low level behaviors, i.e., the transitions logic should not be too complicated. These two steps are to ensure that there are only a few states in each level and the transitions among the states will not be mixed up. In addition, as we keep the sizes of the FSMs in each level to be small, it is not difficult to maintain or extend them. Take the FSM in Fig. 3 as an example, among all the five states, the '*Fight*' state, the '*Flee without Covering Fire*' state, and the '*Flee with Covering*

Fire' state can be further decomposed into FSMs at a lower level. Figure 4 shows the FSM representation of the '*Flee with Covering Fire*'.

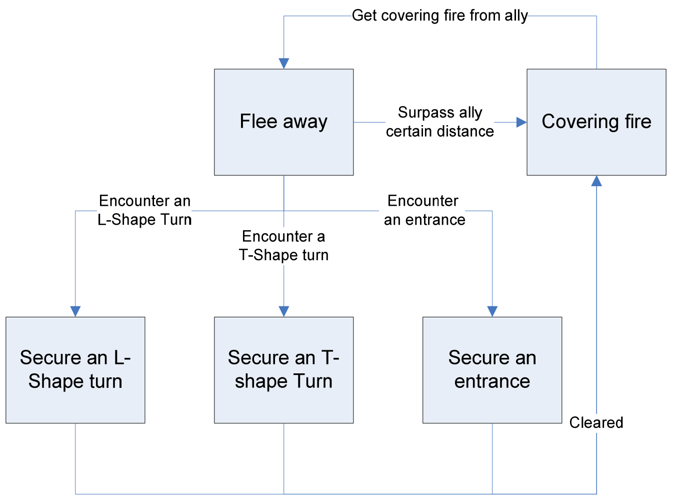
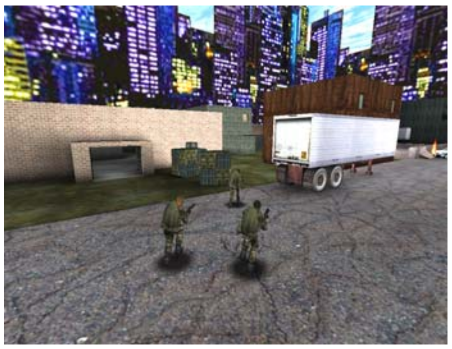
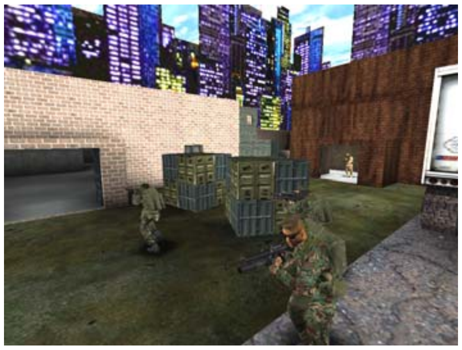


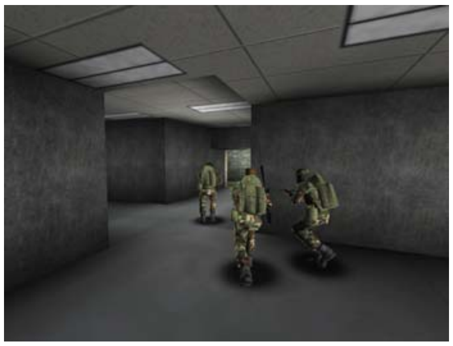
Fig. 4. An FSM example - flee with covering fire

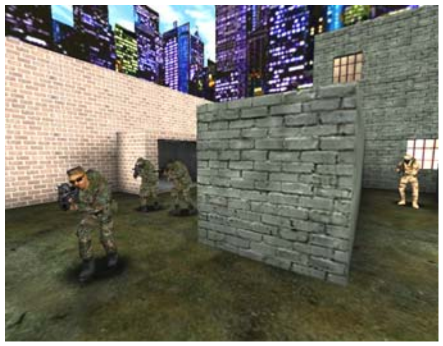
To demonstrate the effectiveness of the proposed framework and its implementation in producing realistic and robust bots behaviors in MOUT simulations, various scenarios are designed for a squad to complete some specified missions. Here we describe the results with one of these testing scenarios. The squad consists of 3 AI-driven bots. The mission of the squad is to get an item in a room. On the way to the item, the squad may encounter some enemies. So the squad members need to adopt various tactics to minimize the potential loss. As shown in Fig. 5, the two bots follow their leader to approach the destination. If an enemy is found at an entrance of the room along the path, the squad changes to another path to reach the destination. When an *L-shape* turn is encountered, the squad performs securing tactics. When the enemy is killed, one bot continues along its path to get the item, and the mission is finished.

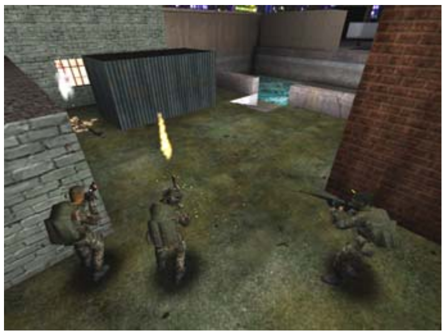
Other testing scenarios have also been used to verify the effectiveness of the proposed AI framework. For instance, different number of squad members and different number of enemies are used in the simulation. We have also tested the case when the squad leader was controlled by a human player. In all these testing scenarios, the bots can not only promptly respond to immediate situational changes but also consistently pursue their goal to the end. Even with the same experimental setup, the bots are able to take different actions and move accordingly in response to the different actions of the human players. It should also be noted that the behavior of the bots in our current implementation are still somewhat hard-coded in the sense that they are *tied* to squad operations and we need to explicitly tell the bots about the locations for executing some tactics, e.g., the locations for securing. Thus, if these locations are occupied by other bots or objects, the bots are unable to execute the proper actions. We hope to solve this problem by some terrain reasoning algorithms so that the bots could identify the suitable tactical key locations based on the current situation. We believe that the development of realistic bot behaviors is an incremental process, and there is still a long way to go for our bots to demonstrate comprehensive intelligence for MOUT simulations.


(a) Follow the leader to destination


(b) An enemy is found along the path


(c) Change to a new path


(d) L-shape turn is encountered


(e) Securing and fighting

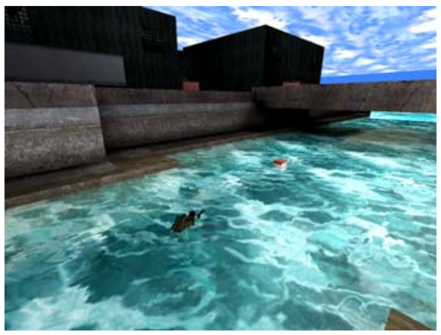

(f) Continue with the mission

Fig. 5. Squad operation

C. Speech Synthesis and Recognition

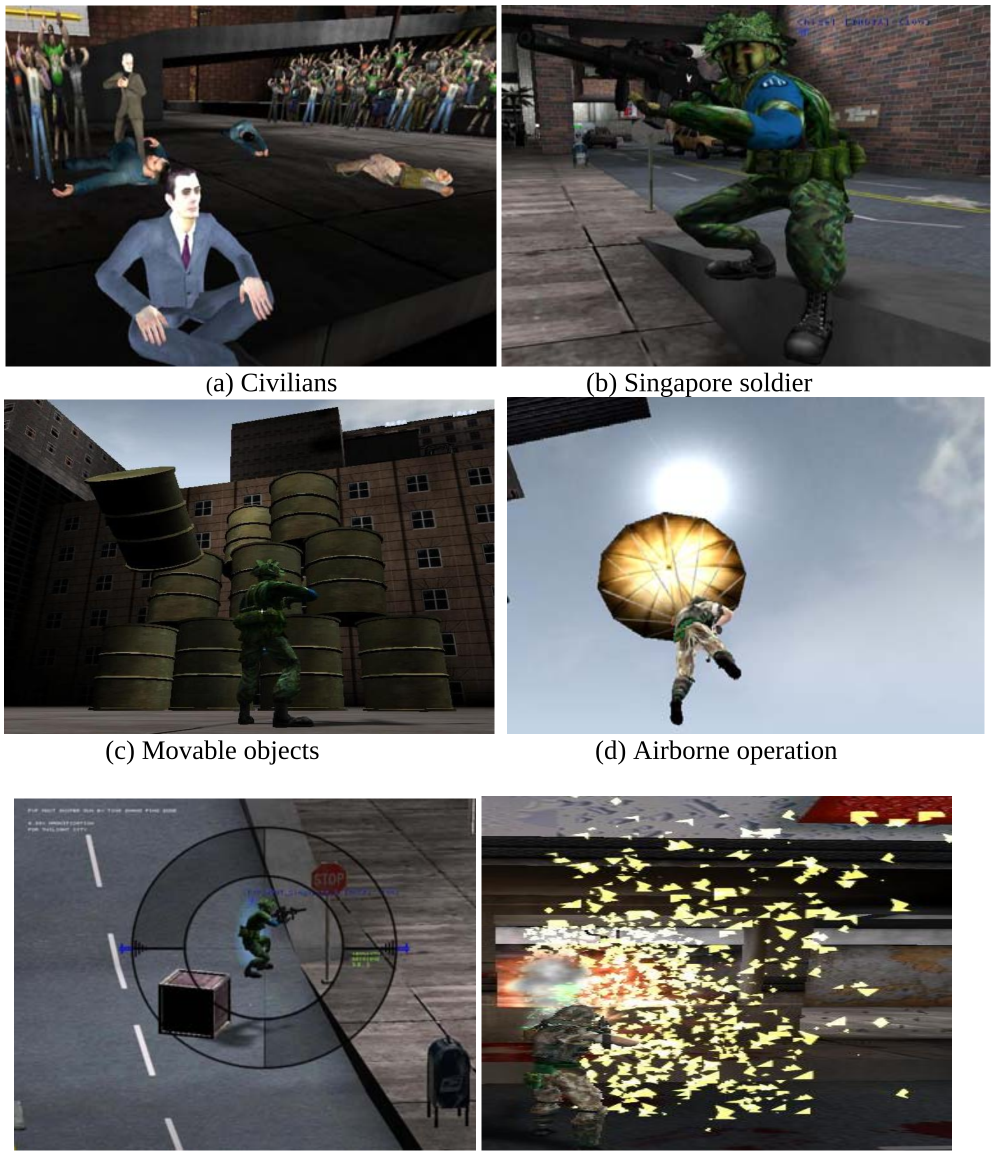
To enhance the interactivity of *Twilight City*, we have implemented a speech synthesis and recognition module using Microsoft's Speech SDK 5.1. We chose this toolkit for speech synthesis and recognition for two reasons. First, it has good APIs and is easy to use. Second, it reduces the code overhead required for an application to use the speech synthesis and recognition. Speech synthesis is straightforward by using the Text-To-Speech module in the toolkit. With the speech synthesis function, the bots can communicate with a human player by some simple words like "Yes, Sir!", "Danger!", and "I am on my way.", etc. We believe that the voice synthesis functionality can greatly enrich the experiences of a human player in the environment, though it is still rather simple at the current stage of our project.

The implementation of the human voice recognition function needs much more effort. First, human voices are received and translated into text messages. To this end, voice training must be done with Microsoft Speech SDK. Then, the built-in *Interaction* class of UT 2004 is extended so that it can intercept not only the keyboard inputs, but also the text commands from the voice module.

A series of tests have been done to evaluate the performance of the voice functionalities. Our first finding is that the voice recognition module performs best with commands of two or three syllables, e.g., "Attack!", "Come to me!", etc. It is difficult for the module to recognize more complicated commands. On the other hand, if the command is too short, like "Jump!", "Run!", etc., the results are more likely influenced by background noises. We also found that there is a positive correlation between the amount of training using the Microsoft Speech SDK and the response time of the bots to human commands. The response time could reach 3 seconds if the voice module is not trained properly. After sufficient and careful training, a bot could react to most of the human commands in the *Twilight City* within 1 second.

D. Customized Animation and Models

The UT game engine has some built-in physics models and 3D graphics models of the characters and other objects in the game. However, some of these models do not seem very realistic for MOUT simulations. In addition, many features and special effects in a real MOUT are not provided by the game engine. Therefore, we have created some customized animations and models. The animations are created frame by frame using a third-party tool, and imported into the UT engine with an *ActorX* plug-in [18]. To model airborne and underwater operations, the force of gravity on an object is reduced once a parachute is opened or the object is within water. For movable objects, e.g., boxes, barrels, we use some simple qualitative physics rules [19] rather than the exact physics laws to describe their dynamic behavior. Qualitative rules have been successfully used in virtual environments to model various dynamic features [8]. With qualitative physics, the computational costs to represent the behavior of movable objects will be greatly reduced, thus more moveable objects could be introduced into the virtual environment to enhance the fidelity of the virtual environment and also to enrich a trainee's experience in the environment. As human beings are more sensitive to the qualitative and causal relations among various objects and factors, and rely on these relations in their reasoning about the environment, we believe that qualitative physics could also be very helpful in enhancing the non-player characters' spatial reasoning abilities so that they could have some *common sense* about the environment and the relationships among various objects and factors. Figure 6 shows some of the models that we have implemented.



(a) Civilians (b) Singapore soldier

(c) Movable objects (d) Airborne operation

(e) Sniper’s view (f) Shattering glasses

Fig. 6. Customized animation and models

IV. EXPERIMENTS

To further evaluate the performance of *Twilight City*, we have installed it on some PCs (DELL with Pentium IV 2.4GHz CPU) in the Parallel and Distributed Computing Center (PDCC) at Nanyang Technological University and the Modeling & Simulation Department of the Defense Science and Technology Agency of Singapore. Twenty six subjects (aged between 20 to 40 years old) were asked to play with *Twilight City* for about half an hour (see Fig. 7). Then the subjects' opinions on various aspects of *Twilight City* were collected. All the subjects have more than 2 years of military training experiences and have experiences with FPS games.

Fig. 7. Mean Opinion Score Tests of *Twilight City*

It should be noted that the emphasis of these tests is not on the behavior of the AI bots that we have developed. Though the bots in *Twilight City* have demonstrated a certain level of tactical intelligence as described in section III.B, we feel that the behaviors of the bots are still very limited and thus the bots are still not intelligent enough to collaborate or fight against real human players in more complex scenarios. Thus, the emphasis of these tests is on a subject's perception of the visual effects of the virtual environment and various artifacts that we have introduced. The subjects were asked to give their opinions on the following questions:

1. *How do you feel the responsiveness of Twilight City?*
2. *Is Twilight City a good urban warfare simulation tool?*
3. *How do you feel the voice module (accuracy and responsiveness)?*
4. *How do you feel the behavior (realism and responsiveness) of the movable objects?*

The subjects answer these questions by giving a score (5-Exellent, 4-Good, 3-Fair, 2-Poor, 1-Bad) to the performance of *Twilight City* on each of these aspects. Figure 8 summarizes the results of these tests.

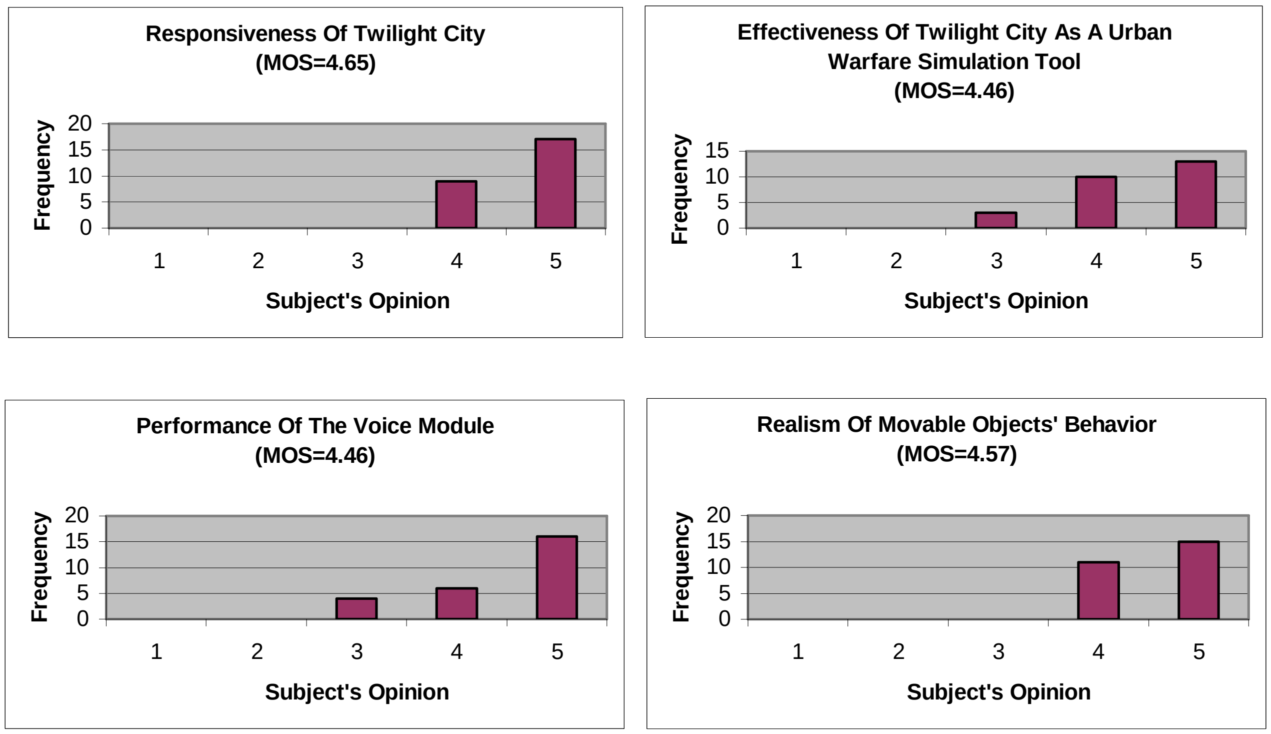


Fig. 8. Test Results

The subjects are satisfied with the responsiveness of *Twilight City* and the behavior of the movable objects that we have introduced. They also feel that these movable objects greatly enhance the realism of *Twilight City* as compared to other FPS games they have played. Though some subjects suggest that the voice module has noticeable delays and failed to produce correct results in some cases, the results show that the voice module is generally acceptable by most subjects. In general, the subjects feel that *Twilight City* is a good prototype system for MOUT simulations. The movable objects, various artifacts and the voice module are important to MOUT simulations. However, more voice commands need to be added, and more trainings of the voice module are also needed.

V. CONCLUSIONS AND FUTURE WORK

Building virtual training environments for MOUT not only has important practical value due to the high cost of MOUT trainings in real environments and the ever-increasing threat of terrorist attacks on civilians in large cities, it also proposes various challenging research issues in computer graphics/animation, computer-human-interface, modeling and simulation, and artificial intelligence, etc. In this paper, we described our work on *Twilight City*, a virtual training environment for MOUT. The commercial UT game engine is used in the implementation due to its high performance in 3D graphics modeling and rendering. Various modifications to the engine are made to implement various features in MOUT. In particular, a layered AI framework is proposed to implement various tactical behaviors for the bots. Although the bots behaviors are still somewhat hard-coded and specific to squad operations at the current stage, we believe that the framework is flexible and extensible for other behaviors. For example, the *Group Coordination* module may be by-passed if a bot is acting alone.

Experiments with human subjects show that *Twilight City* has good responsiveness, rich object models and behaviors for urban warfare simulations. The results also show that various daily-life movable objects are important for MOUT virtual training environments. Moreover, even with simple speech synthesis and recognition capabilities, the realism of such systems will be greatly enhanced.

In our future work, the AI framework will be further extended to implement more realistic human-like behaviours. A more sophisticated higher level reasoning and planning layer should be designed in order to select and schedule different behaviours. Spatial reasoning algorithms need to be developed to enhance situation-awareness of the bots. Group behaviour models also need to be further investigated so that bots in the same squad or different squads could collaborate to execute a mission. In addition, more tactics in real MOUT operations will be investigated and implemented in *Twilight City*. To make the communication between a human player and bots more natural, more voice commands will be added, and more careful trainings will be conducted using the voices of different people rather than a single person.

REFERENCES

[1] S. B. Griffith, *Sun Tzu: The Art of War,* Oxford University Press, New York, 1963.

[2] http://eidtion.cnn.com/2004/WORLD/europe/09/04/russia.school/

[3] Institute for Creative Technology, *ICT Games Project*, 2003, http://www.ict.usc.edu/disp.php?bd =proj_games

[4] MOVES Institute, *The official U.S. Army game: America's Army*, 2004, http://www.americasarmy.com/

[5] Epic Games Inc., *Unreal Tournament*, http://www.unrealtournament.com/

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

[7] D. Arendash, "The Unreal Editor as a Web 3D Authoring Environment", Proceedings of the ninth international conference on 3D Web technology, pp.119-126, April, 2004, Monterey, California

[8] M. Cavazza and et al, "Alternative Reality: A New Platform for Digital Arts", ACM Symposium on Virtual Reality Software and Technology (VRST 2003), pp.100-108, October 2003, Osaka, Japan

[9] P. Prasithsangaree and et al., "UTSAF: A Simulation Bridge between OneSAF and the Unreal Game Engine," Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics, October 5-8, 2003, Washington, D.C.

[10] G. A. Kaminka and et al, "GameBots: A Flexible Test Bed for Multi-agent team research", Communications of the ACM, Vol. 45, No. 1, Jan. 2004

[11] Sentry Studios, *Infiltration*, http://infiltration.sentrystudios.net/

[12] B. Magerko and et al, "AI Characters and Directors for Interactive Computer Games", Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference, pp.877-883, July 2004, San Jose, California

[13] E. Lewis and M. Barlow, "The Use of Games to Investigate Tactical Decision-making", SimTecT 05, 10-12 May, 2005, Sydney, Australia

[14] R. A. Brooks, “Elephants Don’t Play Chess”, in *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pp.3-16., the MIT Press, Cambridge, MA

[15] J. E. Laird, “It Knows What You're Going to Do: Adding Anticipation to A Quakebot”, Proceedings of the 5th international conference on Autonomous Agents, pp.385-392, May, 2001, Montreal, Canada

[16] J. P. Müller, “The Design of Intelligent Agents: A Layered Approach”, Lecture Notes in Computer Science, Vol. 1177, 1996

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

[18] Animation Export Tools, http://udn.epicgames.com/Two/ActorX

[19] K. D. Forbus, “Qualitative Process Theory”, Artificial Intelligence, vol. 24, 1-3, pp. 85-168, December, 1984