An Emotional Engine for Behavior Simulators

Santiago García Carbajal[1*], Fabio Polimeni[2], Jose Luís Múgica[3]
[1]*University of Oviedo, sgarcia@uniovi.es
[2,3] Signal software S.L.
{jlmugica, fpolimeni}@signalsoftware.es

Abstract

Interpreting, modeling and representing emotions is a key feature of new generation games. This paper describes the first version of the Emotional Engine we have developed as a component of more complex behavior simulators. The purpose of this module is to manage the state and behavior of the characters present in a scene while they interact with a human user. We use preexistent language recognition libraries like Windows™ Speech API, and Kinect™ devices to communicate real humans with artificial characters participating in a virtual scene. The Emotional Engine works upon numeric variables extracted from such devices and calculated after some natural language interpretation process. It then produces numerical results that lead the behavior, modify both the verbal and body language of the characters, and influence the general evolution of the scene that takes place inside the simulator. This paper presents the system architecture and discusses some key components, such as the Language Interpretation and the Body Language Interpreter modules.

Keywords: *Behavior Simulators, Fuzzy Logic, Unity3D, Serious Games;*

1. State of the Art. Motivation for this work

Human Computer Interaction has become a growing demand on simulation software, which is the main activity of our company. Unfortunately, although there has been significant improvements in fields like Speech Recognition [7], or Body Detection, the effective use of such techniques has not led to a real interaction between humans and synthetic characters, sometimes due to technical problems [4,5], but also because of the psychological implications of communication between humans and virtual entities [14].

Many authors have dealt in the past with the difficult task of building believable virtual characters [8] and their associated animations, or generating visual attending behaviors [9]. The definition of a standardized representation for perceived nonverbal behaviors [10] is also a key research field when developing interactive human simulation software. Some other authors [12] have focused their work on the verbal communication between virtual agents inside a video game.

Developed by the Institute for Creative Technologies at the University of Southern California, Virtual Human Toolkit [6] provides a set of tools that aids in the construction of virtual human conversational characters, and is the closest work to the one presented in this paper. In [13], a dialog model for virtual humans is proposed. Different approaches used when modeling embodied agents are also discussed in [11].

Many of these problems remain open, and so does the possibility of having a complete architecture where the definition of the behavioral properties of conversational characters, together with the context where they will be interacting with humans was possible. Integration of both in a realistic three dimensional engine like Unity3D™ is also a desirable feature. This is what our system was designed for, and although every single task involved in the process (Speech Recognition, Natural Language Interpretation, Body Gesture Labeling and Scoring) can be certainly improved, we ended up with a set of programs that cooperate inside a functional architecture in a straightforward manner that can be effectively understood and used by the average computer user. Possible application domains of our system are Conflict Avoidance Training, Questioning Simulation and Games.

This paper is organized as follows. Section 2 describes the system's requirements. Section 3 presents the complete system architecture and introduces terminology used throughout this paper. In Section 4 we present our conclusions and future works.

2. System Requirements

This project is focused on the development of a tool that aids in the definition of different virtual characters, in order to perform Conflict Avoidance Training in Medical, Military, Police environments, but in general in any interactively trainable situation. We also need the system to allow easy design and modification of the contextual situation where these virtual characters will act. Such situations are what we call Exercises, or Training Situations, defined by experts in some field, and formally introduced in the system in a way discussed in following sections. We expect the system to be used in different contexts, as we designed it to be easily rebuilt on top of a small number of elements, mainly a Situation Graph (described in Section 3.3) and some sets of expressions whose role we will discus later on this paper, that must be introduced in the form of text files.

The main characteristics of the system are:

• High quality, immersive Graphics: to achieve this we have employed Unity3D™ engine, and realistic scene modeling.

• Natural Language Interpretation: we use Windows™ Speech API to recognize words and sentences, together with our own Language Interpretation Module, that assigns labels to the recognized sentences, and some statistical processes to improve recognition and labeling.

• Easy, configurable Exercise definition: to do this, we ask the instructor to provide a formal definition of the situation to be trained, in terms of possible intermediate states, success and failure conditions, acceptable language (on the human side), and the sets of expressions to be used by the virtual characters.

As a result, we propose a modular architecture where any component can be changed whether a new release of these modules is launched, or the definition of the exercises implies changes in the way that nonverbal language needs to be interpreted. In this context, we started using Windows™ Speech API for Speech Recognition, but we are now rewriting the code to use Google Speech Recognition API, in order to port the whole system to mobile devices, as we think that a multi device version of the system will increase the number of potential users.

3. System Architecture

In this section we describe our system's architecture, focussing in the following elements:

• The definition of of the exercise to be executed by the user, named Training Situation or Exercise.
• The modules that form the system, and data flow between them.
• The formal definition of Training Situations as labelled, directed graphs.
• The Emotional Engine and its components.
• The Soul Equalizer, a tool that we have developed to help in virtual character definition.

3.1 Training Situation, or Exercise

Before presenting the system architecture is explained, we define what a Training Situation or Exercise is – these terms will be used interchangeably throughout the paper. Since the goal of our system is to help users in the design and training of potentially conflictive situations, an Exercise or Training Situation is a scene where a person (the trainee) must interact with other persons, avoiding conflict, and carrying the conversation to a normal, non potentially dangerous state. For example, a nurse working at the Emergency Room inside an hospital must deal with nervous persons demanding information about their relatives. The goal in this case is to manage the situation, being able to convince them to remain at the waiting room, without starting a fight, and taking rude, and sometimes offensive language. The Situation is defined by an expert with experience in the field, and must be introduced into the system in a format that we will explain in following sections.

3.2 Behaviour Simulator Pipeline

Once the situation is designed, the whole Behavior Simulator works on top of the following elements:

• A formal description of the Training Exercise, named Situation Graph. In this sense, our role does not involve the design and modification of the exercises, but the development of the tools that permit a person with skills in some area to do it. See section 3.3 for a complete explanation.

• A set of sentences associated to each graph's node. Whenever the situation enters a state, represented by a node inside the graph, the system will randomly choose any of the sentences associated to that node, and the virtual character will say it. The higher the number of sentences is, the lower the probability of repeating a sentence, increasing the perception of talking to a real person. These sets of sentences are stored in text files where the name is the same of the graph node, with .talk extension to distinguish them from other configuration files of the system.

• One or more sets of sentences the system must recognize when the graph is at each state. The matching between the expressions said by the user and these sets can eventually trigger a transition to a different state. Each node can be connected to one or more nodes in the graph. The arcs representing these connections will be labelled with names like `"Right Answer", "Wrong Answer", "Insult", or "Expulsion", for example. Each one of these labels is associated to a unique set of sentences that the human user can say in order to trigger that transition, and is stored in a file named RightAnswer.lang, for the first one, and following the same convention for the rest of them. The extension ".lang" is interpreted by the system in this way.

• An Emotional Engine that drives the emotional state, language and behaviour of the virtual character, described in Section 3.4.

• A Body Language Interpreter, developed using Microsoft™ Kinect sensor, which analyses the body gesture of the human interacting with the simulator, in order to give advice about good/bad practices while interacting with real humans. This module is at preliminary state at the moment, reporting only about some body positions considered inappropriate when dealing with nervous, potentially violent persons. This positions are described by an expert in the field.

• A Language Interpretation Module, built on top of Microsoft Speech API, that assigns a label to any sentence said by the human user. At the moment, this module only checks for coincidence between the string returned by the Speech Recognition library and the sets of sentences associated to the graph's current state. Each sentence inside these sets has been tagged with a number between 0 an 10 that means the associated violence, or rudeness. In future versions, this number is expected to be automatically assigned, based on aspects like the presence of words indicating bad manners, but is yet dynamically adjusted depending some other parameters like the perceived level of the input signal.

The whole pipeline works as pictured in Figure 1, where LIM stands for Language Interpretation Module, SR is the Speech Recognition Library and BLI means Body Language Interpreter.

Step 1. Speech Recognition library supplies a string to Language Interpretation Module.

Step 2. Language Interpretation Module converts the string into a Label, and recovers the number representing the violence level assigned.

Step 3. This duple (label,violence) is provided to Situation Graph Manager and Emotional Engine, respectively.

Step 4. Emotional Engine gets information about Body Language and sets output volume and additional language modulations.

Step 5. Data concerning inappropriate body gesture is used to carry slight increments on the label assigned by the Language Interpretation Module.

Step 6. Situation Graph Manager chooses one sentence to be said by the virtual character, and sends it to Text to Speech and Unity3D modules. Information about current graph state is sent also to the Unity3D module. The sentence can be modified, including some interjections,depending on the emotional State.

Step 7. Unity3D module manages animation, FACS changes in character's face, and the whole scene. Character speaks, and the system returns to listening state, until human user speaks again (Step 1).

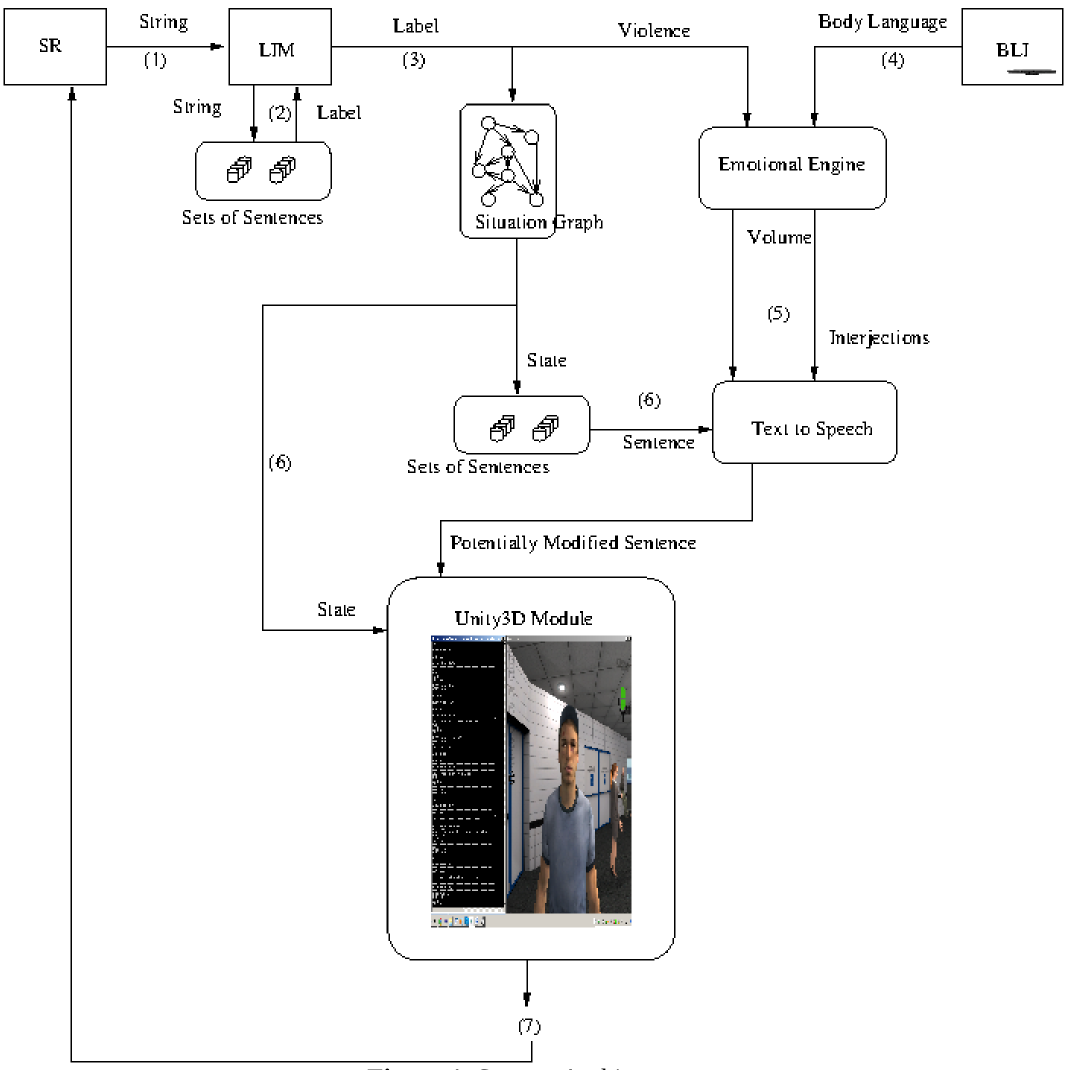


Figure 1. System Architecture.

3.3 Situation Graphs

A situation Graph is a directed, labeled graph that describes all the possible states the Training Situation can be in, and all the possible transitions from one state to another, in terms of what sets of sentences trigger each movement inside the graph. The description of the Situation Graph must be introduced in GraphViz™ [3] format, and the resulting file will be parsed by the simulator in order to generate all the logic for the Exercise. The best way to explain the role of this graph inside the simulator is through the simplest exercise we can think in:

Let's imagine a situation where the virtual character wants to know what time is. We will call the main state of the situation “Asking The Time”. While in this state, the virtual character will keep asking about the hour, using a sentence picked at random from a set of options, for example: “What time it is?”, Can you tell me the time? “, “Can you please tell me what time it is? “, etc. This is the simplest set of acceptable expressions for such a state.

Additionally, there will be two more states: Success, and Failure, associated with the fact that we can give a correct or wrong answer to the character (an insult, for example, would be a wrong one). There is two more states that we need in order to keep consistence between all the possible Situation Graphs, i.e. the initial (Init) and final (End) state of the Exercise. The former is abandoned as soon as the exercise starts and the virtual character enters the scene, with no human interaction needed. The later is reached as soon as the virtual character says one of the sentences associated with Success or Failure States. When entering End state, the system will launch the final animation, with the character leaving the scene and the Exercise ends.

A valid set of sentences that could trigger a transition from AskingTheTime state to Success could be the one formed by “It sis six o'clock”, “Six o'clock”, and “Six”, for example. A transition to Failure from the same state would be triggered by any rude expression containing insults, and finally,

what in Figure 2 is labeled as "Anything", actually means "Anything Else" (neither correct nor incorrect) being this an expression that does not match any other label, and that triggers an empty transition from a state to the same one. The asterisks in the graph mean that the graph makes this movement independently of any user's action.
This situation could be described by the graph represented in Figure 2.

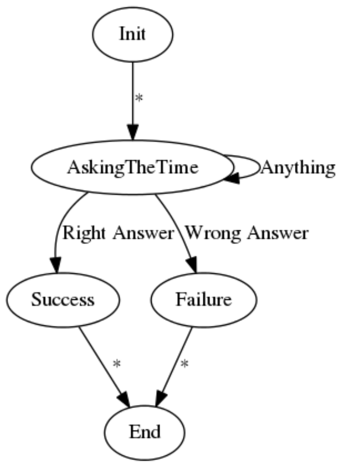


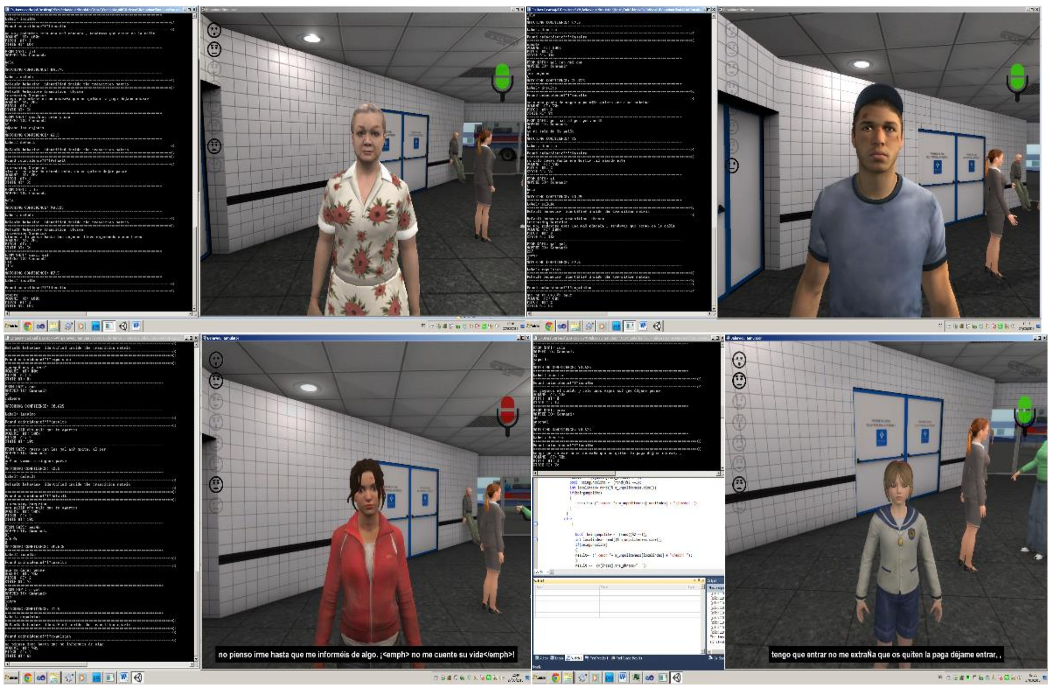
Figure 2. Simplest Situation Graph.

After expressing this graph in Graphviz™ [3] format, and in order to complete exercise definition, the user must define some sets of expressions. Each state in the Situation Graph has a finite, but potentially huge number of sentences that the character will say at that moment (i.e. associated with the current active node of the graph) chosen at random to increase the user's perception of humanity. In the previous example, the set of sentences for AskingTheTime state have yet been commented. For Success state, the set should contain something like "Thanks", "Thank You", "Thank you Very Much", etc. Failure would have a similar set of associated expressions, indicating that the training ended up as a failure, and the character will leave the scene in both cases, ending up the Exercise.

Finally, the instructor must specify the Acceptable Language: each state inside the situation graph has also a set of sentences that can be interpreted by the character, triggering the movement from one state to another. Our Language Interpretation Module allows slight variations in the same sentence (order changes, use of synonyms) to trigger the same transition. This means that what we call "Right Answer", for example, can be a set containing each sentence that we consider as a valid response for the virtual character's question, but slight differences between the specified valid sentence and what we say will have no effect. The Language Interpretation Module converts expressions into labels that trigger changes of state inside the Situation Graph.

The description of the Situation Graph, and all the sentences to be said by the synthetic character, plus all the possible understandable sentences can be a long process, depending on the complexity of the situation we want to model, but only implies manual edition on a number of text files. On top of this, our system builds automatically a virtual environment where the character reacts to the user actions according to the defined exercise. At the moment, we have a set of four different predefined exercises, named Pimp, Crazy Woman, Drunk Teenager, and Child, the first of these being a situation where the goal is to send back to the waiting room a violent young man who wants information about one if his relatives, admitted to the hospital. See Figure 3.

Once the definition process has been completed, the simulator can be launched. The virtual character will start listening to what the user says, and the situation will change from one state to another, following the Situation Graph definition, and eventually ending up in a Success or Failure state. But meanwhile this happens, we need to be able of controlling the Emotional State of the characters, depending on what we tell them. This is the aim of the Emotional Engine, which we describe in next section.



no pienso irme hasta que me informéis de algo. ¡<emph> no me cuente su vida</emph>!

tengo que entrar no me extraÑa que os quiten la paga déjame entrar, ,

Figure 3. Behavior Simulator Running. Crazy woman, Pimp, Drunk Teenager and Child.

3.4 The emotional Engine

To explain how The Emotional Engine works, we will describe each element that defines a virtual character. Each one of these elements is one attribute inside a C++ class, named Soul.

3.4.1 Emotional State, E

We label the possible emotional states of a virtual character from E0 to E10, but only one of them, that we will call E, will be really active at any moment. The higher this number is, the more violent its behavior can be. This state will have influence also in small changes on the character's face according to FACS [1], the inclusion of insults in the sentences said, and the volume of the speakers, among other things (Fig. 4). The maximum number of states is currently eleven (from E0 to E10), but is set by a constant in the code, and can easily be changed without altering any other module. We chose this number of states because higher numbers determine imperceptible changes in virtual character's face from one state to its neighbors.

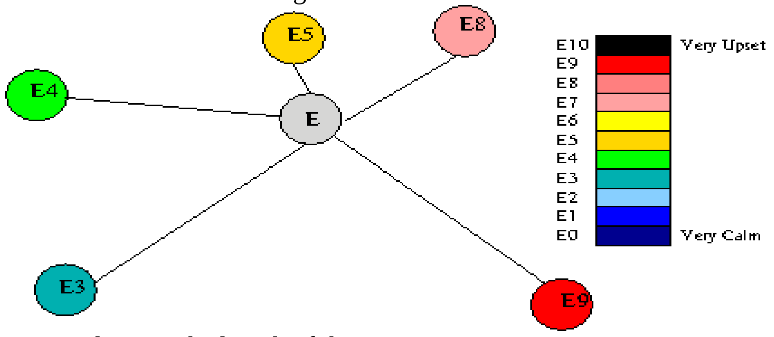


Figure 4. Emotional state. The length of the segments represent represents activation probability.

The meaning of fig. 4 is that the virtual character is as likely to be in state Ei as the length of the segment between Ei and the real current state, E. In fig. 4 we can see that states E3, E4, E5, E8 and E9 are stronger at the moment. All the remaining states have lengths equal to zero and would have no possibility of being activated in such a situation. When the emotional engine receives its inputs from Language Interpretation Module and Body Language Interpreter (Step 3 of the pipeline described in section 3.2), it changes the lengths of these segments according to Dynamic State Activation Matrix (described in section 3.2.3), and the current emotional state of the character, E, is determined as a result of the following procedure:

0. Let |Ei| be the length of the activation segment associated to the ith state,
$|E_i|$ = length of the i-th segment

1. Calculate the sum of the lengths of the different activation segments, being MAXSTATES the constant defining the total number of states the character can adopt. We are currently using eleven different states.. Let S be this value

$$S = \sum_{i=0}^{MAXSTATES-1} |E_i|$$

2. Generate a random number between zero and S-1. Let it be x.

x = rand(0, S-1)

3. Determine the current emotional state according to the following rule, that is applied on the segments with non zero lengths:

$$E = E_i \Leftrightarrow (x \geq \sum_{0}^{i-1} |E_j|) \wedge (x < (\sum_{0}^{i-1} |E_j|) + (|E_i|))$$

3.4.2 Verbal Violence Level, VVL

Each time the user speaks, our Language interpretation Module assigns a level from C0 to C10 to the string returned by the Speech Recognition library. We chose this name, CI, as sometimes what the user will say is something to be interpreted as a command by the virtual character. Higher values of this number mean that the command (expression) is more authoritarian, rude or disgusting. A complete explanation on this module is out of the scope of this paper. The role of this assignment is to direct the manipulations on the current lengths of the segments that describe the Emotional State, E. We also use information about body gesture to increase or decrease this value. At the moment, we have a tool were the expert points out inappropriate body positions that the user must avoid while talking to the virtual character and we increment the initial Verbal Violence Level when some conditions are fulfilled. Some of these rules are (see also fig. 5):

Rule 1:

RightHandHeight ≥ RightShoulderHeight ⇒ VVL = VVL +1

Rule 2:

LeftHandHeight ≥ LeftShoulderHeight ⇒ VVL = VVL + 1

Rule 3:

RightAnkleWidth > RightKneeWidth ⇒ VVL = VVL + 1

Rule 4:

LeftAnkleWidth < LeftKneeWidth ⇒ VVL = VVL + 1

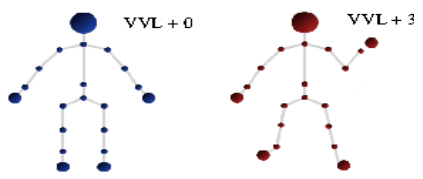


Figure 5. Blue Model (on the left) represents a position where Verbal Violence Level remains unchanged, as opposite to Red Model (on the right), that implies an increment of three units in VVL (two for the feet position and one more for keeping one raising hand: rules 2, 3, and 4).

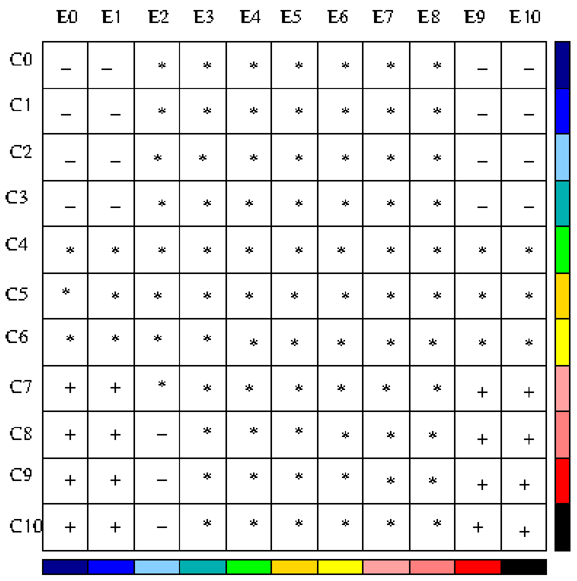
3.4.3 Dynamic State Activation Matrix, M

Each time the user speaks, the situation graph will change (or not) its current state, but the character will adopt one the possible emotional states. To do this, we must first change the length of each segment connecting E to Ei, according to what is stored in this matrix. Dynamic State Activation matrices have as many rows as different Verbal Violence Levels we admit, and as may columns as MAXSTATES. Both of these values are now eleven, but are defined as constants in the code, so can be easily changed without affecting the logic of the program.

The meaning of the values that we store inside this matrix is the following. For any position EI, CJ:

- A + symbol means that we must increment the length of EI segment when receiving CJ.
- A - symbol means that we must decrement the length of that segment in the same case.
- An * symbol means that the associated segment must remain unchanged.

The Dynamic State Activation Matrix pictured in Figure 6 shows a hypothetical situation where the character needs to be treated rudely when calmed, and politely when excited, if we want to keep him under control. Such a character would remain emotionally dull, except when receiving polite commands (expressions) being in violent states, or receiving strong ones when calmed.



	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
C0	_	_	*	*	*	*	*	*	*	_	_
C1	_	_	*	*	*	*	*	*	*	_	_
C2	_	_	*	*	*	*	*	*	*	_	_
C3	_	_	*	*	*	*	*	*	*	_	_
C4	*	*	*	*	*	*	*	*	*	*	*
C5	*	*	*	*	*	*	*	*	*	*	*
C6	*	*	*	*	*	*	*	*	*	*	*
C7	+	+	*	*	*	*	*	*	*	+	+
C8	+	+	–	*	*	*	*	*	*	+	+
C9	+	+	–	*	*	*	*	*	*	+	+
C10	+	+	–	*	*	*	*	*	*	+	+

Figure 6. Dynamic State Activation Matrix. Plus symbols mean incrementing segment length. Minus symbols imply decrementing, and asterisks mean that the length remains unchanged.

3.4.4 Wrapper Functions

Although the Asking Person Example described in previous sections does not need what we call Wrapper Functions, many of the Exercises we are dealing with include graph states where the user can express an order for the virtual character. For example, “Please, show me your identification”. In such moments, the virtual character can decide whether to follow the order or not. When this happens, we say that the virtual character Collaborates or Resists. This is what Wrapper Functions are for. Let’s suppose we are working with a maximum of eleven states, from E0 to E10. We keep a different probability of following an order when the character is at each different Emotional State. Consequently, we have as many Wrapper Functions, or probability values, as different Emotional States. In practice, this is stored as a real number in the [0,1] range, and when the character receives an order we generate a random real number between 0 an 1. If the generated value is below the associated probability for the current state, the character Collaborates. If not, the character Resists, and the possible transition inside Situation Graph is cancelled.

The Soul Equalizer Tool (we describe it later) allows the manipulation of these probabilities when defining new characters, but also permits dynamically changing their values once the Exercise has started if the instructor wants to increase the difficulty level, for example. Figure 7 shows the Soul Equalizer Tool. Faders on the right part of the main window are used to change probability values.

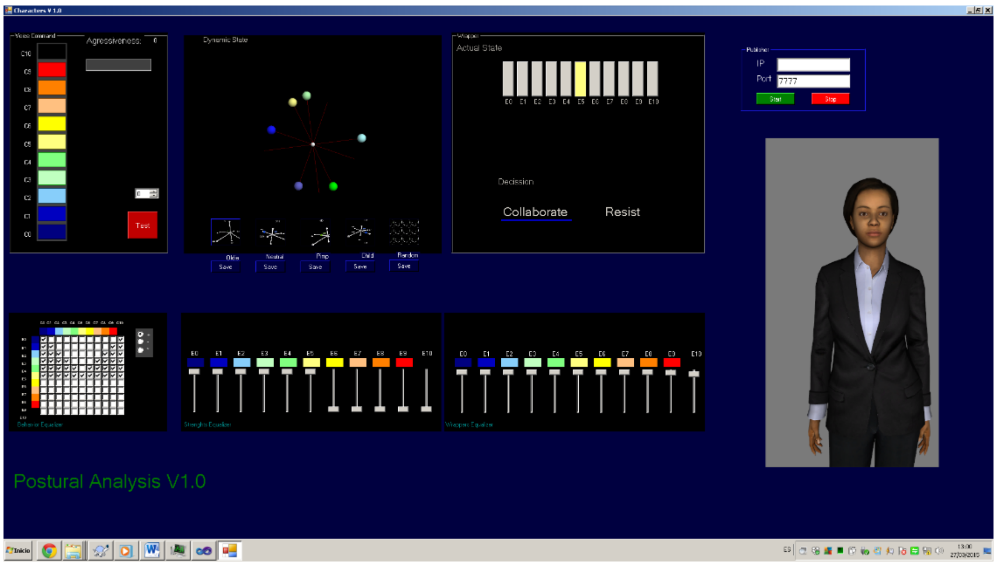


Figure 7. Snapshot of the Soul Equalizer.

3.4.5 Inertia

When setting the next state for a virtual character, we take the length of the segment associated to each possible Emotional State, EI, calculate the sum of these values, and generate a random number between zero and this sum. Giving each state a possibility of activation that is proportional tho the length of its segment, we can expect that longer segments determine more activations for that state. But there is an important problem with such a method. Due to its randomness, harsh changes in the virtual character's state can happen. In such a situation, the modifications to be done on the character's face according to Facial Action Code System would be too extreme. To prevent this, we included one more parameter into character's definition (one more attribute inside Soul Class): inertia. The role of this attribute is to module the changes from the current state to the next, limiting the maximum distance between them. Let's suppose that the random process determined that we are changing from E0 to E10. If Inertia value is 3, the change will occur, but from E0 to E3.

3.5 The Soul Equalizer

The definition of the emotional settings of any virtual character is implemented in a C++ Class, which we call Soul. To keep the character design process simple, we have also developed a tool which allows manipulating the different personality characteristics, in order to create virtual characters with different levels of inherent violence, more reactive to commands. The tool allows an initial definition of default segment lengths, Activation Matrix, and some other attributes like the IP address where the Soul class will perform listen/write actions. (Fig. 7). We call this tool "Soul Equalizer" because it looks like a Sound Table made up from Equalizers, Faders, and knobs with different functions.

Once all parameters have been set, the Soul class is instantiated according to the current settings and connected to the other modules inside the simulator. This class is really what we call the Emotional Engine, and can interact with the Natural Language module, the Body Language Interpretation Module (using Kinect™), or both of them. In Figure 7, the first eleven faders (on the left side of the main window) are used for the setting of segment lengths that describe the Emotional State. The faders on the right side can be used to set the initial values for the Wrapper Functions (when defining a new character), or to dynamically change the behavior of the character when the Exercise has started. Dynamic State Activation Matrix is also accessible from the tool, and the instructor can monitor what happens each time the user speaks (calculated Verbal Violence Level, Collaborate/Resists events). The picture shows a sophisticated Unity3D model that we bought recently to improve the procedures that determine facial expression of the virtual characters.

4. Conclusions and Future Work

4.1 Conclusions

Summarizing our work, we highlight that the primary contribution is the definition of a modular architecture which can be easily rebuilt whenever a new version of any of its modules is available. Different Speech Recognition libraries are directly plugged into and out of the system, depending on the target device.

The system's main capabilities include:

- Easy Exercise design and manipulation: with the use of Situation Graphs, creation and discussion of a new Exercise is an agile process. Graphical representation of Exercises also permits early stage detection of inconsistencies.
- Intuitive design, manipulation and storage of different virtual characters: the Soul Equalizer visual tool supports users (e-learning professionals are frequently not expert in programming) in manipulating the different attributes that define a virtual character.
- Integration of nonverbal language as a variable that modifies the statically assigned Verbal Violence Levels. Data from Kinect style sensors can be used as a way of conditioning nonverbal language when interacting with virtual characters.
- Use of Facial Action Code System: in order to increase the user's perception of talking to a real human, we have introduced slight changes in character's face to show emotions like surprise or fear, that are not explicitly managed by our Emotional Engine, as it is mainly oriented to violence levels.

We are now working on the definition of common behavioral patterns using Dynamic State Activation Matrices, and exploring the fields where such a system would be of use. At the same time, we are finishing the development of an app that integrates Speech Recognition. The goal is to be able to communicate with the simulator using a mobile phone, or tablet, as a microphone, without being necessarily in the same room. Additionally, we plan to improve each module when new Kinect™ style devices are available, or new updates of speech Recognition libraries appear.

4.2 Future Work

Although fully functional, this is an ongoing, open project, and we keep working on the improvement of each one of the modules described in this paper. Our future research lines are:

• Communication between one virtual character and several humans. This would be useful for Questioning Simulation Software, for example, and is technically possible, although it will imply some changes in the definition of Situation Graphs.

•On the theoretical side, it would be interesting to extend the Emotional Engine to manage feelings like surprise or fear, as the tool is currently focused on violence levels only. However, we have already introduced surprise as an effect on virtual character's faces when Speech Recognition process is not working properly, or the expression said by the user does not match any of the expected ones, but it is a simplistic approach. Another future research direction will concern the possibility of allowing virtual characters to speak and interact directly between each other in the scene.

The Simulator can be seen at work at the following url: https://www.youtube.com/watch?v=l4gP97_Lvrc (since min 2:00).

References

[1] Ekman P. and Friesen W., Facial Action Coding System: A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, Palo Alto, 1978.

[2] Shotton J., Fitzgibbon A., Cook M., Sharp T., Finocchio M., Moore R., Kipman A., Blake A., "Real-Time Human Pose Recognition in Parts from Single Depth Images". Microsoft Research Cambridge & Xbox Incubation. DOI: 10.1007/978-3-642-28661-2_5

[3] Gansner E. R. and North S. C.. "An open graph visualization system and its applications to oftware engineering". Software-Practice and Experience, 2000(30)pages1203—1233. DOI: 10.1002/1097-024X(200009)30:11<1203::AID-SPE338>3.0.CO;2-N

[4] https://www.movesinstitute.org/research/human-behavior-simulation/

[5] Tardón C. G. Behavior Simulator. Generación y Aplicación de un Ser Humano Simulado para el Estudio de la Interacción social didáctica. InteligenciaArtificial. Revista Iberoamericana de Inteligencia Artificial, 12 (38), 2008.

[6] Hartholt A., Traum D., Marsella S. C., Shapiro A., Stratou G., Leuski A., Morency L.P., Gratch J.. All Together Now. Introducing the Virtual Human Toolkit. International Conference on Intelligent Virtual Humans, 2013. DOI: 10.1007/978-3-642-40415-3_33

[7] Leuski A, Kennedy B, Patel R, Traum DR. Asking questions to limited domain virtual characters: how good does speech recognition have to be? ASC, 2006.

[8] Jung Y., Kuijper A., Fellner D., Kipp M., Miksatko J., Gratch J., Thalmann D., Believable Virtual Characters in Human-Computer Dialogs. Eurographics 2011.

[9] Chopra S., Badler N., Where to look? Automating attending behaviors of virtual human characters. Proceedings of the third annual conference on Autonomous Agents. 1999. DOI: 10.1145/301136.301152

[10] Scherer S., Marsella S., Stratou G., Xu Y., Morbini F., Egan A., Rizzo A., and Morency L.-P., Perception Markup Language: Towards a Standardized Representation of Perceived Nonverbal Behaviors. The 12th International Conference on Intelligent Virtual Agents, 2012. DOI: 10.1007/978-3-642-33197-8_47

[11] Ribeiro T., Vala M., and Paiva A., Censys: A Model for Distributed Embodied Cognition. Intelligent Virtual Agents. Lecture Notes in Computer Science (8108), 2013. DOI: 10.1007/978-3-642-40415-3_5

[12] van Oijen J. and Dignum F., Agent Communication for Believable Human-Like Interactions between Virtual Characters. Lecture Notes in Computer Science (7764), 2013. DOI: 10.1007/978-3-642-36444-0_3

[13] Traum D., Swartout W., Gratch J. and Marsella S., A Virtual Human Dialogue Model for Non-team Interaction. In Dybkjær L. and et al., editors, Recent Trends inDiscourse and Dialogue, volume 39 of Text, Speech and Language Technology, pages 45–67. Springer, 2008. DOI: 10.1007/978-1-4020-6821-8_3

[14] Tinwell A., Sloan R.J., Children's perception of uncanny human-like virtual characters. Computers in Human Behavior. 36: pp. 286-296. 2014. DOI:10.1016/j.chb.2014.03.073