Kirby Banman
Jared Rewerts
Ryan Thornhill

ECE 493
Test Plan

8 April 2016

Test Plan

This is the test plan for the polyball project. It is designed to demonstrate conformance to the functional requirements of polyball as specified in the SRS document of February 8, 2016.

The white box section is a set of instructions for validation testers to follow. Each minor heading of the white box testing section is instructions to follow that corresponds to the functional requirement of the same name as found in the SRS. If the instructions may be followed all the way through, the test is considered passed.

The black box section is a set of implementation details that are critically important to the functionality of polyball on both its server and its client. The white box tests are designed to sweep through as much of the codebase as possible from a single points of entry, and verify that the final consequences of testing inputs are correct. In the case where a test could be carried out in a white box (automated format) we opted for that over a black box alternative in the interest of rapid development.

Table of Contents

Black Box Testing

This section covers the black box testing portion of our test plan. These tests are designed to be run by a tester in a “play testing” format. That is, many of the tests require the game to be played to be fulfilled.

Black Box - User Interface

Server Command Line Interface

Requirement: **3.1.2 Server Command Line Interface**
Pre-req: Server setup has been completed and all dependencies installed.

Action	Desired Outcome
Run “node polyball/Server.js”	Server is started without any errors. Server prints “Polyball server starting.”

Landing Page

Requirement: **3.1.3.1 Landing Page**
Pre-req: Server is started

Action	Desired Outcome
Navigate to server address	You are prompted to choose a nickname.

Spectating

Requirement: **3.1.3.2 Spectating**
Pre-req: A round has started

Action	Desired Outcome
Navigate to server address, pick nickname, press spectate	You can see the game being played.

Playing

Requirement: **3.1.3.3 Playing**
Pre-req: 2 other players have queued to play

Action	Desired Outcome
Navigate to server address, pick nickname, press play.	You are added to a 3 player game as a player.

Black Box - Server

Configure Server

Requirement: **3.2.1.1 Configure Server**

Action	**Desired Outcome**
Make changes to the configuration file. Start the server	Server should log the configuration values including those which you have configured..

Start Server

Requirement: **3.2.1.2 Start Server**
Pre-req: Server setup has been completed and all dependencies installed.

Action	**Desired Outcome**
Run “node polyball/Server.js”	Server is started without any errors. Server prints “Polyball server starting.”

Server Log

Requirement: **3.2.1.3 Server Log**
Pre-req: Server setup has been completed and all dependencies installed.

Action	**Desired Outcome**
Run “node polyball/Server.js”	Server is started without any errors. Server prints “Polyball server starting.”
Connect a client	Server should print out a log message that a client connected
Queue as a player	Server should print out a message that a spectator attempted to queue

Game Clock Synchronization

Requirement: **3.2.1.4 Game Clock Synchronization**
Pre-req: Server setup has been completed and all dependencies installed.

Action	Desired Outcome
Start a game with 3 players	Ensure that the clients start at almost an identical time when the round begins

Client Application Service

Requirement: **3.2.1.5 Client Application Service**
Pre-req: Client bundle has been built (JS and static files).

Action	Desired Outcome
With a browser connect to the polyball server	The application is shown on the screen correctly

Bidirectional, Real-Time Client Communication

Requirement: **3.2.1.6 Bidirectional, Real-Time Client Communication**
Pre-req: Server is started, 2 other players connected

Action	Desired Outcome
Start a game	Ensure paddle responds to mouse inputs and balls can be interacted with on the screen.

Client Identification

Requirement: **3.2.1.7 Client Identification**
Pre-req: Server is started

Action	Desired Outcome
Queue to play with a client	Make note of the number printed in the server console: “Client <number> requests to play.”
Queue to play with another client	Ensure that the number printed in the server console is unique from the first one

Game Model Simulation

Requirement: **3.2.1.8 Game Model Simulation**

Pre-req: Server is running

Action	**Desired Outcome**
Start a game	There is motion on the screen indicating that the game is running. (balls moving etc.)

Game Model Snapshot

Requirement: **3.2.1.9 Game Model Snapshot**
Pre-req: Server is running

Action	**Desired Outcome**
Start a game	There is motion on the screen indicating that the game is running. (balls moving etc.)

Game Rounds

Requirement: **3.2.1.10 Game Rounds**
Pre-req: Server is running

Action	**Desired Outcome**
Start a game with at least 3 players	
Let the round finish	A new round begins after a short intermission

Player Client Input

Requirement: **3.2.1.11 Player Client Input**
Pre-req: Start a round

Action	**Desired Outcome**
With 2 clients open apply paddle input to one of them.	Ensure the paddle moves on the “local client” and the “remote client”

Spectator Client Broadcast

Requirement: **3.2.1.12 Spectator Client Broadcast**

Pre-req: Round is started

Action	Desired Outcome
Open an additional client, select a nickname and choose “watch”	The game play can be seen from the client.

Player Queue

Requirement: **3.2.1.13 Player Queue**
Pre-req: Round is started with 3 players

Action	Desired Outcome
Open an additional client, select a nickname and choose “Queue”	Client’s name is shown in the player queue.
Open an additional client, select a nickname and choose “Queue”	Both names are shown in the queue on both clients
Let the round finish.	Upon the start of the next round both clients are brought into the game

Black Box - Client

Choose NickName

Requirement: **3.2.2.1 Choose Nickname**
Pre-req: Client connected to server with browser

Action	Desired Outcome
Choose “Watch” on greeting box	User can see game in progress, or waiting for players message if game is not in progress

Join Spectators

Requirement: **3.2.2.2 Join Spectators**
Pre-req: Client connected to server with browser

Action	Desired Outcome

Type name into name choice box	Greeting message changes to reflect new name.

Join Player Queue

Requirement: **3.2.2.3 Join Player Queue**
Pre-req: Client connected to server with browser

Action	**Desired Outcome**
Choose “Queue” on greeting box	Add to queue button is no longer clickable, user is added to game when round ends and/or sufficient players have queued
Choose “Watch” on greeting box, then click the add to queue button	Add to queue button is no longer clickable, user is added to game when round ends and/or sufficient players have queued

Join Game

Requirement: **3.2.2.4 Join Game**
Pre-req: Client queued to play

Action	**Desired Outcome**
Round ends and sufficient players are queued or already playing	User is added to game in arena with paddle when round starts

Arena Setup

Requirement: **3.2.2.5 Arena Setup**
Pre-req: A game is starting

Action	**Desired Outcome**
Round begins	An polygon arena is generated with a side for each player and a bumper between each side.

Paddle Setup

Requirement: **3.2.2.6 Paddle Setup**
Pre-req: Arena exists

Action	**Desired Outcome**

Round begins	User is added to game in arena with paddle between arena bumpers when round starts

Game Start

Requirement: **3.2.2.7 Game Start**
Pre-req: Arena and paddles exist

Action	**Desired Outcome**
Round begins	Round timer reads time remaining in round in MM:SS format

Ball Introduction

Requirement: **3.2.2.8 Ball Introduction**
Pre-req: Game has started

Action	**Desired Outcome**
Round begins	Balls are added once per second until there are an equal number of balls and number of players.

Move Paddle

Requirement: **3.2.2.9 Move Paddle**
Pre-req: Game has started

Action	**Desired Outcome**
Player has paddle, player moves mouse cursor while on game page.	Player’s paddle moves in the direction of mouse movement.

Goal Scoring

Requirement: **3.2.2.10 Goal Scoring**
Pre-req: Game has started

Action	**Desired Outcome**
Player strikes ball with paddle into the goal line of another player.	Player’s score increases by one.

Collecting Powerup

Requirement: **3.2.2.11 Collecting Powerup**
Pre-req: Game has started, powerup has spawned inactive as ball

Action	**Desired Outcome**
Player strikes ball with paddle into the powerup ball.	Powerup becomes active on that player.

Ending Game

Requirement: **3.2.2.12 Ending Game**
Pre-req: Round has started

Action	**Desired Outcome**
Round timer expires	Scores are fixed, top 3 scoring players are displayed.

Player Disconnect

Requirement: **3.2.2.13 Player Disconnect**
Pre-req: A round has started.

Action	**Desired Outcome**
Player disconnects, either by closing the browser window or losing internet connection.	Player’s paddle is removed from arena, score is no longer counted.

Round Intermission

Requirement: **3.2.2.14 Round Intermission**
Pre-req: Round ends

Action	**Desired Outcome**
Round ends and player scores are shown	Another game does not start for several seconds.

Powerup Activated

Requirement: **3.2.2.15 Powerup Activated**
Pre-req: Round is in session, powerup introduced to arena

Action	Desired Outcome
Player strikes ball into powerup with paddle	Player is awarded active powerup

Bullet Time Powerup

Requirement: **3.2.2.15.1 Bullet Time Powerup** and **3.2.2.18.5 Bullet Time Zoom**
Pre-req: Server has been started and minimum number of players have joined. Bullet Time powerup has been voted into arena.

Action	Desired Outcome
Activate the Bullet Time powerup	Balls “lock” at players goal. Particles fire out in a pulse. After a desired time, the balls are shot back out.

King Midas Powerup

Requirement: **3.2.2.15.2 King Midas Powerup** and **3.2.2.18.6 King Midas Sparkle**
Pre-req: Server has been started and minimum number of players have joined. King Midas powerup has been voted into arena.

Action	Desired Outcome
Activate the King Midas powerup	A trail of particles is left in the wake of activated balls. Balls are now worth double points. The player paddle has a particle effect designating it as the active King Midas user.

Black Hole Powerup

Requirement: **3.2.2.15.3 Blackhole Powerup** and **3.2.2.18.7 Black Hole Swell**
Pre-req: Server has been started and minimum number of players have joined. Black Hole powerup has been voted into arena.

Action	Desired Outcome
Activate the Black Hole powerup	A gravitational attractor has been added that warps ball trajectory. Users should notice a twist effect as well as a texture resembling a black hole in the middle of the arena.

Powerup Spawn

Requirement: **3.2.2.17 Powerup Spawn**
Pre-req: Server has been started and minimum number of players have joined.

Action	**Desired Outcome**
Spawn the powerup	A circle spawns in the arena with an icon relating the type of powerup it is.

White Box Testing

The format of the white box testing spec is simple:

- The top levels of indentation, like ***Synchronizer*** or ***Configuration***, represent major modules implemented in the polyball project.
- The next level of indentation represent individual operations performed within the modules. For example, the ***Synchronizer*** has a ***#synchronizeSnapshot*** operation.
- The final level of indentation represents the expected behaviors of that operation. For example, the ***Synchronizer*** operation ***#synchronizeSnapshot*** , should add balls to the model.

The following white box testing specification are for critical components of the polyball subsystems. The tests are categorized as integration tests and unit tests.

Note that these tests are are fully implemented in an automated test suite in the project folder "polyball/test/".

Integration Tests

These tests are designed to sweep through as much of the codebase as possible, like the ***Synchronizer*** module tests, which touch much the model and client side synchronization logic from top to bottom.

Synchronizer

#synchronizeSnapshot

√ should add balls to the model
√ should update balls in the model
√ should delete balls from the model

Server Comms

#newClientConnection

√ should add a spectator to the model (55ms)

#clientDisconnect

√ should remove a spectator from the model

#on(ClientToServer.vote)

√ should fire the ServerToServer.newVote comms event

Engine

#handleAddPlayerToQueue

√ should queue players when players < minPlayers
√ should start a game when enough players queue
√ should queue Spectators while a round is in progress

#initializeGame

√ Should add a paddle to each player
√ Should add an arena index to each player

Note that the following tests for the ***`Model`*** module are designed to test the isomorphic creation and deletion of each major model object. Hence, they are not full integration tests, but are instead small-scale integration tests, as they integrate the ***`Model`***, which is a subsystem of most of the rest of the game.

Model

#addOrResetArena

√ should add an arena to the model.
√ should reset the arena.

ball CRUD

#addBall
√ should add a queryable ball to the model.
√ should add a second, distinct queryable ball to the model.
#getBall
√ should get a ball by its id
√ should get a ball by any predicate
#getBalls
√ should get all balls when passed nothing or null
√ should get only a ball specified by a predicate

#deleteBall
√ should delete a ball and only that ball.
√ should delete a second ball.
#clearBalls
√ should delete all model balls.

spectator CRUD

#addSpectator
√ should add a queryable spectator to the model.
√ should add a second, distinct queryable spectator to the model.
#getSpectator
√ should get a spectator by its id
√ should get a spectator by any predicate
#getSpectators
√ should get all spectators when passed nothing or null
√ should get only a spectator specified by a predicate
#deleteSpectator
√ should delete a spectator and only that spectator.
√ should delete a second spectator.

player CRUD

#addPlayer
√ should add a queryable player to the model.
√ should add a second, distinct queryable player to the model.
#getPlayer
√ should get a player by its id
√ should get a player by any predicate
#getPlayers
√ should get all players when passed nothing or null
√ should get only a player specified by a predicate
#deletePlayer
√ should delete a player and only that player.
√ should delete a second player.
#addPaddleToPlayer
√ should add a paddle to a player
#getPaddle
√ should get a paddle for a player

Player Queue CRUD

#addToPlayerQueue
√ should add a single player to the queue.
√ should not add the same player twice.
√ should have playerQueue.length = 2 when 2 spectators added.

√ should increase the playerQueue by 1 for each added spectator.
√ should not allow clients that are players to add themselves to the queue
#removeFromQueue
√ should remove a single player from the queue.
√ should not throw an error when a player is removed that does not exist
#popPlayerQueue
√ should remove the top spectator from the queue.
#setPowerupElection
√ should add a powerup election.
#toSnapshot
√ should not contain client sockets

Powerup CRUD

#addPowerup
√ should add a queryable powerup
√ should add a distinct powerup
#getPowerup
√ should get a powerup by its id
#getPowerups
√ should get all balls when passed nothing
#deletePowerup
√ should delete a powerup from the list and remove from world (75ms)
#deletePowerup
√ should delete a powerup from the list and remove from world (59ms)
#clearPowerups
√ should delete all powerups from the list and remove from world

Unit Tests

These tests verify small units of code, like the ***PowerupElection*** tests, whose execution does not escape the model classes for ***PowerupElection***.

Configuration

#Construct

√ should read in a json file
√ should overwrite default values
√ should error on malformed JSON
√ should not allow min players < 3
√ should not allow round intermission < 1
√ should not allow max players < min players
√ should not allow powerups dir to be inaccessible

## Arena

#Arena(config)

√ should create bumpers in the correct spots with 3 players
√ should create bumpers in the correct spots with 4 players
√ should create goals in the correct spots with 3 players
√ should deserialize and serialize in a reversible way

BodyCollider

√ should pass through collisions on impulseCollisions:detected channel when in default state

#addIgnoredBody

√ should pass through collisions unignored bodies on impulseCollisions:detected channel
√ should filter collisions of ignored body regardless of which body is ignored

#removeIgnoredBody

√ should pass previously ignored collisions on impulseCollisions:detected channel

Powerup Election

#addVote()

√ should increase the size of the votes array by 1
√ should increase the size of the votes array by 1 for each added vote
√ should replace the current vote with new vote
√ should replace the current vote with new vote for each additional vote from same spectator

#removeVote()

√ should decrease the size of the votes array by 1
√ should decrease the size of the votes array by 1 for each additional call

#getWinner()

√ should return the powerup with the most votes
√ should return any powerup when there are no votes