CMPT 371 – Team 3 Testing Document

Virtual Reality Medical Imaging Software with Luxsonic Technologies Inc.

1 Introduction

The testing plan outlines what types of tests we will be running and how we will be running them each iteration. This document also contains the delta changes from the last iteration as well as issues that we had with the chosen frameworks and solutions. We will be using a top-down approach, as we will start with smoke tests and then work our way down through system and integration testing. Additionally, regression testing will be documented as issues in the code base arise and fixes are implemented. User testing will be used in later iterations once we have a functioning system and acceptance testing once we are confident in the delivery of the requirements of the system in said later iterations. This document will not contain sections for Coverage testing or for a testing matrix as they have been triaged out for this iteration due to our framework issues.

2 Delta Changes

2.1 ID 2 Changes

Many tests in ID1 are no longer useful since architecture of our system changed as a result of how certain unity features behaved with the Oculus Rift. Tests that were removed are removed because they no longer fit the design of the system, and other tests are still relevant, but had to be updated to reflect newer methods. Many of the tests in the document this iteration are being left as //TODO for ID3 as they are not currently relevant to our system. Lastly, we updated the language of most every test to make it more clear on what we are testing and how we are testing it. Any other changes will be marked as such and given a description.

- Updated the Purpose description
- Added Delta Changes
- Added Testing Framework Issues
- Smoke Tests
  - Made verifying dependencies more detailed instead of just one line each
  - Deleted checking for the main menu
  - Check for registered default scene instead of workspace
  - Delete camera check because that is associated with the default scene check
  - Delete button alias check because that is associated with the Oculus libraries
- System Testing
  - Removed must/should/could haves
  - Removed Initialize the program
  - Removed Quit from the menu
  - Removed load DICOM file
  - Moved exit workspace to acceptance testing

        - Updated select images to display
        - Updated remove images from display
        - Updated adjust contrast
        - Updated adjust brightness
        - Updated apply filter
        - Updated zoom
        - Updated move
        - Added copy image
        - Added load file from file browser
        - Added next/prev image in display
- Integration Testing
    - Fixed grammar in introduction
    - Deleted everything else as a result of restructuring project. Will be in ID3 when we have pieces integrating.
- Regression
    - Added issues as regression tests
- Removed coverage testing for this iteration
- User Testing
    - Grammar
    - post -ask them after to see how they feel
    - look for studies and better questions
    - read Oculus warnings to them
- Coverage testing
    - Deleted for this iteration
- Acceptance Testing
    - Added methods for internal acceptance testing
- Testing Matrix
    - Deleted for this iteration

3 Testing Framework Issues

Originally, we had researched that NUnit and NSubstitute were good testing and mocking frameworks to use for a C# project. With Unity, there is an official bundle called "Unity Test Tools" which integrates these 2 specific frameworks into a Unity project. This gave us a lot of reassurance that we had picked the right frameworks and made the mistake of thinking that using them would be easy. This section outlines all of the issues that we had setting up the testing framework and is in this document for the purpose of documentation and troubleshooting as we have already spent over 30 hours making this Framework function.

3.1 Issue #1: NUnit could not be found

The first issue that we ran into was that even though our unity project contained Unity Test Tools and Unity Test Tools contained NUnit, when we were writing scripts, they could not find NUnit. To try and fix this we tried importing NUnit into the project, but then we got an error saying that there were 2 NUnits in the project. so we had to take out the one that we just put in because when we took out the one from Unity Test Tools, everything in the package broke. The second thing that we tried was to add the NUnit library to the Visual Studio and Monodevelop editors because as it turned out, Unity Test Tools only applied NUnit to the Unity editor, and not the C# editor.

Solution: add NUnit to Visual Studio and Monodevelop.

3.2 Issue #2: NUnit cannot be added to C# editor

Visual Studio/Monodevelop cannot add NUnit to the C# project because the project was not actually .NET Framework v3.5 or greater. It is a special type of project labelled Unity Full 3.5 and because Visual Studio and Monodevelop are set to compile to .Unity Full v3.5, they were not allowing us to add a .NET add-on for C# to a Unity project. On top of that, the best part was that the Visual Studio editor would not allow us to edit the project's settings to change it to .NET so that we could finally include NUnit.

Solution: We fixed this by compiling the project as Unity Full v3.5 and once it was compiled, the .csproj that the editor created could be unloaded from the project and then edited. When we edited the .csproj files we changed the version from Unity Full v3.5 to.NET Framework v4.5.2 (latest release to work with NUnit 3) and deleted a few other lines in the header that defined this as a Unity project. The project was now a .NET Framework v4.5.2 project and we then reloaded the .csproj files and went straight to NuGet extension manager to add NUnit to all of the project files. once NUnit was added we then rebuilt the project and NUnit had been added, it reestablished itself as a Unity Project, and our C# scripts now recognize the NUnit.Framework as usable.

3.3 Issue #3: NUnit does not discover tests

Now that NUnit is Usable, we can use the tests runner, but no tests are discovered.

Solution: NUnit3TestAdaptor had to be added to the C# editor using the same steps as the solution for Issue #2. Once we added the NUnit3TestAdapter to the project and then we finally ran tests, but they would only run sometimes and when they stopped working. When this happened the build would have to be cleaned and the version had to be reset to 4.5.2 manually every time (See Issue #5 for solution).

3.4 Issue #4: NUnit3TestAdapter does not discover tests

Some team members computers still could not see NUnit and discover tests even after applying fixes for issue #3.

Solution: This was fixed by reinstalling Unity with the editor as there were some packages that they exchange that were only put in when they were installed together using the newest Unity installer. We tried manually fetching these files and adding them in, but there were too many versions and serial numbers to deal with as a result of Visual Studio and Unity both being licensed products.

3.5 Issue #5: NUnit3TestAdapter STILL does not discover tests

We still could not run tests perfectly every time and had to consistently clean and reversion the build.

Solution was that we were using .NET Framework 4.5.2 and NUnit3TestAdapter only needed to use 3.6 and NUnitTestAdapter.

3.6 Issue #6: C# editor could not access because of Unity permissions

We could not access any Unity specific functions or classes because the C# editors did not have the permissions to access them from the Unity portion of the project.

Solution: This is fixed by running the tests from the Unity editor which makes it so that we have the proper permissions and it runs our tests from there. A small additional issue to this problem is that now some of our tests that passed in the C# editor are now failing in the Unity editor. luckily, Cloud Build builds the Unity project and runs tests from the Unity editor, so we just have to program so that our tests pass using the Unity Editor, not the C# editors.

3.7 Issue #7: Mock objects only return null objects

When using NSubstitute the objects cannot properly mock methods and just throw exceptions

Solution: there is currently no solution for this and will be looked at further in ID3

4 Smoke Testing

These tests are aimed at ensuring the most basic functionality of the system. They are the first set of tests that will be run during a cloud build so that if they fail, the build fails early before it begins running more exhaustive testing.

4.1 Verify all dependencies (libraries, imports, etc)

Test1: Check that OVR asset exists in the project

- Obtain the absolute path to the directory “Luxonic Project” by calling the Directory.GetCurrentDirectory () function.
- Use the Path.Combine() function to add the path “Assets/OVR” to the absolute path.
- Assert that the directory exists (Directory.Exists()) and that it contains files (Directory.GetDirectories(targetPath).Length > 0)) and it contains directories ((Directory.GetFiles(targetPath).Length > 0).

Test2: Check that OVR Avatar asset exists in the project

- Obtain the absolute path to the directory “Luxonic Project” by calling the Directory.GetCurrentDirectory () function.
- Use the Path.Combine() function to add the path “Assets/OvrAvatar” to the absolute path.
- Assert that the directory exists (Directory.Exists()) and that it contains files (Directory.GetDirectories(targetPath).Length > 0)) and it contains directories ((Directory.GetFiles(targetPath).Length > 0).

Test3: Check that OVR Avatar Settings exists in the project
- Obtain the absolute path to the directory “Luxonic Project” by calling the Directory.GetCurrentDirectory () function.
- Use the Path.Combine() function to add the path “Assets/Resources” to the absolute path.
- Assert that the directory exists (Directory.Exists()) and that it contains files (Directory.GetDirectories(targetPath).Length > 0)).

Test4: Check that OVR Gamepad Bundle Plugin exists in the project
- Obtain the absolute path to the directory “Luxonic Project” by calling the Directory.GetCurrentDirectory () function.
- Use the Path.Combine() function to add the path “Assets/Plugins/OVRGamepad.bundle/Contents” to the absolute path.
- Assert that the directory exists (Directory.Exists()) and that it contains files (Directory.GetDirectories(targetPath).Length > 0)) and it contains directories ((Directory.GetFiles(targetPath).Length > 0).

Test5: Check that ImageEffects standard asset exists in the project
- Obtain the absolute path to the directory “Luxonic Project” by calling the Directory.GetCurrentDirectory () function.
- Use the Path.Combine() function to add the path “Assets/Standard Assets/Editor/ImageEffects” to the absolute path.
- Assert that the directory exists (Directory.Exists()) and that it contains files (Directory.GetDirectories(targetPath).Length > 0)).

Test6: Check that Effects standard asset exists in the project
- Obtain the absolute path to the directory “Luxonic Project” by calling the Directory.GetCurrentDirectory () function.
- Use the Path.Combine() function to add the path “Assets/Standard Assets/Effects” to the absolute path.
- Assert that the directory exists (Directory.Exists()) and that it contains files (Directory.GetDirectories(targetPath).Length > 0)) and it contains directories ((Directory.GetFiles(targetPath).Length > 0).

*NOTE: we are not checking for any testing framework dependencies as they are not critical to the system.

4.2 User gets into the default scene

Test1: Check that a default scene is present in the build

- Obtain the default scene from the list “Scenes in Build” in Unity by calling the EditorSceneManager.GetSceneByBuildIndex(0) function.
- Assert that returned scene isValid().

5 System Testing

The purpose of these tests is to ensure that the system is in compliance with the stakeholder’s requirements.The tests are all black-box and use the same options that the user would have in the user interface.

5.1 Copy Image//TODO

Test that the copy is deep, not shallow, and that all modifications made to the clone do not affect the original at all.

5.2 remove image//TODO

Be sure that the copy and all related memory no longer exist.

5.3 Load from file browser//TODO

Test that the loaded images have been added to the doubly linked list of DICOM files

5.4 Select image in tray//TODO

Test that the doubly linked list is currently positioned at the same element as isselected in the tray and is showing on the display

5.5 Remove from tray//TODO

Doubly linked list should no longer contain selected element and should select a new element

5.6 Remove all from tray//TODO

Doubly linked list should be empty.

5.7 Next/Previous image in display//TODO

Sets the next or previous element to the current node in the doubly linked list

5.8 Change contrast//TODO

Test that the contrast value changes.

5.9 Change brightness//TODO

Test that the brightness value changes.

5.10 Move Copy//TODO : describe these in more detail - give inputs and outputs

Test 1: Move the object to a new spot and check transform.position
Test 2: Rotate the object to a new rotation and check transform.rotation
Test 3: check delta position to make sure it is 0 after a the player lets go

5.11 Change filter//TODO

Test that the enum changes.

5.12 Change zoom//TODO

Test that the value changes.

6 Integration testing//TODO add changes for ID3

Involves testing the interactions between integrated components to ensure that their behaviours are proper. These tests a should cover the interactions of functions that call each other from the same class or separate classes via white-box testing and without the use of mocks. The UML diagram should help show some of the relationships between components.

7 Regression testing

These tests will be added to this document as bugs/errors arise. Once a bug report is made, the appropriate tests will be added here to ensure that the bug does not arise again anywhere in the code.

This iteration we did not have any code issues logged because we were just redoing what we did in our spike prototype and re-constructing our project

8 User testing

User testing will require pre-test and post-test documentation to ensure the user understands the risks and requirements of the using VR software. All questions will be optional for the user to answer. These questions will be formatted in a separate document when that becomes necessary.

Pre documentation elements will include:

- The tester informing the user that they must:
    - Use hand sanitizer that is supplied
    - Accept assistance in taking the headset on and off
- How much have you used VR before?
- Are you prone to motion sickness?
- How is your current health?
- How is your eyesight?

During the testing the tester is to:

- Ask the user questions
- Take notes on their responses and comments

Post documentation will include:

- How does your head/stomach/ eyes feel?
- Were there any interactions that you did not enjoy?
- Additional comments

9 Acceptance Testing

9.1 Internal Acceptance Testing

Manual tests to be done with the Project Manager once development is near completion to ensure that the system is meeting the project's requirements, as well as by testers and developers at regular intervals.

Acceptance tests will be performed:

- By developers after working on an affected feature.
- By testers after a merge of a feature branch into the development branch
- By testers after a development freeze
- By the Project Manager just prior to turning in a deliverable build

Record of the test being performed, and its results, will be recorded in the project wiki.

9.1.1 Quit Workspace

Test to ensure that clicking on the “Quit” button closes the workspace at anytime.

9.1.2 Show/Hide Dashboard

Test that when the “Show Dashboard” button is pressed that the dashboard is displayed with all elements which include:

- “Quit” button
- “Load” from File browser button
- The Tray with any images it had on it still loaded and highlighted
- The Display with any images it had still on it that match the highlighted images in the Tray
- “Hide Dashboard” button

Test that when “Hide Dashboard” button is pressed that all of the above elements are hidden and the only element that remains is the “Show Dashboard” button

9.1.3 Copy Image

Test that when the user presses “A” or “X” on an image in a displaying dashboard that the image is pulled out and represented as a new unity game object in front of the user separate from the dashboard. Ensure that “A” and “X” cannot be used to copy an image when the dashboard is hidden

9.1.4 Remove Image

Test that when the user presses “B” or “Y” on a copied image displaying in the workspace that the image is removed and no longer represented. Ensure that “B” and “Y” cannot be used to remove any object from the dashboard.

9.1.5 Show file browser

Test that when the “Load” button is clicked that the file browser is displayed. Ensure that the file browser exists over top of everything else in the workspace.

9.1.6 Close filebrowser

Test that when the “Close” button is pressed that the file browser completely closes.

9.1.7 Load from file browser

Test that when the “Load” button is click and a directory/images are highlighted that all the images that are in the directory are loaded to the tray. Ensure that “Load” does nothing when nothing is selected.

9.1.8 Select image in tray

Test that when an icon in the tray is selected that it is shown on the display along with the icons around it.

9.1.9 Remove from tray

Test that when “Remove” is clicked that the current item that is being displayed is removed from the tray and the display is updated to reflect this.

9.1.10 Remove all from tray

Test that when “Remove All” is clicked that all items are removed from the tray and the display is updated to reflect this.

9.1.11 Slide through images in display

Test that the user can grab the slider and move it back and forth to browse through images in the tray. Ensure that the tray’s selection indicator is updated to reflect this.

9.1.12 next/previous image in display

Test that the user can press the next and previous arrow to browse through images in the tray. Ensure that the tray’s selection indicator is updated to reflect this.

9.2 External Acceptance Testing

To be done by the Stakeholder to ensure that the system is meeting the requirements.