Emergent Architecture Design StandUp Game

f 22nd May 2015

Delft University of Technology

Nick Cleintuar	4300947
Martijn Gribnau	4295374
Jean de Leeuw	4251849
Benjamin Los	4301838
Jurgen van Schagen	4303326

Contents

1 Introduction

This document will provide insight into the architecture and the design of our product. It contains both the goals we seek to accomplish as well as an explanation of how the product is constructed.

1.1 Design Goals

In order to maintain the quality of the product during the development certain design goals are set. During the development these goals are kept in mind and should always be prioritised. These design goals are described in the next subsections.

1.1.1 Reliability

The product has to be reliable. This means that it should always be playable.

We guarantee the reliability of our product through testing. Testing is done using unit testing, integration testing, and real life testing. We also seek reliability by preventing mistakes made during development. This is enforced by programming in pairs as well as using version control with Pull request as well as static analysis tools: Checkstyle, Findbugs, and PMD.

1.1.2 Manageability

The product has to be manageable. Changes or extra features should be easily integrated in the product with minimal effort.

Manageability is obtained by using design patterns in the product. Design patterns provide structure to the project. Depending on the design pattern used it becomes really simple to make changes to the code.

Another approach to reach manageability is by using KISS. By keeping the product and implementations simple it becomes easier to change and add to it in the future.

1.1.3 Usability

The product has to be easy to use and have a minimal learning curve. This is important for new players to feel attracted to the game and prevent frustration by unclear instructions.

The game should be fun to play for most users. Finding what is fun and what is not is done by having real users test our game, as well as gathering their feedback. By doing this during the entire production, there are many chances to make changes or rethink the concept of our product.

2 Software Architecture Views

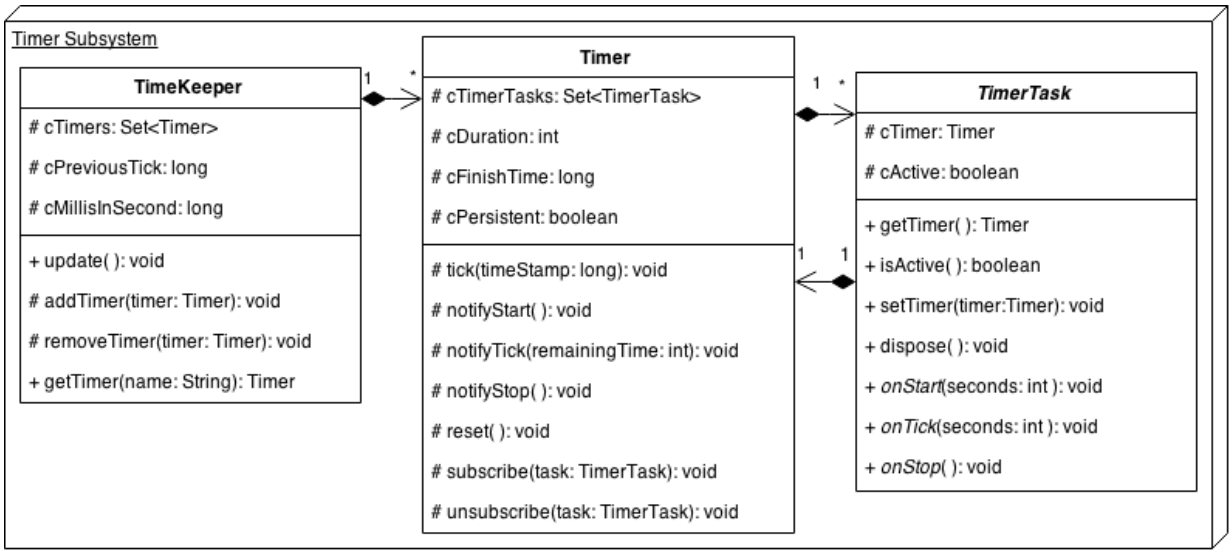
This section provides insight in the architecture of our product. It is divided in four paragraphs. The first paragraph explains how the subsystems of our product work together. In the second paragraph we dive into how the hardware and the software of our product work together. The third paragraph provides insight in how the data is stored and handled. The last paragraph we discuss where collisions between subsystems might occur and how we prevent them.

2.1 Subsystem decomposition

Our product consists out of multiple subsystems. Each subsystem is explained in its own subsection. After that the interaction between the subsystems is explained.

2.1.1 Timers

The UML below has some fields and methods missing to make it fit within the document. The most important methods and fields to understand how the system works are shown.



The Timer subsystem consists of three major components. The first part is the TimeKeeper. The TimeKeeper maintains a list of the Timers that exists within the application. The TimeKeeper calls the *update* method on every render cycle. Within the *update* the TimeKeeper checks to see if it has been a second since the last update. If that is the case all the TimeKeeper calls the *tick* on all Timers with the current system time.

Each Timer then checks to see if the timestamp is larger than the finish time. If the timer is finished it will notify all TimerTasks using the *onStop* method. If the timer is not done yet it will call the *onTick* including the time left that the Timer has to run before finishing.

The TimeKeeper can then perform tasks that are specified in its creation. Because the TimerTasks are created on the fly, they can still access methods and fields from where it is created. The TimerTask does however need to remember to which Timer it has subscribed in order to be able to unsubsribe.

2.1.2 Assets

This part will be written once implemented.

2.1.3 Graphical User Interface

Since the game mechanics are going to be refactored we decided to wait before writing this part.

2.1.4 Game mechanics

This part will be refactored in the next iteration. We left it out for now.

2.1.5 Interaction between subsystems

This part will be written once more subsystems have reached their final stage.

2.2 Hardware/Software mapping

For the product to be useful the user needs a device with access to a gyroscope and an accelerometer. These are needed to complete tasks and to measure activity. Hardware that is not necessary, but still very important to the functionality, is an internet connection and Bluetooth. These are used to communicate with other devices and the server. The server is used to store group information. In the subsections we will explain how the hardware is used in the software.

2.2.1 Gyroscope

This part will be written once implemented.

2.2.2 Accelerometer

This part will be written once implemented.

2.2.3 Internet

This part will be written once implemented.

2.2.4 Bluetooth

This part will be written once implemented.

2.3 Persistent data management

Data also has to be stored for the users. This is both user data as well as group data. For the user data local storage is used. For group data a server is used. These will now be explained in further detail.

2.3.1 Device storage

This part will be written once implemented.

2.3.2 Server storage

This part will be written once implemented.

2.4 Concurrency

Collisions between subsystems can occur on multiple levels throughout our product. On server level two users might try to change group specific data at the same time, this will be mediated by the database management system. Collisions might also occur within the application, when two subsystem would like to access the same recourse.

Glossary

KISS Keep It Simple, Stupid. 2

Pull request Pull request is a way to inform others of the changes you have made on a git repository and provides the opportunity for others to review your work and suggest changes. 2