An Intelligent Agent Who Teaches the Basic Operations with an Abacus inside Virtual World "Second Life"

Israel Guzmán, Darnes Vilariño, María J. Somodevilla, and Fabiola López

Benemérita Universidad Autónoma de Puebla, Mexico
ig89852005@hotmail.com, {darnes, mariasg} @cs.buap.mx, fabiola.lopez@siu.buap.mx

Abstract. Virtual worlds become more popular day by day. For many people virtual world comes to be a work place or a place for knowledge acquisition or entertainment. Universities have joined the virtual worlds by offering common places to students and professors for education-learning process. In this work, we implement our algorithms as agents in the virtual world. The agent, named Israel, should be capable to provide skills to an avatar (also named Israel). Consequently, avatar Israel will teach abacus operations to other avatars. Emphasis is placed on natural language processing techniques in parallel using CUDA.

Keywords: Natural Language Processing, Mobility, Second Life, Virtual World, Theory of Agents, Metaverse, Parallel, CUDA.

1 Introduction

Virtual worlds become more popular day by day. For many people virtual world comes to be a work place or a place for knowledge acquisition or for entertainment. Universities have joined to the virtual worlds by offering common spaces to students and professors for education-learning process [11].

A metaverse is a virtual world. Second Life (SL) [4] is a metaverse developed by Linden Research Inc. It was launched in 2003 and from 2006 has gained a crescent international attention. SL was inspired by "Snow Crash" science fiction novel written by Neal Stephenson and the literary movement "Cyberpunk" [1].

Agents' theory is booming from the last days and it is inside of education-learning process, financial and TAC CAT y TAC SCM [2] market simulations and information retrieval among others. One of the most ambitious projects of the last years is the Edd Hifeng [3] and it is a result of the intelligent software development incorporation to the virtual world. A group of researchers from Rensselaer Polytechnic Institute is working to change that by engineering characters with the capacity to have beliefs and to reason about the beliefs of others. The characters will be able to predict and manipulate the behavior of even human players (avatars), with whom they will directly interact in the real, physical world, according to the team [5].

Following this research line, this work proposes an intelligent agent development be able to manage an avatar and also be able to teach abacus the basic operations (sum, subtraction, multiplication and division) to other avatars inside of Second Life. This agent has four basic modules to provide the avatar with capabilities of

Advances in Soft Computing Algorithms
Research in Computing Science 49, 2010, pp. 165-176

communication, visibility, mobility and interaction with other avatars in the virtual world; these four modules are presented in Figure 1.

Module 1. Natural Language Processing. Agent responds messages from other avatars, according the situation and the receiving text. The NLP uses CUDA (Compute Unified Device Architecture) is a parallel computing architecture developed by NVIDIA.

Module 2. Reasoning. Agent Israel should be able to determine the virtual world state and then chooses which action to carry out.

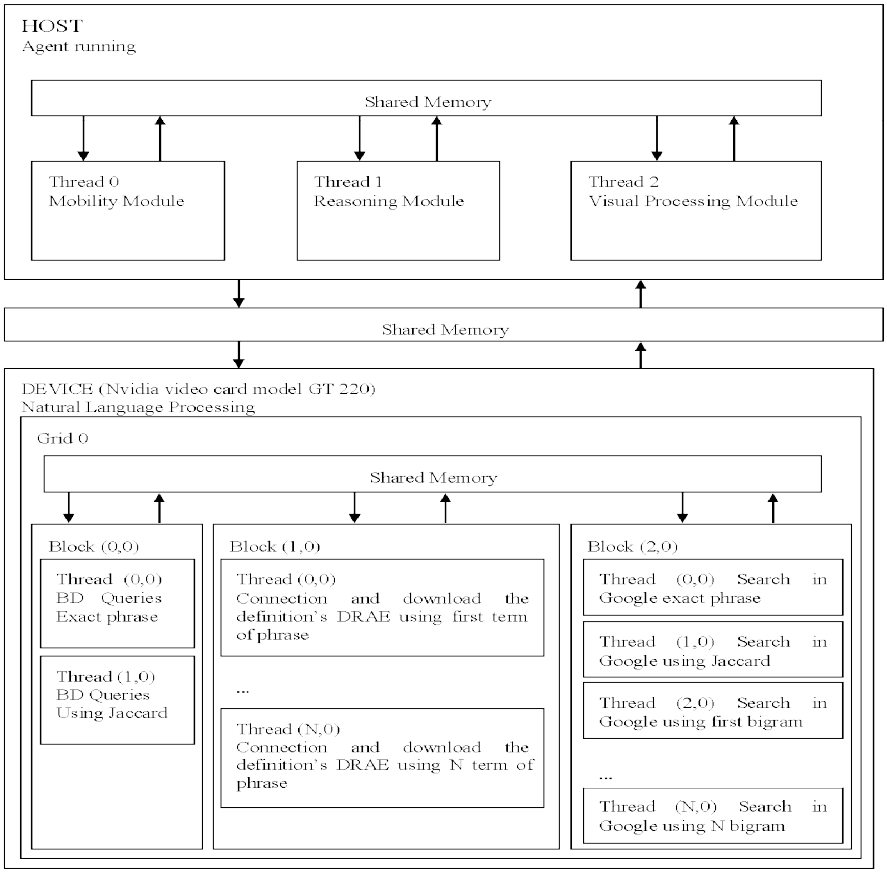

Fig. 1. General architecture of the agent [15].

Module 3. Visual Processing. Agent processes information coming from two sensors, in order to determine the direction to be taken for the avatar which it is controlling.

Module 4. Mobility. It is used for visual processing, since agent could not find an avatar inside its visual field, it has to move to another area.

2 Description of Modules

Agent's algorithms developed for each module are described in the following sections.

2.1 Natural Language Processing Module (NLP)

This module interacts with a database in order to process natural language. A class to permit the connection with the database was created. This class defines methods for receiving and sending messages in the virtual world. For receiving each message the agent starts a kernel (using CUDA) that handles the natural language processing the kernel starts a grid with three blocks (see Figure 1).

NLP module uses eSpeak 1.4.3 [7] to make a conversion from text to voice (only in Spanish), thus the agent is endowed with voice, and also solicitudgetCLR 1.0.0 is used to make http connections with (Google and DRAE (Dictionary of the Real Academic Spanish)).

2.1.1 Receiving and Sending Messages. Agent responds messages from other avatars, in doing that, it should be able to identify the avatar with who it is talking to, since agent can receive messages not only from avatars but from objects as well. Platform provides public or private channels for sending and receiving messages, both managed by Easy Talk protocol [6]. The protocol is implemented using only one string whose parameters are: channel, id, name, and message. Once the agent had the message, it will process it. The agent searches in its database for a message to respond to the received message.

2.1.2 Database. Agent manages a database which was designed according the entity relationship model in Figure 2.

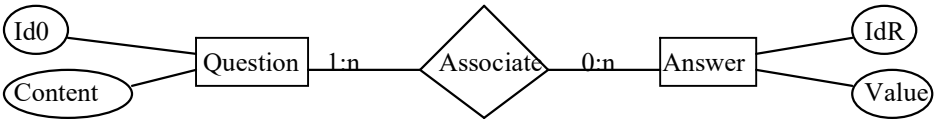

Fig. 2. Entity relationship model for natural language processing database.

The database in figure 2 is normalized until third normal form, in that way data integrity is totally guarantied. Agent Israel updates the database using algorithm 1.

```
Algorithm 1: Update of the DB
Step 0: Receive a message, and extract its terms-
Starts kernel(searchers threads)(terms-), using CUDA
Thread (0,0) databases queries
Thread (0,1) connection and download the definition's DRAE. Thread (0,2)
connection and download the search's Google. FLAGS[3];
If (FLAGS[0]) then Kill(Threads()).
If (FLAGS[1]) then Kill(Threads()).
If (FLAGS[2]) then Kill(Threads()).
Thread (0,0)
```

```
String string = recv(message)
Answer = For each (Question.IdO=associate.IdO) and
(associate.IdR=Answer.idR) where(Question.content = string)
If (Answer <> null) then sendMessage(Answer); activate(FLAGS[0]);
Finish.
Otherwise continue;
  Terms = Process(string) // database enhancement
  flag = 0;
  For each Oj in Question do:
    Jaccard_Sym(Term,Oj)= |o_i ∩ o_j| / |o_i ∪ o_j|
    if (Jaccard_Sym(Term,Oj) >= 0.5) then:
    flag = 1
    New_Answer = For each (Question.IdO= associate.IdO) and
(associate.IDR=Answer.IDR)
      where ( Question.content = Oj).
  End For
If (New_Answer <> null) sendMessage(New_Answer); activate(FLAGS[0]);
Finish.
If (flag == 0) then:
Synonyms_list=search(WordNet, Oi)
For each Ok in Synonyms_list do:
  Answer_list = For each (Question.IdO= associate.IdO) and
(associate.IDR=Answer.IDR)
               where ( Question.content = Ok)
  End For
End For
If longitude (Answer_list) > 1 then
  Greater = 0.5; Term = "";
  For Each Ti in Answer_list do:
    For each Oj in Question do:
      Jaccard_Sym(Ti,Oj)= |T_i ∩ o_j| / |T_i ∪ o_j|
      If (Jaccard_Sym(Ti,Oj) >= Greater ) then:
        Greater = Jaccard_Sym(Ti,Oj)
        Term = Ti
     End For
   End For
   New_Answer = For each (Question.IdO= associate.IdO) and
(associate.IDR=Answer.IDR)
   where ( Question.content = Term)
   If (New_Answer <> null) sendMessage(New_Answer); activate(FLAGS[0]);
Finish. End If
End If //flag =0
```

Thread (1,0)

```
  String string = recv(message)
Terms = Process(string) // search in RAE
For each TMi in Terms
  Category = gramatical_category(TMi) // in Spanish Real Academy
  New_phrase= ∪ (TMi ), ∀ TMi ∈ {adjetive, substantive, verb, adverb}
End For
// Create New_phrase using DRAE.
  If (Check(FLAGS)) then
    updateDB();
      sendMessage(New_phrase);
      activate(FLAGS[1]);
```

```
    Finish.

Thread (2,0)
   Step 1: Search with Google for the String definition and
   Step 2: If it is found return it as Answer and Finish.
   Step 3: It uses the New_phrase and search with Google,
         If it is found return it as Answer and finish.
   Step 4: Searching similar phrases to the New_phrase by constructing
bigrams phrases
           using the New_phrase terms. Pick the bigram with the
greatest degree
         of Jaccard similarity and greater than 0.5.
Step 5: sendMessage(New_phrase); activate(FLAGS[2]);Finish.
```

2.2 Reasoning Module

Agent Israel applies deductive reasoning to react to diverse avatar's behavior. Logic is used to codify a theory which establishes the best action to be taken for a given situation. A set of rules and a logic database were defined in order to describe the world current state and the set of actions to be performed by agent (see Table 1 for a complete set of predicates).

Table 1. Predicates of agent Israel.

Predicate	Description
LocateI(Isle)	Agent is located in an (Isle)
Locate(X,Y,Z)	Agent is located in (X,Y,Z)
Facing(d)	Agent is facing to direction (d)
Teach(j)	Agent is teaching abacus' operations to an avatar (j)
Leisure()	Agent is in leisure state ()
Move(X,Y,Z)	Agent is moving to (X,Y,Z)
MoveI(Isle)	Agent goes to the (Isle)
Search(j)	Agent searches an avatar (j)
Identify(u)	Agent identifies objects or avatars (u)
RMessage(m)	Agent receives messages from avatars(m)
SMessage(n)	Agent send a message to other avatars(n)

Using predicates in Table 1 possible actions of the agent are defined as follows:

Ac = {move_forward, move_back, turn, write_Message, send_Message, teletransport, interact, sum, subtraction, multiplication, division}

Nine rules were defined using these actions of the agent:

Rule 1. LocateI (I) ^ Leisure() ^ ¬Search(A) ^ ¬Identify(A) > Do(teletransport)

Rule 2. LocateI (I) ^ ¬Leisure() ^ Search(A) ^ ¬Identify(A) ^ Move(X,Y,Z) > Do(move_forward)

Rule 3. LocateI (I) ^ ¬Leisure() ^ Search(A) ^ Identify(O) ^ Move(X,Y,Z) > Do(move_back)

Rule 4. LocateI (I) ^ ¬Leisure() ^ Search(A) ^ Identify(O) ^ Move(X,Y,Z) > Do(turn)

Rule 5. LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ RMessage(M) ^ Facing(D) ^ Locate(X,Y,Z) > Do(write Message)

Rule 6. LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA1) ^ RMessage(M) > Do(sum)

Rule 7. LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA2) ^ RMessage(M) > Do(subtraction)

Rule 8. LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA3) ^ RMessage(M) > Do(multiplication)

Rule 9. LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA4) ^ RMessage(M) > Do(division)

Where ^ means "and", V means "or", > means "then" and
I = {I1, …, In } accessible isles
A = {A1, … , Am} visible avatars
(X,Y,Z) = {(0,0,0),…,(a,b,c) } possible coordinates inside an isle
O = {O1, …, Ok} accessible objects inside an isle
M = {M1, …, Mp} messages from public channel
D = {north, south, east, west, northeast, northwest, southeast, southwest}
Op = {sum, subtract, multiplication, division} abacus possible operations
MA1 = {teach to sum}
MA2 = {teach to subtract}
MA3 = {teach to multiply}
MA4 = {teach to division}

2.3 Visual Processing Module

Agent Israel resides in the server and continuously tries to detect an avatar arrival in order to begin the learning process. SL provides some sensors to establish the agent's visibility range. These sensors return an avatars and objects list found in a predetermined range. Algorithm 2 is applied by the agent to provide visibility to avatars.

```
Algorithm 2: Vision_Agent()
Boolean taught=false;
Step 1: For each sensor i. (i = 1,2)
          J=0 , K=0; initial_minute= Time (Minute);
          final_minute = initial_minute + 3;
          While (!Empty (List. Sensor (i))
            and   abs(minute_final – inicial_minute <=3)
              AV [J] = Extract_Avatar (list)
              Object [K] = Extract_Object(list)
              taught=Move(AV[J], Object[K])//applying Algorithm 3
               J++;
               K++;
               Final_Minute = Time(Minute);
          End while
Step 2: If (!taught)
           Teletransport();
```

It is important to note, that sensors are the ones which really provide visibility to the avatars, when an object, created inside virtual world in a defined action field, is

registered. Once, avatar Israel simulates to see any other avatar tries to move toward it.

2.4 Mobility Module

Avatar should be continuously moving inside a determined action range. In doing that, it moves from an initial point within a prefixed angle, as it is shown in Algorithm 3.

```
Algorithm 3. Move (Avatar, Object)
Step 1. Global variable coordinate
Step 2. Thread 1 (establish direction)
         While (!Found (Avatar))
                coordinate = Establish_Direction(Avatar)
         End while
         If(Found(Avatar))
            Teach Abacus_operations()
             Return true;
Thread 2 If (verify(coordinate)) walk(avatar, coordinate, move_forward)
Thread 3 If (verify(coordinate)) walk(avatar, coordinate, move_back)
Thread 4 If (verify(coordinate)) walk(avatar, coordinate, move_right)
Thread 5 If (verify(coordinate)) walk(avatar, coordinate, move_left)
Thread 6 If (verify(coordinate)) walk(avatar, coordinate, turn_right)
Thread 7 If (verify(coordinate)) walk(avatar, coordinate, turn_left)
```

Avatar walks from (x,y,z) position, in four different directions (northeast, northwest, southeast, southwest), to have visited all the avatars found in the sensors' list. In case avatar does not find any other avatar inside its action range, agent has to teletransport to another isle (any other server). In certain cases, it will be necessary agent moves to another metaverse using the Open Grid Protocol (OGP) [8]. OGP was created in Summer 2008 by IBM and Linden Labs. Using OGP it is possible an avatar moves from a metaverse to another, for example from OpenSim [9] to Second Life. OGP consists of a set of protocols, among them count OGP Authentication and OGP Teleport.

3 Abacus Management Algorithm

To understand algorithm 4 is needed to know, abacus is split into two sections. One of the sections only has two beads, named upper_beads; at the other side, the other one has at most five beads, named lower_beads.

It is important to note abacus has 49 beads, along its 7 columns. Each lower_bead at the first column has a value of 0.01, and each upper_bead a value of 0.05. Second column represents 0.1 for the lower_beads and 0.5 for the upper_beads. Third column represents a value 1, and so on both types of beads increase their values multiplying them by 10.

```
Algorithm 4.
Step 1: Agent initializes quantity to 0.0
Step 2: Agent receives and recovers Message(Message,Type).
        If (Type = abacus)
```

```
        Recover value (bead)
        Update environment variables
    If not
         Analyze Message, using LNP Module
         Determine operation type (sum, subtraction, multiplication,
                                   division or representation)
        If (operation_type = representation)
           extract from Message its representation number,
           after that reset abacus.
        Otherwise,
           represent the operation by moving beads
           and follow abacus rules.
```

3.1 General Algorithm

Algorithm 5 is built from algorithm 4 and agent's actions through its four modules.

Algorithm 5:

```
Step 1: Login virtual world, using avatar Israel (for default).
Step 2: Apply Algorithm 2.
Step 3: After a while close application.
```

Algorithm 5 calls algorithm 2 just one time, since algorithm 2 will control totally the application from now on.

4 Discussion

Israel agent uses Visual C++ 2008 programming language since it provides facilities for using Second Life visor. Sensors must be placed inside virtual world because they are programmed in LSL language [10].

For debug testing an OpenSim server was installed due to its similarity with Second Life. Also, the agent was run inside Second Life for final testing, and the results were the same that inside OpenSim [9]. OpenSim was installed in one computer whereas a Second Life visor, controlled by a human, was running in another computer, meanwhile, the agent was running in another different computer.

In this section some examples of use according functionality of the agent´s modules are provided.

4.1 Visibility and Mobility

In order to reach visibility, some messages are sent between the agent and sensors. Messages are received in asynchronous way; considering times for Internet connection and server execution scripts as the two factors affecting this communication.

During testing, the agent performance was medium, due to the influence of the two factors previously mentioned, showing a little delay on sensors updates which causes a little shift between avatars (see figure 3).

4.2 Natural Language Processing

A thread is created to process each message from the virtual world, in that way, N threads are created to process messages coming from objects and avatars. Creation of threads is dynamic and the number of active threads depends on the power of the computer where the agent is running. Therefore, agent is able to respond many messages at the same time, but not necessarily at the same order in which they were received, since each thread consumes a different amount of time, depending on information it has to process (see Figure 4).

Fig. 3. Avatar Israel (man shaped) controlled by the agent.
The agent is trying to place avatar Israel close to avatar Darnes.

The agent consumes much time for Internet connection but results are still satisfactory. As an example the agent can respond to answers like this: *¿De qué color es el caballo blanco de Napoleón?*[1] The given answer is "*pues na guita es blanco son preguntas de agilidad mental y capciosas*", as you can see the right answer would be "es de color blanco". However, answer given by the agent was the one found on Internet using Google, and it is a good approximation to the real answer.

In spite of eSpeak is used to endow with voice to the agent, still it presents pronunciation errors, but they can be fixed improving eSpeak, since it is an open source software.

[1] In English *What colour is the white horse's Napoleon*? In this moment the agent just can speak Spanish. In the future the agent could speak other languages.

Fig. 4. Avatar Israel (man shaped) controlled by the agent.
The agent responds an avatar Darnes question where avatar Darnes is controlling by a human.

Fig. 5. Avatar Israel (man shaped) controlled by the agent.
The agent is representing the number five on the abacus.

4.3 Use of Abacus

The agent uses EasyTalk protocol to manage the abacus. Agent is able to know the abacus state (see Figure 5), but it is too much time consuming to update this state, due to the factors already mentioned in section 4.1. This elapsed time depends totally on the Second Life server and it cannot be measured, however the agent can update abacus state, but that delay, not controlled by it, in certain occasions can lead to misinterpret the abacus state.

The agent can process multiple abacus messages at the same time, as we said before. This is necessary in order the agent keeps its state as current as possible.

5 Conclusions

Some experience was available about agents development under JADE [16] platform. However, agents development inside a virtual world is totally different, since it is necessary the help of sensors where efficiency depends on sending and receiving messages, and continuous changes in virtual world which should be detected by agents as soon as possible. Another important aspect to consider is the fact, that this type of application is very high resources consuming.

Searching on Internet was a new feature annexed to NLP module in order to get better results. However, the agent response time was affected significantly. There are exist tree different agent versions considering the implementation way of the NLP module. The version 1, which is the most stable, has been described in

this paper. However, the other two have been very useful as a comparison point. Version 2 uses Gecko for the

agent internet connections when it should answer a question posted by an avatar which is managed by human. This protocol is more reliable, but it is very high resource consuming. Version 3 uses DDL for database connection, which causes more running time and more resources are needed.

6 Future Work

In order to improve the system capabilities, a 3D vision system is proposed. This substitution benefits teletransportation of the agent, since it no longer will depend on objects which have been programmed in LSL, then agents can teletransport between metaverses using OGP protocol.

Context analysis can be incorporated in the NLP module. Undoubtedly, the agent will increase its response time.

Another improvement respecting to NLP module would be about solicitudgetCLR (for Google connections). This improvement would allow a better analysis on Web pages or another option would be replace completely this part by Gecko [14]. Undoubtedly, Gecko would be the best option. However, a great amount of computational resources would be required. An agent beta version showed these results by using this navigational engine.

At this time the agent already has an own voice but is not able to analyze audio yet. This functionally could be reached by using eSpeak but making improvements to this application in order it could be used by the agent in an adequate way. Besides, it should take in consideration that OpenSim, Asterisk [12], and other additional modules for the correct operation of audio in the virtual world need to be installed. This installation could be necessary in order to understand how audio sending works in Second Life and so could make a better design and implementation of the needed algorithms.

Finally, mobility algorithms could be improved in order avatar mobility looks more human instead of so robot appearance.

References

1. Definition of Metaverse. http://en.wikipedia.org/wiki/Metaverse (2010)
2. Reyes-Farfán, N., Sánchez, J. A.: Personal spaces in the context of OAI. In: Proceedings of the Joint Conference on Digital Libraries (JCDL 2003), Houston, Texas (2003)
3. Bringsjord S., Shilliday A., Clark M., Werner D., Taylor J., Bringsjord A., Charpentier E.: Toward Cognitively Robust Synthetic Characters in Digital Environments. In: Artificial General Intelligence 2008, Proceedings of the First AGI Conference. 171 (2008)
4. Virtual World Second Life. http://www.secondlife.com
5. Conference on Artificial General Intelligence. http://www.agi-08.org/ (2010)
6. Protocol EasyTalk. http://wiki.secondlife.com/wiki/LSL_Protocol/EasyTalk (2010)
7. Speech SyntheIfzer. http://espeak.sourceforge.net/ (2010)
8. Protocol OGP, http://wiki.secondlife.com/wiki/OGP_Base (2010)
9. What is OpenSim? www.openIfmulator.org (2010)
10. Linden Scripting Language. http://wiki.secondlife.com/wiki/LSL_Portal (2010)
11. Institutes in Second Life. http://secondlifegrid.net/casestudies (2010)
12. Voice over Internet Protocols. http://www.asterisk.org/ (2010)
13. Parallel Computing NVIDIA. http://www.nvidia.es/object/cuda_home_new_es.html (2010)
14. Navigational Engine Firefox. http://developer.mozilla.org/es/Gecko (2010)
15. Characteristics of multiprocessors and cores of the video card GT 220 de Nvidia. http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf (2010)
16. JADE (Java Agent DEvelopment Framework), http://jade.tilab.com/ (2010)