Eurographics Symposium on Virtual Environments (2004)
S. Coquillart, M. Göbel (Editors)

MANDALA: A Reconfigurable VR Environment for Studying Spatial Navigation in Humans Using EEG

Pierre Boulanger, Daniel Torres and Walter Bischof

Department of Computing Science
University of Alberta
2-21 Athabasca Hall, Edmonton, Alberta, Canada, T6G 2E8

Abstract

This paper describes a reconfigurable VR environment and a markup language for creating experiments aimed at understanding human spatial navigation. It permits the creation of high-quality virtual environments and the recording of behavioral and brain activity measures while observers navigate these environments. The system is used in studies where the electroencephalographic activity is recorded while observers navigate virtual environments. The results of the study reported here confirmed previous finding that theta oscillations (electroencephalographic activity in the 4-8 Hz band) are linked to the difficulty of spatial navigation. Further, it showed that this activity is likely to occur at points where new rooms come into view, or after navigational mistakes have been realized and are being corrected. This indicates that theta oscillations in humans are related to the encoding and retrieval of spatial information.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism: Virtual Reality

1. Introduction

As we navigate through the environment, real or virtual, we rely on a multitude of cues for planning large-scale movements, for combining trajectories of previously travelled paths into a mental representations, and for determining heading. These cues include both static cues (e.g. landmarks, displacement information, depth information, etc.) as well as dynamic cues (e.g. optic flow patterns). Recent behavioral studies indicate that the difficulty of navigating through virtual environments is closely related to the number and types of visual cues that are available for guiding navigation. For example, Kirschen et al. [KIR00] found that landmark information and optic flow information significantly reduced the time participants take to navigate through a virtual maze. These studies suggest that rendering of perceptually rich VR worlds may make navigation much easier than perceptually impoverished renderings, and thus can help to reduce the cognitive load in navigation tasks. However, perceptual enrichment of VR worlds may not necessarily lead to easier navigation. In a perceptually impoverished VR world, we may rely much more on internally represented, cognitive maps of the environment, whereas in perceptually rich environments one can rely more on landmark-based guidance only. This, in turn, may prevent the building of an efficient and easily accessible mental representation of the environment. Many of the past EEG studies on navigation in VR worlds have imposed fairly severe constraints on the movements that could be made. In the studies of Kahana et al. [KAH99] and Nishiyama & Yamaguchi [NIS01], for example, participants were transported at fixed velocity through maze of hallways and viewpoint changes were instantaneous and limited to 90 degrees. This permitted equating movement patterns across different participants, but it also lead to highly artificial movement patterns that were not under control of the observers. The present study was aimed at overcoming these limitations by studying navigation with more natural movement patterns.

A *maze* is an excellent test-bed to address these questions. A virtual maze can be designed to explicitly test certain navigation abilities of the human user inside a controlled environment. Features like wall color and texture, topological structure and pattern, noise and navigation aids can be strate-

gically placed and varied to analyze the behavioral response. At the same time, different recordings like the user's navigation pattern, time to find the exit and the number and type of committed errors, (such as reaching a dead-end corridor on a previously walked section) can provide information about the processes that take place in our natural navigation system (See Figure 1).

For this purpose a reconfigurable Virtual Reality tool named MANDALA, and its markup authoring language, was created. This paper describes the architecture, features and overall characteristics of this system.

The paper is organized as follows: A general overview of the spatial navigation experiment is described in order to illustrate the requirements of the system. The architecture of MANDALA is then shown and a more detailed description of its elements and features is provided. Some illustrative examples are shown along with their code. The general dynamics of a typical navigation experiment from the point of view of the tool is also described. Finally, we describe the current state of the MANDALA project and some recent experimental results obtained using this system.

Figure 1: *An observer navigating the maze with an EEG cap to monitor brain activities.*

2. The Spatial Navigation Experiment

Given the requirements outlined in the previous section, the VR software environment must be able to perform the following tasks:

1. Allow the construction of virtual environments in a simple way, so that there is not need for complicated data structure or CAD modelling packages to the setup of the maze geometry.
2. Present the user with a simple, clean, and realistic interface that behaves in a believable way, in real-time (60Hz), using a variety of input devices like a keyboard or a joystick.
3. Offer a flexible authoring and scripting language so that mazes geometry can be made interactive, including the parameters that need to be measured during the experimentation.
4. Provide means for automating usage of external hardware devices like the electroencephalographic reader when certain events inside the virtual world are triggered.
5. Provide the ability to keep a log of events and track the navigation of the user for off-line analysis.
6. Allow real time communication with external agents so that other systems can react to the events during maze navigation, review the navigation of the user and evaluate his performance, design new mazes on-the-go and present them to the user without suspending execution of the simulation.
7. Work with conventional computer monitors as well as with specialized VR hardware display devices such as an auto-stereo displays or a CAVE like environment.

With these constraints, the available options are greatly reduced. Languages like VRML cannot provide all the required flexibility with the expected performance, and adapting some existing graphic engine would have implied additional time to understand and restructure (when possible) other architectures that are frequently unfinished and unsupported. Buying a commercial product would imply additional economic resources and would still have to be adapted to the requirements of the experiment.

3. MANDALA: A Virtual Reality Tool

The two main objectives of the MANDALA project are:

1. To create a flexible markup language that will:
   - Allow the definition of virtual worlds while encapsulating all complex 3D details so that experimentors without prior experience in computing science or Virtual Reality would be able to design and put their own experimental scenarios to work
   - Allow the definition and inclusion of libraries for simplifying the process of constructing the virtual environment.
   - Allow straightforward scripting for defining interactivity.
   - Allow communication routines so that attached devices can be controlled from within the maze without the need of implementing other modules.
   - Provide all required resources to build simple worlds, without restraining the possibility of building more complex one.
   - Although inspired by the requirements of this spatial navigation experiment, this toolkit is domain-independent and can be seamlessly adapted to different experiments.
2. To provide an architecture (and its implementation) that:
   - Is compatible with the MANDALA Markup Language (MML) definition and implements all of its features
   - Provides all the advantages and essential characteristics of a modern graphic engine

- Immediately works with conventional and advanced VR hardware like auto-stereo displays
- Allows concurrent operation and communication with other specialized agents

3.1. Overview of the MANDALA Architecture

The MANDALA general architecture is composed of five layers and two external managers. At the base, we have the *world definition* layer, which contains one or more simple text files written in MML specifying the objects, structures and interactivity of the virtual world. These files may also contain links to various multimedia resources like sounds, textures and other geometric models in different 3D formats. In the next layer an interface that reads, parses and validates the world definition file(s) and converts them into *MANDALA Objects*. In addition there is a series of specialized data structures that actually load external multimedia elements, organize the virtual world resources and keep and optimize the information for quick access from the *real time agents*. In this layer, a collection of agents that handle the realtime logistic of the world is found. Aspects like collision detection, script execution, avatar movement, navigation logging and other dynamic tasks are performed. This layer also administrates information circulating to and from two important external managers, one dedicated to administrate the *input and output devices*, and another that keeps several kinds of communication channels open with the *remote agents*.

Figure 2: *The components of the MANDALA architecture.*

The *external devices* manager reads input from the physical navigation controls utilized by the user to navigate through the maze. This abstraction makes it possible to adapt the application to work with simple or advanced input devices without having to change other systems. Output features include communication with hardware devices controlled by the realtime agents. The *remote agents* manager allow to abstract cooperation and communication with other research-specific agents that do not necessarily reside on the same computer. Numerous external agents can be interacting with the MANDALA environment at the same time, sending and receiving messages to supervise the experiment in many forms. This abstraction layer provide the architecture with a great flexibility for implementing domain-specific modules without having to modify the MANDALA architecture. Finally, the *rendering* layer maintains and updates a graphical representation of the state of the virtual world. It is in this layer where some particular graphic library must be used to render the information contained in the MANDALA objects as it is affected by the realtime agents, the external devices and the remote agents.

3.2. The MANDALA Markup Language (MML)

Let us analyze the structure of a MANDALA file. As mentioned before, all information pertinent to the creation of a virtual world resides in simple text files, just like in html. The structure of a MANDALA file is shown in Figure 3.

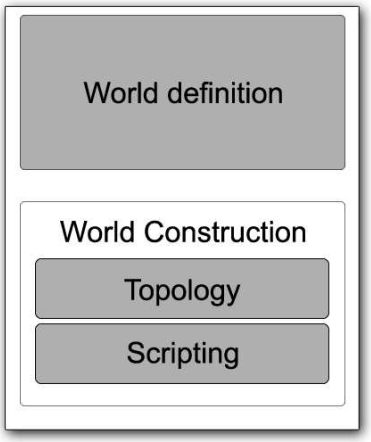

Figure 3: *Structure of a MANDALA file.*

The two main parts of any MANDALA file are:

1. *The Definition Section*: Here all the building blocks required to assemble our world will be defined. All materials, meshes, multimedia elements, included libraries, basic and predefined structures are declared.
2. *The Construction Section*: In this section, we take the building block defined in the Definition Section and use them to define the VR world. It contains two important elements:
   - *The world topology*: This structure define how pieces are to be arranged in order to construct the virtual world.
   - *Scripts*: Define a simple pieces of code that indicate actions to be taken when certain events happen somewhere in our virtual world.

Let's analyze in more detail each element of this format.

3.2.1. Definition Section

The basic construction element for a MANDALA virtual world is a plain *unitary cube*. Imagine an invisible cube in space, an abstract box that occupies an area and waits for things to be placed inside. One can place anything in these boxes and, as they are abstract elements, only what is put

inside will actually *exist*. In the MANDALA language this is known as a *cell*. Once a cell is defined many instances of itself can be put together to form a bigger space. As the cell itself was only defined once, the elements it contains are also declared once. In other words, the cell as an *object* is defined and *instances* are connected on the construction section to assemble the virtual world.

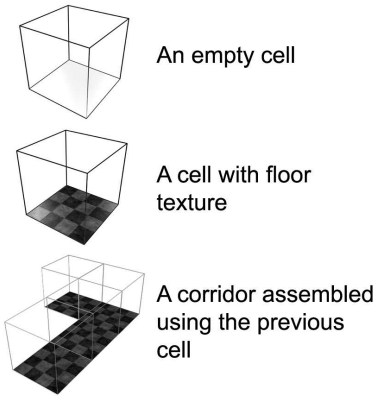

Figure 4: *Using one cell to build a simple corridor.*

Many things can be put inside a cell. Two basic elements are a floor and a ceiling. Additionally one can put walls, furniture and objects designed in some 3D modelling program, sprites and billboards. The object used for this purpose is called a *panel*. A panel is a link to an external image that will be used as a "wallpaper" for any surface in the world, be it a wall, a ceiling or a floor. Like the cells, once a panel is defined its instances can be used anywhere. In Figure 4, a simple cell is created by specifying a panel to use as the floor. Then, four instances of the same cell are concatenated to create a corridor.

The definition of a panel requires at least a unique *id* and the name of the external file. Additionally one can specify *uv* texture coordinates and RGBA values. As for the cell, an *id* is also required, and the floor and ceiling correspond to the *id's* of the desired panels. Note that the cell is actually *empty* as the floor and the ceiling are definition parameters, but do not contained objects. *Walls*, on the other hand, are to be contained because they can be put anywhere inside the cell. Let us analyze a cell with a single wall as shown in Figure 5.

Walls are defined by two 3D coordinates, the lower-left ($p1$) and the upper-right ($p2$) corners. The wall of left cell in Figure 5 would have the pair $p1(0,0,1)$ and $p2(1,1,1)$ (remember that we are working with unitary cells) while the wall on the right cell would be approximately $p1(0,0,0.75), p2(0.25,1,1)$. Following this method, vertical walls can be positioned *anywhere* in the cell.

The wall entry needs a panel to decorate it and both 3D coordinates. A cell can hold as many walls as necessary. There

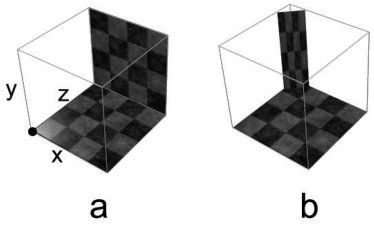

Figure 5: *Two different placements of a wall. Axis and origin shown on left cell.*

are more optional parameters like visibility set to default values. During the simulation walls cause automatic collision response, so if a mesh object is put in the cell, collision can be simplified by putting invisible walls around it.

3.2.2. Construction Section

Cells declared in the definition section will be used to actually *construct* a maze. We have already seen in Figure 4 how a single cell can allow the creation of a whole corridor. It is now necessary to explain how cells are put together. Again, a simple method was considered. In order to create a maze, we use the metaphor of the *watchman*. Imagine a watchman standing at the center of the first cell and deciding where to put the next one, he clearly has four options: *north, south, east* and *west*. Let assume that he places the next cell *west of the current cell* and he now walks to it. He is left with three options since the previous cell remains east. The next cell is placed and he moves again. The watchman can place cells in all available directions at each point and walk to the newly placed cells to put more until the world is finished. To illustrate this concept look at Figure 6.

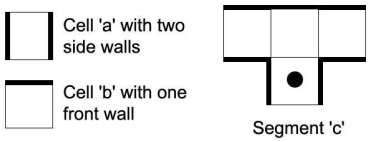

Figure 6: *Two cells building a simple 'T' segment for the maze.*

Assuming we defined cells 'a' and 'b', the MML code to construct the 'T' maze segment is straightforward. Beginning in the dot-marked cell:

```
<!- a simple T-like maze segment ->
    <root cell='a'>
      <north cell='b'>
      <east cell='a'></east>
      <west cell='a'></west>
      </north>
    </root>
```

It is important to note that the 'a' cells placed east and west are *automatically rotated* so that walls fall in its correct position. Notice also that the nesting capabilities of the markup language allows simplicity when designing the maze. This metaphor also relates to known online text games, where at a given point one might choose to look around and describe objects standing at the four cardinal points. Now imagine that it is desired to reference our 'T' object as if it was a *single entity*, in fact, in MML this is called a *sector* and can be declared with a unique *id* at the definition section. It will look like this:

```
<!- assuming cells 'a' and 'b' exist ->
<!- this is our simple 'T' sector    ->
<sector id='simpleT'>
    <draw cell='a' direction='root'>
      <draw cell='b' direction='north'>
        <draw cell='a' direction='east'>
           <endpoint tag='east'/>
        </draw>
        <draw cell='a' direction='west'>
             <endpoint tag='west'/>
        </draw>
      </draw>
    </draw>
</sector>
```

Notice the endpoint tag. It tells MANDALA where to insert new cells or sectors when indicated to place them with respect to our 'T'. Using just simple T sectors, it is possible to define bigger sectors and reach very high levels of complexity while maintaining simplicity of design. The complete MML file to generate the ***mandala-like*** maze shown in Figure 8 is listed in Figure 7. Sectors provide a way for defining complete areas in the virtual world, one can create a sector containing a house and then put several houses to form a street with ease. It is possible to concatenate any number and combination of sectors and cells to create an adequate virtual world.In the navigation experiment, special building sets are contained in files to be included in the maze definition file. Including them and putting them together is completely trivial using this framework.

3.2.3. Scripting

Loading a file like the one shown in Figure 7 will immediately put us inside the 3D maze and let us navigate through it, but since this is a domain for investigation and the maze must be interactive, a scripting system was developed. The chosen approach was, once again, very simple. This can be demonstrated with a couple of examples. Suppose we wanted to set some flag to TRUE if the user goes through a certain cell. Later we want to play a sound if the user successfully set the mentioned flag by the time he reached the exit cell. As every cell or sector put in the maze is an instance of the original, it is necessary to identify some special places where something is to happen. This is done by assigning a *label* when placing them in the maze.

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE maze SYSTEM"mandala.dtd">

<mandala> <!- the definition section ->
  <definition>
    <!include some files with all the cell>
    <!declarations>
    <include file='myCells.xml'/>
  </definition>

<!- the construction section ->
  <construction>
    <root cell='a'>
    <putsector sector='lotsOfT'
    direction='north'>
    </putsector>
    <putsector sector='lotsOfT'
    direction='south'>
    </putsector>
    </root>
    </construction>
</mandala>
```

Figure 7: *The code to produce the maze illustrated in Figure 8.*

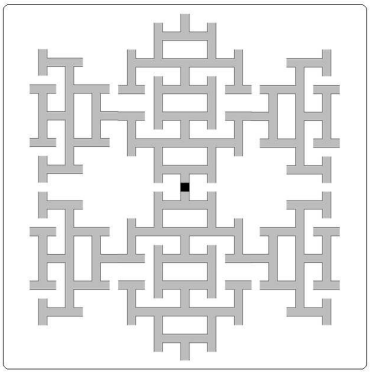

Figure 8: *A more complex maze using repeating patterns.*

Normally the *as* tag would not be needed. Now it is put to exactly reference these two places. We write two instructions after the topology creation of the maze:

```
<action id='action1' at='checkpoint'
event='avatar_enter'>
<execute function='setVariable'
params='flag,1'/>
</action>

<!- check the flag and play some sound ->

<action id='action2' at='exit'
```

```
event='avatar_enter'>
    <condition>
       <require param='flag' value='1'/>
    </condition>
<execute function='soundStart'
params='finale'/>
</action>
```

The first action executes at the cell with label *checkpoint* when the avatar leaves that place, creating a variable called *flag* with value equal to 1. The second action triggers when the user enters the 'exit' cell. If there exists the flag value and its value equals TRUE, then a sound with *id*='finale' will start playing (sounds are created in a similar way than panels, by specifying a file and some unique *id* in the definition section). The *condition* tag tests for several premises required for executing the list of actions. More complicated structures involving AND's and OR's can be also declared. Finally, *avatar_enter* and *avatar_leave* are events that tell when the actions are to be triggered.

3.3. The MANDALA Objects

When the MANDALA application starts, some the virtual world file is read and transformed into *MANDALA objects*, a group of data structures that hold the necessary information for conducting the simulation. This parsing, implemented with help from Xerces [APCH], reads all the definitions and instructions in the MML file and loads into memory all external dependencies like graphics and 3D models. It also generates geometry for supporting the environment and its cells. Many resources like textures and meshes are optimized by keeping just one instance in memory and using it whenever required. At the end of the process the realtime agents find in this layer all the required information to work. The transition from the MML file to the MANDALA objects is shown in Figure 9. Each type of data object is stored in a specialized structure that facilitates the work of agents in the upper layers. Collision geometry, for example, is kept on a dynamic plane shifting using BSP threes[MEL00].

3.4. RealTime Agents: Bringing the Simulation to Life

The fourth layer of the architecture holds a group of dedicated agents that work with the data maintained in the third layer for various dynamic purposes. This control center is where the real mechanics of the engine take place and where the environment is set in motion. Several agents sometimes work with common information but each one is completely independent of each other. Some examples of these agents are:

- *Avatar Agent*: moves the user across the world, reads input from the external control devices (like a mouse or a keyboard) and updates the position of the avatar. It also control factors like pace speed and camera settings.

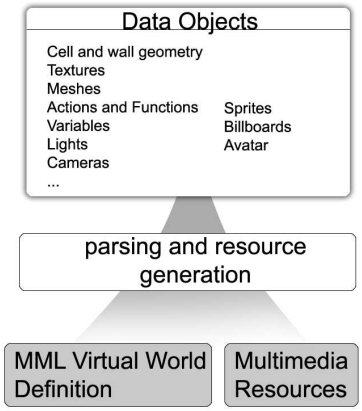

Figure 9: *Process of data generation for the RealTime engine.*

- *Collision Detection Agent*: reviews the position of moving objects at each time frame and calculates collisions. Sends update messages to agents controlling moving objects to rectify position when a collision is detected (avoiding objects to go through walls, for example).
- *Navigation Agent*: this entity is like an invisible character observing everything that happens in the simulation. It is mainly designed to take notes and produce certain reports with data collected during each experiment.
- *Actions Agent*: constantly monitors events in the maze. Should a documented event trigger some action (specified in the MML file), this agent produces a message, reviews the action and takes pertinent measures.
- *Message Agent*: performs callbacks for certain messages and distributes information across other modules.
- *Remote Communications Agent*: establishes and maintains connections with remote agents so that interaction between MANDALA and other domain-specific agents can take place.
- *Rendering Agent*: provides an interface with the fifth layer of the architecture, orchestrating and optimizing the rendering process. This agent is independent from the chosen graphic environment and library.

This architecture allows the complete replacement or addition of agents without altering other elements. Consequentially, it allows scalability and enhancement of functionality.

4. Experimental Setup

This section describes a typical spatial navigation test using the MANDALA system and the EEG recorder. The general structure of the experiment is shown in Figure 10.

The observer sits in front of a stereo display with an electroencephalographic (EEG) cap as illustrated in Figure 1. When the experiment begins, all operations are automated.

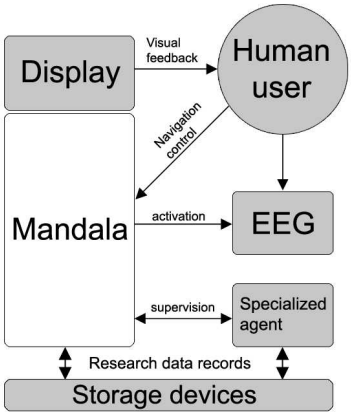

Figure 10: *Structure of a typical setup for the spatial navigation experiment.*

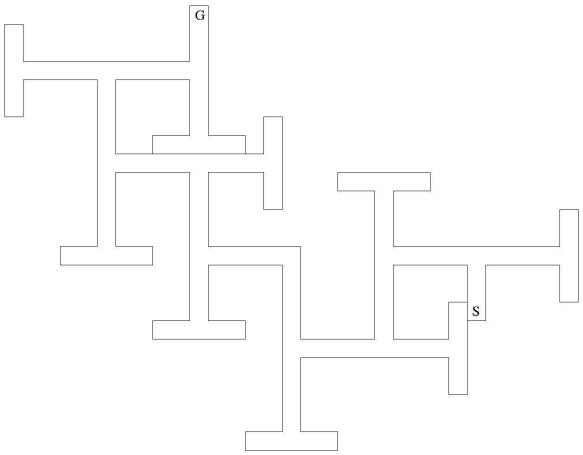

Figure 11: *Layout of the maze used in the experiments.*

An agent specialized for this experiment opens a communication channel with MANDALA and orders the selection of an initial maze for the user. From now on this specialized external agent will be simply referred as the *agent* and unless otherwise noted all operations are performed by MANDALA. The required maze is loaded and the observer can begin the navigation. When the observer reaches the exit a notification is sent to the agent, which analyze the navigation log. Based on certain rules and the user's performance a new maze can be selected from a collection or created in *realtime*, the old maze is terminated and the new one is loaded and presented to the user, who continues navigation. During navigation, the EEG is continuously recorded. These EEG measurements are then saved with their respective position, type of actions, and time stamp for further analysis.

This experimental cycle (the user navigates the maze, MANDALA keeps a record, communicates with the agent and controls the eeg machine, the agent evaluates the user and creates/selects a new maze) repeats until the last maze is unloaded and MANDALA is instructed to close.

All mazes consisted of a set of T-junctions arranged in such a way that the goal position could be reached from the starting position in a sequence of 10 left or right turns. The layout of one maze is shown in Figure 11, with cell S (on the right hand side) indicating the starting position and cell G (at the top) indicating the goal position. In this case, a sequence of left/right turns LLRRLRLRRL leads from the starting position to the goal position. A set of 10 different mazes was created, and random subsets of these mazes were presented in the experiments.

Several maze views are shown in Figure 12. Depending on the experimental condition, the maze walls could be either plain (Figure 12a) or colored (Figure 12b), and could either have an arrow indicating the correct direction (Figure 12c) or not (Figure 12b). Finally, a message on the wall at the end of a hallway indicated the goal position (Figure 12d). Upon entering a new maze, a participant started in the starting cell (cell S in Figure 11) facing the first intersection. As soon as the goal cell (cell G in Figure 11) was entered, the maze was exited and a new maze was entered.

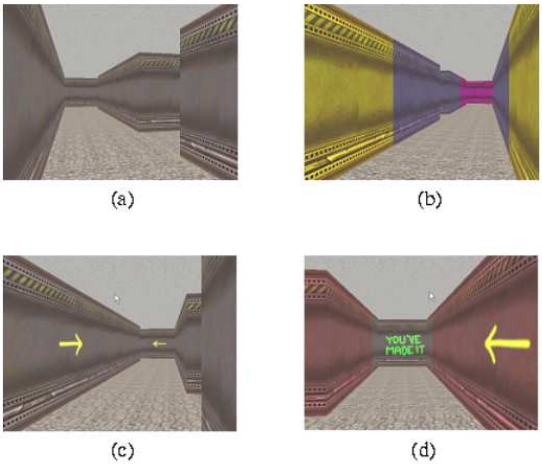

Figure 12: *Four different maze views: (a) view of a hallway in a 'plain' maze, (b) view of a hallway in a 'color' maze, (c) view of a hallway in a 'plain' training maze, with arrows indicating the correct direction of navigation, and (d) view of the goal position in a 'color' maze.*

Two different types of mazes were used, plain mazes and colored mazes. In plain mazes, all walls had the same color (see Figure 12a), whereas in the color mazes, the color of each T-junction was chosen randomly from the set blue, cyan, green, red, magenta, yellow. All walls of a single T-

maze had the same color (see Figure 12b). The texture of all walls, as well as ceilings and floors did not change.

All mazes were generated in two versions, as training mazes and as test mazes. In training mazes, a large yellow arrow indicated the correct direction at each intersection (see Figure 12c), whereas the arrows were absent in the test mazes.

Fifteen observers (12 male and 3 female) participated in the study. EEG signals were collected from 38 gold electrodes embedded in an electrode cap and amplified using a Neuroscan NuAmp amplifier. Recording locations were based on the electrode placement system of the American Electroencephalographic Society. Horizontal eye movements were monitored with bipolar electrodes on the outer canthus of each eye, and vertical movements were monitored from electrodes placed above and below the left eye. EEG and EOG were recorded at a sampling rate of 500 Hz for the duration of the whole experiment.

Each participant traversed five different mazes, once with arrows in a training trial, followed by (at most eight) test trials, so each participant traversed at least 10 and most 45 mazes. These conditions were presented in a between-subject design, i.e. each participant traversed either plain or colored mazes.

An EEG was recorded continuously for a complete experimental session. These data were segmented into episodes corresponding to the traversal of a single maze, beginning at the time the starting position (e.g. cell S in Figure 11) was left, and ending at the time the goal position (e.g. cell G in Figure 11) was entered. Each of these maze episodes was analyzed independently.

For a given EEG channel, short-term spectrograms were computed for each maze episode. To this end, discrete Fourier transforms were computed for Hanning weighted windows of 1000 ms duration, with successive windows overlapping by 900 ms. The spectrograms for one maze traversal is shown in Figure 13a. Brightness in each spectrogram is inversely related to the log power of the short-term Fourier spectrum, normalized over a whole episode, i.e. the darker a spectrogram at a given time point and frequency, the higher is the power of the corresponding Fourier component. Episode with strong activity in the theta (4-8 Hz) band should thus show as dark bands.

The analysis of the EEG spectra related theta-episodes during a maze traversal to locations in a maze. An example of this analysis is shown in Figures 13. In this spectrogram, 10 theta episodes are identifiable if one ignores the very last episode that may be related to exiting the maze. Figure 13b shows the corresponding average power of the theta-band (4 - 8 Hz) over time. The time points of the theta episodes allow determining where in the maze the participant was at the time. As in Figure 11, cell S is the starting position, and cell G is the goal position, hence the goal position could be reached from the starting position in a sequence of turns LLRRLRLRRL. In Figure 14, the arrows indicate the positions in the maze where a theta-episode occurred. The first five episodes are indicated by single arrows immediately after the participant had turned into one arm of a T-junction and saw a new hallway segment. The last theta-episode occurred, shortly before the last T-junction of the maze. The other four theta-episodes occurred when the participant had made an incorrect navigational decision and walked into an invisible barrier. In each pair, the first theta episode occurred at the barrier, and the second episode occurred in the neighboring cell on the way out from the incorrect T-junction arm, and with the correct T-junction arm in view.

The example in Figures 13 indicates that theta episodes are more likely to occur when a new hallway gets into view or, after a participant has realized that a navigational error has been made (after walking into a barrier). If this is so, then the average power of theta episodes should be higher immediately after a turn in a T-junction than immediately before the junction. In Figures 13c, a T-junction is shown together with a definition of cell types (cells 1/2 immediately before a junction, cell 3 the junction, cells 4/5 immediately after a junction in the correct direction, and cells 7/8 immediately after a junction in the incorrect direction). Figure 13d shows the average power of the theta-band for each of these cell types, for the spectrogram in Figure 13a. It can be seen that the power in the theta-band is higher in cells 4-6, immediately after a turn in a junction.

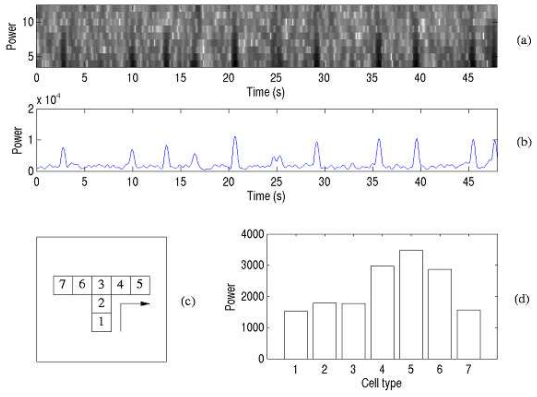

Figure 13: *The top panel shows a magnified view of the spectrogram of a single maze traversal, namely the last one in Figure 12. The middle panel shows the power of the theta band (integrated over the range 4 - 8 Hz) of the spectrogram in the top panel. The bottom panel shows a T-junction on the left, with the direction of correct navigation indicated by the arrow. The histogram on the right shows the average power of the theta components for each of the cell types.*

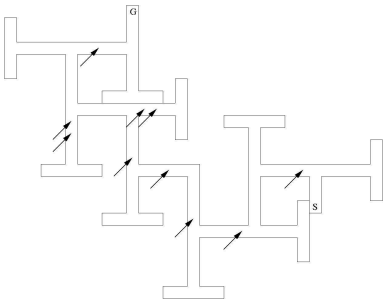

Figure 14: *Layout of the maze that was traversed by the observer. The arrows indicate where in the maze each of the 10 theta episodes in Figure 13 occurred.*

5. Conclusion

The behavioral results show that participants who traversed the colored mazes, spent more time on traversing the mazes, and, at least in tendency, made more navigation errors and required more attempts before managing to traverse a maze without error, than did the participants who traversed the plain mazes. This result stands in contrast to the fact that the participants in the 'color' condition had actually more information available for guiding their navigation than did the participants in the 'plain' condition. This result is consistent with navigation strategies reported by the participants. In the 'plain' condition, all hallways looked the same, and the only way to remember a path was to rely on learning a sequence of left-right turns. Once participants realized and used this strategy, the number of navigation errors dropped sharply. In contrast, only few of the participants in the 'color' condition reported adopting this strategy. Instead, they relied on navigation rules that took color cues into account and appeared to encode a number of local navigational decision rules (e.g. to turn left at the second magenta intersection, or to turn right after to green intersections in a row). As the behavioral results show, this strategy turned out to take more time and to be more prone to errors. The EEG results were consistent with the behavioral results: The power of theta episodes tended to be higher in the in the 'color' condition than in the 'plain' condition.

The most interesting results were obtained in the detailed analysis of the maze points where theta episodes occurred. It was found that these episodes did not occur either uniformly or at random points in the mazes. Rather, theta episodes occurred more likely and the power of these episodes was higher immediately after participants had made a turn in a junction and a new hallway came into view. Second, they were also more likely and more powerful after participants had walked into an invisible barrier and had to revise their navigational rules. In contrast, theta episodes were less likely when the participants were simply moving down a hallway. In the first case, participants were probably retrieving a stored view that could guide their further navigation. In the second case, participants were most likely storing the view with a revised navigation rule. These results provide further evidence on the relation between theta waves and spatial navigation in humans: Theta waves may be directly related to the storage and retrieval of spatial information for navigation, not just in rodents, but also in humans.

These results demonstrate that VR in combination with EEG measurements and some behavioral analysis can be used as a powerful tool for understanding human brain activities as it interact with these world. MANDALA has shown to be a powerful tool to configure rapidly experiments and to perform synchronized recording of EGG with interaction in the virtual world. This is an essential tool to study the type of signals the brain produce in synchronization with various tasks associated to various actions in the VR world. A good understanding of these signals is key for their use in an eventual feedback loop where on day EEG signals can be used to control navigation.

References

[APCH] THE APACHE XML PROJECT xml.apache.org

[MEL00] MELAX S. Dynamic Plane Shifting BSP Traversal Graphics Interfase proceedings, 2000.

[BIS02] BISCHOF, W.F. AND BOULANGER P. *Spatial Navigation in Virtual Reality Worlds: An EEG Analysis* Cyberpsychology and Behavior 2003.

[BOU02] BOURG M. P. *Physics for Game Developers* O'Reilly, 2002.

[KIR00] KIRSCHEN M.P., KAHANA M.J., SEKULER R., AND BURACK B. *Optic flow helps human learn to navigate through synthetic environments* Perception, Vol 29, pp. 801-818, 2000.

[KAH99] KAHANA M.J., SEKULER R., AND CAPLAN J.B. *Human theta oscillations exhibit task dependence during virtual maze navigation* Nature , Vol 399, pp. 781-784, 1999.

[NIS01] NISHIYAMA N AND YAMAGUCHI Y. *Human EEG theta in the spatial recognition task*. Proceedings of the 5th World Multi-Conference on Systemics Cybernetics and Informatics, Orlando, Florida, July 22-25, 2001.

[HOW95] HOWARD, I.P. AND ROGERS, B.J. *Binocular vision and stereopsis* New York: Oxford University Press, 1995.

[EVE99] EBERLY D.H. *3D Game Engine Design* Morgan Kaufman, 1999.