Believable NPCs in Serious Games: HTN Planning Approach Based on Visual Perception

Ibrahim M. Mahmoud*, Lianchao Li, Dieter Wloka
Computer Science, University of Kassel
Kassel Germany 34121
Email:imahmoud@inf.e-technik.uni-kassel.de
Email: lli@inf.e-technik.uni-kassel.de
Email:dwloka@uni-kassel.de

Mostafa Z.Ali
Computer Information Systems, Jordan University of Science & Technology
Irbid, Jordan 22110
Email: mzali@just.edu.jo

Abstract—On the way of the creation of virtual autonomous characters that are lifelike, intelligent and convey empathy, a strategic planning architecture is presented in the domain of serious games, where games are designed and built for an educational primary purpose other than pure entertainment. Our serious game is in the domain of emergency management; firefighters and first responders in accidents situations. Two contributions are presented in this paper; firstly, a visual perception system for non-player characters (NPCs) along with a short-term memory (working memory) are implemented so that NPCs will have access to only limited information and have to build their plans and make their decisions based on what they can perceive from their surrounding environment. Secondly, an HTN planning approach is implemented based on SHOP for the domain of our serious game. Five modules were developed in this part; Controller, World Model, Domain, Interface and HTN Planner. The plan is generated online based on the information gathered from the visual perception system and stored in the working memory. Our experiment shows the effectiveness of the proposed approach to create believable NPC behavior.

I. INTRODUCTION

On the contrary of typical video games, serious games are designed and built for an educational primary purpose other than pure entertainment. They are designed to provide a learning platform for the intended player, and were successfully applied in industries like defense, emergency management, scientific exploration, health care and city planning among other fields by simulating the real-world events or processes designed for the purpose of solving a problem. Combining both entertainment and education is known to be good methodology for learning, the learner will be having fun completing the tasks in the game while implicitly gaining knowledge. On the other hand, using video games as a learning platform will reduce costs, efforts and time spent in real-world trainings. Our serious game is in the domain of emergency management, where firefighters and first responders have to deal with variety of accidents and emergency situations. Nowadays, there are several firefighter training and simulation of emergency situations systems available. RescueSim [1] is a single and multi-agency training software that prepares public safety and security professionals for real-life incidents in virtual environment. RescueSim allows emergency crews to experience the incident as they would in real-life. They assess the situation and determine the best response strategy, implement it and then observe the consequences of their decisions without the dangers, costs and time associated with traditional practical training. E-Semble [2] is European market leader in simulation software for the public safety and security sector. They develop simulation software (serious gaming) for the education, training and assessment of incident response and safety professionals, such as police, fire and medical services. The simulation software is used by educators of police, fire and medical services, industry, traffic and tunnel operators in over 16 countries. VoTeKK [3] is a training and simulation tool that prepares doctors for rare and hard to control emergency scenarios.

Non-player characters (NPCs) are important part for the success of computer games. Most of the current computer games provide NPCs that are completely controlled by the game logic and that react to actions in a predefined action-state mechanism, in other words they will have a scripted and automatic behavior, triggered by certain actions or dialogue with the player character. While the players will have access to only limited information about the game, game developers can provide the game NPCs with full information about the game giving them the advantage over the player. In this way, NPCs will not only be able to defeat the player, but also they will look smarter than what they really are. Players, on the other hand, prefer NPCs that are realistic with life-like behavior. Many game players, and even developers, consider it is impressive to have each NPC in the game as an accurately simulated human, that are life-like, intelligent and convey empathy. In order to provide such type of NPCs, they should act naturally and perceive their environment using their sensing capabilities, such as; visual perception, hear and smell. These sensory information should be limited in order to simulate sensing in real humans. In this work we limit the sensing of the NPC to the visual perception, as vision can provide many useful information about the environment in which the NPC operates.

Social believability in games intends to be an important point that interest developers in aspects of implementing believable NPCs. Adding an extra level of interaction with these games through using a sensory system through which the users can get a sense of intelligence for these NPCs and how they react to different stimulus. An idea that was started through Sony in PS3 where users interact with their game and
see different types of responses to their actions to make more realistic scenarios. Published statistics of such types of games show that they are well-accepted and more interesting to people at different ages, from academic applications for kids to other types of games for older ages. Such types of games are harder to build as one should compile a more intense set of actions and knowledge-bases, but it mimics more realistic responses and a wider set of interactions for higher rate of attraction.

This paper is organized as follows: in section II, we present our visual perception system in virtual environments, which is the basis for our approach. Section III presents strategic planning using HTN in the domain of games. Our general approach is presented in Section IV. In Section V, we present an instantiation of this general approach in our serious game; we present an experiment on one scenario. Finally, in section VI, we conclude with some remarks on future improvements.

II. VISUAL PERCEPTION IN VIRTUAL ENVIRONMENTS

On the contrary to visual perception in robotics, building a visual perception system for NPCs in virtual environments such as video games has not been investigated deeply by researchers nor by game developers because game developers do not want to spend much time and effort on applying sophisticated Artificial Intelligence (AI) techniques for their games and realistic behaviors for their NPCs as they are in the business of selling games, and are therefore likely to be most interested in how intelligent NPC behavior can be channeled into making games more enjoyable. As they are able to build their NPCs with complete information about the game world and using some AI algorithms they can make their games successful and their NPCs look smart and realistic without the need to explore the visual perception for each NPC in the game. Huang and Peng [4] proposed an algorithm of vision simulation of autonomous non-player characters in dynamic scene. Their algorithm is divided into three main steps and is based on scene partition technique and vision cut-out technique. Firstly, the scene is partitioned sectors and potentially visible set (PVS) of each sector is represented as a tree-like structure at stage of level design. Secondly, active entities in each sector are updated while testing collision every frame. Finally, visibility is calculated by the NPCs cut-out frustum for active entities in sector the NPC stays and in PVS the sector includes. In [5] researchers discussed the role and utility of synthetic vision in computer games. They presented an implementation of a synthetic vision module based on two view-ports rendered in real-time, one representing static information and the other dynamic, with false coloring being used for object identification, depth information and movement representation. They demonstrated the utility of this synthetic module by using it as input to a simple rule-based AI module that controls agent behavior in a first-person shooter game.

Simulating vision for NPCs in virtual environments has two main fundamental differences from designing sensory algorithms for autonomous robots. Firstly, the aim of visual perception in robots is to perform a task in the most efficient way possible. On the other hand, NPCs are expected to perform a task in a human-like manner and the visual perception should be the guide for a realistic and life-like behavior. The other main difference from robotics is that robots exist in real environments and NPCs exist in virtual environments. Robots, therefore, must attempt to gain information about the environment through sensors which often give noisy, unreliable and impoverished data. Therefore complex computer vision algorithms are often needed to interpret this data and to translate the information into appropriate behavior. In a virtual environment on the other hand all existing information about the environment is already stored in the computer. This means that intermediary stage of interpreting the input from sensors is unnecessary: information can be obtained much more quickly and reliably from the data structures storing the environment. This makes the task of vision much easier. However, it is still not trivial. Having complete information is not realistic: for example an NPC reacting to events behind its back is not realistic. There need to be filtering mechanisms that ensure that the NPC only receives information that would be available to a real person in an equivalent situation.

There are two main approaches to handling visual perception in the literature. One uses ad hoc solutions for individual behaviors. The other uses computer vision techniques to simulate the low level vision processes.

A. Ad hoc methods

The ad hoc methods are consider simple and easy to be implement. The main idea is to simulate the behavioral pattern by producing realistic perception. In these methods there is no need to build a real virtual environment, but researchers can provide the sensory data about the environment as a row data to the visual perception of the virtual character. An early attempt on this direction was the work of Reynolds [6] on flocking behavior who attempted to provide the same information to his virtual birds (called boids) that a real bird would gain from sensory perception without directly simulating that sensory perception. These methods can produce good results efficiently with simple techniques. However, their originators tend to express worries about their realism.

B. Computer vision based methods

These methods are characterized by rendering the scene from the point of view of the virtual character and then using computer vision techniques on the resulting images. These methods are normally motivated by considerations beyond the needs of merely animating actors in virtual environment. Blumberg and Gaylean [7] use this method in augmented reality system (the ALIVE system) where an actor needs to be able to detect real objects as well as virtual ones. Computer has complete knowledge about the virtual environments but that computer vision type methods throw information away by projecting to 2D and then attempt to recreate it. Another problem is that computer vision deals with rectangular images as the basis of vision but real human vision is quite unlike this model.

Real human vision consists of a relatively small center of attention corresponding roughly to the contents of the fovea, the small high resolution area of the retina. The center of attention is the area in which most intense visual behaviors take place. This is the area of vision where we read and operate tools. Outside the center of attention there is a large periphery of vision where the perceptual awareness becomes progressively reduced. Though it might not be possible to see
the details of objects in the periphery, some features, such as motion, can be picked up.

In our research a visual perception system was developed and integrated as a part of NPC architecture for a serious game. In following subsections we present further details for the visual perception system, NPC architecture and knowledge representation.

C. Visual Perception

Our goal is to provide the NPC with a visual perception system close to the way humans perceive their surroundings. The artificial visual perception have the same constrains humans’ perception has. The constrains of the visual perception system are:

1) Limited view angle (horizontal and vertical).
2) Limited view distance.
3) Not to perceive occluded objects.
4) Perceiving partially occluded objects.
5) Lighting effects the visual perception (limit the perception in dark scenes).
6) Center of attention.
7) Small far objects will have higher probability to be missed.

In our implementation for the visual perception system we used a hybrid approach between the ad hoc and computer vision based methods as follows:

- From the computer vision based methods we built multiple levels from (1 to n) of concave retinas in front of the NPC’s eye (i.e camera) as shown in Figure 1. Number of levels represent the view distance while the dimensions of each retina will be the view angle of the NPC. Retina 1 is the closest to the eye and n is the farthest. Each retina is divided into small squares and ray casting technique is used to hit each square of the retinas. Each ray will try to hit the closest retina if it hits an object in the world, it will stop and the NPC will become aware of that object. If it misses, it will go farther to hit the farther square on the farther retina, and will keep going on until it reaches the last retina, in other words it reaches the view distance. The more squares retina will have, the less the probability will be to miss a smaller object.

- From the ad hoc methods we implemented the idea of encoding the surrounding environment. For our serious game we used virtual 3D model for a virtual city called Burgstadt. In this city there are objects that the NPC will sense using his visual perception, some of these objects will be more important for the NPC because they are related to his task and can effect his strategy such as cars in accident, injured people, firefighting equipment and fire, while other objects can be less important and may not effect his task such as cars, people, trees and houses. These objects will be stored (see knowledge representation subsection) along with some farther information about their states, this information is considered part of the visual perception system because the visual perception system in the NPC should provide the NPC with rich information about the surrounding environment similar to the way humans enrich their knowledge using their vision perception.

- An improvement on the visual perception is achieved through increasing number of squares on a specific portion of a retina to mimic the center of attention in humans’ eye.

D. NPC architecture

In order to build NPCs that reflect humans in the real-world they should be composed of a multiple systems that work simultaneously and communicate transparently with each other. We gathered these systems under one architecture and as follows:

- Physical body with animation: NPCs have physical bodies constructed in 3D virtual world. They look like humans and have the possibility to make different types of gestures with their bodies and move themselves by walking or running.

- Camera: NPC have a camera attached to his head, at the position of the eyes. This camera mimics the role of the eye in humans and represent the visual perception system component in the NPC architecture. The position of the camera is the initial position from which the view distance and view angels are calculated, and the ray casting takes place.

- Memory: is the place where the NPC stores information about the environment, objects, states and plans. An NPC has two types of memories:
  - Long-term memory: mainly for storing the domain knowledge which is the information a firefighter and first responder should have in advance regarding his job, such as how to fight a fire in a trash can or how to rescue bleeding person in a car accident. Also, this memory is used to store past events and achieved tasks.
  - Short-term memory: also known as working memory, it is used to store current state of the environment and tasks. New updates from
the visual perception system are stored in this memory. Strategic planning and task selecting is based mainly on this memory.

- Strategic planning: Based on the information gathered from the visual perception system and the information provided in the short-term and long-term memories a task is selected and a list of actions is generated to execute a given plan. Hierarchical task planning is used to generate the plan (see section IV).

- Primitive actions: a plan is a non-primitive task that the NPC should execute at a given time, and it is composed of multiple primitive actions in some order. A primitive action can not be decomposed any farther. Example: Pick up fire extinguisher. Figure 2 shows a diagram for the architecture.

E. Knowledge representation

NPC will need to process data to understand the environment and to make decisions. There are two main types of data that NPC need to handle:

- Environment driven data: This is the data gathered from the surrounding environment either by the visual perception system or by communicating with other NPCs. The representation is based on the Object Oriented Programming (OOP) mechanism, where a class with multiple attributes and methods is developed, and then a list consisting of instantiated objects from this class is prepared. Each NPC has his own list of data, and he is able to manipulate it and update it during each iteration based on the new input from the visual perception system using his own methods. The visual perception system retrieves a list of seen objects, and the NPC updates his current working memory by adding and removing objects to his own list.

- Task driven data: This is the data that comes from the domain knowledge, and it is encoded as a class into the long-term memory of the NPC. A parent class implements the abstract attributes and methods of a general class, children classes of tasks implement the details of each specific task. An NPC has a list of instantiated tasks. While the NPC operating he will be able to update the attributes of task/s and invoke methods.

Both types of data are passed to the strategic planning system in order to be processed and generate decisions (see section IV).

III. STRATEGIC PLANNING USING HTN

The mechanism of controlling behaviors of the autonomous characters described in this paper is so called hierarchical task network (HTN) planning.

A. Hierarchical task network

HTN is an automated planning method in artificial intelligence (AI) in which the hierarchy among tasks can be given in the form of network. It is a collection of tasks that need to be fulfilled, and tested against constraints from the problem domain. The tasks that satisfies its preconditions will be executed. HTN planning has two types of tasks, primitive tasks and non-primitive tasks. Primitive tasks exist in the leaves of the network, that can be directly accomplished when their preconditions are satisfied. Non-primitive tasks, also named as compound tasks, need to be decomposed into further tasks until they reach to the leaves of the task network in order to be accomplished. Primitive and non-primitive tasks are extracted from a planning domain, which contains methods and operators. Methods are used for decomposing non-primitive tasks, while operators fulfill the primitive tasks. To decompose non-primitive tasks, methods can provide several branches. Those branches provide multiple ways to accomplish the non-primitive tasks and each one of these branches composes a sequence of actions. The planner will check whether the preconditions are satisfied according to the current state of the perspective world before executing a method branch or an operator. Executing operators can change the state of the world by adding or deleting a list of states. Each branch of method leads to a task list, which could be either primitive tasks or non-primitive tasks. Recursively executing the methods, until no farther decomposition is needed. If there exists a linear sequence of operators which could accomplish a given task, a final plan is found by the HTN planner. The planner performs a search by decomposing tasks using the domains methods and operators, gathering valid bindings for logical expressions as it proceeds.

B. Typical techniques for HTN

HTN planning is not a new technique, Sacerdoti firstly brought this idea from NOAH [8] in 1975. Introducing this partial order planning mechanism using procedural net which can easily and directly solve problems. Tate [9] continued the work and invented NONLIN. An interdisciplinary approach, originated from Operations Research (OR) and Artificial Intelligence (AI), was developed to improve NOAH in the aspect of nodes interactions and with back-tracking. To use HTN planner not only in the block world, Tate et al. [10] introduced O-plan2 and Wilkins [11] introduced SIPE-2, several practical domains are tested, such as house construction, etc. Erol et al. [12] introduced the HTN planning as theoretical framework and presented a formal syntax and semantics for it. Therefore, a HTN planning algorithm was invented and the algorithm has also been proved to be sound and complete. To avoid the node interactions in the HTN, Nau et al. [13] presented SHOP, which planed tasks in the same order that they will later be executed. This planning algorithm was relatively simple to be implemented and a Python version was already released from the author. SHOP2 [14] improved SHOP by allowing tasks and sub-tasks to be partially ordered.
C. HTN in games

HTN was not only limited to works in the academic realm, but also widely used in industry, especially games. Avila and Hoang [15] presented a strategic planning for Unreal Tournament (UT) bots based on HTN planner. Gorniak and Ian Davis [16] presented an application that controls the behaviors of a group of in-game characters using HTN planner. HTN was used on a well-known game Killzone 2 [17] and its following series. The HTN planner used in the game could generate strategical combat with enemy squads, which provided players a fascinating game experience in the multiplayer games. Kelly et al. [18] developed a system that applied the HTN to video games, and tested in the game Elder scrolls IV: Oblivion. The script generator convert the linear plans, which applied from HTN, to the script language used by the game automatically. And the script activated the real actions in the game.

IV. OUR GENERAL APPROACH

NPCs in serious games, where a scene is already designed and built for an educational purpose, usually follow a hard-coded behavior and execute the same plan, a change in the environment or in the scene will require changing the implementation of the NPC behavior. A more realistic behavior emerges when building NPC that is capable of perceiving the environment and building a plan according to the knowledge he gather. In our approach we integrate HTN planning with the NPC architecture presented in section II. The HTN implementation is an extension of C# Unity3D HTN-Planner (CUHP) [19], which is based on Dana Nau’s Pyhop implementation (an HTN-Planner written in Python) [20] that provides a basic implementation of the SHOP planner. As shown previously in figure 2, NPC perceives the virtual environment using his visual perception system and forward the gathered data to update his memories, we extended NPC architecture with the following five modules to provide our NPC with strategic planning mechanism; controller, world model, domain, interface and HTN-Planner. Figure 3 shows the extended NPC architecture. Our approach is explained in the following subsections through discussing the modules:

A. Visual perception and short-term memory

There are two types of game objects in the virtual environment; task-related objects that are objects important to the NPC task and can effect his decisions, and task-unrelated objects that are objects do not effect the NPC task. When the NPC start navigating in the virtual environment his visual perception system start to gather data, only the task-related objects are necessary and processed. When the NPC sees an object he will have access to the information related to this object, in other words, when the NPC is able to see an object in his field of view he will get the object’s name, and using the object’s name it can retrieve further information about this object these further information is not retrieved by the visual perception system, but it will be encoded and stored in the game engine as described in the knowledge representation in section II. Two examples of the type of information stored about the objects that the NPC will gain when he see a car and an injured person is shown below respectively:

**CAR:**
- Car_location (Transform),
- Is_on_fire (Boolean),
- Is_in_accident (Boolean),
- Number_of_people_inside (Integer).

**Injured Person:**
- Person_location (Transform),
- Is_burning (Boolean),
- Blood_pressure (Integer),
- Heart_beat (Integer),
- amount_of_blood_lost (Integer).

Gathered data (environment driven data) is processed as objects (object oriented) and forwarded to the short-term memory (working memory) where a list of game objects is maintained and updated periodically. Another list of task driven data is maintained holding information about the current task the NPC is trying to complete (see controller). Long-term memory at the current implementation has simple usage; initialize the main task the NPC will have to solve, for example:

HandleCarsAccident.

B. World-model and state manager

Using information stored in the short-term memory, NPC will start developing an understanding of the virtual environment. The world-model module is a representation of how the NPC sees his surroundings, it will be based on the data that the NPC saw. If the NPC visual perception system fails to see an injured person on the ground, then the world-model will not have any information about this person and this will effect NPC’s decision. The world-model has its own methods to process the data coming from the short-term memory and generate useful information that can guide the NPC and help him in making decisions and taking actions to fulfill his task. Examples of these methods are:

CalculateNumberOfInjuredPeople
CalculateNumberOfHeavilyInjuredPeople
CalculateNumberOfSlightlyInjuredPeople

The state manager is nothing but an image of the world-model at a given time. Real-time planning and decision making is based on the latest values of the state manager.
C. Interface

The interface module is necessary to integrate the planner with the world-model, all the information stored in the world-model must be transferred to the planner and the planner should be able to interrupting these information and processing it to generate a plan. Our HTN-Planner is text-based, which implies that the interface must convert the image of the world-model stored in the state manager to a meaningful text understandable by the planner.

D. Controller

Once the planner generates a plan it needs to be executed by the NPC. The plan will contain a list of primitive actions, the generated plan contains a linear operator sequence (see HTN-Planner). Each operator is associated with a function in the controller which controls the NPC characters. These functions fulfill the plan with navigating the game world, playing relevant animations, avoid obstacles and physics controls. The role of the controller is to make the generated plan understandable to the NPC by converting it to primitive actuator commands the NPC should be able to execute in the virtual environment.

E. Domain

NPC will play the role of firefighter or first responder therefore he has to have the required domain knowledge of a firefighter or first responder in order to handle the emergency situations similar to the real firefighters and first responders do in the real-life. NPC should be able to handle variant situations by processing the information he gathered using his visual perception system based on his domain knowledge. The domain knowledge is presented as tasks, and divided into two types: primitive and non-primitive tasks. A primitive task is called an operator in the HTN-Planner and generate primitive actions while non-primitive task is called a method and can be decomposed into another non-primitive task or a primitive task. Both of the operators and methods of the planning problem are implemented in the domain module as functions. Below are examples of operator and method deceleration:

\[
\text{S <Op_name>}(\text{S},\text{<other_args>})\{\ldots\}
\]
\[
\text{List <Me_name>}(\text{S},\text{<other_args>})\{\ldots\}
\]

The arguments of operators should contain the current state, and other relevant variables that need to be checked against pre and post conditions. The operator will be applied if the state satisfied the conditions of the current operators, and the states should be updated after executing the operator and return an updated state with deleting and adding states. Methods contains similar arguments, different values of arguments can lead to different branches of in the methods, namely different decomposition branches in the HTN. A new task list will be returned after executing the method. Figure 4 presents an example. ExtinguishFire is the goal task and two methods are presented to decompose the non-primitive task. Method Extinguisher-fighting(fire) present four steps to fight the fire. Go to the fire extinguisher and pick it up, then go to the fire and extinguish it. The second branch of decomposition this task is to use the method Water-fighting(fire). Go to the fire fighter car and pick up the water pipe, go to the fire position with the water pipe and turn on water supply that can provide water to fighting the fire.

F. HTN-Planner

The implementation of the planner is the same as the work in CUHP [19], which is based on the algorithm in SHOP [13]. Farther details about HTN-planner can be found by referring to the original paper SHOP that describes the domain knowledge representation and planning algorithm.

V. EXPERIMENT: CAR ACCIDENT SCENARIO

KATIE (Katastrophen, Avatare, Technische Simulationen in virtuellen Environments) [21] is a project carried out by our department with the aim to develop a powerful disaster simulation in the form of a serious game, which maps as learning content tasks and rules of civil protection. A specific purpose is to support and improve the education and training of firefighters. The virtual environments is modeled as a 3D city called Burgstadt. Our scenarios are based on real-life accidents happened in Germany selected from two books [22] [23], the accident situation and the domain knowledge used to handle the situation are extracted from these resources.

To evaluate the proposed visual perception system we designed an experiment to measure the ability of system to perceive virtual objects of different sizes, placed in different angels and distances from the NPC. Two sets of virtual objects were prepared, big objects set consists of 30 objects such as cars and persons, and small objects set consists of 50 small objects such as special tools and fire extinguisher. Each set is placed in far and near situations. Far means that the object is placed in a distance more than 50 units from the NPC, while near means the object is placed within 50 units from the NCP. The angle between the perpendicular line from the NPC eyes (forward vector) and the intended object as shown in figure 5 is divided into three categories: [0 - 60], [60 - 90] and [90 - 130]. The visual perception for the NPC was set to a view distance of 100 unit, and horizontal view angle of 130 degree. Each configuration was carried out three times and table I shows the average of seen objects in each case. Last column is the overall average of each configuration. Results

---

1 In Unity3D distance is measured in units. By default, the units in Unity3D are meters. In our experiment we used the default configuration.
show that the propose visual perception system is effective in most cases. The worst two cases were for perceiving small objects while the angle is wide [90 - 130].

For our experiment of the proposed architecture we selected an accident scenario where two cars with high speed collapse from the front. A fire starts between the two cars, people inside the cars are injured, some of them are heavily injured while others are slightly injured figure 6 shows a picture for the scene.

NPC reaches the accident location and starts navigating, his visual perception system starts to gather information about the environment, and feeds it to the working memory and NPC will start developing his world-model. Once he "sees" the crashed cars a trigger will initialize the strategic planning with the main task HandleCarsAccident. Based on the state presented in the state-manager, HTN-planner will decompose the main task using the methods and operators implemented in the domain module and test against the pre/post conditions. A plan will be generated and presented as a linear operator sequence . The controller will interrupt the generated plan and transform it into primitive actions the NPC can execute in the virtual environment. The list below shows an example of generated plan, tasks surrounded by angel practis <> are non primitive tasks generated by methods and need to be farther decompose into another one or more primitive or non primitive tasks. Tasks surrounded by braces {} are primitive tasks generated by operators and can be mapped directly into primitive actions using the controller to be executed by the NPC. For instance the task: <RescueHeavilyInjuredPerson> is decomposed into two other none-primitive tasks: <ExtractInjuredPersonFromCar> and <MoveInjuredPersonToAmbulance>. While <ExtractInjuredPersonFromCar> is decomposed into another non-primitive task <OpenCarDoor> and the two primitive tasks {PullInjuredPersonOutOfCar} and {ApplyFirstAidToInjuredPerson}. Also, <OpenCarDoor> is decomposed into three primitive tasks: {PickUpSpecialTool}, {GoToCarDoor} and {UseSpecialToolToOpenDoor}.

When comparing the generated plan with the domain knowledge on handling the exact accident in real-life situation as presented in the book [23] our approach was able to generate the perfect plan to handle the same accident. The experiment shows the effectiveness of the proposed approach to generate ideal plans at real-time, a change in the environment is reflected as a change in the generated plan.

<table>
<thead>
<tr>
<th>Size, Location</th>
<th>Number</th>
<th>[0 - 60]</th>
<th>[60 - 90]</th>
<th>[90 - 130]</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>big, far</td>
<td>30</td>
<td>90%</td>
<td>90%</td>
<td>83%</td>
<td>87.6%</td>
</tr>
<tr>
<td>big, close</td>
<td>30</td>
<td>100%</td>
<td>100%</td>
<td>93.3%</td>
<td>97.7%</td>
</tr>
<tr>
<td>small, far</td>
<td>50</td>
<td>82%</td>
<td>78%</td>
<td>42%</td>
<td>67.3%</td>
</tr>
<tr>
<td>small, close</td>
<td>50</td>
<td>96%</td>
<td>86%</td>
<td>66%</td>
<td>82.6%</td>
</tr>
</tbody>
</table>

1. <HandleCarsAccident>
2. <RescueHeavilyInjuredPerson>
3. <FightFire>
4. {PickUpFireExtinguisher}
5. {GoToFireLocation}
6. {UseFireExtinguisherToFightFire}
7. {DropeFireExtinguisher}
8. <ExtractInjuredPersonFromCar>
9. <OpenCarDoor>
10. {PickUpSpecialTool}
11. {GoToCarDoor}
12. {UseSpecialToolToOpenDoor}
13. {PullInjuredPersonOutOfCar}
14. {ApplyFirstAidToInjuredPerson}
15. <MoveInjuredPersonToAmbulance>
16. {BringStretcherNearInjuredPerson}
17. {CarryInjuredPersonToStricher}
18. {MoveStretcherToAmbulance}
NPCs behavior in previous work in our research group in the domain of serious gaming was mainly based on a scripted behavior. Where NPCs follow predefined strategic plan. A change in the scenario or the environment will require modifying the implementation of the NPC, which is time and effort consuming task, the NPC architecture is tightly coupled and no possibility for code reusability. NPCs based on finite state machine mechanism will have a limited set of behaviors or tasks that they can be in at a given time, thus the behavior of the NPC can be expected and will be repeated. Using our new modular architecture, the extensibility of the scenarios is a promising direction, through providing new methods and operators from the problem domain. Also, this will open the door for evolving new unexpected plans which would result in emerging more believable behaviors for the NPCs, that will have a positive effect on the believability and thus the successful of our serious game. The assessment of NPCs behavior believability has not been investigated in our research yet, but we are considering this in our future work.

VI. CONCLUSION AND FUTURE DIRECTIONS

Two contributions were presented in this paper; firstly, a visual perception system for non-player characters along with a short-term memory (working memory) are implemented to mimic the human vision and provide the NPC with only limited information about the environment for a realistic behavior. Secondly, an HTN planning approach is implemented based on SHOP in the area of serious games. Instead of hard-coding the behavior of the NPC, where a change in the scene will require changing the implementation of the NPC behavior, the proposed approach provide more realistic and believable NPC behavior. The strategy and the goals of the NPC are generated online, dependent on the surrounding environment and based on what he can “see”. A change in the environment will be reflected on the NPC actions. Farther future directions will be enhancing the visual perception system by including lighting effect and weather conditions (rain, fog). Also, extending the domain module to include domain knowledge for more complicated scenarios so that more complicated plans and behaviors can be generated. Finally, we are planning to design experiments to asses the believability of our NPCs behavior, and its effect as a factor for the success of our serious game through involving some form of human testing, where external observers evaluate the human likeness of the NPCs.

REFERENCES