Software Requirements Specification

Elie Canonici Merle, Colin González, Sylvain Ribstein, Kevin Sanchis, Anas Aarab, Paul Laforgue

October 17, 2016
Last updated : January 7, 2017

Abstract

This file provides an overview of the requirements for the project of the course POCA 2016.

Introduction

Purpose The goal of the project is to develop a framework in SCALA to implement games such as pokemon-go (i.e a MMORPG plus augmented reality). The project is open source.

Definitions Augmented reality : the concept here is still vague, the main importance is to make use, eventually, of a real life map, i.e. *à la* Google Maps, in which a player is able to move through the World.
MMORPG : Massively Multiplayer Online Role-Playing Game. We expect the software to manage several (to be determined later) users at a time.

System overview In this document we detail the specification of the underlying engine and the client of the software. The engine represents an abstraction of the game, the gameplay and its features. On top of the engine a more precise implementation is to be added. The implementation provides the setup, as well as a gameplay and the features expected by the player. The engine itself is a server application able to manage the clients (players) connected on it. The server is expected to resolve concurrency issues to ensure safety and liveness in order to prevent lost updates as well as insuring coherence between players. For instance, the server is responsible of the information given to the player. More precisely, if a token (i.e. a Pokémon) is spawned in a location l, then all the clients (players) near enough to the position l must be able to interact with the tokens. On the other side the client is the part of the sofware that allows a user to interact with the server application. The client provides a user interface and simple documentation used as an accessible goto reference for any kind of user. The developpement of such software, both client and software, is supposed to be scalable. Therefore an object oriented and modular paradigm is needed.

Contents

1 Overall Description

1.1 Product Perspective

The software presented in this document is a MMORPG game. The game will use virtual reality features to provide a realistic experience of gameplay for its users. This videogame is mainly inspired by the Pokémon Go mobile game. In Pokémon Go a player goes on the hunt after Pokémons everywhere, possibly all over the World. The player uses Pokeballs in order to capture such creatures. The goals are to capture all existing kind of Pokémons and conquer and hold particular spots in the real world. Pokémons are trainable creatures, they are put to fight by their trainers in such way that the player earns experience from won battles. Players are part of a team so that players from the same team can help each other during fights. The software is expected to have similar features.

1.2 Design Constraints

The software must absolutely respect the specifications on figure **??**.

1.3 Product Functions

The software must ensure the functions on figure **??**.

1. The software provides a means to interact with virtual characters.
2. The software functions in real-time.
3. The software is secure.
4. New players can register to game.
5. The software manages the conection of clients to the server.
6. The software ensures a dynamic gameplay.
7. The game difficulty is low and mostly based and random events and progression by regular sessions of gameplay.
8. The software may offer interaction with other between players.

Figure 2: Product Functions

1.4 User characteristics

The user has to be connected to the server (and by extension to internet) in order to play the game. There is no offline version. Since this MMORPG is based on augmented reality, the user needs to use a device allowing him to be located and to move freely, for example a smartphone.

1. The game is about augmented reality Pokémon-like characters with statistics about their level, health points and other statistics.
2. A map like model allowing geolicalisation. In the first version a simple model is acceptable. However it is desirable to add Google Maps like World Map.
3. The game makes use of augmented reality or real life objects (i.e. Pokéballs, Pokéstops, points of interest, Pokémon centers and the like).
4. Interaction between several players.
5. A modular architecture in order to provide extra features using extensions.
6. The software must be implemented using the Scala language.
7. The software must respect a distributed architecture using a server-client architecture.
8. A protocol for the server-client must be defined and used mandatorily. No off protocol communications should be accepted.
9. The synchronization of the client is achieved through the server part.
10. The engine must not need high resources (CPU and memory) so that it is playable in a variety of devices.
11. The software is easy to use by any sort of player.
12. Optmizations are desireable. As an example, characters should not be spawned in useless places. Particularly, a place in the world with no players near enough.
13. The client software doens't store the user data, the client retrieves it when the user signs in.

Figure 1: Design Constraints

2 Specific requirements

2.1 Performance requirements

The application has to be constantly connected with the user and remember its current information. In that sense, if the client-side crashes, the application should be able to restore its previous state. The player should have the possibility to look for its progression and so access databases at anytime.

2.2 Logical database requirement

The database should distinguish between static (age of the user,..) and dynamic informations (its current location,..).
Briefly, the database will need to store numerous informations about the:

- **Users**: name, age, contacts, friends, inventory, current and last locations...
- **Pokemons**: name, race, characteristics, color...
- **Items**: name, price...

An adequate object-relational database choice would be: PostgreSQL. It's open source and has a good reputation. The relational model has to be scalable, and should be able to support a lot of players. Also, it should be easily extended.