2012

# Space Shooter

## Test Plan

COP 4331: Processes for Object Oriented Software Development
Fall – 2012

Team 2
9/7/2012

# Table of Contents

# Prefatory Information

## Modification history:

| Version | Date | Who | Comment |
|---|---|---|---|
| v0.1 | 9/07/2012 | Andre Meireles | Preliminary (alpha) |
| v1.0 | 9/11/2012 | Joshua Thames | Formatting |
| | | | |

## Team Name:

Team 2

## Team Members:

| Name | Email address |
|---|---|
| Andre Meireles | andre.meireles@knights.ucf.edu |
| Alex Banke | banke@knights.ucf.edu |
| Christopher Margol | margol_chris@knights.ucf.edu |
| Chris Lin | christophercklin@gmail.com |
| Josh Thames | jthames88@knights.ucf.edu |
| Thaddeus Latsa | tlatsa@knights.ucf.edu |

## Definitions

CTD = “Crash to desktop”, results when a program exits abnormally.

## Section 1: Introduction

It is critically important for entertainment software to be free of bugs and faults. The purpose of a video game, like Space Shooter, is to provide entertainment in a simulated environment, which would take the player out of their natural environment and place him or her into the world we, the programmers, have created on the PC. Bugs, from poorly written code, distract the user from our end purpose and may manifest themselves in a variety of ways. Some of the bugs which could arise from a video game environment might be fatal to the process (resulting in a CTD), user interface glitches, misrepresented graphics, and all round unintended gameplay distractions. All of these types of glitches remind the user that he or she is still in fact playing a simulated shooting game on a computer and on earth and the programmer's mission has failed.

To ensure the release of a stable product free from common maladies, the developers shall routinely test the code in such a way that is time-effective and provides the development team with meaningful information that will aid in the eradication of software failures.

Reference Documents:

- Concept of Operations <include link here>
- Project Plan <include link here>
- SRS <include link here>
- <any other relevant documents; include full reference information or link>

## Section 2: Description of Test Environment

Hardware: Since Space Shooter is being developed in Java, the development team shall test Space Shooter in any hardware environment that supports the Java Runtime Environment (JRE). This includes, but is not limited to, name-brand desktop and laptop computers (Dell, Asus, Sony), as well as Apple desktops and laptops.

At this time, Space Shooter is expected to have low hardware requirements, so testing on netbooks and tablets is feasible.

Software: Space Shooter will be tested in the Windows, Macintosh, and possibly Android operating system environments. Linux support is not currently planned, so will not be tested.

Testers: The development team members will all be testing Space Shooter. Additionally, outside volunteers with different perspectives may be sought for testing purposes.

Final remarks: The testing environment in regards to both hardware and software is expected to be identical to the environment in which the software will operate after release.

## Section 3: Stopping Criteria

Software testing will be conducted as follows:

Method, function, or unit testing is to be conducted on each discrete component of the software as often as possible by the individual team member that is producing the component. For example, if Team Member A is tasked (by the software lead) to produce a Method B, Team Member A will be responsible for testing the method as far as the expected input and output bounds will allow. Once the component has been declared functional by the team member producing it, it will be sent to the software lead to be integrated into the final product.

Once the component has been integrated, the project itself will be tested for both the old functionality (before the new component was added) to ensure that nothing that was working before is broken, and new functionality (after new component). If testing is successful, further components (that have been tested) may be integrated into the project, and further testing may be completed.

Testing the software will be the responsibility of all group members. This may be as simple as playing the game and noting what bugs or errors are encountered. Once the software is in a playable state, every group member will have the same version of the software to take home and test prior to the next meeting OR software update.

The key idea here is to make sure that everyone is testing either their own components or the same version of the project at the same time, so that any reported bugs will be relevant to the current iteration of the project.

Once the time comes where everyone has to report what bugs they have encountered, serious issues such as CTDs take precedence over other issues, such as arithmetic errors. Fixing the known bugs shall be done in phases, with the higher precedence errors being fixed in the beginning phases, and culminating in minor bug fixes at the latter phases.

Test cases to use:

To be decided as development progresses, but generally: have broad test cases covering most common usage scenarios and specific test cases covering unusual usage scenarios. Test cases must also fall within preset bounds governing the input/output.

When is it “good enough to deliver”?:

When the entire game can be played from start to finish - without encountering any game-breaking bugs.

## Section 4: Description of Individual Test Cases

To be determined.