

Mobile application development

PCRemote - documentation

Table of contents

[1. Introduction 5](#_Toc352873656)

[1.1. About project 5](#_Toc352873657)

[1.2. Purpose 5](#_Toc352873658)

[2. Technologies 6](#_Toc352873659)

[2.1. Why these technologies? 6](#_Toc352873660)

[2.2. Android 6](#_Toc352873661)

[2.2.1. Basics 6](#_Toc352873662)

[2.2.2. Notifications 8](#_Toc352873663)

[2.2.3. Project structure 10](#_Toc352873664)

[2.2.4. Framework 10](#_Toc352873665)

[2.2.5. Unit tests 10](#_Toc352873666)

[2.3. Windows Phone 10](#_Toc352873667)

[2.3.1. Basics 10](#_Toc352873668)

[2.3.2. Notifications 11](#_Toc352873669)

[2.3.3. Project structure 11](#_Toc352873670)

[2.3.4. Framework 11](#_Toc352873671)

[2.3.5. Unit tests 11](#_Toc352873672)

[3. Project analysis and design 11](#_Toc352873673)

[3.1. Functional requirements 11](#_Toc352873674)

[3.1.1. Android client 11](#_Toc352873675)

[3.1.2. Windows client 12](#_Toc352873676)

[3.1.3. Server 12](#_Toc352873677)

[3.2. Non-functional requirements 12](#_Toc352873678)

[3.2.1. Clients 12](#_Toc352873679)

[3.2.2. Server 12](#_Toc352873680)

[3.3. User stories 12](#_Toc352873681)

[3.3.1. Server 12](#_Toc352873682)

[3.3.2. Android client 13](#_Toc352873683)

[3.3.3. Windows Phone client 13](#_Toc352873684)

[3.4. Diagrams 14](#_Toc352873685)

[3.4.1. Server class diagram 14](#_Toc352873686)

[3.4.2. Server use case diagram 16](#_Toc352873687)

[3.4.3. Server sequence diagram 16](#_Toc352873688)

[3.4.4. Android client class diagrams 17](#_Toc352873689)

[3.4.5. Android client use case diagram 17](#_Toc352873690)

[3.4.6. Android client sequence diagram 17](#_Toc352873691)

[3.4.7. Windows Phone client class diagrams 17](#_Toc352873692)

[3.4.8. Windows Phone client use case diagram 19](#_Toc352873693)

[3.4.9. Windows Phone client sequence diagram 19](#_Toc352873694)

[3.5. Layout design 19](#_Toc352873695)

[4. Implementation 19](#_Toc352873696)

[4.1. Network messages 19](#_Toc352873697)

[5. Summary 20](#_Toc352873698)

[6. Dictionary 20](#_Toc352873699)

[7. Bibliography 20](#_Toc352873700)



Mobile application development

PCRemote - documentation

# Introduction

## About project

The whole project consists of 3 applications: server written with C++ and Qt Framework that will run on Windows operating system and two clients, one for Android and one for Windows Phone.

Clients connect with the server over the network via TCP (for general data transfer purposes) and UDP (for mouse movement requests, since UDP doesn’t perform any data correction checks, therefore doesn’t generate any delays) and will take the control over the cursor and the keyboard.

## Purpose

The purpose of the project is to ease user the access to the computer when showing a multimedia presentation.

Using just pc mouse and keyboard might often be unhandy when giving the presentation. The user must stay in front of the computer at all times to switch slides and show additional content on the screen, what makes it difficult to make contact with people that listen to the presentation. PCRemote allows to avoid this necessity. The project allows to take the full control over computer cursor, keyboard, media keys and even files stored on the computer from mobile device that can be simply carried around the room. This way the user can get closer to his listeners and is no longer bound to one place.

# Technologies

## Why these technologies?

You might wonder, why among non-functional requirements for server you can find Windows, but not Linux even though it’s supposed to be written with Qt Framework which is multiplatform framework for C++. Well, the problem is, that even though Qt Framework makes it much easier to write applications with graphical user interface, it still doesn’t provide all the features required by the project, such as simulation of mouse or keyboard button click. What’s more, Linux native libraries do not provide this functionality either while WinAPI does.

Then why is it written in C++ at all? It’s because C++ is very efficient language and efficiency for background applications is very important. You certainly don’t want any background application that doesn’t do anything for most of the time (unless you ask it to) to use up 20 MB of memory. What’s more applications written in C++ are totally stand-alone. Many potential users of the application have very little experience with usage of computers and satisfying a requirement of installing for instance JRE before application can run, might be unachievable task for them (especially since nobody likes reading installation instructions).

As for clients, Android was chosen because it’s currently the most popular mobile operating system, and even though version 2.2 is already pretty old, it’s still the most popular version of Android.

Windows Phone recently started gaining on popularity but yet there are very few native applications for this operating system, what means, there are very few competitive applications.

Why not IOS then? It’s also very popular after all.

Well, there are two reasons for that:

1. In order to code applications for IOS, you need Mac, which I don’t have
2. I don’t know anyone, that could lend me an iPhone for testing
3. I don’t like Apple

## Android

### Basics

In Android operating system there are 3 most basic entities: Activities, Services and Widgets. PCRemote actually uses Activities only, as they’re most commonly used in Android applications as well as the other two do not really provide any useful functionality for the project.

Activity is the most basic view the user can see right after starting the application. It may contain all kind of controls like buttons, labels and pictures. Single app usually contains a multiple of Activities. All Activities are removed from memory right after the application gets closed.

An Activity has a following lifecycle:



Image 1. Android: Activity Lifecycle

http://developer.android.com/reference/android/app/Activity.html

An example of activity:

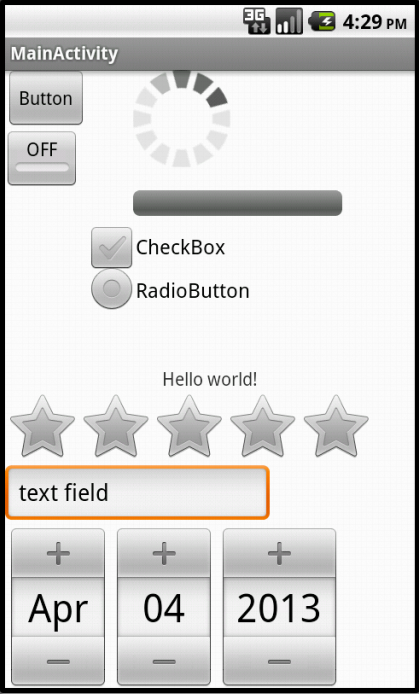


Image 2. Android: Activity Example

Widgets are the functional elements of application that can be easily added straight to the main desktop of the operating system. Unlike Activities, widgets are active at all times, provided that the user actually puts them on the desktop.

Then there are services. Services are nothing more than a kind of processes that may get started along with the application and then remain turned on even when the application itself is closed. They are most commonly used for data synchronization with a remote server. This way even when the application if turned off, they can show the user some notifications, for instance about new email waiting in a mailbox.

### Notifications

In Android there are basically two different kinds of notifications: Toasts and top screen notifications.

Toasts are usually small pieces of text shown at the bottom of the screen for just a few seconds. Only one toast message can be shown at a time. If application tries to show two or more toasts at one time, they will be queued and shown one after another, each for the specified amount of time.

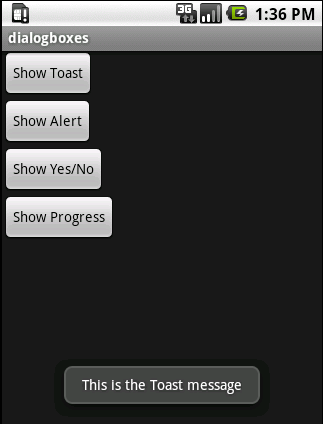


Image 3. Android: Toast Example

The other notifications can be shown at the top of the screen. These are most often created by services whenever they generate some data worth presenting to the user.

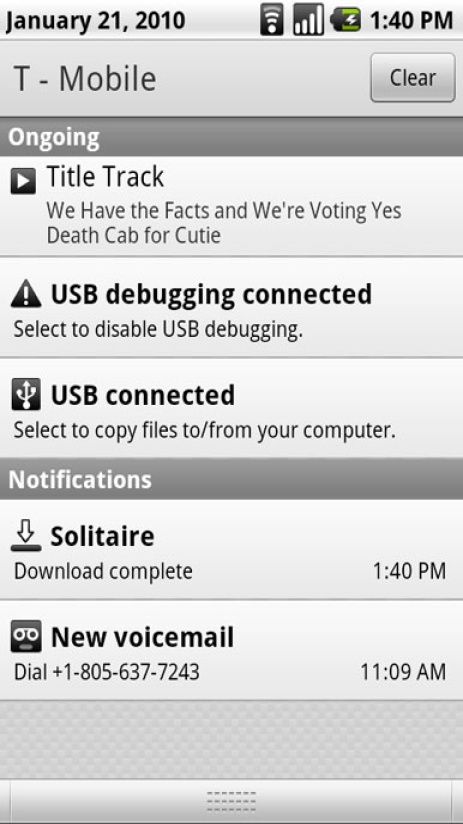


Image 4. Android: Top Screen Notifications example

Unlike toasts, top screen notifications can contain multiple different Views that may present multiple kinds data as well as control the application itself. Also they may be connected to some Activity. This way when user clicks one of the notifications, one the application activities, that would present more data concerning particular notification, may be started.

### Project structure

### Framework

### Unit tests

## Windows Phone

### Basics

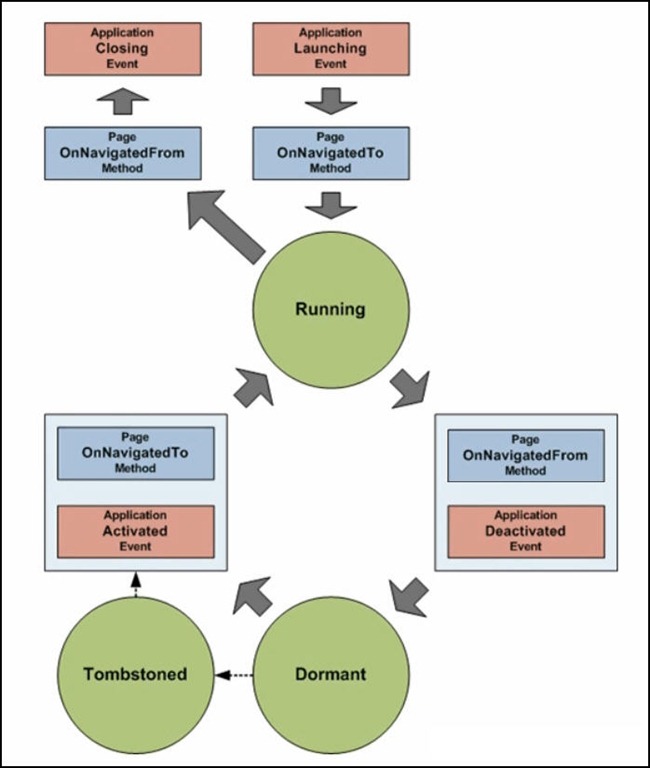
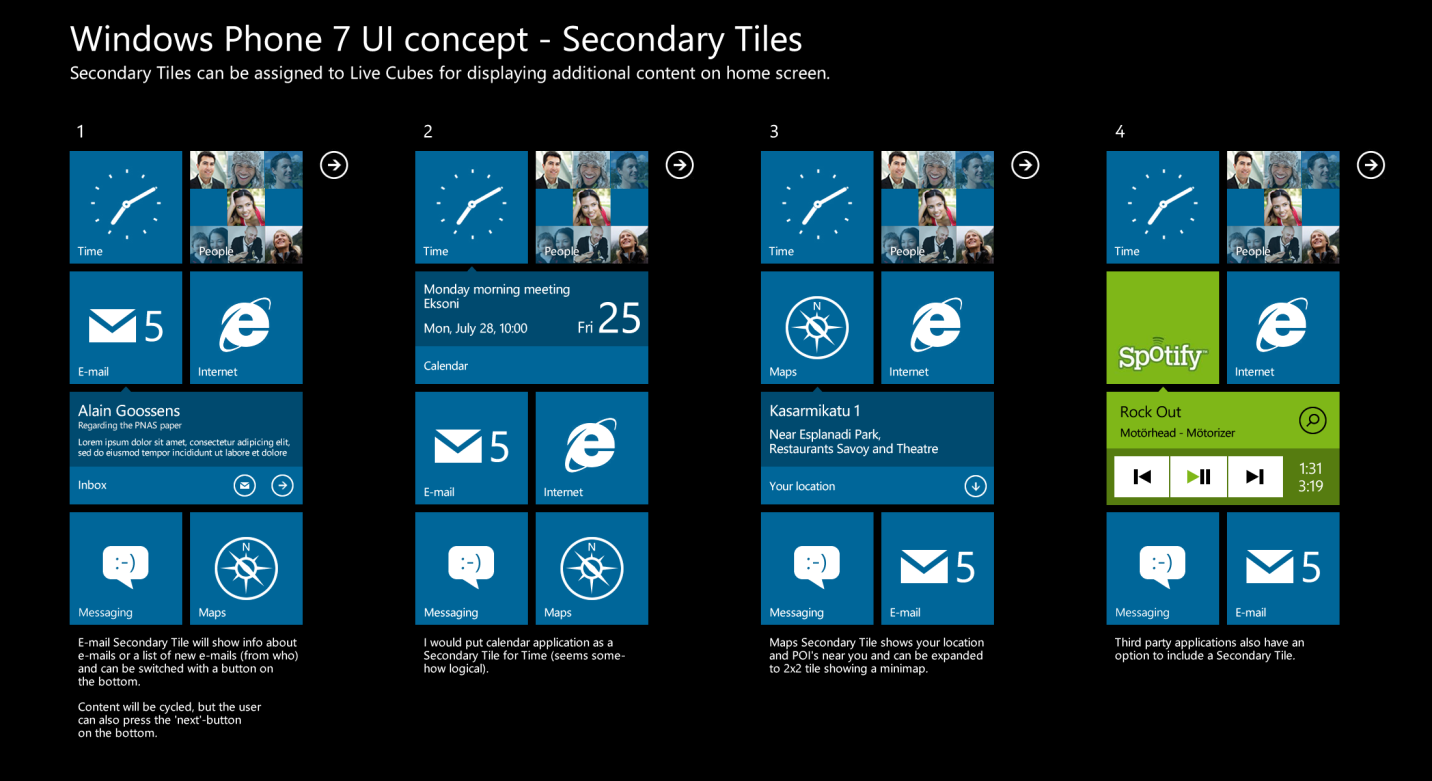


Image 5. Windows Phone: Page Lifecycle

***http://jesseliberty.com/2011/07/27/page-statestep-by-step/***

### Notifications

Image 6. Windows Phone: Tiles

http://www.concept-phones.com/nokia/windows-phone-7-concept-ui-nokia-relies-cubes-brilliant/

### Project structure

### Framework

### Unit tests

# Project analysis and design

## Functional requirements

### Android client

* + - 1. It is possible to use mobile device like a touchpad to control cursor and scrolls
      2. It is possible to use mobile device to control the keyboard (all keys that can be found on English keyboard)
      3. It is possible to add and store data about a new server
      4. It is possible to connect to the server
      5. It is possible to choose sensitivity of a cursor
      6. It is possible to define the time of a long click (how long the control needs to be pressed to call it a long click)
      7. It is possible to define sensitivity of a scroll
      8. It is possible to control media keys
      9. It is possible to download files from computer to mobile device

### Windows client

* + - 1. It is possible to use mobile device like a touchpad to control cursor and scrolls
      2. It is possible to use mobile device to control the keyboard (all keys that can be found on English keyboard)
      3. It is possible to add and store data about a new server
      4. It is possible to connect to the server
      5. It is possible to choose sensitivity of a cursor
      6. It is possible to define the time of a long click (how long the control needs to be pressed to call it a long click)

### Server

* + - 1. Server notifies of each new client connection
      2. Server notifies of each client disconnection
      3. Server processes all client requests concerning cursor movement or key press
      4. Has a tray icon that allows to control the application
      5. It is possible to change default port for listening
      6. It has an option for closing the application from tray icon menu
      7. It can restrict access with password for Android clients

## Non-functional requirements

### Clients

* + - 1. Wi-Fi connection to the internet
      2. Either Windows Phone 7 or Android (at least version 2.2) depending on a version of client application.

### Server

* + - 1. Windows Vista / 7 / 8 operating system
      2. Wireless internet connection

## User stories

### Server

* + - 1. As a user of a server, I want to access all its features through a tray icon.
      2. As a user I want the server to support at least all basic functions of controlling my computer, such as full control over the cursor, just the way I control it with touchpad, control over keyboard and basic media keys (play, previous, next).
      3. As a user of a server I want to be able to set it up, so that it refused connections from certain clients.
      4. As a user I want to server to be able to provide me instructions of how I should use it and its clients.
      5. As a user I want to be able to choose whether my server can be detected automatically by a client.
      6. As a user of a server I want to be able to restrict access to it with a password.

### Android client

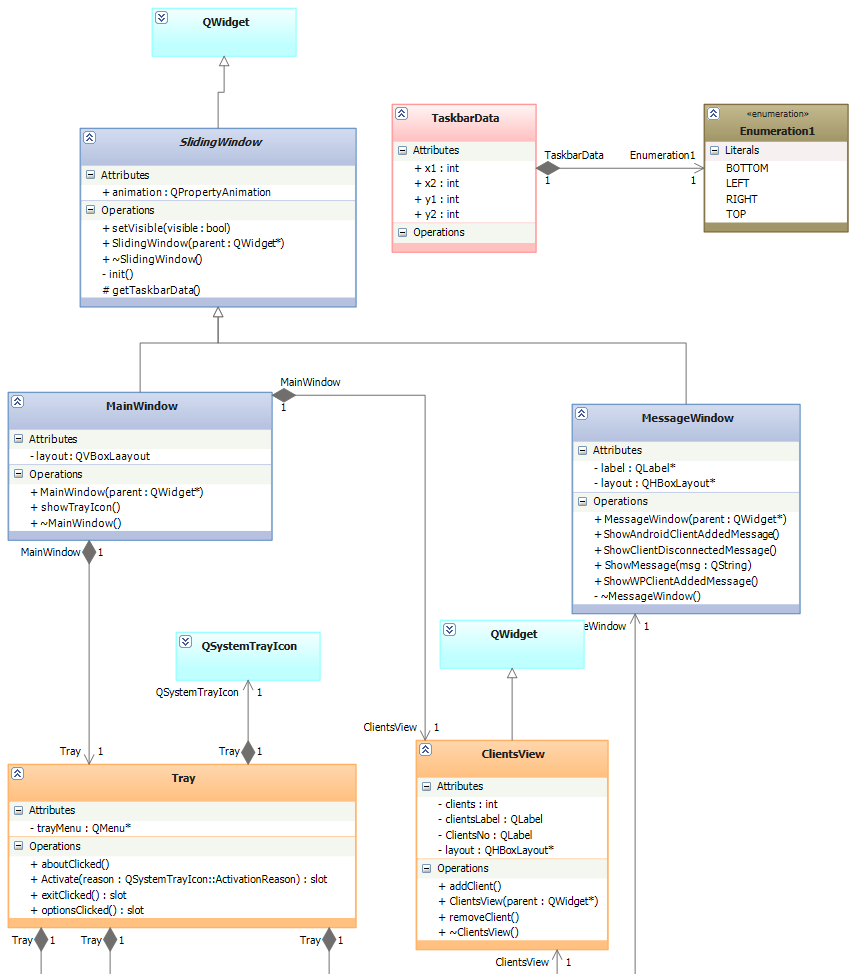
* + - 1. As a user I want to be able to add information about a server manually.
      2. As a user I want to be able to control my cursor as if my device was a real touchpad.
      3. As a user I want to be able to control my keyboard via device with client installed.
      4. As a user I want to be able to control my media buttons: play, next, previous, volume up, volume down.
      5. As a user I want to be able to choose sensitivity of touchpad.
      6. As a user I want to be able to choose sensitivity of scrolls.
      7. As a user I want the application to support multiple languages.
      8. As a user I want to be notified when the server rejects my attempt to connect.
      9. As a user I want to be able to enter password when it’s required to start the connection with the server.
      10. As a user I want to be able to download files from my PC to my mobile device.

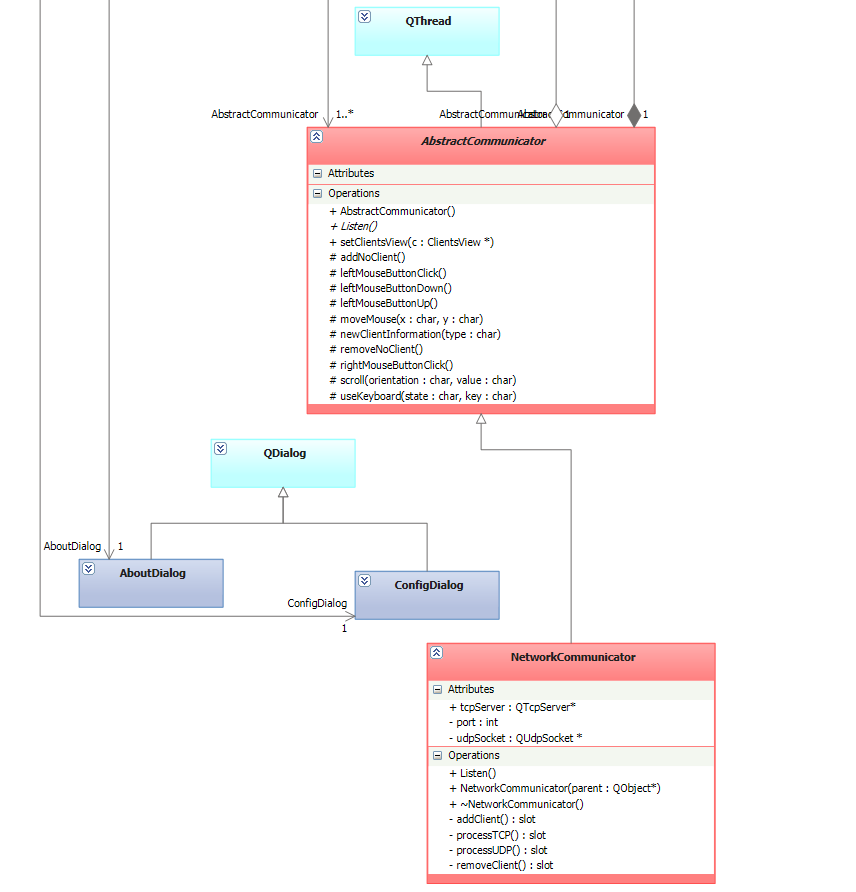
### Windows Phone client

* + - 1. As a user I want to be able to add information about a server manually.
      2. As a user I want to be able to control my cursor as if my device was a real touchpad.
      3. As a user I want to be able to control my keyboard via device with client installed.
      4. As a user I want to be able to control my media buttons: play, next, previous, volume up, volume down.
      5. As a user I want to be able to choose sensitivity of touchpad.
      6. As a user I want to be able to choose sensitivity of scrolls.
      7. As a user I want the application to support multiple languages.

## Diagrams

### Server class diagram





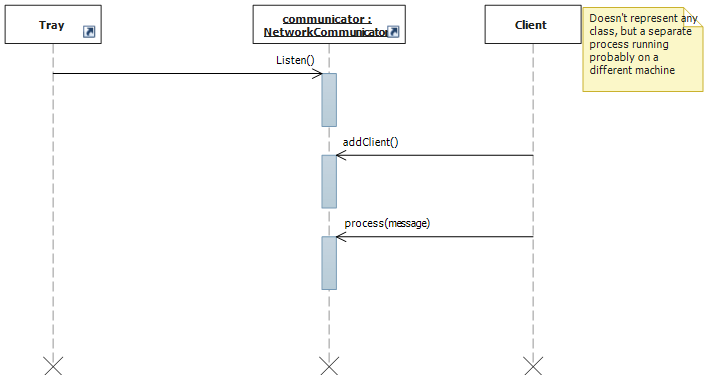
Main part of the whole application is Tray class. Even though it’s still just a part of MainWindow, it is visible all the time, while MainWindow is visible only after Tray is activated (tray icon is clicked by a user). Tray contains a set of AbstractCommunicators, although in this example we’ll use only NetworkCommunicator (inherits AbstractCommunicator) that allows to communicate with the clients via TCP (general events) and UDP (mouse movement event). This is also Tray that calls the Listen method in each AbstractCommunicator. Then Tray also contains the menu (visible after clicking tray icon with right mouse button) for managing the whole application.

### Server use case diagram



As a matter of fact, ClientsView is nothing more than just a Widget consisting of two labels. One that says “Number of clients” and the other one that actually represents the number.

### Server sequence diagram



### Android client class diagrams

### Android client use case diagram

### Android client sequence diagram

### Windows Phone client class diagrams





Data contained and generated by ServersStorage are common for all classes across the application, therefore it’s much easier to make the entire class static, rather than pass a reference to it to all the objects used in the application.



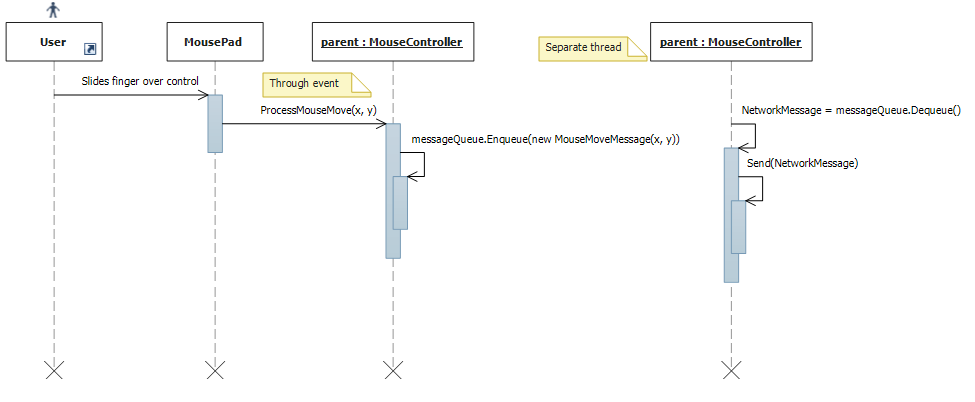
Mouse controller is a UserControl that contains two other user controls: MousePad and MouseScroll.

Each time MousePad or MouseScroll recognize a gesture, they raise an event that is captured later on by MouseController. Then depending on the event MouseController generates a proper NetworkMessage and puts it into a messageQueue. Within MouseController there is a separate thread working in the background, that continuously reads the messageQueue and sends data to the server.

### Windows Phone client use case diagram



### Windows Phone client sequence diagram



## Layout design

# Implementation

## Network messages

In project two kind of transport protocols are used: TCP and UDP.

The reason is because TCP generates small delays (since it performs multiple data integrity checks) that could decrease the comfort of using the application, as well as it requires to establish connection, what is impossible during for instance server detection (it’s easier to simply send one UDP packet to broadcasting address and then wait for responses).

# Summary

# Dictionary

* + - 1. Client – any application that is able to communicate with the PCRemote Server
      2. Mobile device – device running either Windows Phone (at least 7.1 version) or Android (at least 2.3 version) operating system with client application installed
      3. MessageQueue – Queue (First-In-First-Out) containing set of NetworkMessage objects.

# Bibliography

* + - 1. Android developers – official documentation - <http://developer.android.com>
      2. Official JUnit documentation- <http://www.junit.org/>
      3. Microsoft Developer Network (MSDN) - <http://msdn.microsoft.com/en-US/>
      4. Qt Project - <http://qt-project.org/doc/qt-4.8/tutorials.html>