Requirements and Analysis
Document for the Speedtype Project(RAD)

Contents

1 Introduction 1

1.1 Purpose of application 2

1.2 General characteristics of application 2

1.3 Scope of application 2

1.4 Objectives and success criteria of the project 2

1.5 Definitions, acronyms and abbreviations 2

2 Requirements 3

2.1 Functional requirements 3

2.2 Non-functional requirements 3

2.2.1 Usability 3

2.2.2 Reliability 4

2.2.3 Performance 4

2.2.4 Supportability 4

2.2.5 Implementation 4

2.2.6 Packaging and installation 4

2.2.7 Legal 4

2.3 Application models 4

2.3.1 Use case model 4

2.3.2 Use cases priority 5

2.3.3 Domain model 5

2.3.4 User interface 5

2.4 References 5

Version: 2.0
Date 2012-03-22
Author: Robin Hammaräng

This version overrides all previous versions.

1 Introduction

This section gives a brief overview of the project

11.1 Purpose of application

This project aims to create an Android game. The basic idea is that the game displays a word on the screen and the user is meant to type the word on his/her phone as fast as possible. The basic game mode is *Time Attack* where the user starts with a certain amount of time which counts down to 0 and for each word the player types correctly he/she is awarded a few extra seconds. The game is over when the counter reaches 0.

1.2 General characteristics of application

This game will be an Android, which easily is emulated on a PC, standalone single player application. It will feature a graphical user interface based on HTC Sensation phone but is scalable to all Android phones with software of Android 4.0 or newer. The goal is to release the application on Android Market.

The game is made for single player use only. When a game is started a word appears on screen, the player types down the word correctly on the built-in touch-keyboard within x amount of seconds. When the user types a letter correctly the letter displayed on screen turns green to provide visual feedback on the user progress. When the word is completed there's an animation that removes the current word from the screen and displays the next word. Also when the word is completed the player is rewarded with y amount of bonus seconds. After x amount of seconds, when the counter hits 0, the game is over.

1.3 Scope of application

Since the application is made for a single user only there's no need for a computer-player. The application will allow the player to pause a game; the software will also be able to do so if an incoming call is being made.

1.4 Objectives and success criteria of the project

1. It should be possible to play a game of Time attack mode with our own set of rules on HTC Sensation specifically.

2. It should be possible to play a game of Balance mode with our own set of rules on HTC Sensation specifically.

3. It should be possible to play a game of Falling words mode with our own set of rules on HTC Sensation specifically.

4. It should be possible to play a game of Scrabble mode with our own set of rules on HTC Sensation specifically.

5. All game modes should be possible to play in English, and should feature a design that is easy to extend into other languages as well.

1.5 Definitions, acronyms and abbreviations

All definitions and terms regarding the core Speedtype game are as defined in the references section.

- GUI, graphical user interface.

- Java, platform independent programming language.
- JRE, the Java Run time Environment. Additional software needed to run an Java application.
- Host, a phone where the game will run.
- Round, one game ending with game over or possibly canceled.
- Score, the score for the player during one round.
- Activity, a class that holds the GUI, comparable to a JFrame in Swing.
- Service, an application component representing an application's desire to perform a long-running operation in the background, without interaction with the user. I.e. background music.

2 Requirements

2.1 Functional requirements

The players should be able to;

1. Start a new game.
    a) Select a game mode (Time attack, balance, falling words or Scrabble).
2. Play the game by giving input to the application. This can be done by:
    a. Typing letters on the keyboard.
    b. Balancing the host unit by tilting it to the right/left (only in Balance mode).
3. After finished round, submit the score to a global high score list.
4. Change global application settings, such as volume and high score automatic submitting.
5. View global high score.
6. View and unlock local achievements.
    a. Achievements are unlocked by reaching specific goals, such as "Write 30 words in a row, without mistyping a letter".
7. Exit the application.

2.2 Non-functional requirements

2.2.1 Usability

Usability is highly prioritized. A user should be able to play at least a round of Time attack without having to read a manual or tutorial. For all game modes however, a brief instruction will be visible for the player just before starting the game. No further manual will be provided, since it will not be necessary. Test with at least two different non-programmers should be performed to verify usability. Test results will follow below.
The game should support the addition of new languages, simply via reading words from a different source.

2.2.2 Reliability

No sudden errors caused by the application should occur. Testing will be performed to ensure this, both using an emulator and an actual phone.

2.2.3 Performance

The game revolves around its performance, which needs to be very smooth and have a very short response time, which should not exceed 0.4 seconds in the worst case. Power saving actions will be taken during the life of the application, such as elimination of services when they are not in use.

2.2.4 Supportability

The game will be implemented so that the GUI can support more than one screen resolution. Tests will be made to ensure that the GUI is separated from the game logics, so that it is easily reused by future game modes.

2.2.5 Implementation

The application is set to run on Android phones, so the host should be an Android platform. It can however be simulated on a computer through an emulator. This is done by installing Android SDK and AVD in Eclipse, and then creating a new emulator in AVD.

2.2.6 Packaging and installation

Installation on the host will require an .apk file. When running the apk-file on the host it will automatically install and run like any other phone-application.

2.2.7 Legal

When the application reaches Android Market, some legal issues will be invoked. These are however not covered here.

2.3 Application models

2.3.1 Use case model

See APPENDIX for UML diagram and textual descriptions.

2.3.2 Use cases priority

TypeWord and StartGame will be our main priority since it's the foundation of the application.
There other use cases will come in second hand but all serve their purpose so we won't be able to leave any of them out.

2.3.3 Domain model

See APPENDIX.

2.3.4 User interface

The application will use a fixed GUI following Android conventions. To make sure it is runnable on most modern Android phones, the GUI is scalable to different screen sizes.

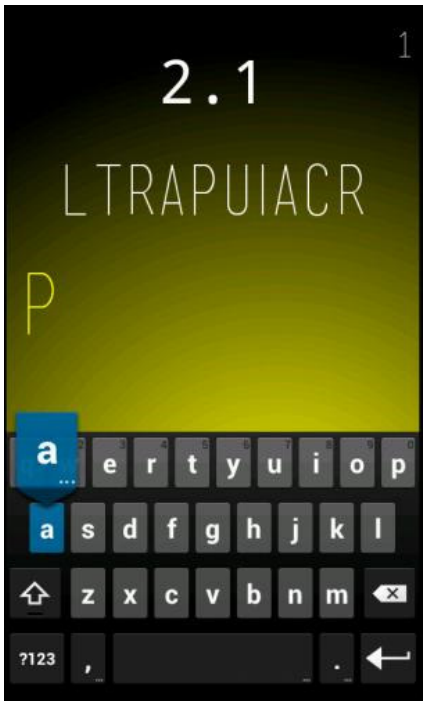
2.4 References

N/A

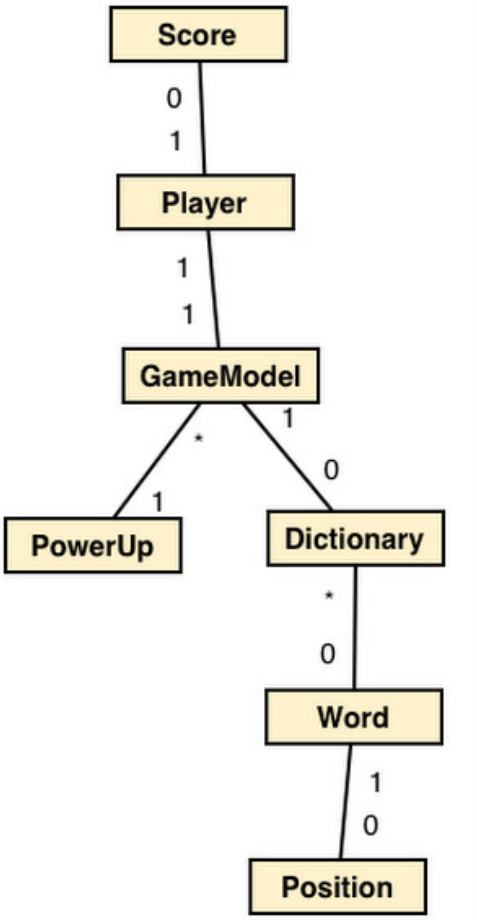
APPENDIX

Use cases, all possible cases

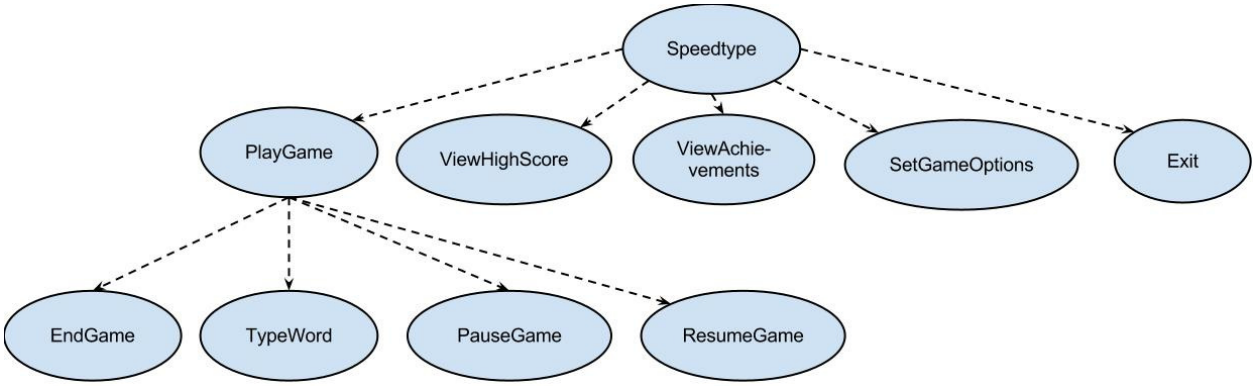
GUI



Domain model (See git repo: Images/ for higher resolution images)



Use cases (See git repo: Images/ for higher resolution images)



Use case: EndGame

Short description: How a player quits a running game.

Normal flow of events

Actor	System
Player clicks physical “back” button on the phone	
	The game pauses.
	A modal panel appears that asks if the player want to quit the ongoing game or not.
Player clicks “yes” or “back” button again	
	Game is ended, without saving any data.
	The menu is displayed.

Alternate flow: Player clicks “no” button in second actor event

Modal panel disappears and after a short delay, the game is continued.

Use case: Exit

Short description: How a user exits the application.

Normal flow of events

Actor	System
Clicks “Exit” button in menu	
	Application exits, without warning

Use case: PlayGame

Short description: How a player starts a new game.

Normal flow of events

Actor	System
Player clicks “new game” button	
	The view is changed to one that displays the different game modes available.
Player clicks on the game mode he/she wants to play	
	Starts a new game. An exited game will not be restored (see UC: EndGame)

Use case: PauseGame
Short description: How a player restarts a game.
Normal flow of events

Actor	System
Player clicks “Back” from the phone buttons.	
	Pauses the game.

Use case: ResumeGame
Short description: How a player resumes a game after it being automatically paused due to incoming call or system notification.
Normal flow of events

Actor	System
Player taps the screen to Resume the current game.	
	The pop -up is closed and the game is continued.

Use case: SetGameOptions
Short description: How to set global application options and in-game options.
Normal flow of events

Actor	System
User clicks “Options” from menu	
	The view is changed to one that contains the options.
	The user can change options by checking checkboxes and/or setting sliders to desired value.

Use case: TypeWord

Short description: How a user types a word in the time attack game mode.

Normal flow of events

Actor	System
System waits for user input to start game with an overlay.	
	Overlay disappears and the application displays a word
Player types the first letter of the word displayed	
	Highlights the first letter in the word by changing the color of it, given that the actor typed the correct letter.
Player types the rest of the word by doing the same as above until all letters have been typed.	
	Highlights the last letter so that the entire word is highlighted. The word is removed from the view and a new word is displayed. Points will also be given.

Alternative flow: Player does not type the correct letter. The letter in question will then very briefly turn red, and then back to white.

Use case: ViewAchievements

Short description: How to view the list of achievements.

Normal flow of events

Actor	System
User clicks “Achievements” from menu	
	The view is changed to one that contains the Swarm interface.
	The user can see the list of achievements, both the ones already unlocked, and the ones that are still locked.

Use case: ViewHighScore

Short description: How to view the global high score list.

Normal flow of events

Actor	System
User clicks “High score” from menu	
	The view is changed to one that contains the Swarm interface.
	The user can see the global high score list and his/her own placement on the list, given that she has submitted at least one score to it.