Cognitive State Representation and Visualization of Human Brain

Simple Labs / 21.03.2014

Iteration Report I

Introduction

This is the iteration report for Iteration I, which was planned to be held between February 17 and March 18. As Simple Labs, we completed all tasks which belong to this iteration. The detailed information about this iteration and tasks are given below.

Iteration I

There were three tasks in the first iteration. Two of them has been completed. One of them spans 2 iterations, thus this task is still ongoing. Aside from these tasks, we re-organized our project structure on SVN so that it can be distributed more easily.

Task 1: Bugs in the MAT File Loader

This file loader plugin both provides a native-managed MATIO bridge and supplies “MAT File Loader” processor. Since MAT file format will be the predominant file format, it is essential that this processor be bug-free. Getting MAT File Loader rid of its bugs was one of this iterations main concerns.

MAT File Loader had problems with garbage collection of .NET runtime. Thus, the system would cause “use after free” problems. Before iteration, these problem were solved by not freeing anything that was received from native-managed bridge. While this approach solved the problem, it would cause memory and file descriptor leaks. Problem was solved with adding a back-reference to necessary classes.

Another problem was with variable name lists. MAT File Loader can (and should be able to) load .MAT datasets with different internal structures. After this iterations additions, this loader can load every dataset that we have access to.

Task 2: Additional Input Processors (File Loaders)

While our main purpose is to read and display .MAT file formatted datasets, we want this product to be used with a variety of file formats. Since we cannot implement loaders for every file format out in the wild, we want to create example loaders from which people can learn how the system works and implement their own.

For this purpose, we implemented a CSV-like dataset loader, which reflects data formats used in the .MAT datasets.

Task 3: Line Drawing Optimizations

Our first prototype used rectangular prisms to show connection between voxels. This is costly regarding draw calls. Since we are drawing a static 3D representation, we can do better. First part is to draw connections with actual lines.

Unity provides a `Debug.DrawLine()` function that can draw a 3D line. However, this function does not conserve depth, since they are only for debug purposes. Thus, we are thinking of a way to draw lines with 2 triangles that always face the camera. By drawing lines this way, we can both run faster and conserve connections depths.

Implementation of this method is left for the second iteration.

Conclusion

As of first iteration end, we completed all tasks that should be completed in first iteration. Our plans on next iterations will continue as written before.