BrainGenix Engine and Game Test Plan

Prepared by: Josh Coldiron

Document creation

date: 1-31-2023

Contributors:

Table of Contents

*TOC to be updated and numbered once initial document is finalized.

Intro:

This document is the describe the test plan for the BrainGenix Environment Rendering System (ERS). The ERS is a high-performance rendering system aimed towards scientific applications such as virtual body simulation. This test plan will later be adapted to the BrainGenix developed game Eternal Souls (working title). Eternal Souls is an interactive experience designed to showcase elements of the Carbon Copies whole brain emulation as well as promote the company's research.

Objective:

The goal of this document is to provide the basis for establishing a test plan that can evolve to be applied to both projects with the aim of releasing high quality software applications.
Due to the limited testing staff the plan is organized in a way where any level of employee can submit an issue. It is organized in a way where each department who works within the applications can log bugs relative to their use. This is to act in place of a dedicated testing staff until a dedicated testing volunteer is available.
Any changes, or additions to the document or related planning documents will be documented and tested at the best of the team's ability within their remaining time on the project.

Definitions and Acronyms:

The following section covers definitions and acronyms used throughout the document.

Integration: You test the integrations of many units together. You make sure your code works when put together, including dependencies, databases, and libraries.

Regression: After integrating (and maybe fixing) you should run your unit tests again. This is regression testing to ensure that further changes have not broken any units that were already tested. You can run your unit tests again and again for regression testing.

Acceptance: You should test that the program works the way a user/customer expects the application to work. Acceptance tests ensure that the functionality meets business requirements.

Bugs: Any error or defect that cause the software/application or hardware to malfunction.

Enhancement:
1) Any alteration or modification to the existing system for better workflow and process.
Enhancement can be added as a new requirement after appropriate Change Management process.

End to end testing: Tests user scenarios and various path conditions by verifying that the system runs and performs tasks accurately with the same set of data from beginning to end, as intended.

API:	Application Program Interface
UX:	User Experience
UI:	User Interface
GUI:	Graphical User Interface
QA:	Quality Assurance
OS:	Operating System
TBD:	To be determined

Scope:

Scope of testing will be a full team effort and relate to the department each is functioning within. As the application grows to certain intervals a set of tests will be scheduled to cover any functionality that has not yet been used. The test scope includes the following:

• Testing all functional, application performance, and requirements listed in feature list.

• Quality requirements and fit for purpose testing. Fit for purpose will determine if the application executes the functions its intended purpose as per the design documentation.

• End to end testing and resting of interfaces of all systems that interact with ERS.

Roles and Responsibilities:

Developer: Responsible for producing the application and systems involved. They will conduct Unit, system, regression, and integration testing.

Adopter: BrainGenix staff who are working with the engine to produce the game will be responsible to run tests related to their department. Examples include, game designer testing out scripting portions, UI designer testing GUI elements, and 3D designers testing the functions of animation and 3d elements.

Testing Process Management Team: There is currently no dedicated staff for testing at this time however during testing cycles someone on the team will manage testing as needed.

Constraints for Test Execution

Below are assumptions for related tests before they are to take place.
Before User Acceptance testing, Developers have completed their tests.
Acceptance testing will be completed by End-Users.
Testing results will be recorded through Trello comments on the related working issues, which will notify the dev team.
Major issues or bugs should be reported immediately.
Any test scripts available or automation should be added to the testing document.

Testing Methodology

The purpose of the test plan is to accomplish the following:
Define test strategies for each department.
Define bug-tracking processes.
Identify required resources and related information.

The purpose of Usability testing is to ensure that features function in a way that is acceptable to the end user. Usability testing will often be the result of prototyping a non-function project through an application such as Figma to test and refine these components.
Unit testing is conducted during code development to ensure to proper functioning of code.
Regression testing will be the process to ensure that previous bugs are tested and do not arise in the next build of the application.

Final release testing is where the development team and end users participate in finding any new bugs and confirming no application breaking bug is still included in the final build. The final quality of the application will be determined when it is acceptable to release the application. Releasing for distribution can only be completed once all levels of the testing are complete.

Test Levels

*This section was taken from a test plan template. This standard should be discussed and modified based on our resources.
Testing of an application can be broken down into three primary categories and several sub-levels. The three primary categories include tests conducted every build (Build Tests), tests conducted every major milestone (Milestone Tests), and tests conducted at least once every project release cycle (Release Tests). The test categories and test levels are defined below:
Build Tests

Level 1 - Build Acceptance Tests

Build Acceptance Tests should take less than 2-3 hours to complete (15 minutes is typical). These test cases simply ensure that the application can be built and installed successfully. Other related test cases ensure that adopters received the proper Development Release Document plus other build related information (drop point, etc.). The objective is to determine if further testing is possible. If any Level 1 test case fails, the build is returned to developers un-tested.

Level 2 - Smoke Tests

Smoke Tests should be automated and take less than 2-3 hours (20 minutes is typical). These tests cases verify the major functionality a high level.
The objective is to determine if further testing is possible. These test cases should emphasize breadth more than depth. All components should be touched, and every major feature should be tested briefly by the Smoke Test. If any Level 2 test case fails, the build is returned to developers un-tested.

Level 2a - Bug Regression Testing

Every bug that was "Open" during the previous build, but marked as "Fixed, Needs Re-Testing" for the current build under test, will need to be regressed, or re-tested. Once the smoke test is completed, all resolved bugs need to be regressed. It should take between 5 minutes to 1 hour to regress most bugs.

Milestone Tests

Level 3 - Critical Path Tests

Critical Path test cases are targeted on features and functionality that the user will see and use every day.
Critical Path test cases must pass by the end of every 2-3 Build Test Cycles. They do not need to be tested every drop but must be tested at least once per milestone. Thus, the Critical Path test cases must all be executed at least once during the Iteration cycle, and once during the Final Release cycle.

Release Tests

Level 4 - Standard Tests

Test Cases that need to be run at least once during the entire test cycle for this release. These cases are run once, not repeated as are the test cases in previous levels. Functional Testing and Detailed Design Testing (Functional Spec and Design Spec Test Cases, respectively). These can be tested multiple times for each Milestone Test Cycle (Iteration, Final Release, etc.).
Standard test cases usually include Installation, Data, GUI, and other test areas.

Level 5 - Suggested Test
These are Test Cases that would be nice to execute but may be omitted due to time constraints.
Most Performance and Stress Test Cases are classic examples of Suggested test cases (although some should be considered standard test cases). Other examples of suggested test cases include WAN, LAN, Network, and Load testing.

Bug Regression

Bug regression testing should be reserved for major bugs at this time and for the final testing phase before release. This is due to the number of volunteers on the project and no dedicated test team. If a dedicated tester is brought on board, then more regression testing can be implemented, and this section will be amended. The entire team should work together before release on regression testing with a total list of bugs from the life span of the project development.

Testing Automation

Automated testing that will come at a later time will be amended to this section.

Bug Severity

These are levels of bugs reported to indicate their priority to the development team for correcting them. The dev team will label the bug as the following based on severity: Critical, High, Medium, Minor.

Critical	The application crashes or the bug causes non-recoverable conditions. System crashes, GP Faults, database or file corruption, potential data loss, program hangs requiring reboot are all examples of a critical bug.
High	Major system component unusable due to failure or incorrect functionality. These bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. These bugs can have a work around, but the work around is inconvenient or difficult.
Medium	Incorrect functionality of component or process. There is a simple work around.
Minor	Documentation errors or signed off bugs.

The following is the Priority rating for reported bugs.

Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.

Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it.
Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed.
Trivial	Enhancements/ Good to have features incorporated but are out of the current scope. May get moved to backlog for future releases.

Test Environment

Hardware: Once system requirements have been determined a range of machines should be tested that covers the minimum to maximum available specs. Until the specs are determined it will be tested on the available machines as used by the project volunteers.

Software: The applications should be tested on the three primary OS they are designed for; Microsoft Windows, Mac OS, Linux. Compatible OS versions to be updated once finalized. Jira for bug reporting and fixes. GitHub for version control and new builds.

Testing Workflow

* Once a system has been established for reporting bugs, this section needs to be amended with more details and instructions.
The following is the process to open a ticket on a bug. If part of the development team a bug will simply be added to Trello in following link: https://trello.com/b/kBWOJBra/ers-dev
Other departments will need a portal for opening a ticket. It can either be done in the general Trello at the following link: https://trello.com/b/KODUwfip/game-dev-design
Or through a ticket system offered by Trello as explained by the following link: https://trello.com/templates/engineering/bug-tracker-qW06fAZE

Bug reports must include the following: Reported by(name), date encountered, bug description, is it repeatable, if so, list steps, what was supposed to happen, what happened instead, platform tested on, version of application tested. Once submitted the dev team will assign a priority to the bug and create a new task to track in Trello.

More detailed way to format a report:
Hardware/ Operating System - List the testing environment hardware. e.g. PC, MAC/ Win X, MAC OS, etc.
Product – ERS or Eternal Souls
Version - Application version
Severity – Only DEVELOPER will update based on the defect
Resolution - Only DEVELOPER will update based on the defect
Assigned to - To be updated by Developer, for whom the ticket is assigned to

Priority - Set a priority based on severity. Only DEVELOPER will update based on the defect
Detailed Desc. - Details of the defect, bug or issue. Include details of steps to reproduce of if it cannot.
Upload file - Attach evidence file

Developing Tests For your Department

Departments working with ERS to further develop the engine as well as those utilizing it to develop Eternal Souls will focus on testing that relates on how they use the applications. The following are suggestions for tests as they might relate to various departments and what they would entail. The primary goal is to test the steps to do an action in the application as well as the end results of that action.
Game/Mechanic/Level Designers: Testing the scripting system as well as compilers or any other tool used to create the interactions utilized in the game.
UX/UI designers/researchers: Testing the steps and results of changing the UI, as well refining the steps required to do any task in the engine and or application.
Sound FX/Music Engineers: Testing the steps and results of sound applications related to game play and feedback.
Artists/ Animators: Test if the elements are represented correctly and if working in engine, the methods in which they are implemented.
Writers: Test for order of elements, tone, pace, and all related story elements are implemented correctly.
Other: For departments not listed here, you would still take a similar approach. Any interaction you have in the engine, you would test the steps in that process and look for bugs, as well as made sure the results performed as expected. These results are also that the intended outcome of your work is correctly represented in the final release.

Change Log

This section covers changes to this document made after the official approved original has been released. The format for change should be the following:
Date, Item changed or added, changed made by (Name).

Changes:

•