Requirements and Analysis Document for BallBuster

Version: 3

Date : 2015-03-29

Author : Joakim Eliasson, Johan Segerlund, Jacob Lundberg, Matthias Andersson

1 Introduction

A computer game application used for entertainment. Users will be have access to a menu, which will let them change the settings of the game, and start the game itself. The game features a board with a set borders and obstacles in the form of blocks which prevent movement. Users will be in control of a ball, an entity which can roll around on the board. The game also features so called Powerups, which users can roll over (with their Balls) to acquire a temporary activatable ability.
The goal of the game is to destroy the enemys ball by ramming into them, bouncing them into a wall, eventually reducing their Shields to zero strength at which point the game is over.

This section gives a brief overview of the project.

1.1 Purpose of application

BallBuster is an entertaining two-player game inspired by an idea from DICE. It gives two players a platform to challenge each other.

1.2 General characteristics of application

BallBuster is a standalone java desktop application. It’s a multiplayer application with support for two players minimum, open for extension allowing more players.

The players share the same keyboard, each player are assigned different controls.

The two players a competing with each other, the goal is the get the other players shield down in order to win. This is possible by colliding with the other player where angle and speed is a factor.

In the middle of the field, superpowers will be frequently spawned. This can for instance affect the speed of the ball or restore parts of the shield.

The application will run on platforms that support java such as Mac, Windows and Linux.

1.3 Scope of application
The application will not support online functions. The application will not support more than two players. The user can either play against AI or against another player on the same device.

1.4 Objectives and success criteria of the project
1. The user can start a game together with another player.
2. The players can accelerate the ball, if they don't the ball will automatically stop.
3. The ball will take damage when the player hit the wall or collide with the other player.
4. The game is started via a menu, in which the user can specify which map to play on.

1.5 Definitions, acronyms and abbreviations
- Powerup - a usable item within the game, which can be acquired by players.
- Map - The board on which the users play.
- Ball - The entities controlled by the users.
- Shield - How many hits a Ball can take before loss? occurs
- Aura - An area, centered on the Ball, meant to cause an attractive or repelling force.
- Key - the keyboard button pressed to take an action

2 Requirements

In this section we specify all requirements

2.1 Functional requirements

Start game
- Choose map.
- Choose player keys.

Accelerate ball
- Use keys to accelerate the ball in a specified direction on the playing field. The key input will apply an impulse.

Activate aura
- Press a key to activate the ball’s aura.

Activate speed boost

- Press a key to activate a speed boost. Mana will be drained during the time the key is pressed.

Quit

- Use the mouse to exit program in the main menu

2.2 Non-functional requirements

2.2.1 Usability

BallBuster should be easy to use for everyone. After the initial setup the game will start. The GUI is supposed to be understandable without further documentation. Images and icons can be recognized from other applications, such as a heart image for a health pack. The game offers immediate feedback in the form of messages on the screen when a player collects a powerup or crashes in to a wall.

For more experienced users a number of extra features, such as harder maps, should be available.

2.2.2 Reliability

The application should always be available and never crash during normal flow.

2.2.3 Performance

The main purpose is to have the application running without delay for all users, therefore it is designed as a light weight application. No GPU or CPU demands are necessary.

Since no network support was added, the application does not send model information between clients.

2.2.4 Supportability

Automatic code testing of the model classes using JUnit and manual GUI tests during the development to ensure a highly functional application.

2.2.5 Implementation

The application was implemented using the libgdx framework, in addition to the Java Development Kit (version 1.8).

2.2.6 Packaging and installation
The application is cloned down from a GitHub repository and run using cmd. To be able to run the application a previous installation of Gradle is required on the computer.

2.2.7 Legal
The application does not come with any legal baggage, and if any such issues arise the user should be held fully responsible, with appropriate consequences to follow.

The application uses copyright protected images, but since we have no intention of selling the application or promoting it to the public we have not asked for any permission from the content owners.

2.3 Application models
In this chapter the use cases, model classes and the belonging UML will be discussed.

2.3.1 Use case model
Accelerate Ball
Start game
Choose AI
Choose map
Rebind key
Activate speed up
Activate Aura
Quit

2.3.2 Use cases priority

- Start game
- Accelerate Ball
- Quit
- Activate Aura
- Activate speed up
- Rebind keys
- Choose map
- Choose AI

2.3.3 Domain model

See figure 2

2.3.4 User interface

See figure 3

- The blue star in the middle: Power-up spawn location.
- Player one bottom left (white ball) and player two top right (black ball). The white ball have a blue shield that have taken a bit of damage. The black ball have enabled its aura which changes the color of the shield to yellow.
- White and black square-obstacles on the map can be used to repel or attract the players when the aura is enabled.

2.4 References

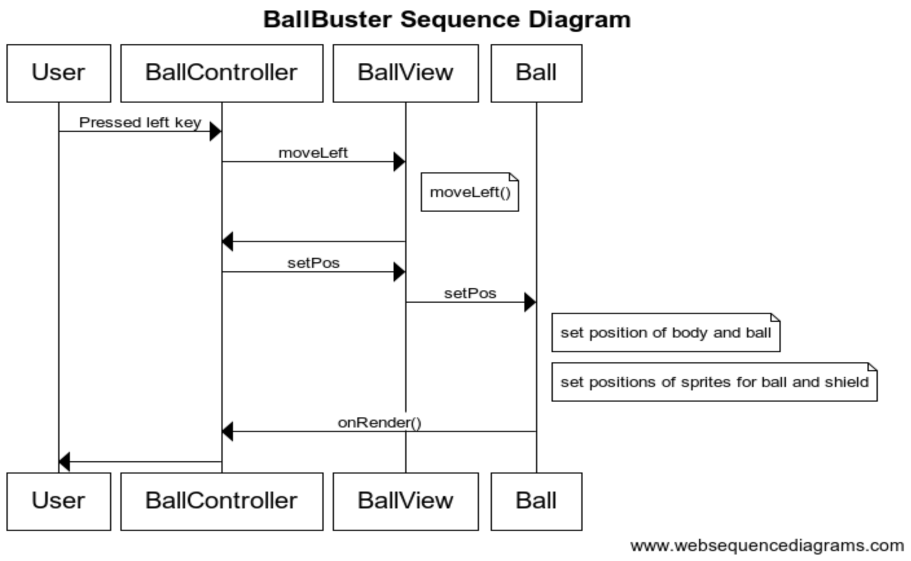
BallBuster Sequence Diagram
User
BallController
BallView
Ball
Pressed left key
moveLeft
moveLeft()
setPos
setPos
set position of body and ball
set positions of sprites for ball and shield
onRender()
User
BallController
BallView
Ball
www.websequencediagrams.com

Figure 1: Sequence Diagram for ballbuster

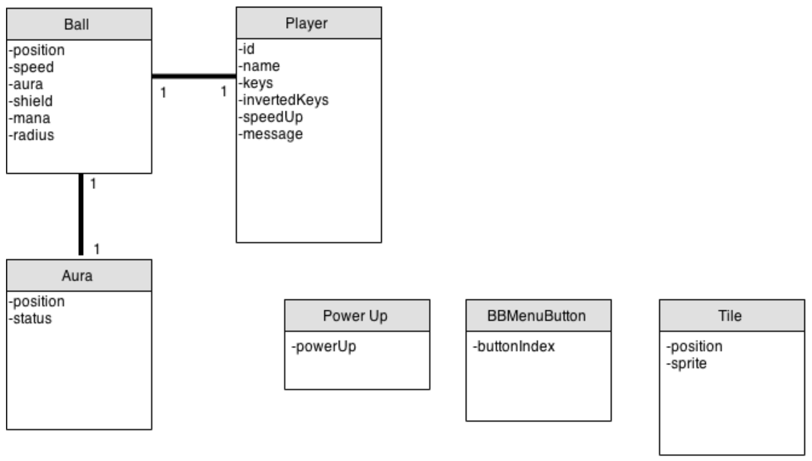

Figure 2: UML Class Diagram for domain model

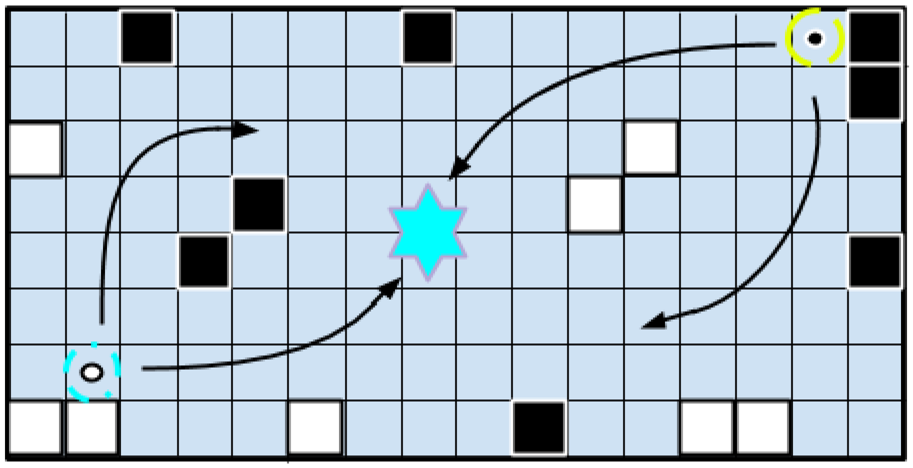

Figure 3: Initial mockup of the GUI