# **VIRTUAL WATER MAZE**

## Introduction

This software simulates the Morris Water Maze task of spatial memory synchronized with magnetic transcranial stimulation to evaluate memory and induce changes in cortical plasticity that would translate into an improvement in the task performance.

In a virtual space, we developed an interface in which the user can navigate through a round pool where a platform is hidden and four symbols are presented on its wall, which will be the reference for orientation. The goal of this task was to find a hidden platform. During the different tests, the symbols and platform remained in the same position; however, the subject appeared in a random place inside the pool. The user can move around the space using arrow buttons on the keyboard and orient the camera using the mouse.

Synchronization with transcranial magnetic stimulation (TMS) methodologies has been shown to neuromodulate the excitability of the motor system and promote the improvement of motor function [1]. TMS is a technique capable of provoking electrical currents in the cerebral cortex in a noninvasive manner by generating short magnetic fields of high intensity, which depolarize a certain population of nerve cells from small cortical regions. A TMS configuration strategy to induce neuronal plasticity is paired associative stimulation (PAS), which is a combination of a single TMS pulse and another stimulus or event that induces activation in the same nervous tract.

Recently, techniques applied to the motor cortex have been shown to improve spatial memory capabilities [2, 3]. Therefore, in the developed program, a single TMS pulse was synchronized with the appearance of the signals necessary for space orientation. As the subject searched for the platform oriented by these symbols, we programmed the application of a simple TMS pulse on the user's motor cortex every time the camera directed towards a signal and entered its field of vision. Therefore, we synchronized motor cortex activation with the endogenous cortical activation of the spatial memory process.

## Technical Description/Support of the Solution

The application was developed on the Unity video game engine v.2019.4.11f. for the StandAlone PC platforms Windows 8 and 10. The solution integrates the “inpout.dll” dependency to communicate with a parallel port. Using this library, we can communicate triggers/pulses from the computer to the parallel port, which is a digital output device. The developed application manages the opening and sending of signals to the parallel port pins. Following the design criteria of the game, pulse sending was performed when viewing a reference in the pool or when reaching an intersection in the maze. The detection of such references in the camera's field of vision is implemented using the raycast method using the Unity 3D Physics engine. The detection of entry in intersections uses collision components, using the Unity 3D Physics engine.

## Visual interface

To avoid possible distractions for the user, the walls maintained a homogeneous color, and the pool water was based on a repeated loop (to avoid possible orientation references in the water). Symbols aimed at being the orientation reference were randomized when initializing the program, and a shape and color were assigned to an already established list of possibilities. This choice was made throughout the experiment until the application was closed and reopened.

## Game design

At the start of the game, the number of attempts and the maximum duration (seconds) of each attempt are set. With these parameters configured, the positions of each reference and platform are defined, and they remain fixed throughout the game. At each attempt, the camera that controls the user is placed at a randomly selected point in the pool, forcing the user to navigate in every attempt from the references. The game view has a Heads-Up Display (HUD) that displays information about the time spent on this attempt and, in addition, by text, indicates to the user whether he has managed to find the platform or if he is in the last 5 s of the current attempt.

The recorded data of the video game are: the coordinates of the camera position in the pool (50 fps), the time to complete each trial (seconds) and the time in which each signal has been viewed since the beginning of the trial (seconds).

The game ends in two ways:

1) Having completed all attempts indicated at the beginning of the session.

2) Reaching the platform position in two consecutive attempts.

# **VIRTUAL CROSS-ROAD MAZE**

## Introduction

This software simulates a maze with 18 crosses and three path choices in each division. The task consists of an initial training phase (a video) in which the subject is shown the path to follow and the TMS pulse is paired with the time of choice of the path (key elements that the subject must remember). In the labyrinth execution phase, the users must independently solve the maze on their own. The objective of this program was to evaluate both spatial memory and the induction of changes in cortical plasticity that translate into an improvement in the resolution of the task itself.

The goal of the user is to make the right choice at each crossroad of the path in the least possible number of attempts.

## Technical Description/Support of the Solution

The application is developed on the Unity video game engine v.2019.4.11f. for StandAlone PC platform Windows 8 and 10. The solution integrates the “inpout.dll” dependency to communicate with the parallel port. With this library we can communicate triggers/pulses from the computer to the parallel port that is as a digital output device. The developed application manages the opening and sending of signal to the parallel port pins. Following the design criteria of the game, pulse transmission was executed when reaching an intersection in the maze. Collision components are used to detect entry into intersections using the Unity 3D physics engine.

## Visual Partition

The overall lighting was homogeneous throughout the experiment to avoid orientation through shadows or color references. Similarly, the walls and floor maintained a texture that was repeated throughout the course to avoid unwanted references. The walls are presented with brick texture and the floor with tile texture to allow the indication of the road and the center of the streets.

## Game Design

At the beginning of the game, the user is located at one end of the maze. Each time it reaches the intersection of three paths (left, front, and right), a pulse is thrown. Once the user has taken a path, if it is incorrect, an error pop-up is displayed after 3 s of progress and returned to the user at the beginning of the maze.

Each time a path intersection is entered, such as when selecting a path, both the times (seconds) and the path selected are recorded by following the code: [Insertion Number]\_[Direction] (for example, Choosing the path to the right at the first cross-section would save the data: 1D). The game ends when the user reached the labyrinth exit.