ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Team Mohan

Milestone 1

Samuel Kim, Brian Collins, Michael Williamson, Kevin Geisler

9/23/2011

Contents

Executive Summary

Developers currently do not possess any means of testing plugins for Minecraft [1], which makes debugging plugins a tedious process.

This document will outline and detail the problem that the *Liza* unit testing framework hopes to solve. This will be the first documentation involved in the project. Included in this document are the requirements, needs and features that Team Mohan has found to be important for the *Liza* project. It also contains some details on how we will approach this problem, and the beginnings of how we may solve it.

Introduction

Minecraft [1] is a sandbox computer game where players can create and remove blocks in a simulated world. These blocks can be arranged in a nearly unlimited number of ways and is only limited by the players imagination. Despite not being officially released, it has gained immense popularity with a user base exceeding 10 million people.

Modifying Minecraft [1] has become increasingly popular, as a number of changes can be made to suit each user's needs. There are currently two methods of modifying Minecraft [1]. Mods require a user to directly modify their game files in order to add or alter functionality in Minecraft [1]. For a server to effectively use a mod, the server requires each user to install that mod. On the other hand, plugins enable developers to make changes to Minecraft [1] without needing each user to directly modify the platform. Only the server needs to be modified.

A common way of introducing plugins to a Minecraft [1] server is to utilize a tool called Bukkit [2]. Bukkit wraps around the official server application and exposes an easy application programming interface (API) for developers to create plugins.

As recommended by Bukkit [2], many developers run Apache Maven [3] alongside the Bukkit server. Apache Maven [3] provides an effective solution for software project management. This includes being able to manage a project's build, as well as recording documentation.

Bukkit [2] currently does not possess any means of testing, which makes debugging plugins a tedious process. *Liza* intends to provide a unit testing framework for plugin developers to programmatically test their code.

Problem Description

Client Background

Tim Ekl and Eric Stokes are software engineers from Rose-Hulman Institute of Technology. Both are active and prominent developers in the Minecraft [1] plugin community. As developers, they understand the importance of creating a strong test base before deploying code, and have become frustrated with the lack of a proper testing framework for Bukkit [2]. If a successful testing framework were available, they would be able to easily test their plugins and those of other developers. They also would like to see the testing framework used by other people in the plugin community, so that more plugins will be of higher quality at their release.

Current System

Currently, there is no support for automated testing. Plugin developers load their code into a test server, and make sure functionality works manually. Doing thorough tests in this way, however, is extremely tedious. As such, some developers only run a few basic cases and therefore may miss some edge cases, or possibly conflict with functionality elsewhere. More daring developers may throw caution to the wind and load their plugins into active servers. This puts the stability of the server and its data at risk.

Users and Stakeholders

Profiles

Players – There are many people that play Minecraft [1] on modified servers. Poorly written and tested plugins can lead to server crashes or even loss of data, which may cause frustration to players. Plugins are made improve the experience of the players on a server, and poorly written plugins will instead create an unsatisfying environment.

Plugin Developers – The developers of these Minecraft [1] modifications are the most directly affected by this problem. The lack of an automated testing framework makes releasing stable and bug-free code much more inefficient. The technical background among developers varies – as some have a lot of programming experience and others have little – so a solution must be able to accommodate the difference in coding levels.

Minecraft [1] Server Administrator – Oftentimes a player or a developer, someone running a server has a particular interest in the stability of his or her server. Many server administrators are cautious to put a plugin onto their servers unless it has been confirmed to be stable. This means that the plugin has been tested thoroughly by other players, with few major bugs.

Bukkit Developers – The Bukkit [2] development team encourages the addition of features in *Minecraft* by making their plugin API public. However, Bukkit [2] will want to ensure that the testing framework interfaces with this API in an appropriate manner, and that the functionality of Bukkit [2] itself remains unaffected.

Mojang AB – Mojang [1] is the company that develops Minecraft [1]. Similarly to the Bukkit [2] developers, Mojang [1] will want a unit testing framework to remain separate from Minecraft [1]. It is a third party modification to the game and therefore unsupported by the company.

Tim Ekl and Eric Stokes – The two will play a major role in the development of the testing framework. As plugin developers themselves, they have a good idea of what the tests need to be able to achieve. In addition, they will undertake the task of maintaining the code once the project is complete.

User Environment

The testing framework will be separate from both Minecraft [1] and Bukkit [2] in the sense that it does not affect either program in any way. However, Bukkit's API will prove useful in communicating data between the testing framework and the server. The program is limited by the capability of Bukkit [2]. As a computer game, many of Minecraft's events are player driven. This means that *Liza* will need to integrate into Minecraft [1] as well, in order to emulate player driven control. Again, *Liza* will be limited to what Minecraft [1] is capable of (plus what the plugin aims to do, of course).

Key Needs

The Problem of...	- Bukkit [2] plugins go untested and therefore are unstable. - Importing plugins to test by trial and error is a lengthy process. -Testing code on a server with informative results is unimplemented
Affects...	Plugin Developers Server Hosts Players
And results in...	-Servers crashes or plugin bugs occur. -Players have a frustrating experience on a server due to frequent crashes and lag. -Developers spend more time debugging and constantly exporting their code to test on a real server.
Benefits of a solution	Reliable plugins Faster develop time Less server crashes Ease of mind

Developers of Minecraft [1] plugins tend to have problems testing their code. There is currently no way of testing their plugins in Minecraft [1] besides running it directly within the game's server. For larger plugins, identifying where issues occur becomes increasingly difficult. Plugins may crash servers if not tested thoroughly, which proves frustrating for both the players and the developers.

A possible solution to this is to create a new testing framework which implements the current Java 6 JUnit [4] testing. A user will be able to create a testing script which will be able to spawn a mock player. The mock player will then run the code written in the file and then listen for events which can be asserted to be true or false.

Product Overview

Product Perspective

Liza sits independent of both Bukkit [2] and Minecraft [1]. However, it is related to Bukkit [2], as plugin developers would be writing unit tests as they write code. Bukkit's API provides a list of events, which will prove useful for asserting desired behavior. However, Bukkit [2] will remain unaffected. The system must also be run programmatically, preferably using an integrated development environment (IDE) such as Eclipse or a software project manager such as Maven.

Elevator Statement

Minecraft [1] is a popular sandbox computer game. The multi-player portion has many 3rd party modifications – one of the most popular being Bukkit [2]. Bukkit is a means for plugin developers to create their code and integrate it with Minecraft [1] servers. However, any testing must be done manually, which is tedious and time consuming. Our team hopes to create a testing framework that eliminates this tedious and time consuming part of plugin development.

Capabilities

An optimal solution for a testing framework for Bukkit [2] will enable a developer to create a testing script that will call upon an API that will test a plugin made for Minecraft [1]. This framework will send and receive information though Bukkit [2] to allow a plugin to be tested in real-time with Minecraft [1] events.

Benefits	Features
Customizable unit testing for Bukkit plugins	Minecraft Unit Testing API
Live unit testing on a sever with Bukkit	Minecraft Unit Testing Framework

Assumptions and Dependencies

We are using the Bukkit API in order to write our testing framework. All testing and code, including Bukkit [2], is directly implemented on the Minecraft [1] server. Much of *Liza*'s code may depend on the structure of Bukkit's API.

Developers often use Maven [3] to compile and test their code. Our testing framework must be compatible with Maven [3] in order to be convenient for the developer.

Rough Estimate of Cost

Most of the software involved in this project (Java 6, JUnit 4 [4] , Bukkit [2]) is free. Minecraft [1] has an associated cost, but each person involved already owns a copy of the game. Because this is a student-developed project, there is no cost related to the development of this project.

Features

ID	Feature	Priority	Effort	Risk
1	Create a mock player	High	High	High
2	Communicate with Bukkit [2]	High	Med	High
3	Emulate player control through mock player	High	High	High
4	Listen for events	High	Med	Low
5	Incorporate JUnit [4]	High	Med	Low
6	Assert Entity/Block attributes	High	Med	Low
7	Create/Remove Entity/Blocks/Items	High	Med	Med
8	Send mock events	Med	Med	Med
9	Enable/Disable other plugins	Low	Med	High
10	Can detect test interference (from other players/entities)	Low	High	Med

- Create a mock player
    - Status: Approved
    - Priority: High
    - Effort: High
    - Risk: High
    - Stability: Medium
    - Reason: Most events in Minecraft [1] are player driven, so the testing framework should be able to create a mock player
- Communicate with *Bukkit* [2]
    - Status: Approved
    - Priority: High
    - Effort: Medium
    - Risk: High
    - Stability: High
    - Reason: *Bukkit* [2] provides a set of events to listen to, and being able to send/receive information with *Bukkit* [2] will prove invaluable for testing.
- Emulate player control through mock player
    - Status: Approved
    - Priority: High
    - Effort: High
    - Risk: High
    - Stability: Medium
    - Reason: A mock player needs to be able to do any action like a human player would

- Listen for events
    - Status: Approved
    - Priority: High
    - Effort: Medium
    - Risk: Low
    - Stability: High
    - Reason: Event listening will be a major component in asserting correct behavior
- Incorporate JUnit [4]
    - Status: Approved
    - Priority: High
    - Effort: Medium
    - Risk: Low
    - Stability: High
    - Reason: As a Java based project, JUnit [4] provides an existing base for asserting code output
- Assert Entity/Block attributes:
    - Status: Approved
    - Priority: High
    - Effort: Medium
    - Risk: Low
    - Stability: High
    - Reason: This feature allows *Liza* verify that an element in the game is at a desired state
- Create/Remove Entity/Blocks/Items:
    - Status: Approved
    - Priority: High
    - Effort: Medium
    - Risk: Medium
    - Stability: High
    - Reason: This feature allows *Liza* create and remove elements in the game, so that the certain Entities, Blocks, or Items, can be tested.
- Send mock events
    - Status: Proposed
    - Priority: Medium
    - Effort: Medium
    - Risk: Medium
    - Stability: Medium
    - Reason: This will allow the developer to simulate some events that may occur in the Minecraft [1] environment

- Enable/Disable other plugins
    - Status: Proposed
    - Priority: Low
    - Effort: Medium
    - Risk: High
    - Stability: Medium
    - Reason: Many servers operate using multiple plugins, which may interfere or conflict with the one being tested.
- Detect test interference (from other players/entities)
    - Status: Proposed
    - Priority: Low
    - Effort: High
    - Risk: Medium
    - Stability: Medium
    - Reason: This will allow the testing framework to detect if the mock player has been affected in any unintended way by an outside entity.

Solution Constraints

Source	**Constraint**	**Rationale**
Customer Preference	It must be open source.	Almost all plugins are open source. Also our clients will want to check our progress.
Customer Preference	It must be transferred via a version control system.	This will make it easily accessible by anyone.
Customer Preference	Program must run through console or command line. No GUI.	Clients prefer power over aesthetics
Systems	It must not modify the code of Bukkit [2] or Minecraft [1].	This may cause the system to crash and cause irreparable damage.
Systems	It must use Java 6 and JUnit 4 [4].	For compatibility purposes.
Systems	The test framework must not use the Minecraft [1] executable or a login to interact with a server.	This would add complication to running it, and putting login data into code would be risky for the user.
Systems	Must be capable of running on any platform, but particularly Mac and Linux.	Plugin developers run a wide variety of operating systems.
Systems	Is limited by what Minecraft [1] and Bukkit [2] can already do.	The framework cannot add any additional methods or classes that do not already exist on the Bukkit Server. If this was done it would create an unstable environment for the framework
Time	The project needs to be in a stable form by the last day of school in the spring.	We are working on this as students.

References

[1] Mojang AB. Minecraft. [Online]. http://www.minecraft.net/
[2] Bukkit. [Online]. http://bukkit.org/
[3] Brett Porter and Jason Zyl. Apache Maven Project. [Online]. http://maven.apache.org/
[4] Oracle. Java. [Online]. http://www.oracle.com/us/technologies/java/index.html

Index

Glossary

Bukkit [2] **–** A wrapper for the *Minecraft* server that exposes a user-friendly API.
Maven [3] **–** A software project management tool that builds and tests Java code.
Minecraft [1] **–** A sandbox computer game where players place and destroy blocks.
Plugin – A server-side modification to the game that alters the behavior of certain actions
Sandbox game – Refers to a style of game that involves an open world and no concrete directive