Notes on the Space Mapping Games Initiative

Covering Period of May-August 2015

Written By
Daniel Tajik tajikd@mcmaster.ca

August 31, 2015

Collaborators:
Dr. John Bandler bandler@mcmaster.ca
Dr. Michael Noseworthy nosewor@mcmaster.ca
Denys Shumakov shumakds@mcmaster.ca

Original Game Concept and Software Developed By:
Drs. Q.S. Cheng and John Bandler

Abstract

This project is based on an original game concept and early software by Q.S. Cheng and J.W. Bandler to test whether people intuitively harness mathematical strategies similar to those associated with space mapping optimization algorithms that have found utility in engineering optimization. Initially the games were simplistic in nature, and needed to be developed to enhance user-friendliness and allow for data collection. As the games evolved, unique pursuits into the realm of fMRI and EEG were discovered, and the games were modified with the intention of using said imaging techniques. This document covers the main games developed, the programming methods relevant to their overall function, the challenges that arose in their enhancement and evolution, and future work that should be considered if pursuing these initiatives further.

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN7239-11 and by Bandler Corporation.

D. Tajik is with the Simulation Optimization Systems Research Laboratory and Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, L8S 4K1 (e-mail: tajikd@mcmaster.ca).

J.W. Bandler is with the Simulation Optimization Systems Research Laboratory and Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, L8S 4K1, and with Bandler Corporation, Dundas, Ontario, Canada L9H 5E7 (e-mail: bandler@mcmaster.ca).

M.D. Noseworthy is with the St Joseph's Healthcare Imaging Research Centre and Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, L8S 4K1 (e-mail: nosewor@mcmaster.ca).

D.S. Shumakov is with the Computational Electromagnetics Research Laboratory and Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, L8S 4K1 (e-mail: shumakds@mcmaster.ca).

1 References

For background information and further references to the art and science of space mapping see:

[1] J.W. Bandler, Q.S. Cheng, S.A. Dakroury, A.S. Mohamed, M.H. Bakr, K. Madsen, and J. Søndergaard, "Space mapping: the state of the art," *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 1, pp. 337-361, Jan. 2004.

[2] J.W. Bandler, "Have you ever wondered about the engineer's mysterious 'feel' for a problem?" Feature Article, *IEEE Canadian Review*, no. 70, pp. 50-60, Summer 2013.

The so-called cheese-cutting illustration of aggressive space mapping is presented in [1] and the shoe-selection illustration is presented in [2] along with cognitive analogies of space mapping that inspire this work.

2 Introduction

When I joined Dr. Bandler in May I honestly had no idea where we were going to go with these space mapping games. I watched the initial iterations of the parachute game (original concept and software developed by Q.S. Cheng and J.W. Bandler, accessible here http://www.sos.mcmaster.ca/SMGame_parachute1D.html) and had no clue to exactly what was happening when the game was being played. Space mapping seemed like an incredibly simple idea, yet I still have troubles wrapping my head around the more challenging scenarios of its use (in fact I still do). The parachute game (in my mind) captures the simplicity of the strategy. Drop a parachute and it will drift, and your adaptation to and adjustment of that drift is the essence of space mapping. I have captured this in a PowerPoint demonstration of a game located in the para_game folder (for access to this folder, talk to Dr. Bandler who should have it saved to the dropbox folder).

The ability to take multiple research paths with space mapping games became immediately apparent as the work on the game progressed. Eventually a discussion with Dr. Noseworthy led to several fMRI scans which revealed some unique and interesting characteristics. Beyond that, EEG has also been considered as a study pathway, and even the fMRI-EEG combination could be tested to pursue a wide variety of research. Near the end of the project term, discussions were also had on the concept of testing space mapping in animals (though this may be extremely far-sighted… perhaps we should work on humans first). Still, the ability to chase all these opportunities is very attractive, and may create exciting possibilities for novel research in mathematical understandings of brain function, mental illnesses, and intellectual performance.

Throughout the following document I will do my best to cover the work that was done over the course of my four month summer here at McMaster. I will go over the major work that was completed on the individual games, including overviews of major components/methods of the games, issues that arose, and further work that could be pursued in order to improve the games. I have not written this document very formally, as I feel that to cover the issues at a high level of detail would not be very helpful for anyone trying to simply understand the work that was done. If you are curious about specific functionality of the games, please look into the commenting of the specific functions, which should delve a little more technically into their behaviour. I have commented most of the functions I have written, so anything left out was either something I did not understand myself, or something I believed was simple enough to understand after debugging the method (or I just forgot…sorry!). If you are having problems with specific things, ctrl+F is most definitely your friend. Thanks for reading!

3 Initial Setup

In order to run these games, a localhost server must be established to enable their operation. To set this up you can use the XAMPP server (https://www.apachefriends.org/index.html). By placing the para_game folder into the XAMPP htcdocs folder and you can access them through localhost/para_game/para_game.html.

Any text editor can be used to develop the code (though what I used was Textwrangler). In order to debug these online documents, Google chromes developer tools were used, which allowed for access to the source code, cookies and local storage, as well as other useful functions. HOWEVER note that PHP code cannot be seen (since it is technically server sided). In order to debug PHP, you need to read the code displayed in the bottom of the game (it is sent via the echo function). If there is an error, the specific error and the line where it happens will appear in the php print out line (which is labeled "txtHint").

To edit the game on the web server (SOS) you will need to have an FTP client, which essentially allows files to be passed to and from the web server. A windows client that is commonly used is WinSCP, which has incredible versatility and simplifies the procedure quite significantly. Since I use OSX, I simply used the transfer capabilities of TextWrangler, though it is not nearly as simple to transfer and update files. Having a virtual windows

4 Parachute Game (Online):

4.1 Overview

Secure admin protection php (para_game.html: ***function checkUser***) In order to maintain access security over the administrative portions of the game, a php file was created (named "pss.php") which would check and see if the name included the phrase "admin123" and would then enable the buttons and controls for the game to the user. In this state, click information would still be returned to the database so that if it was to be analyzed in the future it could be.

Secure database protection php (para_game.html: ***function this.update_db***) The database is created on the server side in a protected file that cannot be accessed through a browser. It must be pulled out of the database through a FTP client (such as WinSCP, Textwrangler, etc).

The database uses a **".csv"** file to store all of the users clicks. The initial plan was to use an SQL Database to store the information, but that seemed to create too many complexities to the simple task we wanted to achieve. The database construction involves the use of PHP (server-sided) code which takes the input parameters of the user clicks and generates a plot of data describing everything from the position of the user clicks to the underlying equation. This data can be found in the "userdat.csv"

The Scheduler (para_game.html: ***function this.calcSched)*** The schedule sets the equations for each round, and is always the same when the cookie is set to the "sched" setting (which happens at the beginning of the games initialization. The original game design is still implemented (via the "distChall" cookie setting) and can be found later on in the code.

Algorithm Function (para_game.html: ***functions this.executeAi, this.animation_complete, function SM_Engine)*** The algorithm is a space mapping algorithm derived from a 2D targeting game which can be found on the sos website (http://www.sos.mcmaster.ca/SMGame_V0.08x.html). It was then stripped down to 1D, though the RSM algorithm remains in the code, it doesn't work in a 1D setting. The algorithmic system can be initiated by typing either "SM" or "ASM" into the name line of the initial pop-up window. It operates based off of a created event being triggered when the parachute makes contact

with the ground. When the parachute contacts the ground, the algorithm computes the new output, and clicks again after a 2 second delay.

Printing the algorithm output (para_game.html: ***function this.storeData, phpfile SM_collect.php/SM_collect.csv)*** Uses the database storage function in a similar manner to the other database storage functions. Outputs a file that displays all the equations and positions that are calculated so that they can be analyzed later.

Movement (para_game.html: ***function moveTimer, keyMovement***) Movement of the game parachute is controlled with the "A" (left) and "D" (right) keys, with the "S" key dropping the parachute. As the key is held down the parachute will move faster until it reaches the side of the window, where it then stops and can only move in the opposite direction. When the user clicks down a key, a timer begins and continually moves the parachute until it is released or reaches a barrier.

Cookie Storage (para_game.html: **function nextround, reset_info, userinfo_popup, getCookie**) The game relies on cookie storage to pass information between each round. The cookie expiry date is set at the top of the page (IMPORTANT: The date needs to be updated in the future in order to keep the game running smoothly, if the game no longer works, seems broken, or is crashing, check the date under the variable cookieExpiry).

Collecting Algorithm (ASM, SM) Data (***function this.storeData***) Current phased out is a function that will print out the calculations made by the space mapping algorithms. By uncommenting that function, you should be able to see the creation of a file (in the data folder) labeled "algorithm.csv". The files creation works very similarly to the database creation, utilizing the PHP file "SM_collect.php" in order to create the data file. If it is not working, remember to make sure that the SM_collect file is in the same folder, and that the access to the algorithm file is allowed. Sometimes the permissions for the folder will be restricted, throwing an error that will make it impossible to create the file.

4.2 Challenges and Issues

Canvas/Screen size or resizing: One part of the game that was developed was the canvas or screen fitting option. This was actually created by Denys Shumakov (who worked on previous iterations of the wedge game). While it was initially implemented in early versions of the game, it immediately became apparent that this would not be a viable option for a behavioural game, due to the desire to have everyone play on the same size screen. The screen size was then locked to 1000 pixels.

Browser optimization: The game was developed in Google chrome, and as such runs most smoothly in the browser. However the game did not run well in Firefox, so it had to have several things added to it in order to ensure that it would work properly. For instance, one of these things is the meta style declared at the very top of the para_game.html. Also, "autocomplete" being disabled is another necessary declaration for Firefox that is not necessary for Google chrome.

File Permissions: A big issue that occurred early on was the permissions that a file contained in order to be written to. For instance, if a file is set to read-only, no one can access that file to write to it. This issue becomes very important in the creation of the database file, as it must contain the correct permissions or else it will not be written to. The SOS server does not allow writing permissions to go across folders for instance, so when I tried to set up the server to write the userdat.csv file in the "data" folder, it would fail every time. This was corrected by simply writing to the same folder.

4.3 Ethics

The game should also be updated to handle the ethics approval board… this will require looking at what their requirements are for data storage. One of those things is how the names are stored in the database… they should be scrambled and stored in a separate file.

It will probably become necessary to have users fill out a declaration of voluntary information when the game is initialized. I believe the psychology department has some standardized versions of this documentation that we could use for our purposes. Contacts for more information on the ethics approval can be found below.

Quote from an email from Dr. Michael Noseworthy:
"In terms of human ethics questions- Dr. Michael Wilson, the university's bioethicist, is the guy to ask <mwilson@McMaster.CA>. Every month they offer a drop in session you can go to for questions. Or just email/call him directly to set up an appointment (x23142). There's also there website: http://reo.mcmaster.ca/. I have dealt with ethics boards in about 10 hospitals over the years. I am pleased to say that McMaster/HIREB (i.e. Hamilton Integrated REB) are the best I've ever dealt with. They are really on side with the researchers. They only give constructive and useful comments, definitely not like some of the other places I've had to deal with."

4.4 Data Processing

In order to look at what exactly was being stored in the database for the website game, I created some really poor automation algorithms to try and sift through the data. Unfortunately these algorithms require both Excel and Matlab to run… so they are not very efficient

The first part of the processing is the excel page, which I use to quickly clean up parts of the file I don't need. I try and remove any names that I use for testing (like "test", "danieltest", etc)

I also remove ASM/RSM beyond the first data entries. We only need one subset of them to do the data analysis, so every time I refresh the database I reset it so that the first two user entries are ASM and RSM

Once I've cleaned them up using the algorithm, ill go through them and make sure there aren't any weird anomalies or irrelevant entries (I usually remove Dr. Bandler's entries because they are generally biased)

After this, I process the data into a new file which I feed into the matlab script. The matlab script then runs through all the info and plots out the scatterplots.

I'd like to shove it all into matlab, but I don't like not being able to easily remove anomalies in the data. Can't really do it in excel because I dislike the graphs excel generates.

Unfortunately, the excel function that I use to clean the data was corrupted, so it has to be rebuilt in order to accurately deal with the database.

4.5 Future Work

Removing the No-Go Zone: The zones in the game beyond round 9 should be removed from the game due to its difficulty to emulate with algorithmic strategies. Removing it should be simple, they just need to be disabled in the CalcSched function.

Reducing Round Number: Another suggestion is to optimize the number of rounds from 20 to 10(?). This would reduce the amount of data collected, and may make the game more challenging. It would be interesting to see how the learning changes between people who play the game 20 times in a row versus

10 times. However right now people get bored of the 20 rounds, it seems as if they have overcome the challenge of the game and are now simply completing it for the sake of completing it.

One Instance of the game: One major issue of this game is its dependence on refreshing the page, and storing information from one round to another. In an ideal setup, this game would run in one instance and each round would happen naturally in the page instead of requiring a refresh. To make this change would take a far amount of effort, as it is very dependent on the cookie system established. To do this, the cookie system would need to be instead changed to a overarching variable system, and a new Parachute function would be generated with each new round.

5 Parachute Game (fMRI compatible):

5.1 Overview:

Two videos were created for each of the fMRI games, which can be found here https://www.youtube.com/watch?v=PXib7NnSRks and here https://www.youtube.com/watch?v=jeoXEjj15x4

These games are… messy. Both were hastily designed so that they could quickly be implemented in the fMRI for pre-funding tests to see if the topic was viable for study. They are reduced versions of the online game, as we needed to remove as much of the excess and irrelevant information as possible. They do not include the no-go zone or anything like that, and only contain 6 playable rounds and 6 non-playable rounds.

There are an enormous amount of redundancies in those games due to the copying/pasting of code from one game to another. As such they should be cleaned up significantly in the future.

Also, they used the same data storage method that the online game used… but the issue with that is it requires a server to operate. Without a server, the game is simply not functional. This needs to be completely overhauled (replacing the cookies with local storage possibly?)

NOTE: In order to run these games, you have to setup a server on your computer. Personally, I use XAMPP (found here: https://www.apachefriends.org/index.html). It lets you run the game from your "localhost" server. Again, it would be much more functional if the cookie system was replaced. See the first section on **Initial Setup** for more information.

The games were "supposed" to run for 30 seconds in an "On" state (where the game is being played) and an "Off" state, where the game does whatever it does (shows pictures, parachutes falling, etc). However the first part of the game doesn't exactly run for 30 seconds because the user is allowed to play for as long as they want. This means that an extra file is created, basically presenting the state of the game at 2-second intervals

MRI timing and update (***function this.MRI_timer, this.MRI_update)*** The major component of the MRI compatible games is the timer sequence implemented to output what position the game is in every 2 seconds (by position I am referring to the On or Off state). The timer itself operates every repeats every 5 ms (cannot be lower due to speed limits set by code execution). I set it this way because I found that there was significant drift (over a second) when the timer was set to 2 seconds. By reducing the timer, I try and catch each 2-second interval by calculating a difference, and then adjusting for the offset created by the speed of algorithm execution (so I write every 1997 ms instead of every 2s). This reduces drift to less that 100ms per game execution.

MRI Updating using local storage (***this.MRI_update)*** Another change that was implemented was the use of local storage items to contain the round start and timings of the game. While cookie implementation was used up to this point, I enjoyed the concept of using local storage because it wasn't dependent on the operation of cookies. This means that if the implementation was ever pulled away from server dependence, it would still work (cookies need a server to operate)(this doesn't correct for the need of PHP to write the file however). The variable this.MRI_onState is used to track exactly what round the game is in, and is a variable used often by other functions to determine what behaviour to exhibit.

Also an important note is that the OFF state is switched after 30 seconds have elapsed in this mode, but the ON state can continue for an unlimited amount of time. We collect the data to MANUALLY decode what has occurred, since the GE algorithms depend on 30 second intervals being exact, which is not possible for someone having unlimited time per round.

Database creation and click tracking (***function this.update_db, php MRI6Min.php***) The creation of the files used to track the position of the game is very similar to the database creation in the online game. However, notably, there are several more files being generated in order to make ease of analysis possible. The least varied file created is the "MRI6Min_time.csv" which plots the standard metrics at 2-second intervals. MRI6min

Scheduling: It is important to note that the schedules (the "model" equations used during each round) are different between the two variations of the game.

5.1.1 Game Specific Methods: para_game_MRI6MIN_image.html

Scattered Image (***function this.plotscattered***) For this version of the game, we needed to plot a scattered image of the screen (according to Dr. Noseworthy's suggestions) which would be used as the OFF state to contrast with the ON state. The file "img_scrmble.m" simplifies the process of creating these scrambled images, and even accommodates the removal of white regions which do not add to the image. There may be some minor adjustments required to use the "im_scrmble.m" function, but it should be relatively straight forward (change the name of the file you want to scramble in the matlab file, and go for it!).

5.1.2 para_game_MRI6MIN_falling.html

Falling Parachute (***function this.Rand in SM_Engine***) For this version of the game, we need to display a parachute repeatedly falling for the user over the 30 second span. In order to do this easily, an algorithm (similar to ASM, SM) was implemented, except that it randomly drops the parachute on the screen. The rest of the images on the screen were removed so that all that is seen is the parachute slowly descending. It continues to do this until the time limit is reached, where the screen is then frozen and the player is sent to the next stage

5.2 Challenges and Issues

One major issue that immediately became apparent in these tests was that there was no "mouse" device that was compatible with the MRI machine. Purchase of a mouse devices would have amounted to over $1000, and was considered (most notably parts from the following site: http://www.curdes.com/mainforp/responsedevices/variabledevices.html

To avoid this, we simply programmed it to be compatible with a button device the lab already had. This device has 5 buttons, mapped to 1,2,3,4,6 keys. The 5 key is also mapped, but it sends a signal every time the MRI machine records a picture.

During the setup for the fMRI test, it was also discovered that the projector did not do two things well, 1) the screen size that was actually visible to the patient was less than 640x480, so it was adjusted to being 600x440 and 2) the part of the screen displayed was not directly in the center of the screen, so the game had to be manually moved in the browser window until the patient confirmed that the screen filled their view.

Again, since the timings of the gameplay were off (not perfectly 30s ON state) we had to manually capture the state of the game every 2 seconds, which was then processed by Dr.Noseworthy who has custom software capable of handling this digression.

During the first test of the game, the controls to move the parachute left and right did not work. This ended up being caused by the fact that the button device did not behave in the same manner as a keyboard device. When the button was clicked, it would send one KeyDown event immediately followed by a KeyUp event (even if the key was held down). This meant that the **function keyMovement** had to be adjusted to instead calculate movement based off of each individual click, rather than run a timer while the key was held down. So in these games, the user has to click repeatedly to move the cursor over, and as they increase their clicks the cursor jumps a larger distance. There is probably a way to optimize this and make it smooth, but I didn't look into it as I was creating these games as quickly as I could.

Local Server Requirement: The game is currently requires a server running in order to operate. However, these games cannot be uploaded to the server because their cookies would conflict with one another and cause issues when they are ran. So they can currently only run on the local server I created.

5.3 Future Work

Update to server independence: In order for these games to continue to be used easily, they should be made independent of the local server. This would mean that the cookie system and the PHP system implemented would have to be replaced by something else, or that the system would have to be updated to a singular instance (no page refreshing, similar to what I referenced for the online game).

These games are generally fine and function reasonably (ignoring a few random bugs). There shouldn't be many more changes to them other than to clean up their code by removing redundancies, and to create them so that they work offline.

6 The Swimming Pool Game (formerly cake/wedge problem)

6.1 Overview

The point of the swimming pool game is to split the pool into two halves of equal weight. The pool does not have a flat bottom, which means that slicing the pool in half does not guarantee the pool will be split into two equal parts. When the pool is separated, an imbalance is presented to the user, who then has to adjust their guess to a more appropriate position.

Controls: The marker position is moved by using the keys “2” (moves left) and “3” (moves right) as well as the key “1” to select the position. These values were selected due to similar reasons noted in the fMRI section above (the MRI compatible device is mapped to these keys specifically).

The files for these games can be found in the “block_game” folder. These games do not use any database structures or PHP, so they can be ran directly as is (assuming there are no errors with the way they store local-storage items.

The swimming pool game (pool_game.html) was created when issues of intuition arose with the previous iterations of the game. In order for us to test out the space mapping algorithms, we need to create a game that has as few intellectual challenges as possible. The reason the parachute game was so easy for players to play is that they had an understanding of where to begin, and therefore advanced much more easily during gameplay. However the wedge problem has presented a challenge that seems harder to grasp (though in reality, the equations are much easier!). Hopefully with the implementation of a more intuitive problem the games overall difficulty can be reduced.

Denys Shumakov developed the original game with the intention of making it a cake-cutting game. The original game can still be found in the block game folder under the name "cake 7.2.html".

While the block and cake games can be played over and over again, the swimming pool game was designed to be used within an MRI machine. Its purpose is to test the fMRI's ability to detect the regions that were also activated in the parachute game. The reason for creating another game to test the same scenario is to see if movement activates regions that are not relevant to the space mapping strategies. Because the parachute is falling, the brain may activate in a manner involving the tracking and prediction of the movement, which is not something we want to directly analyze. Since the swimming pool game has no movement (other than the cursor) it should be a more accurate representation of the mapping that goes on inside the patient's brain.

The swimming pool game has quite a few redundancies due to the speed at which it was ported over. Much of the code (such as the slope determination that occurs around **line 165**) is irrelevant to the game and was simply never removed when the transition was made from a browser playable game to an fMRI only game.

Variables may be moderately confusing to understand (this.h1, this.h2, this.m) which were in place when I (Daniel) took over this project. I tried to implement clearer variables and used them when I can, but I never went through and properly cleaned up all the irrelevant/strange variables that remained in the code.

Pool Drawing (pool_game.html ***function this.plotpool, this.fillpool***) The pool is drawn onto the page in two stages, the first being the "plotpool" stage. This simply draws the pool onto the page with no colour changes. An important note here is that the pool is stretched to keep the marker contained within the boundaries of the water, not of the entire image of the pool. The next stage is the "fillpool" stage, which gives the user the impression of water depth, assisting them in their decision. The reason we implemented this was due to the difficulty of understanding what to do after the user first clicks the pool. Initially we just had numbers describing what was occurring (ex. "The left side contains 76% of the water") but we found that this was difficult for people to quickly process. It was not as intuitive as the parachute game was, which was a major hindrance. In order to correct for this, the application of a colour change gives the user some indication of where the water is without telling them a number they have to slowly process.

The Schedule (pool_game.html ***variable coef***) In order to have the same rounds repeated with every iteration of the game, a schedule was established containing the slope of the bottom of the pool. It operates more simply than the schedule in the parachute game, as each round simply goes through the array and selected the appropriate slope, whereas the parachute game will have to select the two variables and whether the no-go zone is established or not. Note that the slopes set to "0.1" are never actually used. These are the rounds where the scattered pool image is displayed. The odd numbered rounds are the slopes that are actually used. Also note that while the array goes to 20 items, only the first 12 are actually used (the original game had 20 possible rounds before it would return to the first slope and start again).

The Buckets (pool_game.html ***function this.drawbucket***) The buckets were created as another step to trying to reduce the dependency on numbers in order to solve the game. Initially, the game only had two

numbers dictating what was occurring, requiring the user to actively do some math in order to hit the correct position. However with the bucket implementation, a relatively intuitive approach of balancing the two buckets can be used as a strategy, reducing the requirement of math on the patient. This function draws both buckets depending on what the user input, and tries to describe variables in a manner that makes sense to the user. I would have liked to use the variables in this function for the overall game to simplify the organization of it all.

The scattered pool (pool_game.html ***function this.poolscattered***) Since this game was developed with the intention of being implemented in the fMRI machine, it follows the same structure as the "para_game_MRI6Min_image.html" game, in that it uses a pixelated image as the OFF state. This image was suggested for use by Dr. Noseworthy who has worked with these types of games with the psychology department. The scattered pool image was generated with the matlab code "img_scrmble.m" which can be found in the para_game folder. See the previous section talking about this method for more information.

Database Saving (pool_game.html ***function this.MRI_timer, this.MRI_update, this.update_db***) Very similar (if not nearly identical) to the functions used in the fMRI versions of the parachute game. These functions use local storage to save a lot of the data (instead of cookies) and even use the same php file "MRI6Min.php" for the processing of the data. See the previous discussion on these variables for more information

6.2 Challenges and Issues

Intuitive: One major issue with this game was making it more intuitive. In its early iterations (and even the cake game implementation) it was difficult to understand exactly what had to be done to complete the game. Some steps have been taking to fix this, such as changing the colouring of the pool and introducing the buckets (mentioned in overview section). However there are still some intuitive obstacles making the game more challenging than it needs to be.

Variables: A huge issue is the disorganization of the variables in the game. Whether it is their naming or their position in the code, they are very scattered. Even at this point in the development I have no idea what certain variables are doing, I just choose to avoid them and redesign certain things with variables that I do understand.

6.3 Future Work:

Clean up the code: This is a huge change that needs to happen. The code is a mess and really needs to be organized into something that is readable and still has proper functionality. As it stands right now I have no idea what half of it does, and I believe that certain parts of the game need to be reorganized to simplify its structure. Basically I want to move functions around to make it better.

Implement a limited attempt system: One major component still missing from the game is an attempt mechanic that restricts the user from spamming their clicks (similar to the "lives" depicted in the parachute game). I do believe that there need to be more attempts allowed at this game because it seems to be harder than the parachute game. Maybe 10 attempts instead of 5? Also a possibility is increasing the tolerance of success so that you don't have to be so painfully close to the correct answer.

Improve intuitive nature: This is a hard issue to solve because there are no obvious answers as to how to make it more simplified. The state it is currently in may be the most optimized way of depicting the swimming pool game (at least at my basic level of JavaScript understanding… maybe something 3D would be useful? I don't know).

Test with the MRI: This game has not been tested with the MRI machine and the device. In order to make sure it's actually functional, it needs to be run with the machine in **SERVER MODE** to see if it operates correctly and saves the data appropriately.

7 fMRI and EEG Testing

7.1 Overview

In July, the idea of testing these games with EEG became an interesting concept. EEG could be beneficial because of its ability to capture the temporal aspects of the though process, which would mean we would have the capacity to POTENTIALLY see the moment when a space mapping calculation occurs/ is implemented. This could have major benefits in analyzing brain disorders by comparing the speed of the response, and the accuracy of the response. The idea of combining the two strategies did not occur until a discussion with Rami Saab (saabr@mcmaster.ca) and Saurabh Shaw (shaws5@mcmaster.ca), research students working with Dr. Noseworthy. In doing this combinational test, we could collect and compare a huge amount of data which would take some time to sift through but may ultimately point at something interesting.

7.2 Challenges and Issues

No tests have begun so far. One major issues is the synchronization of the games with the EEG machine. Since the EEG machine needs to be accurate to the millisecond it would need to capture the exact moments events occur in the game. However this is extraordinarily difficult because the games were developed in JavaScript and were designed to operate on a server through a browser, while the EEG software is Matlab/Third party software dependent. The EEG data collection requires a signal to place markers at the specific time intervals when things occur, and these games do not have an easy method of implementing this. Three possibilities were discussed to overcome this issue:

7.2.1 Implementing the browser games to communicate with Matlab

If it is possible to integrate the game in matlab and run it in a single thread, we would be able to accurately write markers at the specific ms position. Unfortunately, it does not seem possible to run the matlab code and game in a single thread. While the age can be run in the matlab browser, and it can communicate with the matlab software, it cannot run in a script mode.

However, what it can do is send communication to the matlab code. This means that it can capture the timestamp of events, send these timestamps to the EEG software, and with a buffer retroactively place the marker in the appropriate position. JavaScript has the ability to call matlab functions, so if a function was written to perform this behaviour, it may be able to work.

7.2.2 Implementing browser game to communicate with the Third Party software (brain-vision)

There is software called Brainvision which is proprietary software that can communicate properly with the EEG. The challenge is to get the browser game to communicate with it, since it doesn't have a very visible way of allowing communication.

This scenario may be possible if an exterior script was used to communicate with Brain-vision, perhaps something like python. It has to be looked into by the MRI-EEG guys, who at the time of writing this did not respond to my questions.

7.2.3 Timestamp comparison

This is by far the easiest strategy, but is simultaneously the least elegant. By saving the timestamps of the browser game, and synchronizing the clocks, we can collect each event through the database system already implemented. Afterwards we can manually place each timestamp in the EEG data.

The main issue with this is the lack of perfect synchronization. We would be off certain points by at least a couple milliseconds, which can drastically affect the output of the EEG analysis software.

However, it may be useful to try and implement this way so that a quick test can be completed and analyzed to see if that research path has any merit whatsoever. EEG has significant challenges involving timing, because it takes about an hour to set up (versus the 10 minute prep to get ready for the fMRI scan)

7.3 Future Work

Synchronizing with EEG Setup: This is the most obvious task that needs to be completed. Both the parachute game and the pool game need to have systems implemented that allow them to properly communicate with the EEG setup so that we can easily capture useful data. An independent setup is ideal in case more games are developed in the future. Depending on the speed at which a test would need to be done, any of the three strategies can be attempted, though it will take members of both the MRI lab and the space mapping team to get a functional system.

Test value of EEG: Once the EEG tests have been complete, some basic analysis of what is actually happening would be useful to determine whether further pursuits into this region would be useful. I am most interested in seeing the delays between a decision being made and realizing the error, and when the processing could possibly occur. By this I mean it would be interesting to see when exactly an individual realizes where to place the next parachute (at what point in the fall have they made their next guess?). I would also like to see what happens when the parachute behaves in a manner that there we're not at all expecting (such as Round 19 in the current implementation of the game).

8 Conclusions

A fair amount of headway was made during the period I worked on the space mapping games. After developing multiple modalities of these games, I am confident that a pursuit down any of the paths would be beneficial. Right now there are three main research paths I can see that would be interesting

Behavioural Testing: This is what I initially began work on when I joined the development team in May. I think this may be the easiest thing to test and analyze, and would have the most interesting large scale results. By running through a large number of individuals (>100) a map of selections can be made describing each player's decisions. With this, I believe we should see some form of statistical distribution (Gaussian?) around a space mapping algorithm (which I believe will be the simple SM algorithm, not the ASM algorithm). If the swimming pool game was also advanced and pushed into the behavioural tests as well, I think an interesting comparison between the two games, and the strategies players use, would produce valuable results.

fMRI testing: Another interesting avenue that these space mapping games came be used for. By looking at which regions light up in the brain, it may be possible to localize where exactly certain strategies are being implemented. It may even be possible to show that space mapping is used all across the brain! I know Dr. Bandler was interested in doing the test verbally, by describing a scene to an individual and have them solve it by talking. It could be used as a tool to distinguish mental handicaps within individuals as well, though I am not sure how many more tests need to be made before this can be proven.

EEG testing: EEG testing would also be interesting to analyze. According to Rami, the response time of an individual realizing an outcome is well understood in the field. Comparing response times between individuals may allow for markers of intelligence, disabilities to be established. Also may be interesting to see if specific patterns emerge from the EEG which may be associated with space mapping.

Animal Testing: Another pathway that was discussed but not very much looked into was the idea of doing tests with animals to see how closely they achieve the space mapping algorithms. In order to complete these tests, the games the animals would play must be something intuitive to the nature of the animal. While strategies can be taught to the creatures, it would be more interesting to see their behaviour without any human instruction. One possibility is placing food at a certain distance from the ground and having the animal jump for it… only to move the food away by a certain amount. Then let the animal try again and see how their behaviour is adjusted. Following this strategy, a wide range of animal testing could prove how integral space mapping is to all animals, not just humans. It may also be possible to show the advancement of humans in their use of aggressive space mapping strategies, or perhaps if humans and certain animals can use aggressive strategies equally as efficiently.

Thank you for reading this document, I hope it was able to clarify any issues you were running into, or give you some insight into my thought patterns as I worked through the development of this game.

9 Appendix

9.1 Current Updates.txt file

INFORMATION PRE AUGUST 2015

Documentation Notes - Parachute Game
05/01/2015
Created by: Daniel Tajik

V0.02 (05/01/2015 - 06/03/2015)
- Reorganization of control buttons
- Canvas resizing for different resolutions
- Implementation of new paradigms
    + Distance Challenge: Multiple equations developed to increase the difficulty of landing the parachute on the target
        = Random: Equation is basic (Note: this is the original equation) (0.8-1.2 * A +/- 100)
        = Easy: Equation is easy (0.8-1.2 * A +/- 1/4 canvas width)
        = Medium: Equation varies more (0.7-1.3* A +/- 1/4 canvas width)
        = Hard: Equation varies widely (0.5-1.5*A +/- 1/2 canvas width)
        = Sched: Equation is set to whichever values determined by the scheduler
    + No-Go Zone Challenge: An area which when contacted causes a fail condition for the round. The zone varies in size depending on difficulty
        = Easy: No-Go zone is 1/10th of the total canvas, randomly placed on page
        = Medium: Zone is 1/8th of the total canvas, randomly placed
        = Hard: Zone is 1/5th of the total canvas, randomly placed
    + Speed Challenge: An increase in speed in parachute descent
        = Normal: Original speed (1 Frame = 1 pixel increment)
        = Fast: Double the normal speed
        = Super Fast: Three times the normal speed
        = Ultra Fast: Four times the normal speed
    + Movement Challenge: Alters the trajectory of the parachute in a specific manner
        = Disappearing: The parachute disappears while descending vertically at point of click, and reappears at final destination
        = Exponential: The parachute falls along an exponential decaying pathway towards final destination

- User Info Form generated on initialization of web page
- Cookie storage system used to collect user information, save states
- PHP database storing system, generates a database of user inputs for later analysis
- Implementation of keyboard controls, touch based controls, and mouse controls for universal device functionality
- Optimization for Chrome, Firefox, and Safari browsers (currently no IE support)
- Updated win and lose conditions
- Scoring metric implemented
- Multiple scheduling systems enabling regulated user levels as progression through the game occurs
- Ruler for accurate user input added
- Event handling for AI controls
- Space mapping automation added
    + SM, ASM, and RSM strategies from 2D targeting game implemented, with only 1D characteristics determined

- General upgrades including:
    + Updated messaging outputs
    + Offscreen parachute notification
    + Removal of antiquated code
    + Bug fixing and speed improvements
    + Admin control versus basic user control
- Added "close" no-go zones for larger challenge
- Added scheduled no-go zone locations for mirrored gameplay across all user attempts
- Added "Time Challenge" creating count down timer (30 seconds)

V0.3 (06/04/2015 - 06/15/2015)
- Added "Help" button for explanation of how to play game
- Improved keyboard movement animations
- Modified PHP for proper data collection while on server
- Return parachute to screen after delay if user launches it offscreen
- Provide notification method for user when round ends to continue onward

V0.31 (continuation) (06/16/2015 - 06/29/2015)
- Improved "Help" button to be more intuitive, initializes at beginning of the game
- Added additional text to notify user of changes to the game
- Implemented more secure strategy for obtaining administrator control of game through PHP