Design of 3D virtual neuropsychological rehabilitation activities

Sergi Grau, Dani Tost, Ricard Campeny, Sergio Moya, Marcel Ruiz
Computer Graphics Division of the Center for Bioengineering Research
Polytechnical University of Catalonia, Barcelona, Spain
Email: sgrau|dani|rcampeny|smoya|marcelrf@lsi.upc.edu

***Abstract*—Virtual environments can be valid and efficient tools for neuropsychological rehabilitation of cognitive impairments. However, the specification of the virtual task is not easy for neuropsychologists who are not aware of game design and implementation issues. We present the method that we have followed to help therapists to describe the expected task functioning. Our system is split into different subsystems: interactions, task logic, messages, scoring and feedback. This way, therapists specify the requirements of each subsystem separately, which comes out to be easier and clearer. The game structure manages and composes the input and output of the different subsystems to run the game. All the interactions are specified according to a common syntax. This provides the desired flexibility to add, remove and modify interactions in run-time. In order to control the location of all the objects of the scenario, for each receiver surface, we define a pick-and-place grid that keeps record of the objects piled onto it. The game logic is implemented as a non-deterministic automaton that controls each task state, modifies in run-time the different subsystems and programs new interactions. Moreover, we have designed a graphical editor to describe the game logic in a simple way, focusing only on relevant states of the application.**

I. Introduction

Modern medicine has contributed to an increased survival of patients with brain damage following traumatic brain injury and stroke. Currently, brain damage is one of the first cause of disability in most developed countries. It causes physical, sensory and cognitive impairments along with diverse chronic diseases. Neuropsychological rehabilitation addresses cognitive, emotional and behavioral impairments. Its target cognitive deficits are memory, attention, language and executive functions involving planning, selection and decision making. Rehabilitation is key at a short-term as well as at a long-term to restore functions to pre-injury level and to learn ways to compensate for abilities that have been permanently lost. The ultimate goal of rehabilitation is to help patients to reach the most independent level of functioning possible [1].

In the last decade, neuropsychologists have shown a growing interest in computer graphics technologies [2]. Three-dimensional virtual environments (VE) used for leisure games were early introduced for training, especially in the military context, for instance in flight simulators [3]. Moreover, they have attracted the attention of educators who have seen the advantages of using video and computer-games for learning [4]. The specific features of video-games that have shown to be valuable in the education context are the sense of challenge and competition, the motivation, the feedback, diversity and adaptivity of the difficulty level to learner's aptitudes.

In rehabilitation, VE can be privileged places to re-learn and train activities of daily living (ADL). The advantages of virtual environments in neuropsychological rehabilitation (from now on Neuropsychological Virtual Rehabilitation, NVR) are potentially numerous [2]:

- Ecological value: in psychological terms, it means that familiar, meaningful scenarios can be created to realize tasks, in contrast to traditional 2D decontextualized rehabilitation activities.
- Range and diversity: many different virtual scenarios and a wide range of activities can be designed.
- Flexibility: the difficulty level, the tasks themselves and the environments can be adapted to the patient's capacities.
- Safety: virtual environments are almost totally safe.
- Control: the situations that can arise in the VE can all be foreseen and controlled.
- Timing: the temporal dimension in the VE can be integrally programmed and customized to patient's behavior.
- Reproducibility: situations can be recreated exactly as many times as desired.
- Documentation: reporting successes as well as errors can be done automatically.
- Cost: VE are generally cheaper than equivalent physical scenarios. In attention, because of their high level of automatization, they require less continuous surveillance from specialists while they are being used by patients.

Despite all these potential benefits, and although the first experiments in using VE in rehabilitation started early in the 90's [5], most "serious games" for cognitive rehabilitation are still pilot experiences, and very few institutions use them in the current routine practice [6]. The gap between expectations and reality is often attributed to the fact that computer graphics technology was not enough advanced by the time neuropsychologists were first attracted by virtual environments [7]. However, the spectacular development of computer graphics in the last years, promoted by the apparition of programmable hardware, has not yield to a comparable enhancement of virtual rehabilitation applications. We believe that the difficulty of designing effective rehabilitation tasks using VE has been underestimated. Even if the basic technology can now be

978-0-7695-3986-7/10 $26.00 © 2010 IEEE
DOI 10.1109/VS-GAMES.2010.17

considered available, its use is not trivial. Virtual rehabilitation tasks, their goals, steps and scoring mechanisms are at the core of the therapy. Therefore, it is neuropsychologists who must specify them. Unfortunately, neuropsychologists are generally unfamiliar with the video-game technology. Describing the tasks and their rules so that they could be implemented is complex and tedious. Although technology can provide the desired feedback, timing and control mechanisms, deciding when and how they should be activated is often puzzling. In multi-disciplinary teams, this difficulty is overcome through discussions between technical and clinic participants. However, although the process is fruitful, it is slow.

In leisure games the initial game idea and its further development is carried on by a team of artists, designers, engineers and programmers who are all game specialists [8]. Moreover, game developers are regular game players. It is often their game love that has driven them to orient their professional life towards the game industry. Game development teams have a common language and game structure knowledge. By opposite, neuropsychologists generally play and know little about video-games. They are not well aware of the potentialities and the technological difficulties of game design. In general, there is a big gap between what they believe is feasible in VE and what it is actually possible to implement and do in real time. On the other side, game developers don't have formation in cognitive problems, and are often unable to assist therapists in the tasks specification. The importance and complexity of an action from a neuropsychological point-of-view do not always correspond to the complexity of its implementation in the VE. As a example, in a collecting things activity such as virtual shopping, picking can be considered more important for rehabilitation than dropping the selected object in a bag, because picking involves more attention, memory and decision making. By opposite, from a computer graphics point of view, dropping can be more difficult than picking, because it may involve collision detection and gravity simulation.

A consequence of this situation is that developing NVR tasks is slow and costly. It requires multiple iterations steps with frequent changes in the tasks specification resulting in a lot of code reprogramming. NVR applications are not as stable as desired, and they don't always provide all the benefits they ought to.

In this paper, we describe the system that we have developed for designing NVR tasks. It is intended at bridging the gap between neuropsychologists and game developers, and at providing faster, more versatile and therapists-friendly means of developing neuropsychological contents in VE. The main strength of the system are: (i) the separation between the different subsystems that compose the game, (ii) the categorization of the interactions and their definition according to a common syntax, (iii) the pick-and-place managing system and (iv) the task logic automaton together with its graphical editor.

II. Related work

VE can be used as theaters where to play roles, as well as grounds where to play games. The relationship between psychology and theater finds its roots at the beginning of the 20th century around the *therapeutic theater* concept. The dramatist Stanislavsky [9] showed that actors should find in themselves the thoughts and emotions of their characters through introspection and purposeful physical actions [9]. Then, Jacob Moreno, creator of the psychodrama [10], experienced that people could gain more from acting out their problems than from talking about them. VE can recreate any type of stage and setting adapted to the characteristics of any virtual actor. This is why they have early been used in the treatment of psychiatric and psychological disorders such as panic and agoraphobia [11], sexual dysfunctions [12] and eating disorders [13]. Moreover, they have been used to train instrumental activities such as driving [14] and public speaking [15]. In this type of applications the challenge is acting, being able to be and stay in situations that are feared. In general, the interactions of users with the system are not very complex, being navigation the most important one.

The psychologist Piaget [16] showed the relationship between game playing and personality development. In play, children expand their knowledge of the physical world, they learn to communicate and understand others. VE and, more specifically, video and computer-games have proved to be excellent tools for learning. They bring challenge, competition and feedback and are highly motivating. Last year, De Freitas et al. [17] reported 80 VE-based learning projects and 100 more planned for 2009. Learning applications can use currently existing VE, such as Second-life and OpenSim [18]. Alternatively, specific scenarios can be designed. Rooney et al. [19] relate their experience in designing serious games for higher education. They expose the communication problems caused by the background diversity of the project's partners. To overcome these difficulties, they propose to adopt collaborative work strategies such as periodic meetings, blogs and rapid prototyping.

The complementary function of role playing and game playing can be usefully exploited in NVR. Unlike users of previously described experiences, patients with brain damage have severe cognitive and often motor impairments. Thus, standard video-games and VE cannot not readily be used in NVR. Timing should be slower, and the number of stimuli lower and more progressive than for the rest of people. Moreover, loosing the game or leaving tasks unfinished can be sometimes counterproductive, and a free-of-error learning may be preferable [1]. In this case, systems should complete the task and correct the errors when needed. Moreover, the goal of this type of rehabilitation is to realize very simple tasks, activities of daily living (ADL), that are not considered challenging in commercial games.

There are two main types of NVR systems for patients with brain damage: motor rehabilitation for musculo-skeletal recovery following bone or muscle damage [20] and cognitive

rehabilitation. Motor rehabilitation generally focus on specific functional activities such as grasping and moving objects [21]. Recently, it has been suggested that integrating game features in such systems could improve patients motivation [22]. A deep review of cognitive rehabilitation systems can be found in [2]. VE used in these systems are very diverse: buildings, supermarkets, classrooms, rooms, bathrooms and kitchens [23]. Typical tasks consists of collecting objects, for instance shopping, classifying objects to straighten up a room and following a sequence of step actions, such those needed for cooking a specific dish. During the realization of these tasks, patients are subjected to external disturbing stimuli such as telephone ringing and door knocking.

Papers describing NVR experiences are generally written from a neuropsychological perspective. The description of VE and tasks are brief, and they lack technological issues. Riva et al. [24] expose the need for easy-to-use and open-source VR software. They observe that commercial applications are often too closed to be tailored to each patient's necessity, and that the cost of creating virtual applications from scratch can be very expensive. They describe the NeuroVR open system, that provides a scenario editor including 12 pre-designed virtual scenes and a scenario-through navigating tool that works in immersive (stereoscopic) and non-immersive modalities. However, the system does not allow users performing actions in the scenario, and the implementation of tasks requires a lot software programing. In general, the design of the task logics is not addressed in the bibliography, and the use script-writing assistant tools, frequent in TV and movie is not reported. This is probably because most existing NVR applications have been designed on the basis of hand-to-hand iterative work between neuropsychologists and software developers. It is also the way we realized our first NVR application [25]. However, we found out that, although this methodology was fruitful and probably necessary at a first stage of collaboration, it was too slow, and that it was difficult to stabilize program versions. Applying collaborative work strategies such as those described in [19] for learning-games helps but is not sufficient. Because, in NVR, therapists have the most to say in the task design, they need new tools for specifying NVR tasks. Providing these tools is the motivation of our work and the central axis of the system that we describe in the next sections.

III. Structure

Our system is designed for single-user and first-person-shooter tasks. Patients navigate through the VE and perform cognitive tasks, ADL that require interactions with the elements of the environment. Because of the typology of the patients, we have simplified as much as possible the input mechanisms of the system. We allow navigation with the mouse plus the spacebar to start and stop movement. The interactions are done clicking a mouse button.

Figure 1 shows the general structure of the system. It is composed of different subsystems. The *task manager* is at the heart of the system. It runs the logic and, with the *interaction manager*, it manages the actions. It is also responsible of

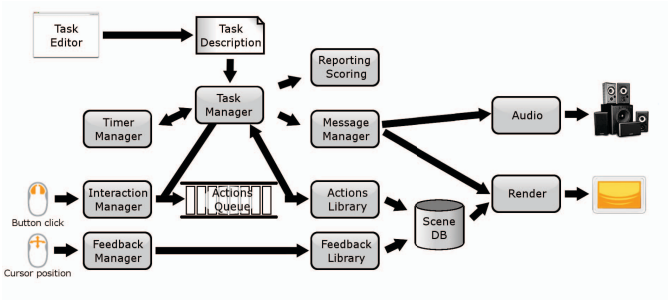

Fig. 1. System internal structure and modules.

scoring all the actions and generating a detailed report of task performing. The *time manager* controls time events, it starts chronometers, watches them and informs their end. The *feedback manager* controls the display modes. Finally, the *message manager* takes care of audio and written messages that contain instructions on the NVR task as well as feedback appreciations. These messages are enqueued in a priority message queue according to their relative importance, that is defined by therapists. Figure 2 shows an example of a message display.

Fig. 2. An example of message display.

We have implemented this structure using Blender Game Engine [26] for the rendering management and the PyGame library to handle sound [27]. We next review more deeply the most relevant elements of the structure.

Before designing the system, we first develop some 3D tasks on a hand-to-hand basis with the therapists. Then we discussed with them the idea of using this new system, and they assessed us through the design process.

A. Scene database

The graphical model of the scene *(scene DB)* represents the scenario. It is composed of fixed structural elements (doors, windows and static furniture) and movable objects. Figure 3 shows a VE representing a children room.

Each object has one or more "states". Objects pass from one state to the other through transformations. In run-time,

Fig. 3. Example of a scenario: a children room.

there can be various instances of an object at the same or at different states. For instance, as illustrated in Figure 4, the object "bottle" has three states: closed, opened and top (`bottle.closed`, `bottle.opened`, `bottle.top`). Figure 4 shows five different states of the object "egg": `egg.fresh`, `egg.shell`, `egg.raw`, `egg.fried` and `egg.burnt`.

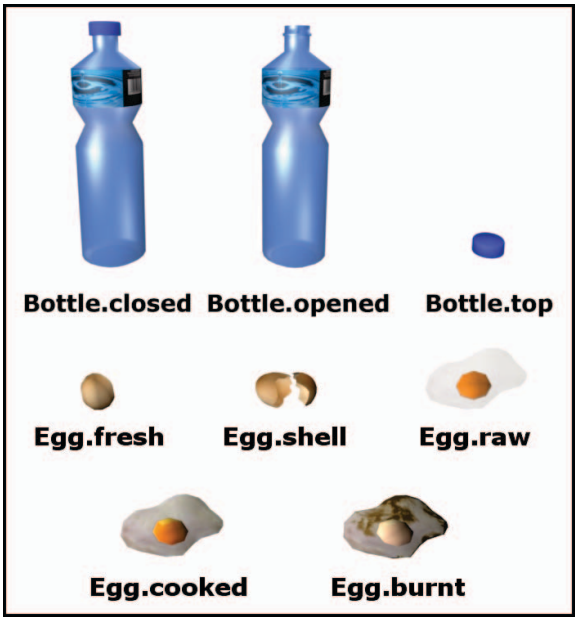

Fig. 4. Illustration of the different states of the objects; top: three states of the `bottle` object; bottom: five states of the `egg` object.

B. Actions library

The *actions library* is able to reproduce any kind of action performed in the real life. Actions consist of one or more of the following processes:

- to launch a pre-recorded animation, for instance, opening a door or animating a water texture when the faucet in turned on,
- to emit and stop a sound, as turning on and off a virtual radio,
- to apply a geometric transformation, as picking and moving,
- to create and/or remove objects instances. For instance, cutting or breaking an object into two pieces deletes the original object and creates two new instances of the object in a different state representing only a half of it.

Pick and place are especially important actions [28]. They have the following requirements:

- to avoid collisions of the picked object with the environment when it is carried by the patient's avatar,
- to avoid penetrations of an object placed on a surface with the surface or with other objects laying on it,
- to provide natural and physically valid orientations of the objects when moved and leaved,
- to allow recursive piling and unpiling of objects on top of others (see Figure 5),
- to provide control on the position of the objects.

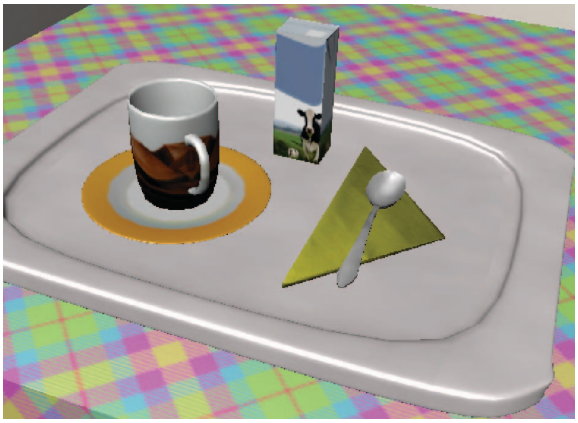

Fig. 5. Need for piling objects and moving them together.

In order to solve these difficulties, for all the objects on which placing is allowed, we create and actualize a grid that stores for each cell the identifier of the object laying on it. The grids are created during the VE design. Some cells can be defined as impracticable to prevent patients from leaving objects on them. Moreover, for each object, we store the geometric transformation that orient it best during transportation and when deposited. Finally, when the patient's avatar picks an object, we move it at the avatar's position to avoid collisions, and we scale it to prevent it from occluding the scene. Figure 6 shows an illustration of the grid concept on a frying pan. Since the pan is circular and the grid is rectangular, the corner cells (in red in the image) are banned.

C. Actions queue and interaction manager

The button click events are handled by the *interaction manager* that interprets them as actions requests. The *interaction manager* determines which specific actions should be done and, if any, it enqueues them in the *action queue*.

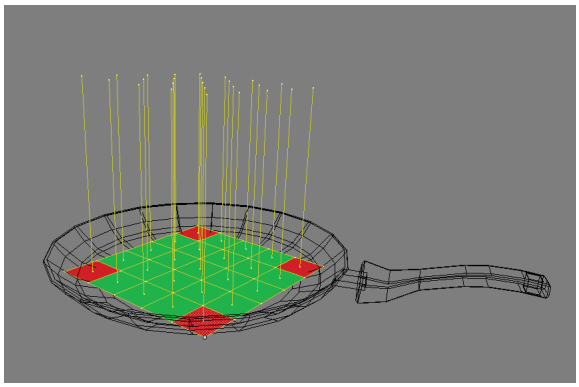

Fig. 6. The pick-and-place grid defined on the `frying pan` object. Each grid cell indicates if it is free or not (green or red) and the maximum height (yellow line) that an object piled on the grid can have.

The actions share the same syntax. They have an emitter, a receiver and the action identifier. Specifically, we define an action as `(emitter,actions-id,receiver)`. When users perform an interaction, the *interaction manager* identifies the emitter as the patient's avatar or as an object carried by it, and it finds the receiver as the first object hit by the vision ray. The *interaction manager* manages a data structure called the *interaction table* that indicates for any pair of emitter and receiver the set of actions that should be performed when an interaction occurs between the pair. For example, in a scene with a closed water bottle object (see in Figure 4 `bottle.closed`), when the user's avatar produces an interaction on `bottle.closed`, the *interaction manager* is able to generate the action `(avatar,open,bottle.closed)`. When the *actions library* receives this action, it replaces the `bottle.closed` instance by two object instances: `bottle.opened` and `bottle.top`. Then, when the avatar produces an interaction on `bottle.opened`, the action `(avatar, pick, bottle.opened)` is launched, and a geometric transformation is performed to put `bottle.opened` on top of the user's avatar.

The *interaction table* is created at the beginning of the task, and it can be modified at any time by the *interaction manager*. This way, some actions can be prohibited during runtime, others can be allowed and even modified. This allows us to regulate the task difficulty level. As an example, patients may start the task being allowed to open any cupboard, but, if they fail to open the one that is needed for their task, the *task manager* can modify the interaction table and allow the action opening only on the right cupboard. Following the example of the bottle, in order to simplify the patient's activity, the *interaction manager* can associate to the pair `(avatar, bottle.closed)` the sequence of actions `open` and `pick`. This way, when the avatar will click on the closed bottle, it will automatically be opened and picked. To avoid disorienting users when the behaviour of the application changes, we can program missages and pauses to warn them.

D. Feedback manager

During navigation, the cursor position is handled by the *feedback manager* which decides the rendering mode of the objects hit by the vision-ray. This way, we can modify the appearance of objects that users focus on. We can highlight them, change their color, size or shading mode to make them easier to detect. Figure 7 shows six different feedback modes of a yogurt pack. The *feeback library* is responsible of applying these changes in the rendering modes.

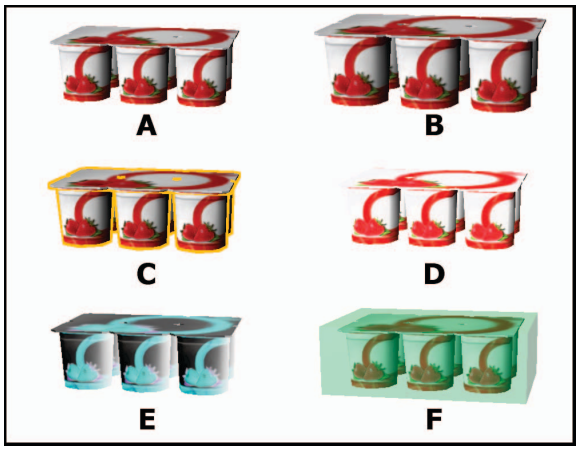

Fig. 7. Six different feedback modes to enhance objects: non-enhanced (A); enlarged (B); silhouette highlighting (C); luminance growth (D); inverse hue (E); semi-transparent bounding-box (F).

E. Task manager

In run-time, at each time clock, the *task manager* processes the actions stored in the *actions queue*. These actions are mainly enqueued by the *interaction manager* as a result of users interactions, but they can also be directly programmed by the *task manager*. In this way, if patients fail to perform a needed interaction, the *task manager* can program it. For instance, if the task goal is to open a wardrobe in order to pick a garment, and the patient doesn't open it, after a time span, the *task manager* will enqueue itself the opening action. Moreover, the task manager can make invisible a certain number of objects not directly related to the scene. This way, depending on the patient's behavior, the scenario is progressively simplified, reducing the number of visual stimuli.

The *task manager* follows the task logic model that is constructed from a *task description file* generated by the task editor. It is based on an implementation of a non-deterministic finite automaton (NFA) that controls in which state is the main task and its subtasks.

The input of the NFA evaluation (tokens) are the actions performed during the execution of the game. The task description is expressed, like in other declarative languages, in terms of relations, represented as rules and facts. The single data type is the *task*. Tasks can be of seven classes: *simple, and, or, seq, rep, not* and *any*:

- `SimpleTask`: it corresponds to an action between two objects of the scene, and it is described as $T(emitter, action, receiver)$. This task is fulfilled when the input token matches the specified action.
- `AndTask`: $AND(subtask_1, subtask_2, \ldots, subtask_n)$ is a list of subtasks that is fulfilled when all the subtasks are fulfilled.
- `OrTask`: $OR(subtask_1, subtask_2, \ldots, subtask_n)$ is a list of subtasks that is fulfilled when at least one of the subtasks is fulfilled.
- `SeqTask`: $SEQ(subtask_1, subtask_2, \ldots, subtask_n)$ is a list of subtasks that need to be fulfilled sequentially.
- `RepTask`: $REP(subtask, N = 0, M = \infty)$ is the repetition of a subtask a number of times between N and M, where N and M default values are 0 and ∞.
- `NotTask`: $NOT(subtask)$ is fulfilled for any task except the specified subtask.
- `AnyTask`: ANY is fulfilled for any input token.

The language also accepts some syntactic sugar designed to make things easier to read or express. For example $RANY$ represents $REP(ANY)$, and $LSEQ(task_1, task_2)$ represents $SEQ(RANY, task_1, RANY, task_2, RANY)$.

In order to show the use of this language, we next define an activity consisting of filling a glass with water. In the scene, users are only allowed to pour water in a glass from a bottle that is closed. The task is defined as:

```
openBottle          = T(avatar,open,bottle.closed)
pickBottle          = T(avatar,pick,bottle.opened)
fillGlass           = T(bottle.opened,fill,glass)

fillGlassWithWater = SEQ(openBottle,pickBottle,fillGlass)
```

The task `fillGlassWithWater` defines the goal of the game. However, if users were allowed to do other things aside from pouring the water into the glass, like picking the phone when it rings, the task should be redefined to be combined with these other tasks:

```
fillGlassWithWater = SEQ(RTN(openBottle),
                         openBottle,
                         RTN(pickBottle),
                         pickBottle,
                         RTN(fillGlass),
                         fillGlass,
                         RANY)
```

where RTN(task) represents REP(AND(ANY,NOT(task))).

Defined in this way, the task allows us evaluating the user actions and determining if the goal of the game is reached. However, it gives neither user feedback, nor report on the user interactions for a later therapists analysis. In order to modify in real-time the other subsystems, like *message manager* and *feedback manager*, we have defined new tasks in the language in addition to the seven basic ones. These new tasks do not distort the logic sequence, and they add game control. For example, $MSG(m)$ allows to send a message to the *message manager*, $INT(emitter, action, receiver)$ substitutes the interaction defined between the emitter and the receiver in the *interaction table* by the given action. $TIMER(id, s)$ allows to control if s seconds have passed since the start of the timer with identifier id. Using the same example as before, we can define a new rule that controls if the patient does not open the bottle during the first five minutes. If he doesn't, the system shows a notification message that recalls to the patient what he should do. In addition, the system packs the actions `open` and `pick` in a single interaction `open+pick`:

```
easyWay = SEQ(AND(TIMER(id,300),
                  RTN(openBottle)),
              MSG("Pick the bottle"),
              INT(avatar,open+pick,bottle.closed))
```

The benefit of this language is the flexibility to create new tasks and to combine existing ones in an intuitive way, avoiding to program a deterministic automaton that contemplates all the possible cases that may happen before the goal is reached. However, for non-expert programmers, in our case neuropsychologists, it is difficult to write such tasks description without errors. We describe the solution to this problem in Section IV-B.

IV. Task specification process

A. Methodology

In order to develop a NVR task, we discuss with neuropsychologists each part of the system separately, following this sequence:

1) Synopsis: therapists provide a brief description of the scenario and the type of activity they want to develop.
2) VE design and validation: using the synopsis, we design the VE, and we fill it with objects related to the projected activities. We have an objects library from which we reuse objects from one scenario to the other. We submit the scenario to the therapists evaluation and after few iterations, the scenario is fixed.
3) Navigation and feedback model: therapists define the sequence of feedback modes they want to be applied for progressively easier navigation. Each feedback mode corresponds to a feedback difficulty level.
4) Visibility: Therapists define a sequence of visibility levels. The visibility level is the percentage of objects non related to the activity that are visible in the scene: the more objects, the more stimuli and the higher difficulty.
5) Interaction table: therapists define the initial state of the interaction table. They define the actions that they want to be realized for all the pairs of emitter and receiver. According to this definition, we complete the *action library* with new procedures for eventual new actions. They define the level of difficulty of the task according to the level of granularity of the actions: the more packed actions (`open+pick`, for instance), the easier the task.
6) Messages: therapists define which messages should be displayed. The message are stored in an audio and textual dictionary from which they can be retrieved.

Therefore, therapists only need to select existing message from the dictionary, and, eventually edit new ones. For each message, they define when it must be emitted, at which frequency and the maximum number of repetitions. The frequency and number of repetitions can be modified during task running. Therefore, associated to these parameters, as for the feedback model and the visibility, therapists define decreasingly levels of message difficulty.

7) Scoring: therapists define the scoring rules for all the actions realized by the patient. Actions related with the task have a different score from those non related to it. Score depends on the level of difficulty of the feedback, the visibility, the message and the interactions.
8) Task description: therapists define the goals and rules.
9) Program development and testing: with all these definitions, we finally compose the NVR activity and test it before submitting it to the therapists validation.

B. NVR Editor

In order to make it easier for therapists to define all the parameters described above, we have designed a graphical editor. It is composed of different flaps, one for each subsystem. It manages the dictionaries that describe the object instances, the actions and messages.

The edition of the initial interactions table starts with default values. Therapists can modify the table entries one pair by one, or using advanced pair selection tools. For instance they can automatically disable all interactions except those directly related with the activity goals.

To edit the tasks and program the messages, users use a graphical interface inspired on Blender nodes. There is a node type for each type of task (`simpleTask`, `AndTask`, `MSG`, ...). Users relate the input and output of these nodes to express the task description. Figure 8 and Figure 9 show the graphical edition example of the rules, described in Section III-E, called `fillGlassWithWater` and `easyWay`, respectively.

After therapists have introduced all these data with the editor, the application itself generates automatically all the information needed by the system, and the NVR activity is ready to use.

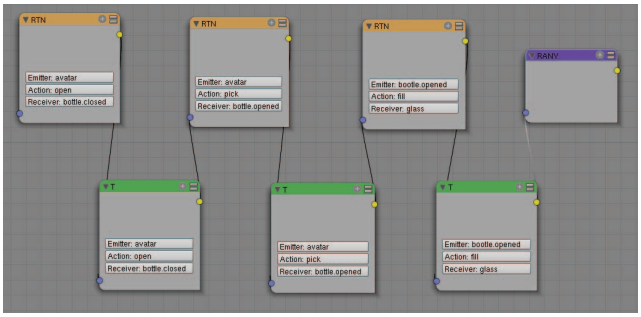

Fig. 8. Graphical edition of the `fillGlassWithWater` task. The task is a sequence of seven nodes. It uses three different kind of nodes: `SimpleTask` (green), `RTN` (orange) and `RANY` (lilac).

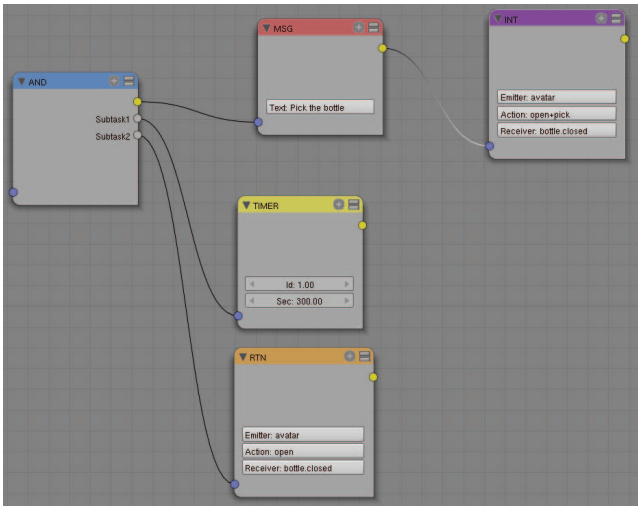

Fig. 9. Graphical edition of the `easyWay` rule. It uses an `AndTask` node (blue), a `MSG` node (red), a `TIMER` node (yellow), a `RTN` node (orange) and a `INT` node (violet).

V. Conclusion and Future Lines

Neuropsychological rehabilitation is an excellent application field for serious games. The development of NVR tasks requires cross-disciplinary teams in which therapists assume the task logic specification, and software developers together with graphics designers, the game programming and implementation. The success of the project depends greatly on how fluent and productive the dialog between the two parts is. In this paper, we have described a working methodology based on a modular software structure and on specification tools that greatly enhance the effectiveness of the NVR tasks design and implementation. We are currently using this system in the definition of NVR tasks. Our feeling is that the process is much faster than the previous rapid prototyping method. Moreover, the cross-team communication is more fluent and productive.

The task logic editor is meant for neuropsychologists involved in task designing. In requires a good knowledge of logics and a training period. Further on, we would like to make it extensive to a wider range of therapists, to allow tasks modifications and variants. For this, we plan to design a second layer on top of the editor, that would allow less skilled users to describe tasks in a more literary way, avoiding to manipulate logic symbols.

Multiplayer games may bring to NVR a useful sense of collaboration [29]. Our system currently supports single-user activities. However, it can be extended to collaborative environments by expanding its logical structure. This is the goal of our future research.

Acknowledgment

This work has been partially funded by project Neurolearning from the Avanza plan of the Spanish Ministery of Industry, Tourism and Trade.

REFERENCES

[1] G. Prigatano, "Learning from our successes and failures: Reflections and comments," *International Neuropsychological Society*, vol. 3, pp. 497–499, 1997.

[2] F. Rose, B. Brooks, and A. Rizzo, "Virtual reality in brain damage rehabilitation: Review," *CyberPsychology & Behavior*, vol. 8, no. 3, pp. 241–271, 2005.

[3] "Serious games in defence education," 2008.

[4] S. Freitas, "Learning in immersive worlds," JISC e-learning programme, Tech. Rep., 2006.

[5] A. Rizzo and J. Buckwalter, "The status of virtual reality for the cognitive rehabilitation of persons with neurological disorders and acquired brain injury," *Study Health Technology Information*, no. 39, pp. 22–33, 1997.

[6] A. Rizzo and K. Jounghyun, "A swot analysis of the field of virtual reality rehabilitation and therapy," *Presence: Teleoperators & Virtual Environments*, vol. 14, no. 2, pp. 119–146, 2005.

[7] P. Standen and D. Brown, "Virtual reality and its role in removing the barriers that turn cognitive impairments into intellectual disability," *Virtual Reality*, vol. 10, pp. 241–252, 2006.

[8] E. Zimmerman and K. Salen, *Rules of play: Game Design Fundamentals*. MIT Press, 2003.

[9] C. Stanislavski, "An actor prepares," 1936.

[10] J. Moreno, "Psychodrama," 1947.

[11] B. Rothbaum, L. Hodges, R. Kooper, D. Opdyke, J. Williford, and M. North, "Effectiveness od computer-generated (virtual reality) grades exposure in the treatment of acrofobia," *American Journal of Psychiatry Behavior Therapy*, vol. 152, no. 4, pp. 626–628, 1995.

[12] G. Optale, A. Munari, A. Nasta, C. Pianon, J. Baldaro, and G. Viggiano, "Multimedia and virtual reality techniques in the treatment of male erectile disorders," *International Journal ofImpotence*, no. 9, pp. 197–203, 1997.

[13] G. Riva, M. Bachetta, G. Cesa, S. Conti, and E. Molinari, "The use of vr in the treatment of eating disorders," *Study Health Technology Information*, vol. 99, pp. 121–163, 2004.

[14] J. Wald and S. Taylor, "Preliminary research on the efficacy of virtual reality exposure therapy to treat driving phobia," *Cyberpsychology & Behavior*, vol. 6, no. 5, pp. 459–465, 2003.

[15] M. Slater, D. Pertaub, C. Barker, and D. Clark, "An experimental study on fear of public speaking using a virtual environment," *Cyberpsychology & Behavior*, vol. 19, no. 5, pp. 627–633, 2006.

[16] J. Piaget, "The origins of intelligence in children," 1952.

[17] S. Freitas, G. Rebolledo-Mendez, F. Liarokapis, G. Magoulas, and A. Poulovassilis, "Developing and evaluation methodology for immersive learning experiences ina avirtual world," in *Conference on Games and Virtual Worls for serious applications*, 2009, pp. 43–50.

[18] A. Gorini, A. Gaggioli, C. Vigna, and G. Riva, "A second life for ehealth: Prospects for the use of 3-d virtual worlds in clinical psychology," *Journal of Medical Internet Research*, vol. 10, no. 3, pp. 1–11, 2008.

[19] P. Rooney, K. Rourke, G. Burke, B. McNamee, and C. Igbrude, "Cross-disciplinary approaches for developing serious games in higher education," in *Conference on Games and Virtual Worls for serious applications*, 2009, pp. 161–165.

[20] M. Holden and E. Todorov, *Use of virtual Environments in motor Learning and Rehabilitation*. Lawrence Elbraum Associates, 2008 2on edition, ch. 49.

[21] S. Lam, D. Man, S. Tam, and P. Weiss, "Virtual reality training for stroke rehabilitation," *Neurorehabilitation*, vol. 21, pp. 245–253, 2006.

[22] *Serious games for upper limb rehabilitation*, 2009.

[23] W. Guo, S. Lim, G. Fok, and G. Chan, "Virtual reality for memory rehabilitation," *International Journal of Computer Applications in Technology*, vol. 21, no. 1–2, pp. 32–37, 2004.

[24] G. Riva, A. Gaggioli, D. Villani, A. Preziosa, F. Morhanti, L. Strambi, R. Corsi, G. Faltti, and L. Vezzadini, "Neurovr: An open source virtual reality platform for clinical psychology and behavioral neurosciences," in *Virtual Reality HCII'07*, ser. LNCS, R. Shumacker, Ed. Springer-Verlag, 2007, pp. 699–707.

[25] D. Tost, M. Ferré, P. Garcia, J. Tormos, A. Garcia, T. Roig, and S. Grau, "Previrnec: a cognitive telerehabilitation system based on virtual environments," in *Virtual Rehabilitation*, 2009, pp. 1–8.

[26] M. K. C. Wartmann, *Blender GameKit 2*, C. Wartmann, Ed. Blender Foundation, 2008.

[27] "Pygame documentation," http://www.pygame.org/docs/.

[28] L. Dovat, O. Lambercy, B. Salman, V. Johnson, R. Gassert, E. Burdet, T. Leong, and T. Milner, "Post-stroke training of a pick and place activity in a virtual environment," in *Virtual Rehabilitation*, August 2008, pp. 28–34.

[29] A. Gaggioli, A. Gorini, and G. Riva, "Prospects for the use of multiplayer online games in psychological rehabilitation," in *Virtual Rehabilitation*, September 2007, pp. 131–137.