Test Plan

Version 1.0

April 10, 2007

Written By:

Sean Bosshardt
Brian Chong
Tom Munro
Jimmy Steorts

Table of Contents

Version Control

Revision	Author	Date	Change Description
0.1	Team	4/5/07	First draft of Test Plan – Through Deliverables by Task Phase
0.2	Team	4/10/07	First draft of Test Plan – Completed remaining sections
1.0	Team	4/10/07	Added estimates and math for bug fixing and finding

Game Concept

"Rules are meant to be fiddled with."

Patchwork Playgrounds is a party game where players are able to collect new rules and objectives as they roam an arena. These rules alter the context of the game and allow for endless hours of exciting gameplay. Objectives are multiplayer in nature, giving the player's extended interactive time. This creates a different play experience each time. Some of the objectives the players will be faced with could include but are not limited to: tag, king of the hill and dodgeball. Throughout the round, players will receive points for completing objectives, pushing opponents and hitting opponents with dodgeballs. At the end of the round, the points are tallied and the winner is determined.

Test Plan Objective

This document is intended to ensure the finest standard of quality in our project. We will be testing our game on the hardware provided by Vancouver Film School. The machines are HP Workstation XW8200's; Intel Xeon 2.8 GHz processor with 1GB ram, an NVIDIA Quadro FX 1300 video card, running Windows XP Professional SP2. We will also be testing our game for Xbox 360 controller support.

Patchwork Playgrounds is developed using the Unreal 2.5 engine. The specific areas of testing will include: gameplay, networking, presentation and audio.

Risks and Assumptions

Approach for Testing

Our testing methodology is an integrated testing process where we perform internal tests on a daily basis. *Insatiable Games* has a process in place allowing easy transfer of builds to streamline the testing process.

Insatiable Games will declare Alpha on April 18th. If the team determines the game is not ready for Alpha by April 18th then we will need to scope down the project. We will begin with scoping down the planned animations and levels. Planned animations include custom animations for each of the five characters. In the event that we do not hit our Alpha date, we will use the same set of animations on all characters. Planned levels include the Mountain and Winter Wonderland. In the event that we do not hit our Alpha date, we will cut the Winter Wonderland level and focus solely on the Mountain level.

All bugs found during the internal testing period will be deemed must fix bugs. The programmers on the project have scheduled time throughout to deal with these bugs. If all of the known issues are not fixed by April 18th then they will be flagged as not fixed and added to the bug database. This database will be analyzed and fixed based on importance during the scheduled bug fix time after each 3rd party testing session.

During Alpha we will open the project for outside testing. We will begin with our classmates. Once there have been 4 bug-free testing hours, we will open testing to the general public. There are four planned dates for brining in outside testers.

3rd Party Testing Dates
April 25th
April 30th
May 4th
May 9th

Build Cycles

Due to the integrated testing process, there will be a new build published every day. These builds will be available at any time during the day, allowing for new features to be smoke tested immediately.

Bug Estimates

Facts

Estimated lines of code (excluding comments) – 8,000
Estimated programming hours spent – 450

Avg. game lines of code – 200,000
Avg. game Function Points – 3,000[1]

Epic's Estimated SEI CMM Level – 3
~3 defects per function point
Our Estimated SEI CMM Level – 2
~4 defects per function point

Avg. hours per Function Point – 14.35
Avg. defects per Function Point – 5[2]

Function Points

$$\frac{Estimated\ Hours\ Spent}{Avg\ Hours\ per\ Function\ Point} = \frac{450}{14.35} = 32\ Function\ Points$$

$$\frac{Estimated\ Lines\ Of\ Code}{Avg.\ Game\ Lines\ of\ Code} = \frac{8{,}000}{200{,}000} = 4\%$$

$$Avg.\ Game\ Function\ Points * 4\% = 120\ Function\ Points$$

Defect Potentials

[1] Software Assessments, Benchmarks, and Best Practices: Capers Jones [Page 63 – Table 3.1 – Defect Removal Efficiency]
[2] Software Assessments, Benchmarks, and Best Practices: Capers Jones [Page 417 – Table 10.3 – Defect Removal Efficiency]

Avg. defects per Function Point = 5

$$\frac{(Avg.Of\ Epic's\ Defects + Our\ Defects)}{Function\ Points} = 3.5$$

Function Points	Defects Potential	Total Defects
32	3.5	112
120	3.5	420
32	5	160
		Avg. defects 230 defects

Note: We disregarded the 120 FP / 5 Defects sample because it skewed results significantly higher than any of the other samples.

Bug Estimate by Spine

Spine	Bugs Estimated
Audio	15
Gameplay	85
Networking	85
Presentation	45

Bug Find Rate

Our find rate will be approximately 12 bugs per day per tester. We reached this number by dividing the average defects by the amount of tester days we will be having.

$$\frac{Average\ Defects}{Tester\ Days} = \frac{230}{16} = \sim 12$$

Bug Fix Rate

95% [3]

In order to come to this number, we took the median of the average defect removal efficiency [91%] and the maximum defect removal efficiency [98%] because we believe that Epic Games (the developer of Unreal) is an SEI CMM level 3 company. This would give Epic Games a defect removal efficiency somewhere in between the average and the maximum. We further increased this number because we are building on an established middleware engine, not developing our own project from scratch.

Budget

Insatiable Games is assuming very minimal costs for the testing process. We estimate a cost of $30 per testing date. This will result in a budget of $120 for testing expenses.

[3] Software Assessments, Benchmarks, and Best Practices: Capers Jones [Page 417 – Table 10.3 – Defect Removal Efficiency]

Stakeholders, Roles and Responsibilities

Producer

Sean Bosshardt is the acting Producer and programmer for Patchwork Playgrounds. His responsibilities include: testing the daily current builds during the internal testing period, fixing all bugs found during the internal testing period, working with the Quality Assurance manager to fit the 3rd party testing sessions in with the overall project schedule, analyzing the bug reports from the Quality Assurance manager to determine how long it will take to fix each bug and to determine if the time required fits with the project schedule. If it will not and the bug is not deemed a must fix it will be flagged as shippable and remain in the game.

Quality Assurance

Brian Chong is the acting Quality Assurance Manager for Patchwork Playgrounds. His responsibilities include: testing the daily current builds during the internal testing period, setting up dates and times for bringing in 3rd party testers, managing the assumed QA Budget and putting together bug reports for the programmers after each 3rd party testing session.

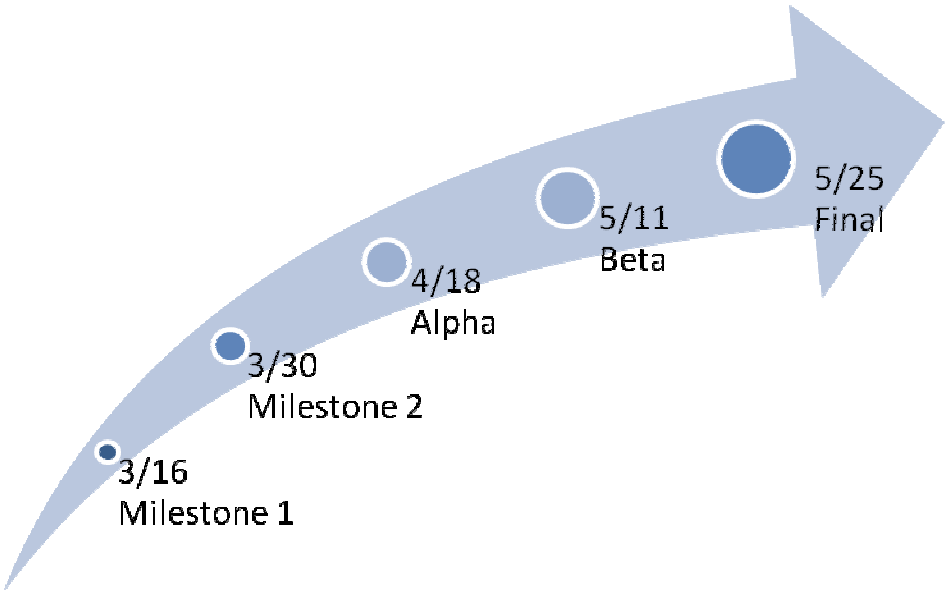
Developers

Jimmy Steorts and Tom Munro are the acting Developers for Patchwork Playgrounds. Their responsibilities include: testing the daily current builds during the internal testing period, assisting the Producer in the bug analyses to determine severity of the bugs and providing input to the Programmers as to possible solutions.

Project Schedule

Key Dates

These are the traditional project dates for developing video games. All of our key dates are determined by the Vancouver Film School Game Design program schedule and are thus unchangeable.

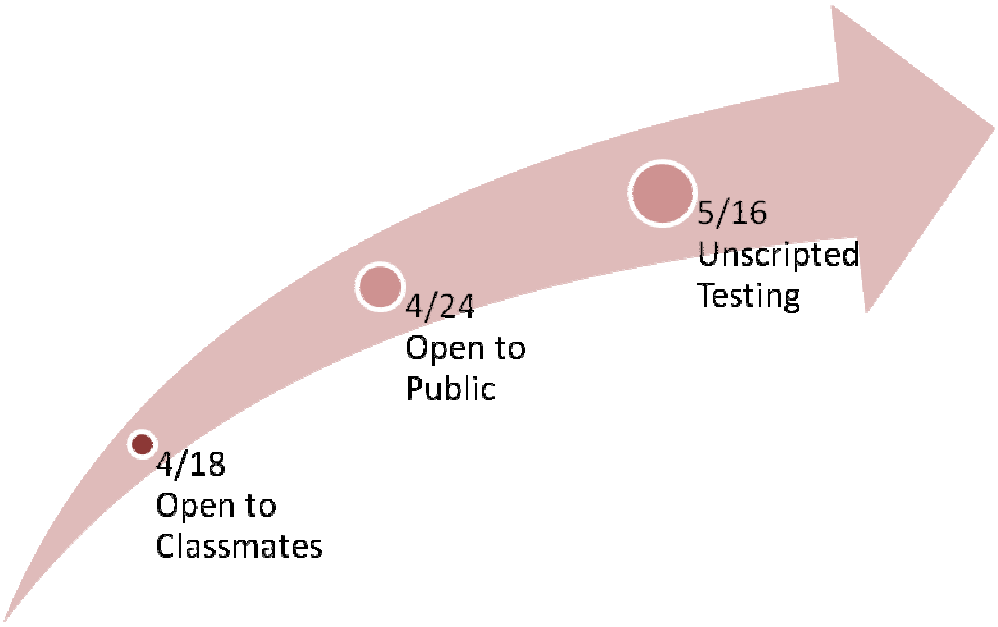


Interim Assessment Points

Insatiable Games will use interim assessment points for determining if the current state of the game is ready for entering these important points.

Open to Classmates is the period of time in which we will shift from internally testing to testing with our classmates

Open to Public is the period after which we determine that testing with classmates no longer produces any new bug findings. This will be declared after 12 hours of successful bug-free testing during the Open for Classmates time period.



Tasks by Phase

Deliverables by Phase

Each phase of our project we will have specific deliverables regarding our testing process.

Road to Alpha

This phase includes Milestone 1, Milestone 2 and Alpha.

During these milestones we will be internally testing. Our goal is to fix all of the problems we find during our daily testing immediately. If our bug list is not empty by the declaration of Alpha we will add the remaining bugs to the database that will be analyzed after 3rd party tests during the Road to Beta.

Road to Beta

Once Alpha has been declared we open testing to our classmates. During this time our goal is to fix all of the problems we find during classmate testing immediately. If our bug list is not empty by 4/25, our Open to Public date, we will add the remaining bugs to the database that will be analyzed after 3rd party tests.

Once Open to Public has been declared we will begin our 3rd party testing. This will include 4 testing sessions spaced apart evenly and containing different testers. The goal of these sessions is to find as many bugs as possible and to add them to a bug database. During the 5 days in between testing sessions *Insatiable Games* will analyze all bugs in the database to determine severity and priority. The programmers will then begin with the highest priority bugs and work downwards until the next testing session.

Road to Final

Once Beta has been declared the remaining time will be spent fixing all database bugs. If the bug database is not empty by 5/16 then we will place the remaining bugs on the list of bugs to fix if there is time. At 5/16 we will begin unscripted testing for any and all that wish to play our game. Bugs should not be found during this time, but in the event that they do they will be analyzed and added to the database.

During Unscripted Testing, the programmers will continue working on any known bugs. These will be fixed in the order of highest priority to lowest priority.

What Will and Will Not Be Tested

Spine Breakdown

This is a list and quick description of each of testing spines. The spines are explained in greater detail in the Features to Be Tested section.

- Networking – All networking, specifically Replication in Unreal
- Audio – All audio related to our game
- Presentation – Includes menu and HUD
- Gameplay – All general aspects of our game

Features to Be Tested

- Networking
    - Replication – Due to the multiplayer nature of our game there will be a lot of replication testing that must occur
        - Visuals
            - Animations
            - HUD information
        - Gameplay
            - Weapon switches
            - Power-ups
- Presentation
    - Menu Travelling
        - All of the menus will be tested so that each button does what it's supposed to and all combinations of menu travel will be attempted
        - Menus will be tested using both a keyboard/mouse combo and an Xbox 360 Controller
    - HUD
        - All of the portions of the HUD must be tested
        - This includes round time, score, current objective, portraits of all players and their position in the game
        - This also includes testing to make sure each character has their own specific color
    - Art

        - Consistency of art style
            - Check to make sure all art looks as if it belongs
        - Texture placement and alignment
            - Make sure there is no clipping or seems in textures
- Audio
    - All audio will be tested so each sound in the game is in fact a custom sound
    - The tester must be able to identify what they are hearing and compare it to what is happening
    - This includes in game music, in game sound effects and speech
    - It also includes menu music and menu sound effects
- Gameplay
    - General gameplay testing includes testing of animations, the throw mechanic, push mechanic, change weapons, grab bag, collision, objective and power-up pickups
        - Animations – test the look, feel and flow of each animation as the character moves around the map
            - This must be done using each character in turn so that all characters will be tested fully
        - Throw mechanic – The look and feel of the ball as it enters and leaves the characters' possession
        - Push mechanic – The look and feel of the push
            - How the animation looks as the character is pushing, the collision radius of the push itself and the character reactions
        - Change Weapons – How the character looks when switching weapons, the speed in which the weapon enters and leaves the players' possession
        - Grab Bag – The randomness of grab bags, the point values obtained when found
        - Collision – The collision of the characters and balls when interacting with the environment and each other
        - Objective and Power-ups – The balance of the objectives and power-ups in addition to the combination of all of the other aspects of Patchwork Playgrounds
            - Objectives
                - Each objective will be tested extensively on its own
                - This is done by playing only one objective constantly
            - Power-ups
                - Each power-up will be tested extensively on its own
                - This is done by spawning only one type of power-up in the level at a time

- This process will be repeated for each objective, effectively allowing a balance test to happen simultaneously

Features Not to Be Tested

- Computer Configurations
  - Our game will be tested using only the computer setup as previously mentioned
  - We will not have the resources nor the time to test all of the possible computer hardware configurations and operating systems
- Game Controller Support
  - No additional controllers will be supported beyond the Xbox 360 Controller

Tools

Code Based Tools

There are a few code applications that we will use in order to help facilitate testing.

- Trac SVN – This is a web based application used to backup revisions of files. Each revision can be looked at separately or compared to see what was changed and what may be causing the problems.
- Client Messages – Unreal has a function that prints messages to the screen, allowing the programmer to see exactly where in the code they are at any given time.
- Logs – These are used when Client Messages are not available, logging specific strings to an external log file.

Reporting Tools

The only reporting tool will be a form handed out and used by 3rd party testers. This form will be created in Microsoft Excel, giving the testers an easy medium in which they are able to log the bugs that they find.

Once completed, the forms will be categorized and assembled into a giant Excel database that will hold all of the bug information for easy access.

Success

Measures of Success

We have created a success percentage level for a number of categories. If our numbers fall within these successful measures we feel that the players will be immersed within our intended gameplay experience.

Estimate Accuracy

We feel that our estimations must be at least 85% of our final numbers in order to measure ourselves successful.

Ship/Defer Rate

We feel that we must fix 95% of bugs and deem less than 5% shippable in order to measure ourselves successful.

Invalid Rate

We feel that the bugs reported by our 3rd party must be 90% valid in order to measure ourselves successful. This means that we must educate our testers so that they are reporting things that we are looking for.

On Time and Budget

In order to be successful budget-wise we must spend less than or equal to the amount budgeted for testing as well as have a completed project by May 25th.