Brain Puzzle:
Arrange Objects in 3D Space

Software Requirements Specification

Version 1.0

October 26, 2012

Bilal Bhakhrani
Emily Li
Tianyuan Qin
Ding Zhao
Mengyi Zhu

Revision History

Date	Description	Author	Comments
Oct 26th, 2012	First draft	Bhakharani, Li, Qin, Zhao, Zhu	

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed name	Title	Date
		Lead software Engineer	

Table of Contents

Document Approval.......... 2
1. Introduction.......... 4
1.1 Purpose.......... 4
1.2 Scope.......... 4
1.3 References.......... 4
1.4 Overview.......... 4
2. General Description.......... 5
2.1 Product Perspective.......... 5
2.2 Product Functions.......... 5
2.3 User Characteristics.......... 5
2.4 General Constraints.......... 6
2.5 Assumptions and Dependencies.......... 6
3. Specific Requirements.......... 6
3.1 External Interface Requirements.......... 6
3.2 Functional Requirements.......... 8
3.3 Use Cases.......... 10
3.4 Class/Objects.......... 11
3.5 Non-functional Requirements 11
3.6 Design constraints.......... 11
4. Change Management Process.......... 12

1. Introduction

1.1 Purpose

This software is a fun, interactive and educational puzzle game to help people learn neuroanatomy. This interactive tool will allow users to study and learn about the different parts of the human brain in an online three-dimensional space.

1.2 Scope

This software is called Brain Puzzle. The goal of the game is to arrange pieces into a human brain. It is a component of the LONI PIPELINE project, and it will based on a web application called brain viewer. The brain viewer app provides file IO and image rendering, while our software will provide the functionality of the game. The game will start off with a brain divided into certain number of regions/pieces, each piece is colored and positioned randomly. During the game, the user will drag each brain pieces together and see if the result is correct. However, to make the game easier, our software does not provide zooming or rotation functionality. The goal of the game is to arrange pieces into a human brain. At the end, a score is given based on how well the user completes the game and shows how much does the user understands about neuroanatomy.

1.3 References

LONI PIPELINE: 2012 Student Project
<http://www.loni.ucla.edu/twiki/bin/view/LONI/Pipeline_2012_StudentProjects>
XTK Library <http://api.goxtk.com/>
Brain Viewer Project <http://users.loni.ucla.edu/~zliu/viewer/>

1.4 Overview

In the next few sections, this SRS document provides a detailed and complete explanation of the requirements. Section 2 gives a general description of the product, including general factors that affect the product and their requirements. Section 3 is the specific requirements which explain both functional and non-functional requirements, and developers should focus on this section. Section 4 explains the change management process that developers should follow upon modification and update of the software or this document.

2. General Description

2.1 Product Perspective

As far as we are concerned, most of the software that are used to learn about neuroanatomy are not free. Therefore, it would be nice for a free and open source software to be created so that it can help people to learn about the knowledge of the brain. The current software to be designed will add features to an existing project (LONI Pipeline) in order to create an interactive learning tool.

2.2 Product Functions

- There are already two levels (13 pieces and 56 pieces) existing for the brain puzzle, and in addition to this, the software will have a customized level: user specifies a level by giving a list of URLs of 3D model files and their corresponding descriptions. Each URL points to a file that can be loaded as a piece. So user can create a game with any number of a valid number of brain pieces.
- The software will provide a mechanism to allow the user to move the piece in the 3D dimension.To move an object, player can select the object, toggle "move" action, and drag the mouse pointer. The object will move in a 2D plane (current viewpoint).
- Textual hints are already built-in with the project given, and the software will add a function to turn it on and off. Also the software will add a Visual hint option that will display the correctly arranged pieces as a shadow, so the player can move pieces into the shadow slots.
- The game should end when all pieces are placed correctly in space. In addition, an "End Game" button is added by our software so user can terminate the game. After a game is terminated, the software will show a score to the player. The score should be determined by time, correctness, hints used, and so on.
- Once a piece is selected, our software will provide the functionality to allow the user to change its color and opacity. The given project already has part of the job done. It is our software's responsibility to complete it.

2.3 User Characteristics

The software is designed for people such as medical students who need to learn about the parts of the human brain. However, since this software is web-based, anyone who has a compatible web browser (Mozilla Firefox, Google Chrome) can access the software. As an web-based educational game, this software can have a lot of potential users.

2.4 General Constraints

This interactive game can only be played on a Google Chrome or Mozilla Firefox web browser with WebGL support.

The additions made to the client's original work must follow the given formatting and structure, including but not limited to any internal packages used.

Since the design process must be in the same pace as the course, the schedule may not be flexible enough for too much creative work beyond satisfying the requirement.

2.5 Assumptions and Dependencies

- It is assumed that the needed files for each brain part are provided and are in the correct format as needed by the application.
- It is not assumed that the URL provided by the user for the custom level is a working link, but it is assumed that the file the valid URL points to is the correct brain file in the correct format.
- It is assumed that the base product the client has provided is correct and error-free.
- It is assumed that the files given by the user, which represent the pieces of brain, will be able to load and combine together.
- Some unused portion on the right corner could be removed from the given project since they are not important for this project.
- This application is dependent on the XTK library. In addition, The software will be written in javascript and no other language will be introduced in order to maintain simplicity of the project.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface for this project will be in two parts: 1) menu page to make options and start the game 2) interface during the actual game play.

Before the game starts, there will be a screen with the following elements:

- A drop-down menu to select between two levels: easy and hard
- A check box to select if a user wants to give their own files for a custom level (functional note: this would disable the drop-down menu above)
- A text box next to the check box that will take the urls the user provide
- Instructions on how the game works
- A button to start the game (functional note: when pressed, this button takes up to the next page)

Once the game starts, the screen the following elements for user interaction will be added:

- A check box option to switch mouse functions between rotation and brain piece movement
- A button to terminate the game (functionality note: this will display final score in center of screen)
- A radio button to display hint
- If the radio button to display the hint with a time limit is selected, a drop down menu with different time option will appear next to it.

Once the game finished, either by connecting all pieces or manually click the finish button, a window will pop up to display the score.

3.1.2 Hardware Interface

Since the whole project is a web-based application, no hardware requirements are explicitly needed.

3.1.3 Software Interface

Since the given project requires WebGL. This software also need browser has to support WebGL. In particular, it has to works in Google Chrome and Mozilla Firefox.

3.1.4 Communication Interface

This software does not interfere with the communication interface that the given project provides. Instead, it will always trust the input files are in correct format if it does not come from the user.
When treating the files the user specified, it must follow the pattern as the game instruction specified.

3.2 Functional Requirements

3.2.1 Dragging/Rotating brain pieces

3.2.1.1 Introduction
The player can select a brain piece by clicking it and then move it. Also, user may rotate the scene with the mouse to change the viewpoint. A check box is used to toggle between dragging a brain piece and changing the viewpoint.

3.2.1.2 Inputs
The state of the action box (move or rotate), the movement of mouse pointer.

3.2.1.3 Processing
Changing the x,y,z coordinate of the object according to the movement of the mouse.

3.2.1.4 Outputs
The position of the selected object is changed.

3.2.1.5 Error Handling
Input/output error should be handled internally.

3.2.2 Display/Hide hints

3.2.2.1 Introduction
To make the game less difficult for less competent players, it will provide visual and textual hints.

- Visual hint will display the correctly arranged pieces as a shadow (transparent without coloring), so the player can move pieces into the shadow slots.
- Textual hints are associated with each pieces, the hints could be the name of the region, or the function of the region (a description of what the region does).
- Both hints can be turn on and off.

3.2.2.2 Inputs
The state of the checkbox regarding visual hints and textual hints.

3.2.2.3 Processing
Reading the hints from the file if textual hint is selected; reconstruct the pieces if visual hint is selected.

3.2.2.4 Outputs
Show the resulting text if textual hint is selected; show the correctly arranged pieces as a shadow if visual hint is selected

3.2.2.5 Error Handling
Input/output error should be handled internally.

3.2.3 Selecting levels

3.2.3.1 Introduction
The game starts off with 13 pieces (easy level) or 56 pieces (hard level) randomly positioned in space. Each piece is 3D model file with correct position, and the program loads the file and applies a random transformation matrix in order to put it in a random position. In addition to the two predefined levels, the game should also allow a customized level: the user specifies a level by giving a list of URLs of 3D model files and their corresponding descriptions. Each URL points to a file that can be loaded as a piece. So user can design any number of pieces with any combination of pieces.

3.2.3.2 Inputs
The state of the radio indicating easy or hard, the state of the checkbox indicating if the user need custom level and the URLs.

3.2.3.3 Processing
If level is set to easy or hard, program will read files from local file system. Otherwise it will read files from URLs specified by the player.

3.2.3.4 Outputs
Displaying objects onto the screen.

3.2.3.5 Error Handling
Input/output error should be handled internally. If URLs are not valid, the program will show error messages in the console.

3.2.4 Ending game

3.2.4.1 Introduction
The game should end when all pieces are placed correctly in space. There is also an "End Game" button so user can terminate the game.

3.2.4.2 Inputs
Pressing “End Game” button.

3.2.4.3 Processing
The program repeatedly checks if all pieces are placed correctly in space.
The program also repeatedly checks if “End Game” button is pressed.

3.2.4.4 Outputs
A text showing the score. The score is determined by time, correctness and hints used.

3.2.4.5 Error Handling
Input/output error should be handled internally.

3.2.5 Opacity change

3.2.5.1 Introduction
When user clicks on a piece, he can change the opacity with the slider.

3.2.5.2 Inputs
Mouse click and slider.

3.2.5.3 Processing
Change the opacity value of the object.

3.2.5.4 Outputs
Display the object.

3.2.5.5 Error Handling
Input/output error should be handled internally.

3.3 Use Cases

3.3.1 Dragging brain pieces
By dragging brain pieces, user will be able to change the position of any brain pieces, the user will be able to rearrange all the pieces into the original brain.

3.3.2 Display/Hide hints
By checking display hints box, the user can get the hints. By unchecking the box, the user can remove the hints.

3.3.3 Selecting levels
The user will be able to select different levels at the beginning of the game.

3.3.4 Ending game
The user will be able to end the game by either click “end game” button or place pieces into correct places.

3.2.6 Opacity change
When user clicks on a piece, he can change the opacity with the slider.

3.4 Class/Objects

The client does not require explicitly our software to have our own javascript file, thus, it is left to the designer to decide whether all the functionality of the software will be in a new file or merged into the given project.

3.5 Non-functional Requirements

3.5.1 Performance

- Moving from top of the screen to bottom will not take more than 2 seconds.
- Displaying hint takes no more than 3 seconds for 15 pieces or less, and no more than 5 seconds for 60 pieces or less.

3.5.2 Reliability

- Game progress will be lost if the browser is closed.
- Probability of crashing should be less than 1/5000.

3.5.3 Availability

- The project is open source.
- Anyone who knows about the LONI project will have access to it.

3.5.4 Security

- The program will have normal behavior as long as user gives valid input.

3.5.5 Maintainability

- All the new code should have the same style as the code that already exists.

3.5.6 Portability

- The program can be run in Google Chrome and Mozilla Firefox when WebGL is enabled, regardless of the environment of the user.

3.6 Design constraints

- All the image files should be stored in the same machine as the one that runs the application.
- The software must be written in javascript based on XTK and WebGL. No other tools should be added.

4. Change Management Process

Since it is a project for a course, it may be possible that one of the design team member of this software will quit half-way. Therefore, any decision change must be agreed by compromise among all the designers. No individual should contact the client and make changes to the SRS. Most of the change in the SRS should be made when all designers and possibly the client have met and discussed potential changes. If a change is inevitable during the time the group cannot meet, emails are allowed to change opinions and come to the agreement. However, the messages must be exposed to all group member, including every designer and the client.