Semantics-based intelligent Human-Computer Interaction

Semantics-based Intelligent Human-Computer Interaction

Valentina Gatteschi, Fabrizio Lamberti, Paolo Montuschi *, Andrea Sanna
Dip. di Automatica e Informatica, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
* corresponding author

Abstract

A recent trend in human-computer interaction is to ease the creation of contents like apps, games, etc. by means of intelligent systems allowing also non-skilled users to define the behavior of a given system through visual programming and/or simplified meta-languages. However, when the number of elements to be controlled increases, the complexity could get comparable to that of traditional coding strategies. This paper addresses this issue, by proposing a framework for automatically configuring a system's behavior based on user's input and context information. Framework effectiveness has been tested in a game creation scenario and used for automatically mapping user's commands on virtual characters' actions based on a natural language description of the game scene. The use of a semantics-based mapping reduces the effort and complexity linked with the configuration of the interaction logic, decreasing also the number of commands for controlling the characters.

Keywords

User Interfaces, Interaction techniques, Interactive environments, Semantics, Natural Language Processing

Introduction and related work

Continuous technological developments produce, in general, a diffuse appreciation, though may be a matter of concern for unskilled users that could find it difficult to get acquainted with them. Hence, it is not surprising that recent research and industry efforts have been extensively devoted to increase the simplicity and usage intuitiveness of devices, appliances and applications in a human-centred perspective based on human-computer intelligent interaction [1]. The goal is to devise interactive systems capable to recognize multiple stimuli deriving from the external context and to adapt their behaviour accordingly.

Application fields are heterogeneous, including the creation of systems supporting everyday activities, such as:

- the creation of intelligent user interfaces, changing the way information is displayed based on user's operation environment habits, etc.; for instance, in [2], an algorithm is designed to predict the next relevant interaction element(s) and adapt the user interface accordingly;
- the development of systems for human-computer interaction considering additional data like, for instance, user's mood; in [3], behavioural cues (like body postures, vocal and facial expressions, etc.) are analysed and matched with the surrounding context in order to identify user's communication intentions;
- the creation of applications able to understand and translate the natural language into commands for a smart home; in [4], context information and speech recognition techniques are considered together with a thesauri to improve command processing performances;

- the exploitation of speech recognition and Natural Language Processing (NLP) techniques to command a robot; in [5], context information is used to improve the accuracy of speech recognition results.

Other application fields deal with the definition of new ways of human-computer interaction, especially devoted to allow users with limited skills to interact with (technological) devices by:

- devising systems to improve webpages accessibility for blind users; systems like that in [6] automatically compare images description with image elements in order to verify the correctness of the information provided before communicating it to the user;
- developing new interaction paradigms, where different sensors acquire data about user's movements and translate them into commands to be sent to a smart wheelchair [7];
- creating Natural User Interfaces (NUI) allowing digital and language illiterates to browse Internet pages; in [8], keyboard and speech recognition are used to gather user's inputs, which are first translated to English, then pre-processed to identify relevant keywords and finally shown in a simplified icon-based interface.

At the same time, prototypes are being developed for easing the work of professionals, e.g.,:

- to support tele-assistance for maintenance applications via object recognition, context capturing and real-scene augmentation with graphical hints [9];
- to explore whether multimodal interaction can be possibly used for communicating with a robotic nurse in surgery rooms [10]; gestures and speech recognition techniques and, in the future, context data represented by the activity of the whole surgical team, could be used to let the system predict the next surgical instrument likely needed by the surgeon.

In parallel to these developments, a new trend is emerging, where users are changing their role from being mere consumers of contents developed by other people, to becoming in first person inventors and creators of contents. Such process is supported by the development of intelligent systems designed to ease the creation phases by hiding the underlying complexity. These systems are becoming rather common in contexts related to:

- mobile apps development, by allowing users to define apps behaviour by using visual programming languages [11];
- virtual worlds/games creation [12], by providing users with graphics interfaces and drag&drop tools that can be used to define both the features of the virtual world/game and the actions to be performed by the virtual characters;
- robotics [13], by making it possible to use programming by demonstration together with speech recognition for performing the training.

In this paper, we will focus on the field of human-computer intelligent interaction, and especially on the definition of system's behaviour in response to user's interactions. Although the above solutions make this process quite easy to be managed, when the number of elements to be controlled increases, existing approaches might be still rather cumbersome even for skilled users. Hence, the objective of this work is to overcome this limitation by proposing an adaptive system capable to automatically identify the more suitable action a system should perform based on commands issued by the user and of the context.

Even though the proposed system could be potentially applied in different fields, we decided to tackle in particular the development of video games and interactive 3D graphics applications. In fact, the economic importance of this sector is continuously growing and the advancements made by frameworks for games creation like Blender (www.blender.org), Unity (unity3d.com) or Project Spark (www.projectspark.com) show the need for systems easing developers' activities.

This field has been historically relevant for the research in the area of artificial intelligence, and is also particularly interesting from the point of view of human-computer interaction, as ever new and complex interaction modalities are often experimented first in this domain with the goal of improving user's experience (more details can be found in the sidebar "Intelligence in games").

To provide intelligence to the system we relied on semantics, a technology usually exploited in the field of information retrieval to support automatic processing of (possibly naturally-expressed) resources. Core pillars of semantics are taxonomical and ontological descriptions, which define relevant concepts for the domain of interest in a machine-understandable format and let computer-based systems exploit relations among them to match resources based on their actual meaning. Semantics has been successfully exploited in different application, from search engines [14], to recommender systems [15], to plagiarism identification [16], etc. In the contexts of interest for this paper, semantics has been mainly used to enable text-based 3D scenes generation [17], to define virtual objects' behaviour and interactions among them [18], to adapt game difficulty to players' ability [19], and to retarget body parts motion in computer animations [20].

In this work, the exploitation of semantics brings two benefits: first, semantic relations are browsed to produce a mediated mapping between commands (that could be gathered by recognizing users gestures performed in front of off-the-shelf gaming sensors, or their voice recorded through microphones, their facial expressions, etc.) and actions a character could carry out in the virtual environment based on their meaning. Second, NLP techniques embedded in the system allow users to freely specify, in natural language, the tasks a character could carry out in a given scene, thus laying the foundations for a dynamic mapping that could change based on the context in which the character is acting into. In this way, not only the complexity associated with aforementioned operations could be hidden thus making them easily accessible also to unskilled users, but also the number of commands required for controlling a character could be decreased, limiting mental effort and improving user experience.

The proposed system has been embedded into the 3D game engine of the Blender open source modeling and animation suite, and has been tested by animating several virtual characters with body gestures. In the case of non-anthropomorphic characters, the system would provide the additional advantage of letting the user control, for instance, a bird character by making it move on the ground or fly with the same walk gesture, depending on the description provided for the task and the context.

The proposed system

The behavior of the proposed system could be split in three different steps:

1. command recognition: user's gestures captured by an off-the-shelf gaming sensor are processed and recognized;
2. semantic processing and mapping (with a set of pre-created actions): mapping can be based just on names assigned to commands and actions or exploit optional text-based descriptions about the actions to be carried out by the character and the context they can be executed into; as a result, the action having a meaning similar to a given command (and task, if specified) is identified, together with the activation conditions defining the particular context (when provided);
3. character animation: recognized gestures activate the corresponding character's action in synthetic worlds created using the Blender game engine.

The architecture of the system is depicted in Figure 1. In the following, a detailed explanation of the functioning of the three key modules implementing the above steps will be given.

Command recognition

In the current implementation, user's commands are represented by body gestures that are gathered by using the Microsoft Kinect, though the system could easily work with other input modalities, e.g., based on speech recognition or face tracking. Microsoft Kinect SDK APIs are used to collect information about the position of user's skeleton joints in the 3D space. Command recognition is performed by comparing 3D coordinates obtained in real time by the sensor with a set of pre-recorded gesture descriptions. Comparison exploits the Dynamic Time Warping (DTW) algorithm [21], which is capable to compute the match between coordinate series with possibly different timings. During the training phase, the user can use a graphics interface to specify body parts affected by the gesture to be recognized. The system will then record only changes affecting interested joints.

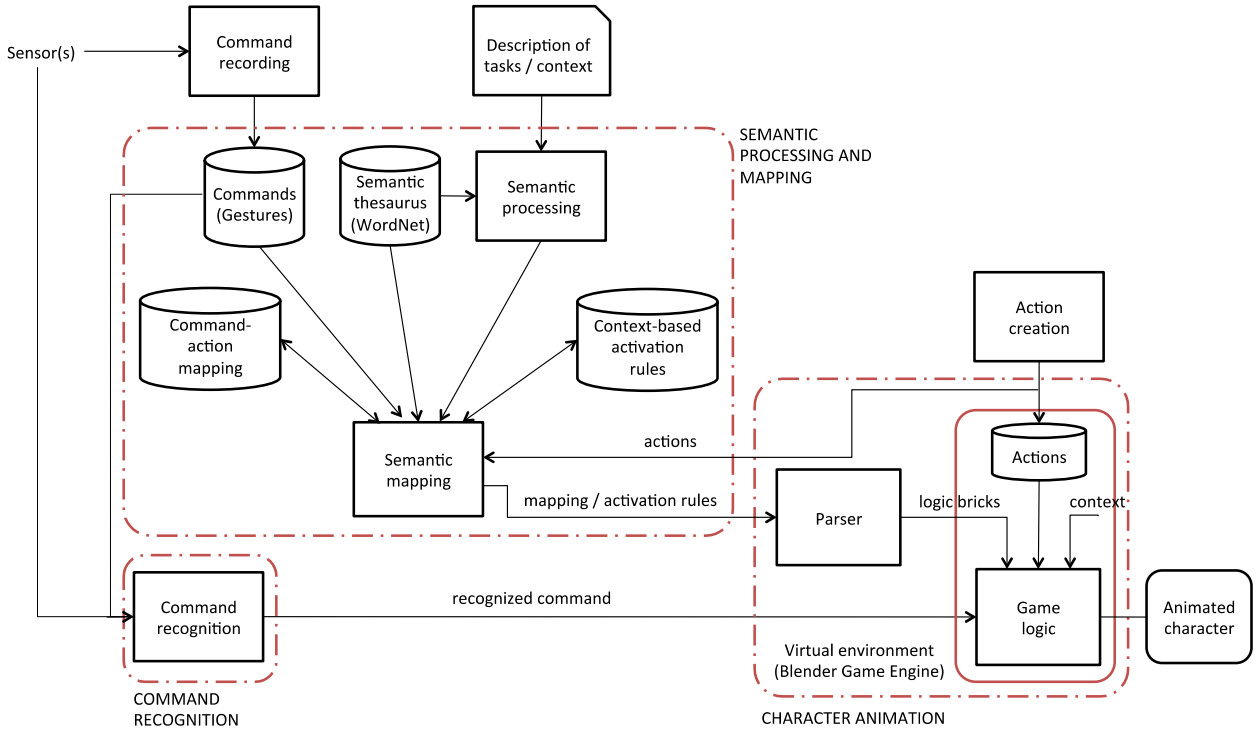

Figure 1. Architecture of the proposed system.

Semantic processing and mapping

In order to identify actions to be activated we designed two approaches, i.e., a *direct* "command - action" mapping and an *indirect* "command - tasks/context - action" mapping.

<u>Direct "command - action" mapping</u>

In the simplest implementation, the identification of the mapping between commands issued by the user, i.e., gestures, and actions defined for a virtual character is performed by directly matching their names. To this aim, a semantics thesaurus is considered, and a similarity score between words (lemmas) used is computed by considering the relations among them. Specifically, we chose the well-known WordNet thesaurus, though other semantic frameworks

could be experimented as well (like, ConceptNet, DBpedia, etc.) even in a combined way, e.g., for improving mapping accuracy.

Since the same word can have more than one meaning (that is, more senses), the user is allowed to annotate gestures and actions by linking a selected subset of senses to them. For instance, a gesture in which user's legs rapidly move could be described by lemma "run", and/or more specifically, linked to the meaning "move fast by using one's feet, with one foot off the ground at any given time", thus ignoring other misleading meanings (e.g., "execute a computer program", "direct a business", etc.).

Mapping is computed every time a new gesture/action is inserted into the system. Firstly, for each action $a_i \in A = \{a_1, \dots, a_I\}$ a character could perform, the similarity $sim(a_i, c_j)$ with each command $c_j \in C = \{c_1, \dots, c_J\}$ that can be issued by the user is computed. Similarity is calculated using the PATH measure [22], an easy-to-compute metric commonly adopted for WordNet implemented in many libraries. In particular, we chose WS4J (code.google.com/p/ws4j), an open source library which includes other similarity measures that could be worth to be experimented in the future. Given two senses, PATH defines a similarity score in the range (0,1] that is inversely proportional to the number of nodes along the shortest path between the two word senses. The closest the senses are (path length equal to 1), the more similar they are assumed to be (score equal to 1).

Similarity scores for all command - action pairs are sorted in a descendent order. Then, starting from the pair showing the highest similarity, action a_i is linked to the best matching command c_j. In case of ambiguities, a random choice is made.

At the end of this step, each command is linked to a character action (a threshold can be used to exclude poor matches). The mapping is stored in a command - action mapping database, which will be queried anytime a command issued by the user is recognized in order to return the most suitable action.

A possible limit of this approach is that each command is linked to one action only. Hence, when a character presents a number of actions higher than the number of commands, some actions could not be activated. To address this issue and further extend system flexibility, an indirect mapping strategy has been explored.

Indirect "command - tasks/context - action" mapping

Indirect mapping additionally considers a description of tasks that can be performed by the virtual character and related context, which is provided in natural language (with context conventionally capitalized). This description is processed to identify a set of tasks $T = \{t_1, \dots t_K\}$ to be matched with the set of actions A defined for the character. When available, context information is used to identify the conditions that, combined with a particular command, will determine the activation of a given action.

The dependency parser and the semantic role labeller in [23] (code.google.com/p/mate-tools) are used to extract the structure (parse tree) of each sentence in terms of dependencies among terms, by identifying both syntactic relations (subject, object, etc.) as well as semantic relations (agent, instrument, goal, manner, time, etc.).

For sentences describing tasks,

- subjects are found, and predicates not related to the character described are discarded;
- for the remaining predicates – each assumed to represent a task – words (and compound words) used are extracted;

- stop-words (adjectives, prepositions, etc.) as well as non-relevant information like temporal data and de-lexical verbs (take, have, make, give, do, etc.) are discarded
- if the sentence does not contain other verbs, objects of de-lexical verbs are used to extract alternative verbs (by considering usage of objects in their definition).

As a result, for each task $t_k \in T$ a set of words $W_k = \{w_1, \dots, w_v, \dots, w_V\}$ to be matched with actions in A is identified. Next step consists in computing $sim(a_i, t_k) = \sum_{v=1}^{V} sim(a_i, w_v) / V$, thus identifying to what extent an action a_i is similar to the set of words describing a task.

Since the number of words to be possibly annotated with senses (and related burden) could be higher than with the direct approach, here we decided to compute similarity by working directly with lemmas. However, we reduced the amount of senses to consider by exploiting words' part-of-speech in the task description and considering for actions/commands only senses belonging to the same category, according to an importance rank defined as verbs-nouns-adjectives-adverbs. This means that, as a matter of example, at first, only verb senses linked to a lemma are taken into account and, if there are no verbs linked to a lemma, all the noun senses are considered (the same for adjectives and adverbs). This choice was based on the assumption that the majority of lemmas used for describing commands and actions are verbs.

Once similarity has been computed, for each task the action with the highest score is selected. In case of ambiguities (more actions with the same similarity value), words in predicates are used to look for additional verbs, similarity is computed again and ranking is adjusted. Possibly remaining ambiguities are managed by simply selecting more than one action.

Selected actions are then matched with the set of commands by following the same approach adopted for direct mapping and results are saved in the command - action mapping database.

When context sentences are found in the description,

- subjects of each clause are recognized and matched with characters' names by using relations in the thesaurus, thus allowing for different words to be used to refer to the same character;
- for each subject, the associated verbs are found and related sub-trees are selected;
- words defining objects and complements in each sub-tree are identified (discarding punctuation, determiners, conjunctions and possessive pronouns);

Subject, verb and object/complement words extracted are used to define activation conditions for the context, which are expressed as (possibly incomplete) triples where personal pronouns are replaced with the actor they refer to. Triples are recorded in the context-based activation rules database, and linked to the corresponding command - action mapping.

Character animation

The last phase of virtual character animation has been experimented with Blender, where a library of actions for the character to be animated is assumed to be available already (e.g., manually developed by a graphics artist or obtained by using either a consumer or professional-level motion capture tool). In an offline configuration stage, a Python script is used to create the interaction logic bricks (variables, sensors, controllers and actuators) in the Blender Game Logic editor and link them to a TCP/IP socket connected to the semantic processing and mapping module. At runtime, when a command is recognized, it is sent to the character animation module, where it is processed based on the actual state of the virtual environment (context) and the action encoded in the mapping is activated (thus animating the character, accordingly).

Practical examples

The functioning of the system could be better understood by considering the case studies illustrated in the following, where an elephant-like virtual character has been animated using the two mapping approaches.

Example of command - action mapping

According to this approach, for each command and action the user could restrict the meaning of names used by specifying one or more senses. Assuming that the set of commands is C = {*walk, bow, drink*}, for command *walk* the user could decide to select sense #v#1 (*use one's feet to advance; advance by steps*). For *bow*, two senses could be chosen, namely #v#3 (*bend the head or the upper part of the body in a gesture of respect and greeting*) and #v#4 (*bend one's back forward from the waist on down*). The user could also decide to leave senses for command *drink* unspecified. In this case, all the senses available in WordNet for the lemma would be used. Now, assuming that set A contains two actions for the elephant, like *amble* and *eat*, the user could assign sense #v#1 (*walk leisurely*) to the former and leave senses unspecified for the latter.

To determine the mapping, the direct mapping algorithm would compute the similarity between command and action senses using PATH (when more senses or just the lemma are provided, it would consider the highest value for each pair). Thus, for instance, for action *amble*, the highest similarity is obtained between amble#v#1 and command sense walk#v#1 (path length equal to 2, similarity 0.50). Thus, action *amble* would be mapped to command *walk*. Similarly, action *eat* would be mapped to command *drink* (path length equal to 3, similarity 0.34). Similarity computation could be experimented at http://ws4jdemo.appspot.com/.

Example of command - task/context - action mapping

Here, three different scenarios are considered (with the elephant in a circus, in a cage and in the savannah), together with a set of nine actions to be performed by the character, namely *amble*, *curtsy*, *eat*, *headstand*, *huddle*, *moonwalk*, *salute*, *ingest*, *walk_around*. A richer set of commands is used, with C = {*walk, bow, run, jump, kick, punch, greet, drink, listen, look*}. The descriptions of the tasks the character could perform and related contexts are as follows:

- AN ELEPHANT IS PERFORMING IN A CIRCUS SHOW. It first makes a sequence of dance steps sliding backward. Then it exhibits itself in an acrobatic feat balancing on its head. Finally it toasts to the health of the audience by raising a glass of wine.
- AN ELEPHANT IS IN A CAGE. It greets the visitors by bowing down on its knees as a sign of obeisance. It nibbles some nuts and stretches its legs by circumambulating without goals inside the cage.
- AN ELEPHANT IS FREE IN THE SAVANNAH. It moseys across the surroundings. It sips some water from a river. Then, it crouches to relax.

Processing, e.g., for the first scenario, starts with the identification of the elephant as the subject of the context sentence. Verb is selected (*is performing*) and complement extracted (*in circus show*). Triple elephant@is_performing@in_circus_show is saved as the context-based activation rule for the considered scenario. Then, the subject in task sentences is identified (it, i.e., *the elephant*). Semantic roles, together with dependency parsing, are used to identify predicates, i.e., tasks. Verbs, together with words (and compound words) they are linked to are added to W_k. Stop-words {*a, of*}, temporal data {*first*} and de-lexical verbs {*make*} are removed.

As a result, W_k = {*sequence, dance, step, dance_step, slide, backward*}. For each word, $sim(a_i, w_v)$ is computed. For *amble*, *curtsy*, *eat*, *huddle*, *salute*, *ingest* and *walk_around* all the verb senses are considered. For *headstand* and *moonwalk*, which in WordNet are nouns, only noun senses are analyzed. For all the commands, the whole set of verb senses is used (since all the considered commands have at least a verb sense).

Results are reported in Table 1. Rows contain the actions, whereas columns show the words w_v. Part-of-speech is reported in brackets, whereas the last column gives $sim(a_i, t_1)$. For example, for the *moonwalk* action, $sim(moonwalk, t_1) = (0.1 + 0.09 + 0.5 + 0.5)/6 \cong 0.2$. After having computed $sim(a_i, t_1)$ for each action a_i, *moonwalk* is selected since no other actions showed a similarity higher than 0.2.
Finally, similarity among actions and commands is calculated. Hence, by considering again the first scenario, for which actions *moonwalk*, *headstand* and *salute* have been identified, selected commands that could be used to activate them are *walk*, *bow* and *drink*, with a similarity of 0.5, 0.125 and 1.0 respectively.

W_k / Actions	sequence (n)	dance (n)	step (n)	dance_step (n)	slide (v)	backward (r)	Sim (a_i, t_1)
moonwalk (n)	0.09	0.1	0.5	0.5	0.0	0.0	0.20
headstand (n)	0.09	0.1	0.1	0.1	0.0	0.0	0.06
ingest (v)	0.0	0.0	0.0	0.0	0.25	0.0	0.04
walk_around (v)	0.0	0.0	0.0	0.0	0.25	0.0	0.04
amble (v)	0.0	0.0	0.0	0.0	0.25	0.0	0.04
eat (v)	0.0	0.0	0.0	0.0	0.2	0.0	0.03
salute (v)	0.0	0.0	0.0	0.0	0.16	0.0	0.02
curtsy (v)	0.0	0.0	0.0	0.0	0.16	0.0	0.02
huddle (v)	0.0	0.0	0.0	0.0	0.12	0.0	0.02

Table 1. Similarity between actions (rows) and words describing the first task t_1 (columns).

By looking at the results of the overall processing that are reported in Figure 2, it is possible to make the following considerations:

- the automatic mapping is able to return results that are reasonable for the given context;
- by using an indirect mapping, some commands could be re-used in different contexts, with different meanings, thus limiting the commands to be memorized by the user;
- in some cases, two ore more commands (e.g., *bow* and *greet*) could activate the same action (e.g., *curtsy*). This could occur when the commands show a comparable similarity with an action, and reflects what happens in the real world.

Experimental results

In order to validate the effectiveness of the proposed technology, we devised an experimental setup centered on the creation of the interaction logic of a game in Blender. We focused on a simple storyboard, where the elephant character introduced in the above examples needs to be controlled in four possible game scenes. In the first three scenes, the elephant is in a cage: at first, it is moving in circle trying to attract visitors' attention; when a boy enters the scene and gets close to the cage, the elephant makes a curtsy to possibly obtain a kind of reward; the boy offers a peanut to the elephant, which takes it with the trunk and eat it. In the fourth scene, the elephant is in the savannah, and crouches down.

Figure 2. Animation of the virtual character in the three contexts: circus (first row), cage (second row), savannah (third row). Context-based activation rules are reported on the left (vertical text).

Since the goal of the proposed system is to hide the complexity associated with the creation of the interaction logic by exploiting natural language descriptions of the scene and character's behavior to automatically choose the action to activate in a given context and the command to be issued by the player for activating it, we first managed to collect text data to process by

means an online questionnaire. Questionnaire was administered to 20 students enrolled in various degrees at our university with no experience in game design with Blender or similar tools.

Students were asked to go through four Web pages each showing a video with the elephant exhibiting one of the above behaviors. For each page, students had to provide a description of the context and, separately, of the main task(s) undertaken by the main character. Time to provide the descriptions was monitored for individual users and pages. The questionnaire and full catalogue of descriptions collected has been included as supplemental material and is also available at http://intelligenthci.altervista.org.

Information extracted by the semantic engine for the descriptions provided by each of the 20 students was used to generate the corresponding logic bricks in Blender. An example reporting the logic created for the descriptions by student S#8 (see catalogue) is illustrated in Figure 3. Dashed boxes identify bricks for the third scene, which was described as THE ELEPHANT IS INSIDE ITS CAGE. A BABY HANDS A PEANUT TO IT. The elephant nibbles the peanut.

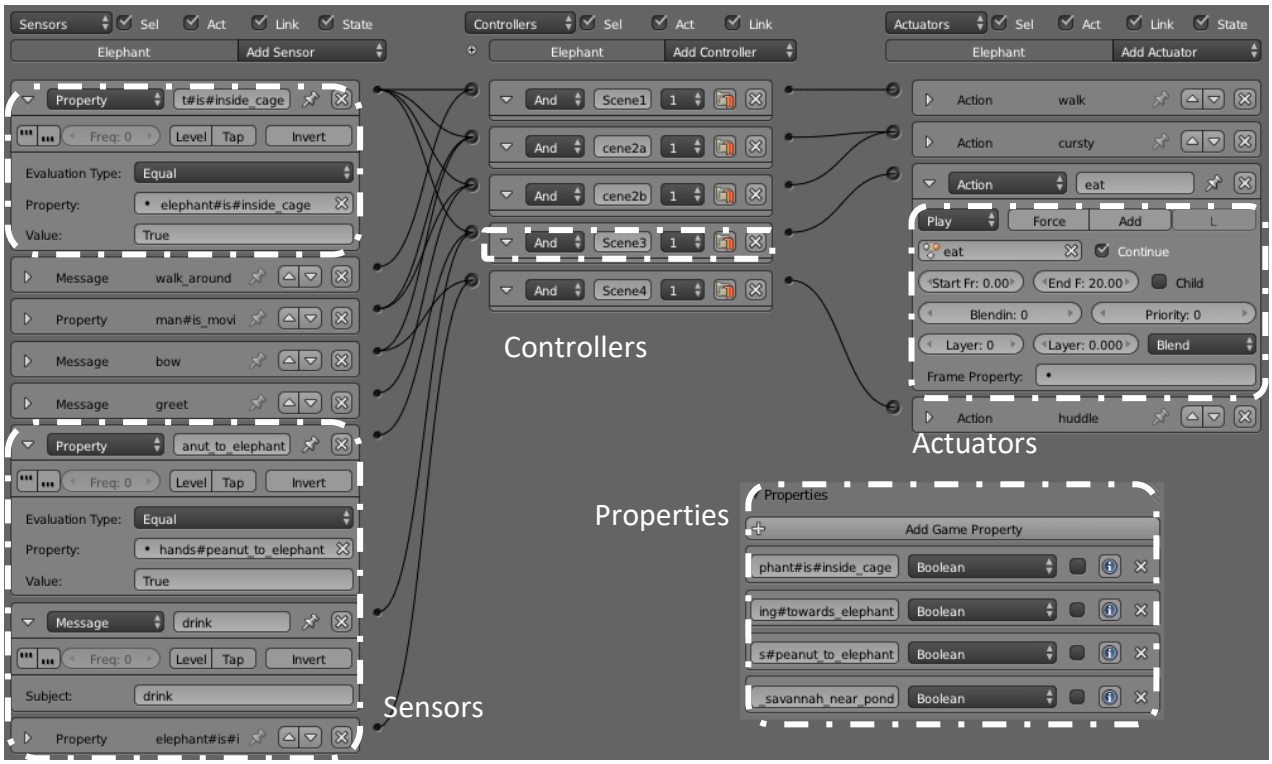

Figure 3. Logic bricks in Blender Game Logic editor created by the semantic system for a given description.

The system identified two variables, elephant@is@inside_cage and man@is_moving@ towards_ elephant, and created as many properties for the Game Engine object named *Elephant*. It then mapped action description onto *Elephant*'s action named *eat*, and linked it to activation command *drink*. Finally, it created two property sensor bricks (one per variable) plus one message sensor brick (for the command). Sensors bricks are connected to an AND controller. Controller's output feeds an action actuator brick that actually plays the correct action when conditions are met.

We started to quantitatively assess the performance of the semantic-based algorithm by measuring the accuracy of indirect mapping. Specifically, regarding context descriptions, we evaluated system's ability to generate the set of activation conditions (i.e., logic variables) to be

used as sensor bricks in the Blender's Game Logic editor. In 95% of the cases, the system extracted the expected number of variables, each with the right meaning. In 5% of the cases, number of variables was either overestimated or underestimated (mainly because of parsing errors), with possible mistakes in the structure of the triples.

Concerning character's behavior, we evaluated system success in automatically selecting, for the considered context, the expected interaction command and the corresponding action (linked to its actuator logic brick). Tabulated results can be accessed from the above link. For all the sentences, when the action was properly identified, the system assigned the correct command. In 18% of the cases, the wrong action was extracted, whereas In 6% of the cases the system was not able to distinguish among two or more actions (and proposed all of them, asking the user to make a decision about the final assignment).

Results obtained in the considered scenario confirmed that semantics could be regarded as an interesting technique for significantly ease tasks related to the coding of the interaction logic, making this pipeline step flexible enough to be approached also by unskilled users.

We then moved to consider how the proposed methodology compares to traditional techniques adopted by game designers. To this aim, we computed the average time required to the students for producing the descriptions of all the four scenes (student looking at the video and writing down the text), approximately equal to 110s (standard deviation σ = 34s). Semantic processing and logic generation required less than 6s per student, on average.

System performance was evaluated by comparing the time needed to obtain the automatically-generated logic with baseline measurements collected for three skilled users using the native coding strategy based on the visual programming interface of the Game Logic editor. The analysis also considered the time required by the users (professors at our university teaching game design in Blender) to fix the mapping errors in the student-generated logics. Average time required by the reference users to create the logic for the four scenes was 338s (standard deviation 63s). By considering the time spent to fix the errors (average 75s, σ = 64s), the average time needed to automatically produce correct logics was 172s (σ = 61s).

Results above quantified the efficiency gain that could be obtained by means of the designed system, which proved to be capable to produce the interaction logic of a game starting from text-based descriptions in a time that is, on average, about half the time required by skilled users operating in a traditional way.

Conclusions

In this work an intelligent human-computer interaction system easing the definition of the behavior of a virtual character in synthetic environments in response to user's interactions has been presented. Users exploiting the proposed system for the creation of games and animations in general could be relieved from the burden of manually processing the library of actions prepared for the character and (learn how to) manually mapping them with possible commands. Moreover, the natural language-based description of tasks to be accomplished by the character in different contexts of the virtual environment could let the system possibly work with a limited set of commands, thus reducing the number of interactions to be learned by the user.

It is worth observing that, although the effectiveness of the devised system has been assessed at present only in 3D animation scenarios, it could be possibly exploited in other contexts of intelligent human-computer interaction like those mentioned in the Introduction.

Future works could be devoted to make the interaction logic even more intelligent, by automatically producing the meta-information to be assigned to commands and actions, and by designing new strategies mixing semantic processing of text-based descriptions with information concerning, among others, command robustness and intuitiveness, users' preferences, etc.

References

[1] Lew M., Bakker E.M., Sebe N., Huang T.S., "Human-computer intelligent interaction: a survey," Human–Computer Interaction pp.1-5,2007.

[2] Hartmann M., "Context-aware intelligent user interfaces for supporting system use," TU-Darmstadt,2010.

[3] Pantic M., Nijholt A., Pentland A., Huanag T.S., "Human-centred intelligent human-computer interaction (HCI2): how far are we from attaining it?," Int. Journal of Autonomous and Adaptive Communication Systems, vol.1, no.2, pp.168-187,2008.

[4] Chandak M.B., Dharaskar R., "Natural language processing based context sensitive, content specific architecture and its speech based implementation for smart home applications," Int. Journal of Smart Home, vol.4, no.2, pp.1–10,2010.

[5] Jusoh S., Al Fawareh H.M., "An intelligent interface for a housekeeping robot," Proc.5th Int. Symposium on Mechatronics and Its Applications, pp.1-6,2008.

[6] Nganji J.T., Brayshaw M., Tompsett B., "Describing and assessing image descriptions for visually impaired web users with IDAT," Proc.3rd Int. Conf. on Intelligent Human Computer Interaction, pp.27-37,2013.

[7] Poosapadi Arjunan S., Hans W., O'Connor J., Kumar D., Sahebjada S., Bastos T., "Towards better real-time control of smart wheelchair using subtle finger movements via wireless (blue-tooth) interface," Proc. Int. Conf. on Intelligent Human Computer Interaction, pp.1-5,2012.

[8] Samanta D., Ghosh S., Dey S., Sarcar S., Sharma M.K., Saha P.K., Maiti S., "Development of multimodal user interfaces to Internet for common people," Proc.4th International Conf. on Intelligent Human Computer Interaction, pp.1-8,2012.

[9] Lamberti F., Manuri F., Sanna A., Paravati G., Pezzolla P., Montuschi P., "Challenges, Opportunities, and Future Trends of Emerging Techniques for Augmented Reality-Based Maintenance," IEEE Transactions on Emerging Topics in Computing, vol.2, no.4, pp.411-421,2014.

[10] Jacob M. G., Li Y. T., Akingba G. A., Wachs J. P., "Collaboration with a robotic scrub nurse," Communications of the ACM, vol.56, no.5, pp.68-75,2013.

[11] Hsu Y., Rice K., Dawley L., "Empowering educators with Google's Android App Inventor: an online workshop in mobile app design", British Journal of Educational Technology, vol.43, no.1,2012.

[12] Smelik R.M., "A declarative approach to procedural generation of virtual worlds", Computer & Graphics, vol.35, pp.352-363,2011.

[13] Artzi Y., Forbes M., Lee K., Cakmak M., "Programming by demonstration with situated semantic parsing," 2014 AAAI Fall Symposium Series,2014.

[14] Lei Y., Uren V., Motta, E., "Semsearch: A search engine for the semantic web," Managing Knowledge in a World of Networks, pp. 238-245,2006

[15] Montuschi P., Lamberti F., Gatteschi V., Demartini C. "A semantic recommender system for adaptive learning," IT Professional, September/October 2015, In Press

[16] Leung C. H., Chan Y. Y., "A natural language processing approach to automatic plagiarism detection," Proceedings of the 8th ACM SIGITE Conf. on Information technology education, pp. 213-218,2007.

[17] Coyne B., Sproat R., "WordsEye: an automatic text-to-scene conversion system," Proc. of the 28th Annual Conf. on Computer Graphics and Interactive Techniques, pp. 487-496,2001.
[18] Tutenel T., Bidarra R., Smelik R. M., Kraker K. J. D., "The role of semantics in games and simulations," Computers in Entertainment, vol.6, no.4, p.57,2008.
[19] Lopes R., Bidarra R., "A semantic generation framework for enabling adaptive game worlds," Proc. of the 8th Int. Conf. on Advances in Computer Entertainment Technology,2011.
[20] Baran I., Vlasic D., Grinspun E., Popović J., "Semantic deformation transfer," ACM Transactions on Graphics, vol.28, no.3, p.36,2009.
[21] Tang J. K. T., Leung H., Komura T., Shum H. P. H., "Emulating human perception of motion similarity," Computer Animation Virtual Worlds, vol.19, no.3-4, pp.211–221,2008.
[22] Pedersen T., Patwardhan S., & Michelizzi J., "WordNet:: Similarity: measuring the relatedness of concepts," Demonstration papers at hlt-naacl 2004, pp. 38-41,2004.
[23] Björkelund A., Hafdell L., Nugues P., "Multilingual semantic role labeling," Proc.13th Conf. on Computational Natural Language Learning, pp.43-48,2009.

Valentina Gatteschi (valentina.gatteschi@polito.it) is a Postdoctoral Research Assistant at Politecnico di Torino, Italy.

Fabrizio Lamberti (M'02-SM'14) (http://staff.polito.it/fabrizio.lamberti, fabrizio.lamberti@polito.it) is an Associate Professor at Politecnico di Torino.

Paolo Montuschi (M'90-SM'07-F'14) (http://staff.polito.it/paolo.montuschi/, paolo.montuschi@polito.it) is a Professor of Computer Engineering at Politecnico di Torino.

Andrea Sanna (http://sanna.polito.it, andrea.sanna@polito.it) is an Associate Professor at Politecnico di Torino.

APPENDIX (supplemental material)

Sidebar "Intelligence in games"

This section and accompanying references could be possibly moved in a sidebar or appendix.

People usually think at intelligence in games as the logic behind the game, i.e., to the so-called artificial intelligence. However, intelligence could be involved also in different stages of games lifecycle [1], from games generation, to players' behavior learning, etc.

Before analyzing the different aspects of intelligence in games, it could be interesting to take a step backward and to examine the game creation process with frameworks like Blender, Unity and Project Spark. Here, game creators initially populate the virtual environment with elements/objects by dragging and dropping them in the scene. In some cases (Project Spark), elements appearance and size could be easily modified with brushes (similar to the ones used in paint programs). Once the scene has been populated, the remaining core step consists in embedding intelligence in the game, by describing the elements (and characters) behavior in terms of reaction to user's commands, or to interactions with other elements of the game. This results in the specification that *if something occurs, the element/character should react in a given manner*, and could be done by means of graphical interfaces and scripts (Blender, Unity), or by using simplified graphics-based meta-languages and ready-made behaviors (Project Spark).

Even though such frameworks proved to be capable of speeding up games creation activities, it is evident that the above steps could still require a considerable amount of time, especially when the number of scenes and elements increases. Hence, a lot of efforts have been devoted to use artificial intelligence for automatically generating 3D scenes and stories [2], e.g., by exploiting natural language processing techniques. More complex approaches involve also automatic object positioning, based on spatial constraints specified for each object [3, 4], or by learning the constraints from similar pre-developed example scenes [5]. Objects' meta-information could describe not only spatial information such as dimension, orientation and space, but also the relations with other objects. Other approaches exploit real data, such as Linked Open Data, to automatically create the scenes [6].

Other research activities aim at investigating whether intelligence could be used to automatically define also the behavior of objects and characters. This is the case of [7], in which user's interactions are processed in order to trigger a chain of cause-effect-based events affecting objects in the scene. In [8] and [9], agents are used to automatically define the actions of crowds and characters. The approach in [10] takes a step forward and enriches non-player characters with emotions, computed based on their goals and on what happens in the scene.

Intelligence could be also used to adapt the game to the player, in order to make it less predictable and more challenging. For instance, the work of [11] models players (in terms of skills, preferences, etc.) starting from their actions in the game and change game's scenes accordingly.

Finally, intelligence could be exploited also in the definition of new strategies for player-game interaction. For instance, [12] proposes a sketching interface allowing the player to interact with the game by drawing shapes on a piece of paper, whereas [13] uses gaming sensors in an intelligent way, letting the user make complex character animations by identifying the object the character is interacting with, and integrating physical constraints.

The present works contributes to the above areas by focusing on another aspect of games creation, that involves automatic modeling of character's behaviors based on commands issued by the user, as done in [13]. Similarly to [13], the proposed system exploits a gaming

sensor to acquire user's commands but, instead of using rules defined by the game creator to chose the right action to be played, it relies on semantics to automatically compute the similarity between user's commands and character's actions, thus making it easier to insert new actions/commands and reducing the efforts associated with the rules definition phase.

[1] Yannakakis G.N., Togelius J., "A Panorama of Artificial and Computational Intelligence in Games,", IEEE Transactions on Computational Intelligence and AI in Games,In press

[2] Hanser E., Mc Kevitt P., Lunney T., Condell J., "Text-to-animation: affective, intelligent and multimodal visualisation of natural language scripts," School of Computing and Intelligent Systems, University of Ulster, Londonderry,2009.

[3] Trinh T. H., "A constraint-based approach to modelling spatial semantics of virtual environments," Doctoral dissertation, Université de Bretagne Occidentale-Brest,2012.

[4] Tutenel T., Smelik R. M., Lopes R., de Kraker K. J., Bidarra R., "Generating consistent buildings: a semantic approach for integrating procedural techniques," IEEE Transactions on Computational Intelligence and AI in Games, vol.3, no.3, pp.274-288,2011.

[5] Dema M. A., Sari-Sarraf H., "3D scene generation by learning from examples," Proc. of the 2012 IEEE International Symposium on Multimedia, pp.58-64,2012.

[6] Warren R., Champion E., "Linked open data driven game generation," Proc. 13th Int. Semantic Web Conference, pp.358-373,2014.

[7] Lugrin J. L., Cavazza M., Crooks S., Palmer M., "Artificial intelligence-mediated interaction in virtual reality art," Intelligent Systems, vol.21, no.5, pp.54-62,2006.

[8] Kraayenbrink N., Kessing J., Tutenel T., de Haan G., Marson F., Musse S. R., Bidarra R., "Semantic crowds: reusable population for virtual worlds," Procedia Computer Science, vol.15, pp.122-139,2012.

[9] Grimaldo F., Lozano M., Barber F., Vigueras G., "Simulating socially intelligent agents in semantic virtual environments," The Knowledge Engineering Review, vol.23, no.4, pp.369-388,2008.

[10] Popescu A., Broekens J., van Someren M., "GAMYGDALA: an emotion engine for games", IEEE Transactions on Affective Computing, vol.5, no.1, pp. 32-44,2014.

[11] Lopes R., Bidarra R., "Adaptivity challenges in games and simulations: a survey," IEEE Transactions on Computational Intelligence and AI in Games, vol.3, no.2, pp.85-99,2011.

[12] Macret M., Antle A. N., Pasquier P., "Can a paper-based sketching interface improve the gamer experience in strategy computer games?," Proc. 4th Int. Conference on Intelligent Human Computer Interaction, pp.1-6,2012

[13] Ishigaki S., White T., Zordan V. B., Liu, C. K., "Performance-based control interface for character animation," ACM Transactions on Graphics, vol.28, no.3, p.61,2009.

Authors full bios

Valentina Gatteschi is a Postdoctoral Research Assistant at Politecnico di Torino, where she received the M.Sc. and the Ph.D. degrees in management engineering and computer engineering, respectively. Her main research interests are in semantics and natural language processing. She has been involved in several European projects on education. Contact her at valentina.gatteschi@polito.it.

Fabrizio Lamberti (M'02-SM'14) is an Associate Professor at Politecnico di Torino, Italy. He received the MS and the PhD degrees in computer engineering from Politecnico di Torino, Italy, in 2000 and 2005, respectively. He has co-authored more than one hundred technical papers in international books, journals and conferences mainly in the areas of computational intelligence, semantic processing, distributed computing, human-computer interaction, computer graphics, and visualization. He has served as General Co-Chair and TPC Chair of the 7th International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN2015) and has been involved in the Organizing and Technical Program Committees of other national and international conferences. He serves as an Associate Editor for IEEE Transactions on Emerging Topics in Computing and for IEEE Consumer Electronics Magazine. He is a member of the Editorial Advisory Board of several international journals and served/is serving as a Co-Guest Editor for six special issues appeared/to appear on Entertainment Computing, Computing and Visualization in Science, Sensors and IEEE Transactions on Emerging Topics in Computing. He is an IEEE Computer Society member and IEEE senior member. Please visit his personal page at http://staff.polito.it/ fabrizio.lamberti and contact him at fabrizio.lamberti@polito.it.

Paolo Montuschi (M'90-SM'07-F'14) is a Professor of Computer Engineering at Politecnico di Torino, Italy, where he served as Chair of Department from 2003 to 2011 and as Chair or Member of several Boards including the Board of Governors. He obtained a PhD in computer engineering in 1989, and since 2000 he has been full Professor. He is serving as Editor-in-Chief of the IEEE Transactions on Computers (2015-16), as a Member of the Advisory Board of Computing Now and as Member-at-Large of the IEEE Publication Services and Products Board. Previously, he served as Chair of the Magazine Operations, of the Electronic Products and Services and of the Digital Library Operations Committees, Member-at-Large of the Computer Society's Publications Board, and Member of the Board of Governors of the IEEE Computer Society. He served as Guest, Associate Editor and Associate Editor-in-Chief the IEEE Transactions on Computers, as Associate Editor of IEEE TETC, as well as co-chair, program and steering committee member of several conferences. His current main research interests and scientific achievements are in computer arithmetic, computer graphics, electronic publications, semantics & education, and new frameworks for the dissemination of scientific knowledge. Montuschi is a Fellow of the IEEE, a Computer Society Golden Core Member and a life member of the International Academy of Sciences of Turin. Visit his web pages at http://staff.polito.it/paolo.montuschi/ and contact him at paolo.montuschi@polito.it.

Andrea Sanna is an Associate Professor at Politecnico di Torino. His research interests include computer graphics, virtual reality, parallel and distributed computing, scientific visualization, and computational geometry. Sanna has a PhD in computer engineering from Politecnico di Torino.
He is a senior member of ACM. Please visit his personal page at http://sanna.polito.it and contact him at andrea.sanna@polito.it.

Complete contact information

Valentina Gatteschi
Dipartimento di Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129, Torino, Italy
Phone: +39 011 090 7169
Fax: +39 011 090 7099
E-mail: valentina.gatteschi@polito.it

Fabrizio Lamberti
Dipartimento di Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129, Torino, Italy
Phone: +39 011 090 7193
Fax: +39 011 090 7099
E-mail: fabrizio.lamberti@polito.it

Paolo Montuschi
Dipartimento di Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129, Torino, Italy
Phone: +39 011 090 7014
Fax: +39 011 090 7099
E-mail: paolo.montuschi@polito.it

Andrea Sanna
Dipartimento di Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129, Torino, Italy
Phone: +39 011 090 7035
Fax: +39 011 090 7099
E-mail: andrea.sanna@polito.it