2011 International Nuclear Atlantic Conference - INAC 2011
Belo Horizonte,MG, Brazil, October 24-28, 2011
ASSOCIAÇÃO BRASILEIRA DE ENERGIA NUCLEAR - ABEN
***ISBN:* 978-85-99141-04-5**

VIRTUAL EVACUATION SIMULATION WITH AUTONOMOUS AVATARS

Antônio Carlos A. Mól[1a,2,3], Ana Paula Legey[2b], Victor Gonçalves G. Freitas[1,2], Vitor Henrique M. G. Soutinho[2], Carlos Eduardo F. Santos[2], Vinicius S. Ventura[2], Luiz B. Montez[2] and Carlos Alexandre F. Jorge[1]

[1] Instituto de Engenharia Nuclear (IEN / CNEN - RJ)
Rua Hélio de Almeida, 75
21941-906 – Rio de Janeiro, RJ
[a]mol@ien.gov.br

[2] Universidade Gama Filho
R. Manuel Vitorino, 553
20740-900 – Rio de Janeiro, RJ
[b]analegey@hotmail.com

[3] Instituto Nacional de Ciência e Tecnologia de Reatores Nucleares Inovadores/CNPq, Brazil

ABSTRACT

This paper describes the use of virtual reality technology for virtual simulation of crowded evacuation from sites. The approach adopted is the reuse of a game engine, thus taking advantage of all its features for virtual environment design. This work upgrades a previously developed one, in which users played simultaneously in a networked environment, each one controlling his or her own avatar. But for crowded evacuation situations, it would require many users playing simultaneously in networked computers. The more crowded the simulation, the more users needed, what could be difficult a task, depending upon the number of avatars needed. Autonomous avatars can surpass this difficulty, so few users can participate, together with as many autonomous avatars as needed, to simulate the desired crowded scenarios. First results show the viability of such an approach.

1. INTRODUCTION

Emergencies can occur in many different environments. Typical ones are industries, which may involve risks for workers or for the general public. Other ones are public environments such as buildings and open public spaces, where a variety of emergencies may suddenly happen, due to natural disasters or human caused threats. There are staffs who are responsible to prepare for these type of situations, training themselves for rescuing people, – as policemen or firefighters –, or for training other people to safely evacuate industries, buildings or similar environments. These staffs usually perform periodic simulation in the real places, to be people prepared for facing emergencies, and also to support evaluation and possible improvements that can be made in emergency response procedures.

Performing training in the real places though, sometimes requires recruitment of a great number of persons, as in the case of training industry personnel or people who occupy buildings, for example. These trainings cause interruption of routine activities in those

environments. Thus, it is interesting to make use of computer-based training practices before training in the real places, for some important reasons:

(i) First, environments such as industrial ones may involve risk or are potentially hazardous for people. Thus, with computer-based training, people can be trained first in a safe environment before entering real ones; when they do it, they would probably have to perform less training repetitions in the potentially hazardous environment, due to skills gained during the computational simulations. It is important to notice though, computer-based training is not intended for eliminating totally training practices in real environment.

(ii) By using first computer-based training, many different emergency scenarios can be performed and analyzed with simple modifications in the computational simulation, what can be easier to perform than in the real places.

(iii) There are high risky scenarios that could be difficult to be simulated in the real places, such as those involving fire, contaminants or other hazardous agents; if not impracticable, due to the danger they could offer people. These scenarios can be easily performed in computational simulations.

Virtual reality (VR) -based simulation find very favorable use for emergency preparedness and response, since people can virtually navigate and interact with both the environment and other people, and virtual environments can be designed with very high similarity with the corresponding real ones. VR comprises techniques for advanced interface development, through which users can navigate and interact, online, in computer generated artificial environments, – sometimes 3D [1]. A very important approach for this purpose is the reuse of game engines, as explained in section 2.

This work continues a R&D for simulation of evacuation from buildings in emergency situations [2]. Currently, a new functionality has been added, that of using autonomous avatars. This later approach lessens the requirement for recruiting great number of persons to perform the simulations. One fact has to be considered in this type of multi-user simulation, when compared to simply playing games: gamers are usually young people highly adapted to playing computer games, but people from a more general public such as workers in an industry or any other company, do not usually have such skills well developed. This would thus require their training for dealing well with the keyboard and mouse, or with joysticks, for performing the virtual simulations. Crowded evacuation simulation would be even more difficult a task, although it is very important for evaluating the environment relatively to crowded exits. Autonomous avatars, guided by some rule, may solve this problem by requiring few well-trained people for performing simulations.

Game engine reuse is described in section 2, along with the particularities of this R&D. Section 3 describes the autonomous avatars' strategy development, while section 5 concludes this work giving also future perspectives.

The simulations take place at some buildings of Universidade Gama Filho (UGF), Piedade campus, specifically the Arquitetura (AR) and Santos Dumont (SD) buildings.

2. GAME ENGINES REUSE FOR SERIOUS APPLICATION

Among the available approaches for virtual simulation, some R&D groups have engaged in the reuse of game engines, because they have already implemented very important features that enable friendly implementation of virtual environments, and have usually low cost for research [3]-[6]. These characteristics are explained in the following:

(i) Game engines perform real-time graphical rendering, in perspective or 3D views;

(ii) They have already implemented the physics needed for simulation, such as Gravity effect and collision handling;

(iii) They are also designed for multi-user simulation, to enable a number of networked users playing it simultaneously, through local networks or Internet connections.

Among the game engines which might be reused for virtual simulation, two of them have been cited elsewhere as very well suited for this task: Unreal from Epic Games and Quake from ID Software [3]. Our staff uses Unreal Engine, but Quake should also work.

Some examples of computer-based simulation related to emergencies and similar applications can be found in [7]-[11]. Other examples of game engine reuse towards serious R&D also for emergencies and similar applications can be found in [12]-[17].

2.1. Unreal Engine

Unreal Engine is free for academic and non-commercial applications, and can be downloaded from the site: http://udn.epicgames.com/Two/UnrealEngine2Runtime.html. This is in fact the Unreal Engine Runtime version 2 that has been used by our staff.

The following sections describe Unreal Engine usage towards the purpose of this R&D.

2.1.1. UnrealEd

Unreal Engine comes with a scenario editor, named UnrealEd, in which users can design new virtual environments. Epic Games do this to enable gamers develop their own gaming scenarios, but researchers can take advantage of this to design serious application environments. It is possible to model environments with high degree of realism. Figure 1 shows UnrealEd interface with an example building model.

One can make designs directly in UnrealEd, by using some embedded tools in this scenario editor. Solid objects can be created with the “Builder Brush” tool, by selecting one of the many geometric models available. Then, using the “Add” tool, one can create the selected object. It is also possible to cut grooves in these objects with the “Subtract” Tool. Figure 2 shows an example of this sequence.

Another alternative is to make designs in CAD software, and then import them into Unreal through UnrealEd, with the “StaticMeshes” tool that consist of 3D meshes that can be replicated as much as needed. This is the best way to represent an object aimed to be replicated [18]. Thus, a designed object such a computer, for example, can be replicated to

full an office with a number of computers. Figure 3a shows an example of this processing, for filling a garden with a plant model. StaticMeshes can be made mobile, such as doors, windows or elevators. This is done by defining them as “Movers”. Figure 3b shows an example of a door creation.

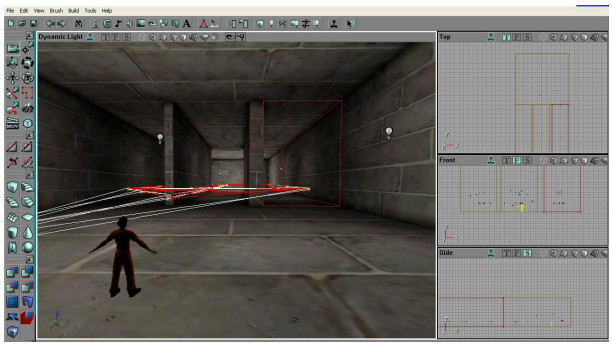


Figure 1. UnrealEd interface with an example building model.

Figure 2. An example of using UnrealEd.

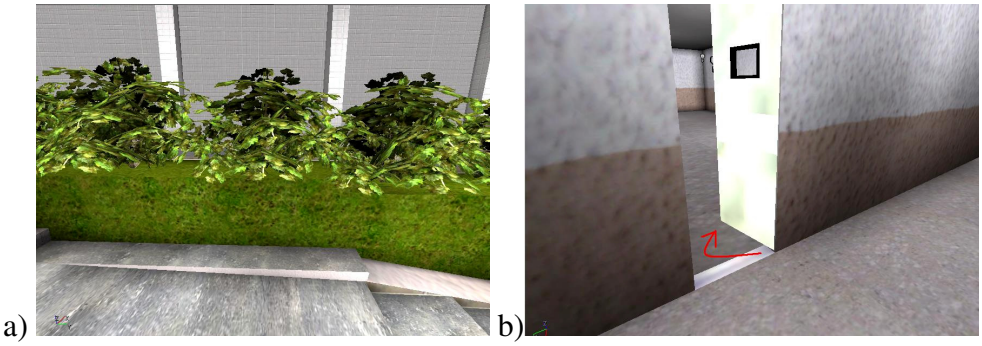


Figure 3. a) An example of StaticMeshes replication; b) an example of Movers’ type StaticMeshes.

2.1.2. UnrealScript

Unreal Engine is written in C/C++ programming language [19], but in this non-commercial version, users have no access to C/C++ source code, but only to a scripting language similar to Java, named UnrealScript. But this enables researchers to modify the Engine, what is sufficient for the creation of serious simulation applications.

Unreal Engine is written following object oriented approach, so there are existing classes that comprise characteristics of such an implementation, as classes' hierarchy, inheritance, and so on. By using UnrealScript, one cannot modify existing classes, but can instead replicate existing ones, and modify these later for the tasks needed. Figure 4 shows a general view of the classes used in our R&D, up to the present. This diagram comprises the classes for different applications, thus the color legend help differentiate among those applications, which classes are general application ones, and which ones are specific for evacuation simulation and for autonomous avatars feature implementation.

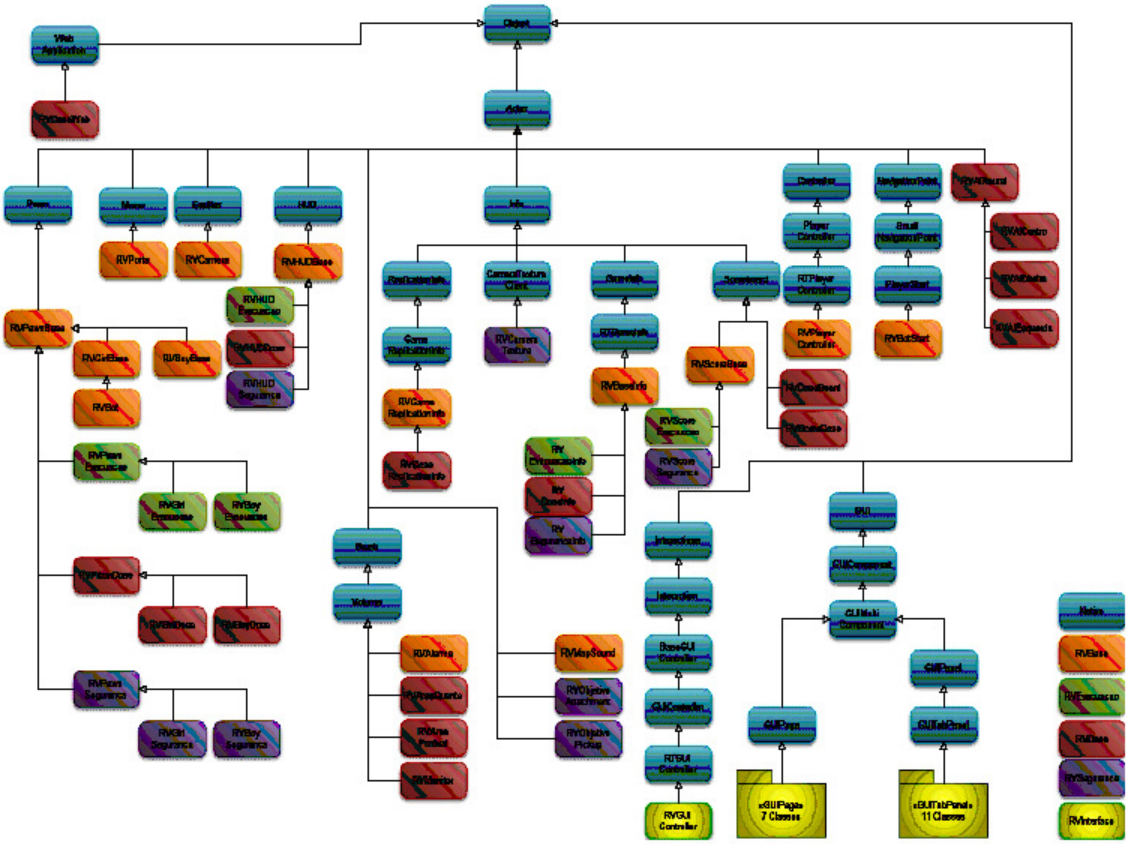


Figure 4. Unreal classes diagram.

UnrealScript programming can be done in the following steps:

1- Package creation: Creation of a directory (folder) and subdirectories' structure of a program that will be done;

2- Coding: Creation of programs that are part of the package, including *.uc source code (uc is the extension of source codes made in Unreal Engine);

3- Editing file *.ini: Enables the interpreter to identify the created package;

4- Compiling: Program execution to generate an intermediary file representing the package;

5- Debugging: Verify possible errors and fix them.

3. IMPLEMENTATION

3.1. Scenario Development

This section shows the scenario development, using UnrealEd functionality. As already mentioned in section 1, some buildings of UGF have been modeled: AR and SD buildings. This has been done with the aid of buildings architectural plants, for their real dimensions' collection. Then, these have been input into Unreal through a scale conversion scheme from meters to Unreal points of 1:60.532, to keep proportion between buildings and person's dimensions and walking velocity, to achieve realistic simulations.

The modeling process has begun with a great groove cut with the "Subtract" tool, to define the place where both buildings should be placed. Then, the buildings have been virtually constructed, from their collected dimensions, considering multi-floor information. Photos collected in the real places have been used as textures pasted to the virtual buildings' walls, to result in as much as possible realistic views, to achieve good immersion for users. Figure 5 shows a comparison between a real building's photo, in Figure 5a, and its corresponding virtual model in Figure 5b.

a) 

b) 

Figure 5. a) A real building's photo; b) its corresponding virtual model.

3.1. Autonomous Avatar Development

Autonomous avatars, also known as "Bots", add good simulation capabilities to cases of evacuation in emergency situations, as treated in this paper, as already explained in section 1. Crowded scenarios can be modeled and simulated, with the need of few participants. Thus, only few user-controlled avatars take part in the simulation, with as many other autonomous avatars as needed. These later have their behavior defined by rules that guide them through some paths along the virtual building.

In this work, the rules are comprised by node points within the building, named "PathNodes", and paths defined by a PathNodes' sequence. Characteristics of these PathNodes are described in the sequel. First, the Bots are not capable of interacting with PathNodes out of their field of view. Thus, intermediary PathNodes must be defined in such a way paths may be traced. This is illustrated in Figure 6: Figure 6a shows a PathNode out of the Bot's field of view that is not connected to form a path. Figure 6b shows, instead, the path formed when an intermediary PathNode (represented as a green apple) is created.

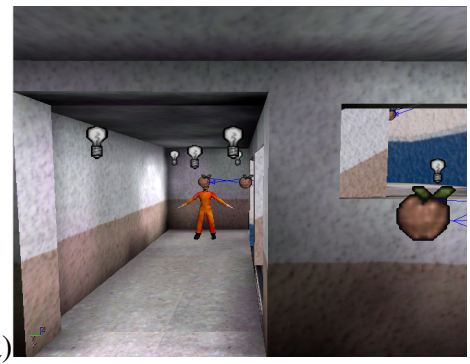
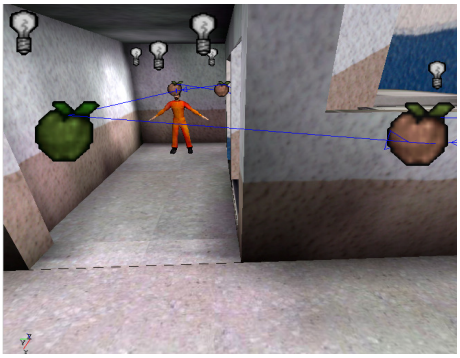
a)  b) 

Figure 6. Illustration of paths formation; a) PathNode out of Bot's field of view: no path; b) Intermediary PathNode added: path formed.

Second characteristic is the sequence of PathNodes through which Bots must walk. When user creates paths, multiple paths may be created, due to the presence of multiple exits within the buildings, or due to multiple PathNodes along one way. Experience showed some paths are correct ones, while other may be wrong, such as paths guiding a Bot back to entering the building, when it should in fact exit it. There are other cases when, among multiple paths formed, some are shorter than others, the shorter ones preferred, for obvious reasons. Other times, some formed paths are directed along a corridor, for example, while other paths may involve entering an intermediary location, when this would be really not desired, because of the resulting longer time to exit.

Therefore, some paths are defined as higher priority ones, over other lower priority paths. Thus, Bots will try to follow high priority paths first. They will take lower priority paths if the higher one is obstructed. Figure 7 illustrate this situation. The Bot of interest in this example is the leftmost one in Figure 7a, and it should reach the point indicated by a red

circle, during evacuation. It should thus follow one direct path, among the many possible paths formed by the PathNodes. But if no priority is defined in the paths it should follow, it could even turn left (considering its reference), taking the paths corresponding to the other rightmost Bot in this figure. Figure 7b shows priority paths though color legend. The yellow path is the higher priority one, as being a direct path from the Bot's current position to the red circle, while blue and red ones are lower priority paths.

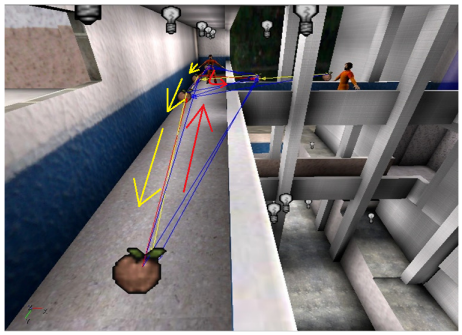
a)  b) 

Figure 7. Multiple paths and paths' priorities; a) Multiple paths; b) Higher (in yellow) and lower (in red and blue) priority paths.

4. RESULTS

Some exit routes have been defined for comparative analysis between real and simulated evacuation procedures. Six locations have been strategically chosen, due to crowded people flow, observed in the real buildings. From these locations, routes have been defined, all converging to the AR building's entrance. The SD building's entrance was under repair at the time the simulations were performed, and was thus blocked. Table 1 shows the defined evacuation routes' descriptions, while Table 2 shows a comparative analysis between elapsed evacuation times in the real and simulated experiments, where times are expressed in the format minutes : seconds : hundredths.

Table 2 shows a good concordance between real and simulated elapsed times during evacuation experiments in all routes. This validates the virtual environment developed with Unreal Engine for evacuation simulation in emergency situations.

Table 1. The evacuation routes.

Route	Description
Route 1	Undergraduate Scientific Research Lab
Route 2	Computer Science Coordination
Route 3	Faculties' Room through AR building
Route 4	Faculties' Room through SD building
Route 5	Fourth floor through AR building
Route 6	Fourth floor through SD building

Table 2. Comparative analysis between elapsed times.

Route	Elapsed times in real experiments	Elapsed times in simulated experiments
Route 1	01:15:04	01:14:19
Route 2	01:01:27	00:57:18
Route 3	01:33:48	01:32:35
Route 4	01:55:52	01:54:21
Route 5	01:38:90	01:42:32
Route 6	01:55:65	01:58:61

3. CONCLUSIONS

Unreal Engine is an important platform for developing virtual environments, due to its friendly interface with users for both scenario modeling and functionality programming, through its UnrealEd and UnrealScript language, respectively. Users can thus take advantage of all Unreal's simulation capabilities, as efficient dynamical rendering, physics representations and multi-user simulation capabilities. Further, this work showed that autonomous avatar usage improves simulation for crowded situations, with few participating users. This lessen the need for user's training for dealing with the gaming interaction, as well as the need of recruiting a large number of users to participate in crowded simulations.

ACKNOWLEDGMENTS

This research was sponsored by Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro – FAPERJ and Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq.

REFERENCES

1. C. Kirner, M. S. Pinho, *Introdução à Realidade Virtual*, Short-course book, 1º Workshop de Realidade Virtual, São Carlos, SP, Brazil, (1997).
2. A. C. A. Mól, C. A. F. Jorge, P. M. Couto, "Using a game engine for VR simulations in evacuation planning", *IEEE Computer Graphics and Applications*, **Volume 28**, N. 3, pp.6-12 (2008).
3. M. Lewis, J. Jacobson, "Introduction," *Communications of the Association for Computing Machinery (CACM)*, Special Issue on "Game Engines in Scientific Research", **Volume 45**, pp.27-31 (2002).
4. A. Rosenbloom, "Introduction," *Communications of the Association for Computing Machinery (CACM)*, Special Issue on "A Game Experience in Every Application", **Volume 46**, pp.28-31 (2003).
5. M. Zyda, "Introduction," *Communications of the Association for Computing Machinery (CACM)*, Special Issue on "Creating a Science of Games", **Volume 50**, pp.26-29 (2007).
6. D. Trenholme, S. P. Smith, "Computer game engines for developing first-person virtual environments," *Virtual Reality*, **Volume 12**, pp.181-187 (2008).
7. N. Pelechano, N. I. Badler, "Modeling crowd and trainer leader behavior during building evacuation", *IEEE Computer Graphics and Applications*, **Volume 26**, N. 6, pp.80-86 (2006).
8. D. Helbing, I. Farkas, T. Vicsek, "Simulating dynamical features of escape panic", *Nature*, **Volume 407**, pp.487-490 (2000).
9. N. Pelechano, K. O'Brien, B. Silvermann, N. Badler, "Crowd simulation incorporating agent psychological models, roles and communication", *Proceeding of the 1st International Workshop on Crowd Simulation (V-CROWDS'05)*, EPFL, pp.21-30 (2005).
10. S. R. Musse, D. Thalmann, "Hierarchical model for real time simulation of virtual human crowds", *IEEE Transactions on Visualization and Computer Graphics*, **Volume 7**, pp.152-1694 (2001).
11. D. L. Tate, L. Sibert, T. King, "Using virtual environments to train firefighters", *IEEE Computer Graphics and Applications*, **Volume 17**, N. 6, pp.23-29 (1997).
12. J. Wang, M. Lewis, J. Gennari, "USAR: a game-based simulation for teleoperation", *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society*, Denver, CO, USA, pp.493-497 (2003).
13. J. Wang, M. Lewis, J. Gennari, "Interactive simulation of the NIST USAR arenas", *Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics,* Washington, DC, USA, pp.1350-1354 (2003).
14. J. Manojlovich, P. Prasithsangaree, S. Hughes, J. Chen, J., M. Lewis, "UTSAF: a multi-agent-based framework for supporting military-based distributed interactive simulations in 3D virtual environments", *Proceedings of the 2003 Winter Simulation Conference*, New Orleans, LA, USA (2003).
15. R. Adobbati, A. N. Marshall, A. Scholer, S. Tejada, G. A. Kaminka, *et al.*, "Gamebots: a 3D virtual world test-bed for multi-agent research", *Proceedings of 2nd International Workshop on Infrastructure, MAS and MAS Scalability*, 2001.
16. G. A. Kaminka, M. M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, *et al.*, "Gamebots: a flexible test bed for multiagent team research", *Communications of the ACM*, **Volume 45**, pp.43-45 (2002).
17. B. G. Silverman, G. K. Barathy, K. O'Brien, J. Cornwell, "Human behavior models for agents in simulators and games: Part II gamebot engineering with PMFserv", *Presence: Teleoperators and virtual environments*, **Volume 15**, pp.163-185 (2006).

18. J. Busby, Z. Parrish, J. V., *Mastering Unreal Technology: The Art of Level Design*, Sams Publishing, 2005.
19. J. P. Flynt, B. Booth, *UnrealScript Game Programming for Teens*, Thomson Course Technology, 2006.